

3.1

CONVERGECFD MANUAL SERIES

CONVERGE MANUAL



Table of Contents

Chapter 1 Introduction	19
1.1 Proprietary Notices and Acknowledgments	21
1.2 Citation and Contact Information	24
1.3 Units in CONVERGE	25
Chapter 2 Release Notes	27
2.1 3.1.9	28
2.2 3.1.8	36
2.3 3.1.7	37
2.4 3.1.6	44
2.5 3.1.5	53
2.6 3.1.4	61
2.7 3.1.3	70
2.8 3.1.2	76
2.9 Major Changes from CONVERGE 3.0 to 3.1	81
2.10 Manual Reorganization from CONVERGE 3.0 to 3.1	85
Chapter 3 Governing Equations	87
3.1 Mass and Momentum Transport	88
Multiple Reference Frame Approach	89
3.2 Equation of State	91
3.3 Energy Transport	93
3.4 Species Transport	96
3.5 Scalar and Passive Transport	99
3.6 Turbulence Transport	100
3.7 Electric Potential	101
Chapter 4 Numerics	103
4.1 Finite Volume	104
4.2 Solution Procedure	106
4.3 Iterative Algorithms	107
PISO Algorithm	108
SIMPLE Algorithm	113
Pressure-Based PISO and SIMPLE	116
4.4 Rhie-Chow Algorithm	116
Legacy Rhie-Chow Scheme	117

Generalized Rhie-Chow Scheme	118
4.5 Non-Local Convective Flux Schemes	121
MUSCL Scheme	121
Limiter Functions	122
4.6 Iterative Linear Solvers	126
SOR Algorithm	127
CONVERGE BiCGSTAB Method	129
HYPRE BiCGSTAB Method	132
Preconditioners	132
GPU-Enabled Solvers	135
Nonlinear Krylov Acceleration Method	136
Chapter 5 Time Control Methods	139
5.1 Solvers	140
Transient Solver	140
Steady-State Solver	144
5.2 Time-Step Control	152
5.3 Super-Cycling	156
Super-Cycle Stages	160
Super-Cycle Output	160
Cylinder Duplication	161
5.4 Fixed Flow Method	162
Fixed Flow Setup	164
Fixed Flow Applications	166
Chapter 6 Optimization Techniques	167
6.1 CONGO Optimization and Model Interrogation	168
Genetic Algorithm	168
Design of Experiments	171
Running CONGO	171
Chapter 7 Parallel Processing	181
7.1 Parallelization for the Transport Equations	182
7.2 Parallelization for the SAGE Solver	185
7.3 Hardware Considerations	185
Chapter 8 Surface Preparation	187
8.1 Surface Geometry	188
8.2 Surface Defects	188
Open Edges	189

Nonmanifold Edges	189
Intersecting Triangles	191
Overlapping and Sliver Triangles	192
Normal Orientation	195
8.3 Periodic Surfaces	195
8.4 Boundary Identification	196
8.5 Moving Boundaries	196
Surface Defects Caused by Motion	196
Surface Alignment Issues Caused by Motion	198
8.6 Sealing of Boundaries	199
Sealing Methods	201
Sealing Logic	203
Sealing Considerations and Recommendations	205
Chapter 9 Boundaries and Boundary Conditions	207
9.1 INFLOW and OUTFLOW	208
Velocity Boundary Conditions	209
Pressure Boundary Conditions	213
Temperature Boundary Conditions	216
Species Boundary Conditions	216
Passive Boundary Conditions	217
Turbulent Kinetic Energy Boundary Conditions	217
Turbulent Dissipation Boundary Conditions	218
Specific Dissipation Rate Boundary Conditions	219
OUTFLOW Backflow Boundary Conditions	220
Navier-Stokes Characteristic Boundary Conditions	221
9.2 WALL	227
Sliding WALL Boundary Conditions	227
Moving WALL Boundaries	228
Reloft WALL Boundaries	234
Linked WALL Boundaries	236
Velocity Boundary Conditions	237
Pressure Boundary Conditions	241
Temperature Boundary Conditions	241
Species Boundary Conditions	249
Passive Boundary Conditions	249
Turbulent Kinetic Energy Boundary Conditions	249
Turbulent Dissipation Boundary Conditions	250
Specific Dissipation Rate Boundary Conditions	250

Electric Potential Boundary Conditions	251
9.3 PERIODIC	251
9.4 SYMMETRY	253
9.5 TWO_D	253
9.6 INTERFACE	254
9.7 GT-SUITE	258
9.8 MIXING_PLANE	258
Chapter 10 Regions and Streams	263
10.1 Regions	264
Dependent Regions	264
Solid Regions	265
Region Connection and Disconnection	265
Events	270
10.2 Streams	273
Stream-Based Overrides for Multi-Stream Simulations	273
Setup for Connected Fluid Streams	275
Chapter 11 Grid Control	279
11.1 Grid Generation	280
11.2 Cell Pairing	282
11.3 Grid Scaling	283
11.4 Fixed Embedding	284
11.5 Adaptive Mesh Refinement	286
11.6 Inlaid Meshing	292
Chapter 12 Initialization	301
12.1 Restarting	302
12.2 Mapping	302
12.3 Specified Initial Values	304
Velocity Initialization in the Piston	304
Velocity Initialization in the Cylinder	305
Hydrostatic Pressure Initialization	307
Chapter 13 Turbulence Modeling	309
13.1 RANS Models	311
k- ϵ Models	312
k- ω Models	323
Reynolds Stress Models	329
Spalart-Allmaras Model	335

RANS Boundary Conditions	336
Typical Parameter Values for RANS Models	352
13.2 LES Models	354
Zero-Equation LES Models	356
One-Equation LES Models	360
Two-Equation LES Models	362
LES Boundary Conditions	363
Typical Parameter Values for LES Models	364
13.3 Hybrid RANS/LES Models	364
13.4 Turbulence Statistics Output	365
Chapter 14 Discrete Phase Modeling	369
14.1 Parcel Equations of Motion	370
14.2 Parcel Time-Step Control	372
14.3 Parcel Post-Processing Parameters	373
14.4 Parcel Settings	374
Liquid Parcels	374
Solid Parcels	417
14.5 Parcel Introduction	422
Nozzle Injection	422
Boundary Injection	431
14.6 Parcel-Parcel Interaction	433
Adaptive Collision Mesh	433
Collision Models	435
Outcome Models	441
14.7 Parcel-Wall Interaction	445
Liquid Parcels	445
Solid Parcels	478
14.8 Eulerian-Lagrangian Spray Atomization	486
Bidirectional VOF-DPM-VOF	490
Chapter 15 Volume of Fluid Modeling	491
15.1 Void Fraction Solution Method	494
15.2 Individual Species Solution Method	496
15.3 Front Capturing Schemes	498
15.4 Mixture Model	504
15.5 Surface Tension and Wall Adhesion	507
15.6 Cavitation Modeling	509
15.7 VOF-Spray One-Way Coupling	512

15.8 Dissolved Gas Modeling	517
Chapter 16 Combustion Modeling	519
16.1 Premixed Combustion Models	520
16.2 Non-Premixed Combustion Models	521
16.3 SAGE Detailed Chemical Kinetics Solver	522
Thickened Flame Model	526
Three-Point PDF Method	531
16.4 Chemical Equilibrium (CEQ) Model	533
16.5 Flamelet Generated Manifold (FGM) Model	534
16.6 G-Equation Model	537
16.7 Extended Coherent Flame Models	544
Extended Coherent Flame Model (ECFM)	545
3-Zone Extended Coherent Flame Model (ECFM3Z)	559
ECFM for LES	562
ECFM/ECFM3Z Setup	566
16.8 Shell+CTC Model	572
16.9 Representative Interactive Flamelet (RIF) Model	590
16.10 Eddy Dissipation Model (EDM)	597
16.11 Flamespeed Calculation	599
16.12 Surface Chemistry Model	604
Two-Step Surface Chemistry	607
Coupled Gas-Surface Chemistry	609
Reaction Options	610
Effectiveness Factor	611
16.13 Adaptive Zoning	613
16.14 Skip Species	617
16.15 Combustion Time-Step Control	618
16.16 Combustion-Related Output	618
Chapter 17 Emissions Modeling	621
17.1 NOx Modeling	622
Thermal NOx Model	622
Prompt NOx Model	628
Detailed Chemistry NOx	630
17.2 Soot Modeling	630
Empirical Soot Model: Hiroyasu-NSC	632
Phenomenological Soot Models	635
Detailed Soot Models	641

Soot Modeling Case Setup	655
17.3 Emissions Post-Processing	656
Chapter 18 Source Modeling	659
18.1 Energy Source Modeling	661
Thermal Runaway Source Modeling	662
Energy Sources in Engine Models	668
18.2 Momentum Source Modeling	671
18.3 Turbulent Kinetic Energy Source Modeling	671
18.4 Turbulent Dissipation Source Modeling	671
18.5 Specific Dissipation Source Modeling	672
18.6 Species Source Modeling	672
18.7 Passive Source Modeling	673
18.8 Porous Media Source Modeling	673
18.9 Baffle Media Source Modeling	680
18.10 Battery Heat Source Modeling	680
18.11 Relaxation Zone Source Modeling	682
Chapter 19 Conjugate Heat Transfer Modeling	685
19.1 1D CHT	686
19.2 3D CHT	688
Solid Properties	689
Fluid and Solid Streams	689
Boundary Conditions	690
19.3 GT-SUITE CHT Coupling	694
Chapter 20 Fluid-Structure Interaction Modeling	695
20.1 Force and Moment Calculations	696
20.2 Equations of Motion: Rigid Body Dynamics	696
20.3 FSI Setup	699
20.4 FSI Beam	705
20.5 FSI Events	710
20.6 FSI Stiction	710
20.7 FSI Spring	711
20.8 GT-SUITE/FSI Coupling	714
20.9 Abaqus/FSI Coupling	714
Chapter 21 Radiation Modeling	717
21.1 Spherical Harmonics Method	719
21.2 Finite Volume Method	720

FVM Governing Equations	720
Spatial Discretization	721
Scattering Phase Function	723
21.3 Gas Models	723
Gray Gas Model	723
Nongray Gas Radiation Models	724
21.4 Radiation/Spray Coupling	725
21.5 Radiation/Energy Coupling	726
21.6 Non-Transport Passives and Post-Processing	726
Chapter 22 Aeroacoustic Modeling	729
22.1 Direct Methods	731
22.2 Reduced-Order Models	731
Far-Field Methods	732
Near-Field Methods	736
Chapter 23 Utilities	739
23.1 Pre-Processing	740
Make Surface	740
Mesh Convert	741
Extract Profile	742
Thermodynamic File Cleanup	745
Mechanism File Cleanup	745
Transport File Cleanup	746
Fluid Property Calculator	746
Extract Sub-Mechanism	746
23.2 Post-Processing	747
Post Convert	747
HTC Mapping	750
Proper Orthogonal Decomposition	751
Sensitivity Output Convert	753
23.3 Chemistry	754
Zero-Dimensional Chemistry Utilities	754
One-Dimensional Chemistry Utilities	768
Mechanism Reduction	775
Mechanism Merge	781
Mechanism Tune	786
Surrogate Blender	790
Pathway Flux Analysis (PFA)	791

Chapter 24 File Overview	795
24.1 Input and Data File Overview	796
Check Inputs	797
Hidden Input Parameters	797
24.2 Simulation Logs	797
Screen Output	797
Log File	800
Active Variables Log	807
Process ID Log	808
Latency Log	809
24.3 Output File Overview	809
Echo Files	811
Restart Files	812
Map Files	812
Cell-Averaged Output Files	813
Cell-By-Cell Post Output Files	813
Chapter 25 Input and Data Files	815
25.1 Spatial and Temporal Profiles	817
Spatial Profile Format	817
Temporal Profile Format	818
Spatial and Temporal Profile Format	819
25.2 Applications	820
Engine Parameters: engine.in	820
Crevice Model: crevice.in	821
Non-Engine RPM: rpm.in	830
Variable RPM: var_rpm.in	830
Generic Motion: motion_sets.in	831
Region-Specific Setup Considerations for Engine Modeling	834
25.3 Materials	835
Gas Properties: gas.dat	836
Gas Properties: transport.dat	836
Gas Properties: schmidt_species.dat	837
Species-Dependent Gas Properties: species_transport.in	838
Species-Dependent Critical Properties of Gases: crit_cond.dat	838
Liquid Properties: liquid.dat	840
Species Data and Reaction Mechanism: mech.dat	843
Custom Fluid Properties: fluid_properties.in and fluid_properties.dat	856

Solid Properties: solid.dat	858
Anisotropic Conductivity: aniso_cond.in	859
Species Definition: species.in	866
Composite Species: composite.in	870
Thermodynamic Properties: therm.dat	872
Thermodynamic Properties: tabular_therm.dat	877
Lower Heating Value: lhv.in	878
Battery Properties: battery_properties.dat	879
Open Circuit Voltage: vocv.dat	880
25.4 Simulation Parameters	880
Surface Geometry: surface.dat	880
Surface List: surface_list.in	882
Stream Setup	884
Simulation Control	887
Region-Dependent CFL Number: max_cfl.in	907
Solver Parameters: solver.in	908
Steady-State Monitor: monitor_steady_state.in	924
Multiple Reference Frame Approach: mrf.in	928
Fixed Flow Method: fixed_flow.in	929
Skip Species: skip_species.in	931
Mixing Plane: mixing_plane.in	933
25.5 Boundary Conditions	935
Boundary Conditions: boundary.in	936
Arbitrary Motion Profile	959
Valve Motion Profile	960
Boundary Motion Profile	961
Pump Mass Flow: pump_massflow.in	962
Wall Values: wall_value.in	963
Co-simulation: cosimulation.in	964
25.6 Initial Conditions and Events	968
Domain Initialization: initialize.in	969
Events: events.in	972
Mapping Variables: map.in, map.h5, map_parcel.h5	976
Initialization Perturbation Modeling: initial_perturbation_region.in	982
25.7 Physical Models	984
Discrete Phase Modeling	984
Combustion Modeling	1029
Emissions Modeling	1062
Turbulence: turbulence.in	1071

Source Modeling	1079
Volume of Fluid Modeling	1099
Super-Cycling: <i>supercycle.in</i>	1104
Surface Duplication for CHT: <i>supercycle_surface_map.in</i>	1107
1D Conjugate Heat Transfer Modeling	1108
Radiation: <i>radiation.in</i>	1112
Fluid-Structure Interaction Modeling	1116
Surface Chemistry Modeling	1132
Nucleate Boiling: <i>nucleate_boiling.in</i>	1139
Urea: <i>urea.in</i>	1140
Aeroacoustics: <i>acoustics.in</i>	1142
Erosion: <i>erosion.in</i>	1144
25.8 Grid Control	1148
Adaptive Mesh Refinement: <i>amr.in</i>	1148
Fixed Embedding: <i>embedded.in</i>	1167
Grid Scaling: <i>gridscale.in</i>	1173
Inlaid Mesh: <i>inlaid_mesh.in</i>	1174
Minimum Volume Ratio: <i>min_vol_ratio.in</i>	1175
Proximity Boundaries: <i>proximity_boundaries.in</i>	1177
25.9 Pre-Processing	1178
Make Surface: <i>makesurface.in</i>	1178
25.10 Output/Post-Processing	1180
Post Variable Selection: <i>post.in</i>	1180
Wall Output: <i>wall_output.in</i>	1193
Swirl, Tumble, and Angular Momentum: <i>dynamic.in</i>	1194
Flow Between Regions: <i>regions_flow.in</i>	1196
SMD and Drop Distribution: <i>smd_pdf.in</i>	1199
Parcel Radius Distribution: <i>radius_pdf.in</i>	1199
Phase Doppler Particle Analyzer: <i>pdpa.in</i>	1200
Monitor Points: <i>monitor_points.in</i>	1201
Monitor Cloud Data: <i>cloud.in</i>	1205
Uniformity Index: <i>uniformity_index.in</i>	1207
Custom Species Output: <i>species_output.in</i>	1209
Mapping File Frequency: <i>write_map.in</i>	1210
Proper Orthogonal Decomposition: <i>pod.in</i>	1211
In Situ Post-Processing: <i>paraview_catalyst.in</i>	1212
25.11 User-Defined Functions	1215
User-Defined Functions: <i>udf.in</i>	1215
25.12 Chemistry	1215

Zero-Dimensional Chemistry Utilities	1215
One-Dimensional Chemistry Utilities	1241
Mechanism Reduction	1255
Mechanism Tune: mechanism_tune.in	1259
Surrogate Blender: blender.in	1264
Pathway Flux Analysis: pfa.in	1266
25.13 Heat Transfer Mapping	1267
Heat Transfer Mapping: htc_map.in	1268
Heat Transfer Output Control: transfer.in	1274
25.14 CONGO	1275
GA/DoE Setup: congo.in	1275
Case: case.in	1278
Execution: execute.in	1284
Merit: merit.in	1286
UDI: udi.in	1289
GA to DOE: gatodoe.in	1290
Chapter 26 Output Files	1293
26.1 3D Simulation Output Files	1294
acoustic_receiver_<ID>_stream_<ID>_pressure.out	1294
amr.out	1294
area_avg_flow.out	1295
area_avg_regions_flow.out	1296
az_info<dump number>_<dump time>.out	1299
battery_num_<number>_stream_<stream ID>.out	1301
beam_<name>_monitor_point_<ID>.out	1302
borghi.out	1302
bound<boundary ID>-wall.out	1303
cell_count_ranks.out	1305
cell_count_regions.out	1306
cht1d_point<number>_bound<boundary ID>.out	1306
crevice.out	1307
crevice_rings.out	1308
dmr_mech_info.out	1308
drill_bit_profile<boundary ID>.out	1309
dynamic.out	1310
ecfm.out	1314
ecfm3z.out	1317
emissions.out	1320

equiv_ratio_bin.out	1320
erosion_boundary_<boundary ID>.out	1321
film_urea.out	1322
fsi_object_<name>.out	1322
fsi_spring_<index>.out	1325
g_eqn.out	1325
gti_interface.out	1326
issim_ignition_<number>.out	1326
lhv_info.out	1327
liquid_parcel.out	1328
liquid_parcel_ecn.out	1330
liquid_parcel_film.out	1330
liquid_parcel_film_accum.out, liquid_parcel_film_accum_net.out	1331
liquid_parcel_film_scrape.out	1331
map_<time>.h5	1332
map_bound<boundary ID>_<time>.out	1333
map_parcel_<time>.h5	1334
mass_avg_flow.out	1335
mass_avg_regions_flow.out	1336
mass_balance.out	1339
memory_usage.out	1340
mixing.out	1340
monitor_line_<number>_<average type>_avg.out	1342
monitor_line_<number>_coordinates.out	1343
monitor_point_<number>_<average type>_avg.out	1343
numerics.out	1344
parcel_flow.out	1345
passive.out	1346
phenom_soot_model.out	1346
piston_profile<boundary ID>.out	1347
planes_flow.out	1347
progressive_cavity_rotor_profile<boundary ID>.out	1348
radius_pdf.out	1349
react_ratio_bin.out	1349
residuals.out	1350
rot_object<number>-wall_power_torque.out	1351
scalar_diss_rate.out	1352
skip_species.out	1352
smd_pdf.out	1353

solid_parcel.out	1353
solid_parcel_ecn.out	1354
soot_hiroy.out	1355
soot_pm_model.out	1356
soot_psm_model.out	1357
source_energy.out	1358
species_mass.out	1359
species_mass_frac.out	1359
species_mass_src.out	1360
species_mole_frac.out	1360
species_std_masfrac.out	1361
species_vol.out	1361
spray_rate_inj<injector ID>.out	1362
spray_urea_file.out	1364
steady_state.out	1364
supercycle_point<number>.out	1365
supercycle_stream<number>_balance.out	1366
surface_species_cov.out	1366
temperature.out	1367
thermo.out	1367
time.out	1367
transfer.out	1368
turbulence.out	1370
user_pdpa_<time>.out	1372
vof_spray.out	1373
volumes.out	1374
wall_stress<number>_<time>.out	1374
walltime.out	1375
zero_d_cases_rank<number>.out	1376
zero_d_solver.out	1376
26.2 Thermodynamic Output File	1377
thermo.out	1377
26.3 Mechanism Check Output Files	1378
mech_check.out	1378
surface_mech_check.out	1379
26.4 HTC Mapping Output Files	1380
htc_triangles.map	1380
htc_vertices.map	1381
26.5 Proper Orthogonal Decomposition Output File	1382

energy_fraction.out	1382
26.6 0D Simulation Output Files	1383
extinction_limit.out	1383
ignition_det.out	1383
sens<case ID>.out	1385
sens<case ID>_var<number>_<name>*.out	1386
species_info.dat	1387
zero_d_adjoint.out	1387
zero_d_adjoint_case<case ID>.out	1388
zero_d_adjoint_rank.out	1389
zero_d_end_time_species.out	1389
zero_d_sol_case<case ID>.out	1389
26.7 1D Simulation Output Files	1392
flamespeed.out	1392
one_d_flamespeed.out	1392
one_d_end_distance_species.out	1393
one_d_sens.out	1393
one_d_sens_case<caseID>.out	1394
one_d_sens_rank.out	1395
one_d_sol_case<case ID>.out	1395
one_d_strain_rate.out	1396
schmidt_species_case<number>.dat	1396
tlf_table_failed_cases.out	1396
26.8 Mechanism Reduction Output Files	1397
ignition_det.out	1397
ignition_ske.out	1397
mech_lump.dat	1398
mech_ske.dat	1398
one_d_flamespeed_det.out	1398
one_d_flamespeed_ske.out	1398
26.9 Mechanism Tune Output Files	1399
mechanism_tune.out	1399
nominal.out	1400
nominal_perform.out	1400
tune_group_reac_sens.out	1401
tune_perform.out	1401
tune_reaction.out	1401
tune_reaction_factors.out	1402
tune_reaction_results.out	1402

tune_target_results.out	1403
26.10 Surrogate Blender Output File	1403
blender.out	1403
26.11 Pathway Flux Analysis Output Files	1404
graphviz_input_<number>.out	1404
rate_tab_<number>.out	1404
spec_mol_<number>.out	1404
26.12 CONGO Output Files	1405
Individual Run Directory Output Files	1405
Main CONGO Folder Output Files	1405
Chapter 27 References	1409

Chapter



Introduction

1 Introduction

CONVERGE is a revolutionary computational fluid dynamics (CFD) program that eliminates the grid generation bottleneck from the simulation process. CONVERGE was developed by engine simulation experts and is straightforward to use for both engine and non-engine simulations. Unlike many CFD programs, CONVERGE automatically generates a perfectly orthogonal, structured grid at runtime based on simple, user-defined grid control parameters. This grid generation method completely eliminates the need to manually generate a grid. In addition, CONVERGE offers many other features to expedite the setup process and to ensure that your simulations are as computationally efficient as possible.

This manual describes how to use CONVERGE to simulate three-dimensional, incompressible or compressible, transient or steady-state, chemically reacting flows in complex geometries with stationary or moving surfaces. CONVERGE can perform calculations with any number of species and chemical reactions, as well as transient liquid sprays and laminar or turbulent flows.

Innovative Gridding Methods in CONVERGE

Traditionally, boundary-fitted grids morph the vertices and cells in the interior of the domain to conform to the shape of the geometry. There are two significant disadvantages to using a traditional boundary fitted grid. First, whether structured or unstructured, fitting a grid to a complex geometry prevents the use of simple orthogonal grids, which in turn eliminates the benefits of numerical accuracy and computational efficiency associated with orthogonal grids. Second, generating a traditional boundary-fitted grid for a complex moving geometry can be time consuming and difficult. Often the grid generation difficulties and significant time requirements are a roadblock to simulating complex moving geometries such as an internal combustion engine.

CONVERGE uses a different, better strategy: an innovative boundary-fitted approach eliminates the need for the computational grid to coincide with the geometry of interest. This method has two significant advantages. First, the type of grid used is chosen for computational efficiency instead of geometry. This process allows the use of simple orthogonal grids, which simplifies the numerics of the solver. Second, the grid generation complexity and the time required are greatly reduced, as the complex geometry needs to be mapped only onto the underlying orthogonal grid. You are required to provide only a file containing the surface geometry represented as a closed triangulated surface. This file is easily written in Stereo Lithography (STL) format in most CAD packages. Given a proper STL file for the geometry of interest, it will take mere minutes to prepare a surface for even complex geometries. Note that this user time is not spent creating a grid, as CONVERGE performs the grid generation internally at runtime. Your time is spent uniquely identifying various portions of the surface so that you can specify mesh motion and boundary conditions.

At runtime, CONVERGE uses the given triangulated surface to cut the cells that are intersected by the surface. There are many benefits of generating the grid internally by the code at runtime rather than requiring you to generate the grid as an input to the code.

Runtime grid generation allows the grid to be changed during the simulation. Possible changes include scaling the cell size of the entire domain, locally refining or coarsening during the simulation, and adaptively refining the mesh. Another major advantage of runtime grid generation is the ability of CONVERGE to regenerate the grid near moving boundaries during the simulation without any input from you. This regeneration means that setting up a case with a moving boundary is no more difficult than setting up a stationary case.

Powerful Physical Models in CONVERGE

In addition to its novel approaches to grid generation and boundary treatment, CONVERGE includes state-of-the-art numerical techniques and models for physical processes including turbulence, spray, combustion, conjugate heat transfer, and cavitation. With these models, CONVERGE can simulate a wide variety of flow problems.

1.1 Proprietary Notices and Acknowledgments

Limitation of Warranty

Except as provided in any written software license agreement with the licensee, the licensed software is provided "as is," "where is," and "as available" and without any representations, warranties, or conditions of any kind, express or implied, including without limitation, any implied warranties of merchantability, fitness for a particular purpose, title, or non-infringement.

Licensee's sole remedy for any dissatisfaction with licensed software is to stop using the licensed software.

To the fullest extent permitted by applicable law, under no circumstances shall licensor or its officers, employees, successor, or assigns be liable for direct damages or any incidental, special, consequential, exemplary, punitive, or other indirect damages whatsoever (including damages for loss of profits, goodwill, use, data, or other intangibles) arising out of or in any way connected with licensed software provided hereunder whether based on contract, tort, negligence, strict liability or otherwise, even if licensor or its related parties have been advised of the possibility of such damages.

Anti-Piracy Notice

Software piracy is a crime, and Convergent Science reserves the right to embed a software security mechanism within its CONVERGE software to monitor usage and thus verify compliance with this license. Such a security mechanism may store data relating to the usage of CONVERGE and the number of times it has been copied, or this security mechanism may communicate with computers controlled by Convergent Science over any type of communications link to exchange communications and report data relating to the usage of the software and the number of times it has been copied. Convergent Science reserves the right to use a hardware lock device, license administration software, and/or a license authorization key to control access to CONVERGE. Taking steps to avoid or defeat the purpose of these measures violates the terms of this license. Using CONVERGE without any required lock device or authorization key is prohibited.

Chapter 1: Introduction

Proprietary Notices and Acknowledgments

CONVERGE License Types

CONVERGE is primarily designed as a commercial research and development engineering product. Full commercial CONVERGE licenses support commercial and proprietary usage without scale limits, subject to the number and type of commercial licenses available on your system.

Academic CONVERGE licenses are available for eligible institutions. Academic licenses are full-capability licenses and are not subject to core or cell count limits. They cannot be used for commercial or proprietary purposes, but open-literature research is allowed.

The CONVERGE Explore program is a professional development program that helps established and aspiring engineers expand their simulation skill sets. CONVERGE Explore licenses support all CONVERGE features and capabilities, except that CONVERGE Explore is limited to 500,000 cells and 8 cores. CONVERGE Explore licenses cannot be used for commercial or proprietary purposes, but open-literature research is allowed.

For more details about the terms of your license, please consult your specific license agreement. For commercial and academic licensing information, please email licensing@convergecfd.com. To inquire about the CONVERGE Explore program, please email explore@convergecfd.com.

Other Licenses

CONVERGE is enhanced with visualization software from Tecplot, Inc. of Bellevue, Washington, USA. Tecplot for CONVERGE is licensed to be used in conjunction with CONVERGE applications and not for any other use. Use of Tecplot for CONVERGE software for plotting data not generated from CONVERGE applications is prohibited.

The CEQ equilibrium solver is included under license agreement with Ithaca Combustion Enterprise, LLC.

The ParMETIS parallel domain decomposition library is included under license agreement with the Regents of the University of Minnesota.

NLOpt is dynamically linked to CONVERGE in accordance with the terms of the LGPL license.

CONVERGE includes software packages and libraries that are included under open-source licenses. The following table lists the package or library name, the license type under which it is included, and the license file name. You can find the additional license information in the *doc/licenses* directory of your CONVERGE installation, or in the *CONVERGE 3.1/Documentation/ThirdPartyLicenses* directory on the Convergent Science Hub (hub.convergecfd.com/downloads). Note that a login is required.

Chapter 1: Introduction

Proprietary Notices and Acknowledgments

Table 1.1: Software packages and libraries included in CONVERGE under an open-source license.

Package or library name	License type	License file name
BLAS	None	<i>BLAS.txt</i>
Boost	Boost Software License	<i>Boost.txt</i>
Coolprop	MIT	<i>Coolprop.txt</i>
CGNS	zlib/libpng	<i>CGNS.pdf</i>
cxxopts	MIT	<i>cxxopts.txt</i>
GCC runtime	GPLv3 with exception	<i>GPL_v3.pdf, GPL_v3_exception.pdf</i>
Intel Data Analytics Acceleration Library (daal)	ISSL	<i>ISSL.pdf</i>
glibc	GPLv2.1	<i>LGPL_V2.1.txt</i>
Portable Hardware Locality (hwloc)	New BSD License	<i>New_BSD_License.txt</i>
HDF5	Modified BSD	<i>HDF5.txt</i>
HYPRE	GPLv2.1	<i>HYPRE.txt</i>
Lib C++	GPLv3 with exception	<i>GPL_v3.pdf, GPL_v3_exception.pdf</i>
METIS	Apache 2.0	<i>Apache_2.0.txt</i>
MKL	ISSL	<i>ISSL.pdf</i>
NLOpt	GPLv2.1, MIT	<i>LGPL_V2.1.txt, NLOpt.txt</i>
Nonlinear Krylov Acceleration (nlkain)	MIT	<i>nlkain.txt</i>
NVIDIA AMGX	BSD	<i>AMGX.txt</i>
NVIDIA CUDA	Custom	<i>CUDA.pdf, cuda_routines.txt</i>
OpenMP Runtime	GPLv3 with exception	<i>GPL_v3.pdf, GPL_v3_exception.pdf</i>
OpenMPI	BSD	<i>OpenMPI.txt</i>
ParaView	Custom BSD	<i>Paraview/License_v1.2.txt</i>
PETSc	BSD	<i>PETSc.txt</i>
Sphinx	BSD	<i>Sphinx.txt</i>
SUNDIALS	BSD	<i>SUNDIALS.txt</i>
SuperLU	BSD	<i>SuperLU.txt</i>
YAML	MIT	<i>YAML.txt</i>
zlib	Custom	<i>zlib.txt</i>

ParaView includes additional sub-modules that have their own license terms. These sub-modules are compliant with ParaView's BSD license. Refer to *doc/licenses/ParaView/<sub-module name>* for details on these individual licenses.

Chapter 1: Introduction

Proprietary Notices and Acknowledgments

Acknowledgments

Convergent Science acknowledges combustion modeling expertise from IFP Energies nouvelles.

Convergent Science acknowledges computing resources and industry expertise from Cray Inc., Intel Corporation, and NVIDIA Corporation.

Convergent Science acknowledges collaboration on adaptive zoning with Lawrence Livermore National Security, LLC.

1.2 Citation and Contact Information

Citations for Standard Licenses

If you are publishing work that includes CONVERGE results in conference proceedings, journal articles, book chapters, etc., please cite CONVERGE as follows:

Richards, K. J., Senecal, P. K., and Pomraning, E., CONVERGE 3.1, Convergent Science, Madison, WI (2023).

*Specify the release year of the minor version that you ran. This information can be found at the top of the [log file](#). If you do not have this information, specify the year in which you completed your simulations.

If you need to cite this manual, please use the following citation:

Richards, K. J., Senecal, P. K., and Pomraning, E., CONVERGE 3.1 Manual, Convergent Science, Madison, WI (2023).

Citations for Non-Commercial Licenses

If you are publishing CONVERGE results obtained via a free academic CONVERGE license or CONVERGE Explore license, please cite CONVERGE as shown above and include the following statement in your acknowledgments:

Convergent Science provided CONVERGE licenses and technical support for this work.

Contact

For licensing information, please email licensing@convergecfd.com. For CONVERGE support, please email support@convergecfd.com (US), supportEU@convergecfd.com (EU), or support.in@convergecfd.com (India). For general assistance, please contact Convergent Science at (608) 230-1500 or contact@convergecfd.com.

Chapter 1: Introduction

Units in CONVERGE

1.3 Units in CONVERGE

CONVERGE uses SI units (*meters, kilograms, seconds, Kelvin*) almost exclusively. Deviations from SI units are noted in this manual.

Note that, unless otherwise specified in the [reaction mechanism file](#), the chemistry inputs in that file contain reaction rate information with CGS (*centimeter, gram, second*) units. Also note that in this file, the activation energy is given in units of *calories/mole*. The reason for these units is to match the units used by CHEMKIN.

There are also some parameters in [combust.in](#) that use CGS units to match CHEMKIN. These units are specified in this manual.

Chapter



2

Release Notes

2 Release Notes

New builds of CONVERGE 3.1 are released regularly. When a new build is released, the accompanying release notes are appended to the beginning of this section.

CONVERGE Studio 3.1 releases are often concurrent with CONVERGE 3.1 solver releases. Release notes for CONVERGE Studio can be found in the CONVERGE Studio 3.1 Manual.

2.1 3.1.9

CONVERGE 3.1.9 is a minor release that includes enhancements and bug fixes.

Solver

Enhancement: For the v2-f and zeta-f turbulence models, you can now specify SOR for a preconditioner when solving the f transport equation with the [CONVERGE BiCGSTAB](#) linear solver.

Enhancement: Sped up the solution of the [energy transport](#) equation.

Enhancement: Enabled statistics calculations for all field variables that are available to be specified in [post.in](#). Previously, CONVERGE defined these variables only if they were listed in *post.in*. Now, CONVERGE defines these variables if they are listed in *post.in* or for [turbulent statistics](#) output.

Enhancement: Altered the logic of some [species](#) renormalization routines to reduce the frequency of temperature extrapolations.

Enhancement: You can now run a simulation with *inputs.in* > *solver_control* > [energy_solver](#) = TOTAL and with *solver.in* > Numerical_Schemes > strict_conserve_level greater than 0.

Enhancement: CONVERGE now applies a non-orthogonality correction when calculating the [Rhe-Chow](#) correction. This improves convergence behavior for grids that are non-uniform and/or non-orthogonal.

Bug fix: Fixed a bug in [load balancing](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug in the total energy [energy transport](#) solver that could cause CONVERGE to crash.

Bug fix: Fixed a bug in the time-step calculation for [GT-SUITE coupling](#) cases that could cause the time-step to be smaller than *dt_min* when consecutive recoveries occurred.

Bug fix: Fixed several bugs that could cause CONVERGE to crash during co-simulations with [GT-SUITE](#).

Bug fix: Fixed a bug that caused CONVERGE to calculate inter-region mass flow rates incorrectly for simulations using [MUSCL](#) as the global flux scheme.

Bug fix: Fixed a bug that caused the [energy transport](#) solver to run slowly for incompressible VOF simulations with CHT.

Bug fix: Fixed a bug in how CONVERGE calculated residuals with the [CONVERGE BiCGSTAB](#) linear solver. CONVERGE could run more BiCGSTAB iterations than necessary for the desired residual. This bug fix improves simulation speed.

Bug fix: Fixed a bug that caused CONVERGE to print unnecessary warnings to the [log file](#) for some [mixing plane](#) cases.

Bug fix: Fixed several typos in messages printed to the [log file](#).

Bug fix: Fixed a bug that prevented CONVERGE from correctly ending a co-simulation with [GT-SUITE](#) when *inputs.in* > *simulation_control* > *end_time* = GT.

Bug fix: Previously, CONVERGE printed a warning message about potential solver instability when *mult_dt_move* was greater than 0.5. To avoid unnecessary warnings, this message now appears only when *mult_dt_move* is greater than 1.5.

Bug fix: Fixed a bug that could cause CONVERGE to incorrectly recover when running a simulation with a [fixed time-step](#).

Input

Enhancement: You can now use a profile to set a spatially varying pressure or velocity boundary condition at [GT-SUITE](#) boundaries.

Enhancement: CONVERGE now exits with an error if a [stream](#) is defined with no cells assigned to it. Previously, this caused CONVERGE to crash.

Bug fix: Fixed a bug that caused CONVERGE to crash when a temporal profile was specified for *inputs.in* > *temporal_control* > *max_cfl_nu*.

Bug fix: For sealing cases, CONVERGE no longer reads the deprecated *removal_indicator* input from the [surface geometry file](#).

Output/Post-Processing

Enhancement: Reorganized some [MPI](#) function calls to improve runtime for cases with many regions or boundaries.

Enhancement: CONVERGE now supports ParaView 5.11.0 for simulations using in situ post-processing with [ParaView Catalyst](#). ParaView 5.11.0 is included in the CONVERGE 3.1.9 installation package. ParaView 5.10.0 is still supported, but it is not included with CONVERGE 3.1.9.

Enhancement: [ParaView Catalyst](#) inputs now support stream-based overrides.

Enhancement: If you use [ParaView Catalyst](#) with a script exported from CONVERGE Studio 3.1_10May2023 or a later version, the results are written in HDF5 CGNS format to *outputs_*/paraview_catalyst*.

Enhancement: When a simulation exits because of [geometry errors](#), such as intersecting triangles, CONVERGE now writes the current geometry to a surface (*.dat) file, which you can load into CONVERGE Studio for debugging purposes.

Enhancement: A new optional flag in *monitor_points.in*, *monitor_point_coords_out*, enables the reporting of coordinates for monitor points attached to a moving boundary.

Enhancement: You can now specify *specific_entropy* in *post.in* to direct CONVERGE to write specific entropy to *post*.h5* files. If and only if this is specified, CONVERGE will also write specific entropy to a new column in [thermo.out](#).

Enhancement: You can now output species mole fractions at [monitor points](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when writing [restart files](#) for simulations that use film modeling for liquid parcels.

Bug fix: Fixed bugs related to [ParaView Catalyst](#) that caused incorrect cyclic output timing and incorrect indexing when passing variables.

Bug fix: When CONVERGE started a simulation from a [restart file](#), the restart number was incorrectly included in the file names for *cht1d_point<number>_bound<boundary ID>.out* and *map_<time>.h5*. This issue has been fixed.

Chapter 2: Release Notes

3.1.9

Bug fix: The [monitor point](#) IDs in the file names for `cht1d_point<number>_bound<boundary ID>.out` did not match the IDs in `monitor_point_<number>_<average type>.avg.out`. This issue has been fixed.

Bug fix: Fixed a bug that prevented [ParaView Catalyst](#) from correctly deinterlacing N-dimensional field arrays.

Bug fix: Fixed a bug that would cause CONVERGE to crash if two [steady-state monitor](#) parameters had the same name in the same stream.

Bug fix: Previously, when `source_value` was specified in [post.in](#), CONVERGE would output the total source value. CONVERGE now correctly writes the source value per unit volume.

Bug fix: Fixed a bug that could cause CONVERGE to crash when a [post.in](#) variable was specified in a stream where it could not be calculated.

Bug fix: Fixed a bug that caused CONVERGE to crash when total pressure or total temperature were specified as [uniformity index](#) variables.

Bug fix: Fixed a bug that caused CONVERGE to write incorrect `diff-mean` and `std-dev` to `steady_state.out` when `monitor_steady_state.in > variables > variable > variable_name = MASS_FLOW_RATE_NET`.

Boundaries and Boundary Conditions

Enhancement: Some [law-of-the-wall](#) heat transfer models are now treated implicitly when heat flux is dependent only on temperature.

Enhancement: Reworded several error and warning messages. The reworded messages are more clear.

Bug fix: Fixed a bug that caused CONVERGE to crash on a simulation with a [supersonic INFLOW](#) boundary.

Bug fix: Fixed a bug in semi-implicit treatment of Dirichlet [temperature boundary conditions](#).

Bug fix: If you specify the same [reference pressure](#) location for multiple boundaries, and the location is not located on any of those boundaries, CONVERGE no longer moves the reference pressure location. Previously, CONVERGE moved the reference pressure location to the closest point on one of the boundaries, which would cause the pressure distribution to be calculated incorrectly for the other boundaries.

Bug fix: Fixed a bug that could cause CONVERGE to crash when part of the domain was fully enclosed by [flow-through INTERFACE](#) boundaries.

Bug fix: Fixed a bug in error detection logic for reloft [WALL](#) boundaries. The error is now detected and written correctly.

Bug fix: Fixed a bug that caused CONVERGE to perform [electric potential](#) calculations incorrectly on curved walls.

Regions and Streams

Bug fix: Fixed a bug that could cause CONVERGE to crash when a solid [stream](#) had certain prescribed motion settings.

Grid Control

Enhancement: Previously, CONVERGE [printed a warning](#) when there were six scales of [embedding](#) or greater. The warning condition is now set to an embedding scale of twenty.

Enhancement: Added logic to detect some common setup mistakes for case setups with an [inlaid mesh](#).

Chapter 2: Release Notes

3.1.9

Enhancement: Altered some internal data structures to speed up simulations with moving [inlaid meshes](#).

Enhancement: Altered some [inlaid mesh](#) creation subroutines for improved performance.

Bug fix: Fixed a bug that caused CONVERGE to not apply region or shape [embedding](#) correctly in 2D simulations with multiple [streams](#).

Bug fix: Fixed a bug that could cause CONVERGE to not terminate normally after completing a simulation or running [check inputs](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with an [inlaid mesh](#) and proximity AMR.

Bug fix: Fixed a bug that caused CONVERGE to not apply [AMR](#) correctly in simulations using the steady-state monitor with the transient solver.

Bug fix: Fixed a bug in [sealing](#) that could lead to different triangulations on different ranks, causing CONVERGE to produce inaccurate results or crash.

Bug fix: When [gridscale.in](#) was used to set up [grid scaling](#) as a function of time, CONVERGE would crash if the file did not include a grid scale value for the start of the simulation. This issue has been fixed.

Bug fix: Fixed a bug in [sealing](#) that could unexpectedly remove disconnect triangles and cause CONVERGE to crash.

Bug fix: Fixed a bug that could cause CONVERGE to crash when [generating](#) a mesh.

Bug fix: Fixed a bug that could cause CONVERGE to not release [AMR](#) embedding when it should be released. This bug particularly affected gas-liquid interfaces in VOF simulations.

Bug fix: Fixed several bugs that could cause CONVERGE to crash on a simulation with [inlaid meshes](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with moving [inlaid meshes](#).

Bug fix: Fixed a bug in [sealing](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug that could cause CONVERGE to incorrectly detect negative cell volumes and print associated warning messages to the [log file](#).

Bug fix: Fixed a bug that caused CONVERGE to apply excess [AMR](#) in some [MRF](#) simulations.

Materials

Bug fix: Fixed a bug that caused CONVERGE to calculate the [Herschel-Bulkley non-Newtonian viscosity](#) incorrectly.

Bug fix: Fixed a bug that prevented CONVERGE from correctly reading NASA9-formatted [therm.dat](#).

Initialization

Enhancement: You can now initialize a pressure field with a [hydrostatic gradient](#).

Enhancement: Improved the logic of a case setup warning check. The previous logic printed an unnecessary warning for some cases.

Bug fix: Fixed a bug that caused CONVERGE to hang on engine cases with mapping enabled and one or more boundaries with [dependent regions](#).

Bug fix: Fixed a bug that could cause CONVERGE to read incorrect restart or map file data when [inputs.in](#) > [mpi_control](#) > [shared_memory_flag](#) = 1.

Discrete Phase

Enhancement: Updated an error message for the [KH-ACT](#) breakup model to improve clarity.

Enhancement: [liquid_parcel.out](#) now includes a column for Eulerian-to-Lagrangian mass transition.

Enhancement: Added logic to detect liquid parcels that are configured to use the [detailed decomposition](#) urea model but do not have all of the required urea species in the parcel setup. If detected, CONVERGE will exit with an error.

Enhancement: Improved warning and error messages for cases with improper configuration of parcel variables in [post.in](#).

Bug fix: Fixed bugs related to [post*.h5](#) output for solid parcels.

Bug fix: Fixed a bug in the [Bai-Gosman](#) film splash model that could cause inaccurate results in some circumstances.

Bug fix: Fixed a bug that caused CONVERGE to hang in certain cases when a parcel became stuck in the corner of an [inlaid mesh](#) cell. Now, CONVERGE prints a warning and removes the stuck parcel.

Bug fix: Fixed a bug in the [Phase Doppler Particle Analyzer](#) (PDPA) where parcels that passed through multiple overlapping PDPA planes were missing from [user_pdpa_<time>.out](#).

Bug fix: Fixed a bug that caused CONVERGE to hang in certain cases with boundary-specific [parcel-wall interaction](#) settings.

Bug fix: For variable RPM cases that use [tabular profiles](#) to control liquid parcel injection, CONVERGE now prints a warning if the injection is set to end after the simulation end time.

Bug fix: For spray cases with both [drop temperature discretization](#) and [film temperature discretization](#) active, CONVERGE used the wrong re-discretization function when a parcel splashed, causing a crash. This issue has been fixed.

Bug fix: Fixed a bug that caused an incorrect loss of parcel mass.

Bug fix: Previously, at the end of injection for a [boundary injector](#), if insufficient mass remained to form an additional parcel, this mass was not injected. CONVERGE now injects this remaining parcel mass at the end of injection.

Bug fix: Fixed a bug that caused CONVERGE to crash while zeroing out a parcel heat source.

Bug fix: Fixed a bug that caused some parcel [mapping](#) cases to crash.

Bug fix: Fixed a bug that caused CONVERGE to crash when [overall_equiv_ratio](#) was included in [post.in](#).

Bug fix: Fixed a bug that caused CONVERGE incorrectly calculate [spray-wall heat transfer](#).

Bug fix: Fixed a bug that caused CONVERGE to write size distribution for a single location when multiple locations were configured for [radius_pdf.out](#).

Bug fix: Fixed some problems with the boundary injector INJECT_WITH_FLOW option that caused too many parcels to be injected or parcels to be injected at the wrong location.

Bug fix: Fixed a bug that caused a fatal error when writing [surf_temp](#) output for [solid parcels](#).

Bug fix: Removed a warning message that incorrectly indicated that [parcel model sub-cycling](#) was not available.

Chapter 2: Release Notes

3.1.9

VOF

Enhancement: Added a warning message to indicate that the [dissolved gas model](#) is not recommended for compressible flows.

Bug fix: Fixed a bug that caused CONVERGE to calculate the initial density incorrectly for [incompressible VOF](#) streams with more than two species.

Bug fix: Fixed a bug that could cause excess temperature extrapolations in [VOF](#) simulations.

Bug fix: Fixed the calculation of the Rhie-Chow correction for [incompressible VOF](#) simulations with body force and $vof.in > face_density_approximation = ARITHMETIC_AVERAGE$. The previous way of calculating the Rhie-Chow correction caused numerical oscillations near embedding interfaces.

Bug fix: Fixed a bug that caused CONVERGE to split [vof_spray.out](#) into two files if the path to the output directory was very long.

Fixed Flow

Bug fix: Fixed a bug that caused mass to not be conserved during fixed flow intervals when species and density are solved during [fixed flow](#).

Combustion

Enhancement: CONVERGE will now print a warning if $combust.in > adaptive_zone_model > nox_flag = 1$ and N2 is not included in the simulation.

Enhancement: The [ECFM](#) and [ECFM3Z](#) combustion models can now treat H2 and CO as fuel species.

Enhancement: NH3 is now considered a combustible gas for comparison against [combust.in > general_control > hc_minimum](#).

Enhancement: The [FGM](#) combustion model now includes an option to output total energy.

Enhancement: The [surface chemistry](#) model now includes the thermal contributions from surface species when solving the energy equation.

Bug fix: The 0D [CEQ](#) solver now correctly writes [ceq_table.h5](#) to the [outputs_original](#) subdirectory instead of to the root Case Directory.

Bug fix: Fixed a bug in [surface chemistry](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug in the [surface chemistry](#) temperature calculation.

Bug fix: Fixed a bug that caused CONVERGE to crash while calculating the unburned temperature in non-combusting regions with [ECFM](#).

Bug fix: Fixed a bug in the [SAGE](#) combustion variable reaction multiplier that caused incorrect results.

Bug fix: Fixed two bugs that caused erroneous results in an LES [ECFM](#) case.

Bug fix: Fixed a bug in adaptive zoning that could cause CONVERGE to crash when running a simulation with non-hydrocarbon combustion.

Bug fix: Fixed a bug in the [ECFM](#) combustion model that caused an erroneous calculation in the transport of unburned enthalpy.

Bug fix: Fixed a bug in [G-Equation](#) initialization that caused erroneous output when multiple ignitions occurred during the same cycle.

Emissions

Enhancement: [Phenomenological soot models](#) are now supported when running a simulation with [G-Equation](#), as long as at least one of $combust.in > g_eqn_model > burned_region$, $g_eqn_model > on_flame$, or $g_eqn_model > unburned_region = SAGE$.

Enhancement: Removed a molecular weight limit to allow mechanisms with heavy [soot species](#).

Bug fix: Fixed a bug in the [PM soot](#) model that caused erroneous output.

Sources

Enhancement: You can now specify a sequential [source](#) with a start time that is before the simulation start time. CONVERGE will print a warning message and continue. Previously, CONVERGE would exit with an error under these conditions.

Enhancement: For [porous media](#) sources, *source.in* > *sources* > *porosity_data* > *solid_density* can now vary spatially. Previously, it was constant over the entire porous source.

Bug fix: Removed some restrictions that caused CONVERGE to crash when reading [battery properties.dat](#).

Bug fix: Fixed a bug that caused CONVERGE to crash when reading unused entries for [porous media](#) when *is_directional* = 0.

Bug fix: Fixed some bugs to make short-circuit [BATTERY](#)-type heat source modeling more robust.

Bug fix: Fixed a bug that caused CONVERGE to incorrectly exit with an error when the [temporal profile](#) of a source value was specified as *type* = PER_UNIT_TIME.

Bug fix: Fixed a bug that caused CONVERGE to crash when a [source](#) had prescribed motion and a distribution specified by a file.

Bug fix: Fixed a bug that caused CONVERGE to not correctly apply [thermal runaway model.in](#) > *ren_mode* > *lumped_temperature_flag*, causing incorrect results.

Bug fix: Fixed a bug that caused [BATTERY](#)-type sources to be incorrectly initialized to a non-zero value.

CHT

Enhancement: Sped up [super-cycling](#) for cases that are run with a large number of cores per node.

Bug fix: Previously, CONVERGE did not read the *time_control* > *stream_dependent_data* sub-block if it was specified in [supercycle.in](#). Now, CONVERGE reads *stream_dependent_data* only if *supercycle.in* is in the root Case Directory. If *supercycle.in* is in a directory specified in *stream_settings.in* > *stream_list* > *stream* > *stream_overwrite*, CONVERGE exits with an error.

Bug fix: Fixed a bug that caused CONVERGE to calculate energy sources incorrectly for cases using transient [super-cycling](#).

Bug fix: Fixed a bug that caused CONVERGE to crash on simulations with [super-cycling](#) if sources were activated, but none of the sources were energy sources.

Bug fix: Fixed a bug that caused CONVERGE to crash when writing *supercycle_point<ID>.out* for simulations with [super-cycling](#) and sources.

Bug fix: Fixed a crash that occurred when the [analytic wall treatment](#) was used in conjunction with CHT.

Bug fix: Previously, when a [contact resistance](#) event was CLOSED, CONVERGE would calculate erroneous heat transfer. After the CLOSE event, CONVERGE now changes the temperature boundary condition to DIRICHLET and prints a message to the log file. The temperature boundary condition is restored to its previous value when the event changes to OPEN.

Bug fix: Fixed a bug that caused CONVERGE to crash on a simulation without [super-cycling](#) but with [CHT1D](#).

Chapter 2: Release Notes

3.1.9

FSI

Enhancement: [INTERFACE](#) boundaries can now be part of co-simulation objects.

Enhancement: [Co-simulation](#) objects can now be multi-stream.

Bug fix: Fixed a bug that could cause CONVERGE to write erroneous data to map and restart files for simulations that used [FSI beams](#).

Bug fix: Fixed a bug that caused CONVERGE to crash when [FSI](#) cases specified a file for *applied_force* for a rigid object.

Radiation

Bug fix: Fixed a bug that caused CONVERGE to incorrectly calculate [post.in](#) > *parcels* > **parcels* > *physical* > *radiation_energy* as an instantaneous quantity rather than as a cumulative quantity.

Bug fix: Fixed a bug in [radiation modeling](#) that could cause incorrect results near boundaries for simulations with multiple regions.

Utilities

Enhancement: Added API to allow mechanism merge utility to operate on [transport.dat](#).

Enhancement: Added error checks and increased robustness of double [ignition delay](#) calculation.

CONVERGE Explore

Enhancement: Added a new CONVERGE Explore license type for non-academic, non-commercial use.

UDF

Enhancement: Added APIs to access molecular diffusivity, conductivity, and viscosity for individual species.

Enhancement: Updated API documentation to correctly mark some functions as deprecated and to remove declarations for functions that did not work.

Enhancement: Added several APIs and data types for accessing unique face data.

Enhancement: For simulations with sources, the <*source description*>_*source_value* field variable is now available for UDFs.

Enhancement: Added the *CONVERGE_get_boundary_id_from_index* and *CONVERGE_get_boundary_index_from_id* APIs.

Enhancement: Completed the implementation of the *CONVERGE_table_load_file* API.

Enhancement: UDFs can now access the global variable *hidden.ds_update_freq* via *CONVERGE_get_double*.

Enhancement: Added example UDFs for customizing parcel-wall interactions (*liquid_spray_parcel_rebound_slide_velocity.c* and *solid_parcel_rebound_slide_velocity.c*).

Enhancement: Added APIs for accessing solid parcel data.

Enhancement: Added an API to retrieve solid density from *source.in*.

Enhancement: The *realizable_keps_cmu* field variable is now available for UDFs.

Enhancement: UDFs can now access the surface species coverage fraction via the *surface_species* field variable.

Bug fix: Fixed a bug in the *CONVERGE_boundary_position* API that caused CONVERGE to always detect zero boundary motion.

Chapter 2: Release Notes

3.1.9

Bug fix: Fixed a bug in UDFs that could cause mass to not be conserved if a variable turbulent Schmidt number was used.

Bug fix: Fixed a bug that caused the `CONVERGE_clouds_get_kh_flag` API to return the wrong flag.

Bug fix: Fixed a bug that caused CONVERGE to crash on cases using Abaqus coupling with UDFs activated.

Bug fix: Fixed bugs in the `absorption_coef.c` example UDF that caused species mole fractions to be calculated incorrectly.

Bug fix: Fixed a bug that prevented users from registering passives as `ND_FIELD` arrays in UDFs.

Bug fix: Fixed a bug that caused motion UDFs to crash.

Bug fix: Fixed bugs in memory allocation for Lagrangian variables that could cause liquid spray and film UDFs to crash.

Bug fix: Fixed a bug that caused the `CONVERGE_source_get_description` API to incorrectly append extra characters to the source description.

Bug fix: Fixed a bug that caused CONVERGE to not allow the rotational speed of a boundary to be set via UDF. Now, you can specify rotational speed with UDFs based on the `CONVERGE_BEFORE_TIME_STEP` macro.

2.2 3.1.8

CONVERGE 3.1.8 is a minor release that includes enhancements and bug fixes.

Solver

Bug fix: Fixed a bug that caused CONVERGE to calculate incorrect results when running a simulation with a [MUSCL](#) flux scheme and `inputs.in > feature_control > mrf_flag = 2`.

Bug fix: Fixed a bug in how CONVERGE calculates inter-region flow rates when running with the [MUSCL_CVG](#) flux scheme.

Input

Enhancement: Improved the logic of a geometry motion case setup warning check. The previous logic printed an unnecessary warning for some cases.

Bug fix: Fixed a bug that prevented CONVERGE from correctly ending a co-simulation with [GT-SUITE](#) when `inputs.in > simulation_control > end_time = GT`.

Boundaries and Boundary Conditions

Enhancement: You can now specify [rotating boundary motion](#) via a file for steady-state simulations. CONVERGE will rotate the boundary to the specified angle before beginning the simulation. This allows you to alter some aspects of your case setup (e.g., rotating a throttle body plate to a different position) without altering `surface.dat`. This is equivalent to translating motion for steady-state simulations, which was already available.

Regions and Streams

Bug fix: Fixed a bug that could cause CONVERGE to crash when writing `passive.out` for multi-stream simulations.

Turbulence

Chapter 2: Release Notes

3.1.8

Bug fix: Fixed a bug in the [analytic wall treatment](#) that caused heat transfer to be calculated incorrectly in simulations with CHT.

Discrete Phase

Bug fix: Fixed a memory leak that occurred when writing [*vof_spray.out*](#) for VOF-spray one-way coupling.

Bug fix: Fixed a bug that caused CONVERGE to write incorrect header data to [*vof_spray.out*](#) when running a simulation with multiple region pairs.

Sources

Bug fix: Fixed a bug that caused CONVERGE to calculate source values incorrectly on a simulation with [*super-cycling*](#) and *inputs.in > simulation_control > crank_flag* greater than 0.

CHT

Bug fix: Fixed a bug that caused CONVERGE to calculate heat transfer incorrectly on simulations with [*super-cycling*](#) and [*MRF*](#).

UDF

Bug fix: Fixed several bugs in the *film_evap.c* example UDF that could cause CONVERGE to crash.

Bug fix: Fixed a bug that caused CONVERGE to crash when a UDF tried to write a post variable that was only defined within the UDF.

2.3 3.1.7

CONVERGE 3.1.7 is a minor release that includes enhancements and bug fixes.

Solver

Enhancement: CONVERGE now prints a warning message for CONVERGE + [GT-SUITE](#) co-simulations that use the *CONVERGE_LEAD* coupling scheme. This coupling scheme has been deprecated in GT-SUITE. We recommend using the *GT_LEAD* coupling scheme instead.

Enhancement: Refactored some PISO subroutines to speed up the solution of the pressure equation when this is solved using the [*SOR*](#) linear solver.

Enhancement: Altered numerical treatment at [*steady-transient*](#) coupling interfaces to reduce non-physical results.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a multi-stream simulation with fixed flow and *solver.in > Transport > wall_dist > solver_type = CONVERGE_BICGSTAB* or *HYBRID*.

Bug fix: Fixed a bug that could cause CONVERGE to crash when specifying *WALL_DIST* for passive-based AMR. *WALL_DIST* is defined as an [*internal passive*](#) and was previously initialized only if this variable was specified in *post.in* or required by a specified turbulence model. Passive-based AMR can now serve as an initialization trigger, which was the intended behavior.

Bug fix: Fixed a bug in the [*CONVERGE BiCGSTAB*](#) linear solver that could cause temperature extrapolations.

Bug fix: Fixed a bug that caused CONVERGE to end a CONVERGE + [GT-SUITE](#) co-simulation prematurely when *inputs.in* > *simulation_control* > *end_time* = *GT*.

Bug fix: Fixed a bug that would cause CONVERGE to crash when running a single-cell simulation with the [CONVERGE BiCGSTAB](#) or [HYPRE BiCGSTAB](#) linear solver.

Bug fix: Fixed several bugs that could cause CONVERGE to crash or write incorrect output when performing [statistics calculations](#).

Bug fix: Fixed a bug that caused CONVERGE to solve the transport equation unnecessarily for [passives](#) that are constant throughout the domain.

Bug fix: Fixed a bug in the implementation of the [Rhie-Chow](#) algorithm that could cause CONVERGE to experience temperature extrapolations on a simulation with a body force.

Bug fix: Fixed a bug that caused CONVERGE to calculate momentum transport inaccurately when solving this transport equation with the [MUSCL](#) flux scheme. This bug only affected cases with moving boundaries.

Bug fix: Fixed a bug in [load balancing](#) that could cause CONVERGE to crash.

Input

Enhancement: Some CONVERGE features should not be used in a simulation with [non-Newtonian liquids](#). Added several warning checks to detect such combinations.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a simulation in Windows when a Case Directory path included non-ASCII characters.

Bug fix: Fixed a bug that could cause CONVERGE to apply incorrect settings when an input file contained an exclamation point, causing behavior other than what the user specified.

Output/Post-Processing

Enhancement: Added *Pres_Rot_Power* and *Visc_Rot_Power* to [bound<boundary ID>-wall.out](#).

Enhancement: For VOF simulations, CONVERGE now writes species names to the headers of [volumes.out](#).

Enhancement: CONVERGE now computes boundary power and torque for [WALL](#) boundaries with *motion* = *ARBITRARY*. Previously, only *motion* = *ROTATING* was supported.

Enhancement: For VOF-spray one-way coupling simulations, CONVERGE now writes [vof_spray.out](#) much more quickly.

Enhancement: CONVERGE now writes data for PERIODIC boundaries to [area_avg_flow.out](#) and [mass_avg_flow.out](#).

Bug fix: Fixed a bug that produced different simulation results when different values were specified for *inputs.in* > *output_control* > *twrite_files*.

Bug fix: Fixed a bug that caused CONVERGE to write incorrect heat flux values to [supercycle_stream<number>.balance.out](#) when a solid stream was stationary.

Bug fix: Fixed a bug that caused CONVERGE to write incorrect header units to [*.out](#) files in rare instances.

Bug fix: Fixed a bug that could cause CONVERGE to write NaN data to [transfer.out](#).

Bug fix: Fixed a bug that could cause CONVERGE to output NaN and infinite values to [post*.h5](#).

Bug fix: Fixed a bug that could cause CONVERGE to write incorrect parcel species mass fractions to [post*.h5](#).

Chapter 2: Release Notes

3.1.7

Bug fix: Fixed a bug that could cause CONVERGE to write the incorrect *dt_limiter* to [*time.out*](#).

Bug fix: Fixed several bugs in how CONVERGE sets up statistics calculations in [*passive.out*](#).

Bug fix: Fixed the column header units in [*passive.out*](#).

Bug fix: Fixed a bug that could cause corrupted [*post*.h5*](#) output in case setups with sealing.

Bug fix: Fixed a bug where certain keywords specified in [*surface_mech.dat*](#) were not correctly written to [*surface_mech_check.out*](#).

Boundaries and Boundary Conditions

Enhancement: Altered the logic for [**PERIODIC**](#) boundary match error checking. The new logic detects some invalid setups that were not previously detected.

Bug fix: Fixed an error in calculating energy balance in the [**nucleate boiling model**](#) by setting an upper limit, *qmax*.

Bug fix: Fixed a bug in the [**Poinsot-Lele INFLOW**](#) boundary condition. The previous implementation did not damp out pressure waves correctly.

Bug fix: If you specify a reference center that is not located on the corresponding [**INFLOW**](#) or [**OUTFLOW**](#) boundary, CONVERGE now prints a warning and moves the reference center to a point on the boundary. Previously, CONVERGE used the specified location, which could cause the boundary pressure distribution to be calculated incorrectly.

Regions and Streams

Bug fix: Fixed a bug that could cause [**multi-stream**](#) cases to crash.

Bug fix: Corrected errors in [**cross-stream species matching**](#).

Grid Control

Enhancement: Improved some surface handling subroutines to improve the robustness of the [**sealing**](#) feature.

Enhancement: Improved the algorithm for matching [**inlaid**](#) faces at PERIODIC boundaries. This fixes an issue where the periodic matching could fail for some cases.

Bug fix: Fixed a bug that could cause CONVERGE to crash when event triangles were created near an [**inlaid**](#) mesh and a [**seal**](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation in parallel with moving [**inlaid**](#) meshes.

Bug fix: Fixed a bug that caused CONVERGE to crash on a simulation with [**inlaid**](#) meshing and *inputs.in > prox_check_flag* greater than 0.

Bug fix: Fixed a bug that caused [**inlaid**](#) mesh deformation behavior to be dependent on the order in which boundaries were defined.

Bug fix: Fixed a bug that caused CONVERGE to exit with an error in some cases with a moving [**inlaid**](#) mesh.

Bug fix: Fixed a bug that caused CONVERGE to crash in some cases with large deformations in an [**inlaid**](#) mesh.

Bug fix: Improved the [**load balancing**](#) procedure to avoid environment-dependent crashes caused by ParMETIS.

Bug fix: Fixed a bug that caused CONVERGE to calculate *dt_move* incorrectly in some simulations with [**surface lists**](#).

Chapter 2: Release Notes

3.1.7

Bug fix: Fixed a bug in [grid initialization](#) that could cause CONVERGE to crash when starting up a simulation.

Bug fix: Fixed a bug that caused CONVERGE to crash if [load balancing](#) was triggered while one or more ranks had zero cells.

Bug fix: Fixed a bug that caused CONVERGE to exit with an error when controlling value-based [AMR](#) with the *LSR_PARAM* internal passive.

Bug fix: Fixed a bug that could cause CONVERGE to crash during mesh generation in some cases with an [inlaid](#) mesh.

Materials

Enhancement: Added some warning checks in [material file](#) reading subroutines.

Enhancement: For simulations that use [tabular fluid properties](#), CONVERGE now uses a real gas relation to compute the specific heat at constant volume for the mixture. Previously, CONVERGE used a mass-weighted average of the specific heats of the individual species.

Enhancement: In [surface mech.dat](#), you can now add the MWON keyword to the REACTIONS line to apply the Motz-Wise correction to every STICK reaction in the file.

Bug fix: Fixed a memory leak in simulations with *inputs.in > property_control > tabular fluid prop flag* greater than zero.

Initialization

Bug fix: Fixed a bug that caused CONVERGE to read [map.in](#) even though mapping was disabled in *inputs.in*.

Bug fix: Fixed a bug that caused CONVERGE to initialize velocity incorrectly when running a simulation with a specified *rotation_axis* in [boundary.in](#).

Bug fix: Fixed a bug that caused CONVERGE to initialize the domain incorrectly when initializing a super-cycling case from a [map](#) file.

Turbulence

Enhancement: Updated the RANS k-epsilon [analytic wall treatment](#) to improve speed and accuracy.

Bug fix: Fixed several bugs in the turbulence [statistics](#) calculation. These bugs affected error messages and restart parameters.

Discrete Phase

Enhancement: Added *k_kuhnke*, the K number from the [Kuhnke splash model](#), as an available post-processing variable in *post.in*.

Enhancement: You can now specify [custom parcel-wall interaction behavior](#) based on boundary ID.

Bug fix: Fixed a problem with error checking of the [spray injection](#) location that caused inaccurate results.

Bug fix: Fixed a bug that caused some boundary injection cases to crash when *parcels.in > liquid parcels > combine parcels > active = 1*.

Bug fix: Fixed a bug that caused the [temperature of spray parcels](#) to not change after injection if the injection temperature was lower than 270 K. Now, CONVERGE clips parcel temperatures only if the injection temperature is lower than a default cutoff of 200 K. In this scenario, CONVERGE prints a warning with instructions on how to change the cutoff temperature.

Chapter 2: Release Notes

3.1.7

Bug fix: Fixed a bug that caused CONVERGE to crash on simulations using [boundary injection](#) without an evaporation model.

Bug fix: Fixed a bug that caused CONVERGE to exit with an error if [parcel mapping](#) was activated and cell mapping was deactivated. This error check is unnecessary and has been removed.

Bug fix: Fixed a bug that caused CONVERGE to exit with an error when attempting to restart a simulation with [boundary injection](#).

Bug fix: Fixed several bugs that caused CONVERGE to calculate properties incorrectly for [composite](#) parcel species.

Bug fix: Fixed a bug that caused CONVERGE to calculate [film](#) thickness incorrectly.

Bug fix: Fixed several bugs related to [film](#) interaction with [sealing](#).

Bug fix: Fixed a bug that could cause CONVERGE to calculate heat fluxes incorrectly when [inputs.in](#) > *feature_control* > *nucleate_boiling_flag* = 1.

VOF

Enhancement: Improved the accuracy and stability of VOF simulations using the flux-corrected transport ([FCT](#)) scheme with a central difference scheme.

Enhancement: You can now use the high resolution interface capturing ([HRIC](#)) scheme with the species density solver.

Enhancement: You can now use the species density solver for [non-Newtonian liquids](#).

Bug fix: Fixed a bug that caused CONVERGE to set liquid properties incorrectly for cases using the species density solver with [tabular fluid properties](#).

Bug fix: Previously, CONVERGE used the ideal gas equation to initialize the density of the gas phase for all incompressible VOF cases. Now, CONVERGE uses the equation of state specified in [inputs.in](#).

Bug fix: Fixed a bug that prevented CONVERGE from printing an error message when [post.in](#) contained invalid VOF variables.

Fixed Flow

Enhancement: Altered some *dt_chem* calculation logic to improve the stability of [fixed flow](#) simulations.

Bug fix: Fixed a bug that caused some settings in [fixed_flow.in](#) to not be correctly applied after CONVERGE [reread inputs](#) from [solver.in](#). This could cause mass transport to be calculated incorrectly.

Bug fix: Fixed a bug that caused a flow variable to not be properly initialized at the beginning of a [fixed flow](#) interval. This could cause mass transport to be calculated incorrectly.

Bug fix: In cases that used both erosion modeling and [fixed flow](#), the simulation time calculated by the erosion model would become inconsistent with the fixed flow solver, leading to inaccurate results. This issue has been fixed.

Bug fix: For steady-state streams, fixed flow would start at the beginning of the simulation even if a different start time was specified in [fixed_flow.in](#). This issue has been fixed.

Combustion

Enhancement: Modified the adaptive zoning MPI function call to improve run times for HPC-X and Intel builds with InfiniBand.

Enhancement: The RIF model can now be used with Eulerian-only cases. Previously, it required Lagrangian parcel modeling to be activated.

Bug fix: Fixed a bug in how CONVERGE calculated convergence criteria for some internal data structures. This caused perturbation-level result differences when running a simulation with [SAGE three-point PDF](#) method.

Bug fix: Fixed a bug where CONVERGE did not apply the Motz-Wise correction to individual reactions for which it was specified. You can now add the STICK MOTZ keyword to individual reactions in [surface_mech.dat](#) to apply the Motz-Wise correction to those reactions.

Bug fix: Fixed a bug that prevented CONVERGE from writing adaptive zone output for multiple streams. To write adaptive zone output from all streams, set [stream_settings.in > post_file_option = STREAM_BASED](#).

Bug fix: Corrected the wording of the error message that CONVERGE prints when a case has an invalid species name in [combust.in > adaptive_zone_model > bin_options](#).

Bug fix: Fixed a bug that caused some reaction coefficients to be written in a floating-point format without scientific notation.

Bug fix: Fixed a bug that could cause inaccurate results when running a simulation with [SAGE](#) adaptive zoning, and surface chemistry.

Bug fix: Fixed a bug that caused the [CEQ](#) solver to crash when the reaction mechanism file contained an element that is not a part of any species.

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation with the [G-Equation](#) combustion model and an inlaid mesh.

Bug fix: Fixed a bug in [FGM](#) that caused the *zvar* post variable to not be clipped to a minimum value of zero.

Emissions

Bug fix: Fixed a bug that caused CONVERGE to write incorrect data to *phenom_soot_model.out* when the [Waseda soot model](#) is active.

Sources

Enhancement: For [source.in > source > moving_control > moving_control > MOVE_WITH_BOUNDARY](#), you can now specify an FSI boundary.

Bug fix: Fixed a bug that caused [BATTERY](#)-type sources to be incorrectly initialized to a non-zero value.

Bug fix: Fixed a bug that caused CONVERGE to crash when a case setup included an [EPS](#) source.

Bug fix: Fixed a bug in [porous media](#) sources with anisotropic conductivity. Previously, if the LTNE model was active, CONVERGE applied the anisotropic conductivity only to the fluid energy equation. Now, CONVERGE correctly applies the anisotropic conductivity to both the fluid and solid energy equations.

CHT

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a cyclic simulation with [super-cycling](#) and a variable RPM.

Bug fix: Fixed a bug that could cause Windows Intel MPI builds of CONVERGE to crash at the end of a case with [super-cycling](#).

Chapter 2: Release Notes

3.1.7

Bug fix: Fixed a bug that could cause CONVERGE to calculate inter-region heat transfer incorrectly when a simulation included a [contact resistance](#) event.

FSI

Enhancement: Altered implicit [FSI](#) calculations to make CONVERGE more robust.

Enhancement: CONVERGE 3.1 now supports [co-simulation with Abaqus](#) 2022 and retains existing support for Abaqus 2019-2021. The 2017 and 2018 versions of Abaqus are no longer supported.

Enhancement: Added error checks for implicit [FSI](#) to ensure that the specified maximum number of iterations is equal to or greater than the minimum number of iterations, and that the minimum number of iterations is at least zero.

Bug fix: Fixed a bug that caused CONVERGE to calculate viscous stress incorrectly for [FSI](#) objects.

Bug fix: Fixed a bug that caused CONVERGE to ignore the specified minimum number of iterations for implicit [FSI](#). This parameter is now respected, which might cause minor perturbations in implicit FSI results.

Bug fix: Fixed several bugs related to rigid [FSI](#) objects with INTERFACE boundaries.

CONVERGE did not correctly identify whether these objects were open or closed and, for some case setups, miscalculated their volumes and moments of inertia.

Bug fix: Fixed a bug where CONVERGE calculated the position and velocity of an [FSI](#) object incorrectly in simulations using FSI mapping.

Radiation

Bug fix: Fixed a bug that caused CONVERGE to solve the radiation transport equation in the first PISO iteration when [radiation-energy coupling](#) was activated, slowing down the simulation.

Aeroacoustics

Bug fix: Fixed a bug in the [Ffowcs Williams-Hawkins](#) aeroacoustic model that could cause incorrect results.

Utilities

Enhancement: The [cleantransport](#) utility now checks *transport.dat* for missing species and checks the first three values for each species.

Enhancement: Altered the random seed generation algorithm to make mechanism reduction more repeatable in 0D [well-stirred reactor](#) cases.

Enhancement: Added logic to detect a [1D](#) case setup with the sparse solver with less than 10% N2 by mass in the unburned mixture. This combination is not recommended.

CONVERGE will print a warning to the log file if this is detected.

Bug fix: Fixed a bug that could cause Intel MPI builds of CONVERGE to crash when running a [0D](#) simulation.

UDF

Enhancement: Updated the *spray_evap.c* example UDF to incorporate a flash boiling model. Note that this UDF does not calculate the drop size reduction that occurs due to flash boiling during injection. You can use the new *size_scale_flash_boiling.c* example UDF to calculate the drop size reduction during injection.

Chapter 2: Release Notes

3.1.7

Enhancement: You can now use source UDFs to add heat sources to solid streams in simulations with super-cycling.

Enhancement: Added several APIs for multi-phase modeling.

Enhancement: Updated the default settings for user-defined non-transport passives. By default, these variables are now stored on boundaries as mass-like quantities instead of as surface-density-like quantities.

Enhancement: Added the `CONVERGE_source_get_description` API.

Enhancement: The `WALL_DIST` field variable is now available for UDFs if any turbulence model is activated. Previously, it was available only when specific models were activated.

Enhancement: Added the `CONVERGE_get_parent_index` API.

Enhancement: Added APIs to access data for porous sources.

Enhancement: Added APIs for combustion.

Enhancement: Added the Kuhnke and Bai-Gosman film splash models to the `film_splash.c` example UDF.

Enhancement: Added APIs for the Particulate Size Mimic (PSM) detailed soot model.

Enhancement: Added the `CONVERGE_parcel_evap_species_lookup` API, which replaces the `CONVERGE_parcel_species_lookup` API. `CONVERGE_parcel_species_lookup` is now deprecated.

Enhancement: Updated the `vof_cav_condensation.c` example UDF to include cavitation modeling for multiple species.

Bug fix: Fixed a bug that caused CONVERGE to not correctly apply the force or moment calculated by an FSI UDF.

Bug fix: For FSI cases with general constraints, CONVERGE now passes constrained values of variables to FSI UDFs instead of unconstrained values.

Bug fix: Fixed a bug in the `CONVERGE_mech_get_species_entropy` API that incorrectly caused the API to return enthalpy.

Bug fix: In `active_variables.log`, field variables that previously had a `".` in the name now have `"_DOT_"` instead. For example, `BAR_X.U` is now `BAR_X_DOT_U`. This prevents compilation errors caused by the `".` in the variable name.

Bug fix: Fixed a bug that caused minor perturbations in results when the `film_gradp` UDF was activated.

Bug fix: Fixed a bug in the `film_splash.c` example UDF that caused film thickness to be calculated incorrectly.

Bug fix: Added a missing argument to the `CONVERGE_spray_coalesce` function declaration. Runtime errors occurred if UDFs called the function without this argument.

Bug fix: Fixed a bug that caused CONVERGE to write incorrect variable names for species volume fractions to `active_variables.log`.

Bug fix: Fixed a bug where temperature boundary conditions calculated by profile UDFs were not applied to WALL boundaries when specified.

Bug fix: Fixed a memory leak that occurred when running UDFs based on the `CONVERGE_TRANSPORT_SYSTEM` macro.

2.4 3.1.6

CONVERGE 3.1.6 is a minor release that includes enhancements and bug fixes.

Solver

Chapter 2: Release Notes

3.1.6

Enhancement: When running with [*inputs.in*](#) > *simulation_control* > *check_grid_motion_flag* greater than 0, CONVERGE previously performed some fluid-dynamical calculations that do not affect grid generation. CONVERGE now skips these calculations.

Enhancement: Improved solver stability for large values of [*mult_dt_move*](#).

Enhancement: Added logic to the [CONVERGE_BICGSTAB](#) linear solver that allows CONVERGE to recover in some circumstances that previously caused a crash.

Bug fix: Fixed bugs that caused CONVERGE to miscalculate the turbulent kinetic energy production term in the [total energy](#) equation.

Bug fix: Fixed a bug in the implementation of the [Rhie-Chow](#) algorithm that could cause CONVERGE to experience temperature extrapolations on a simulation with a body force.

Bug fix: Fixed a bug that caused CONVERGE to crash when using the [MUSCL](#) scheme for VOF simulations with two incompressible fluids.

Bug fix: Fixed a bug that caused CONVERGE to use the global [MUSCL](#) blending factor (*solver.in* > *numerical_schemes* > *muscl_blend_factor* = *GLOBAL*) instead of the momentum blending factor (*muscl_blend_factor* > *MOM*) in simulations that used the MUSCL scheme only for momentum.

Bug fix: Fixed a bug that caused CONVERGE to calculate the minmod [slope limiter](#) incorrectly when the gradient was zero.

Bug fix: Fixed a bug that could cause CONVERGE to crash when running with a large [mult_dt_move](#).

Bug fix: Fixed several inconsistencies in the [total energy](#) solver. This improves accuracy and convergence performance.

Bug fix: Fixed several bugs that could cause CONVERGE to crash when running a co-simulation with [GT-SUITE](#).

Bug fix: Fixed a bug that caused wall distance to not be defined when there were no [WALL](#) boundaries. This could cause CONVERGE to crash when running a simulation with a turbulence model that requires wall distance.

Bug fix: Fixed a bug in the [anisotropic conductivity](#) solver that caused inaccurate results when using this feature.

Bug fix: Fixed several bugs in how CONVERGE calculates residuals. This could result in suboptimal convergence behavior.

Bug fix: Fixed several memory leaks.

Bug fix: Fixed a bug that caused CONVERGE to [report](#) an incorrect location for *max_omega_cut_ustar*.

Bug fix: Fixed a bug in how CONVERGE calculates [PISO](#) error in simulations that include only a solid stream.

Bug fix: Fixed a bug in [MRF](#) that could cause minor perturbations in results.

Bug fix: Fixed a bug in the [total energy](#) solver that could cause CONVERGE to crash on a simulation with moving boundaries.

Bug fix: Fixed a memory leak in wall distance calculation for the [CONVERGE_BICGSTAB](#) and HYBRID solvers.

Input

Enhancement: You can now use *.h5 [map](#) files with limited information where phase cannot be determined (for example, from experimental data). CONVERGE maps all streams in the map file to all streams in the current case setup.

Chapter 2: Release Notes

3.1.6

Enhancement: Added error checking for reactions with negative or zero A-factors in [surface_mech.dat](#).

Bug fix: Fixed a bug that caused CONVERGE to print the wrong line number when checking for lines with negative or zero A-factors in [mech.dat](#).

Bug fix: Fixed a bug in reading inputs that could cause CONVERGE to crash on Windows.

Bug fix: Fixed a bug that prevented CONVERGE from reading a [discharge coefficient](#) from a file.

Bug fix: Fixed a bug that caused CONVERGE to crash when running [check_inputs](#) for a case in which adaptive zoning was active and VOF was inactive.

Output/Post-Processing

Enhancement: In simulations using the [steady-state monitor](#), *duration_size* is now printed to the log file along with *sample_size*. This makes it easier to identify the criterion CONVERGE is using to determine whether the variable has reached a steady state.

Enhancement: If simulations using [fluid_properties.dat](#) reach a temperature outside the defined range, CONVERGE extrapolates a temperature value from the file and logs a warning. Now, these warnings are logged only once per time step to reduce clutter in the [log file](#).

Enhancement: For LES simulations, CONVERGE now prints *logic_ijk*, *volume*, and *idreg* for any simulation that uses LES in at least one stream. These variables are required for the averaging feature in [post_convert](#).

Enhancement: If CONVERGE calculates a negative uniformity index, it will print a warning to the [log file](#). This often indicates a problem with the case setup.

Enhancement: Added *species_diffusion_velocity* as an available post-processing variable in [post.in](#).

Enhancement: [borghi.out](#) is now stream-based.

Bug fix: Fixed a bug that caused CONVERGE to print a spurious temperature extrapolation warning to the [log file](#).

Bug fix: Fixed a bug that caused CONVERGE to write incorrect data to [spray_rate_inj<ID>.out](#).

Bug fix: For periodic cases, fixed a bug in how CONVERGE calculated mass flow rates in [mass_balance.out](#).

Bug fix: Fixed an issue where, after [ParaView Catalyst](#) scripts were triggered, they were incorrectly triggered again at the next iteration.

Boundaries and Boundary Conditions

Enhancement: Implemented a semi-implicit formulation for total pressure at [INFLOW](#) boundaries. This improves simulation stability when taking large time-steps.

Enhancement: Added a reverse flow treatment for NSCBC [OUTFLOW](#) boundaries.

Enhancement: Added logic to detect disallowed [surface normal](#) flipping caused by boundary motion. If detected, CONVERGE will print a warning to the [log file](#).

Bug fix: Fixed a bug in the [flux](#) boundary condition for the electric potential solver. This bug caused inaccurate results.

Bug fix: Fixed a bug that caused CONVERGE to incorrectly apply [OUTFLOW](#) boundary reverse flow logic when grid scaling changed. This bug could cause inaccurate results.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [flow-through INTERFACE](#) boundaries.

Chapter 2: Release Notes

3.1.6

Bug fix: Fixed a bug in the [GT-SUITE](#)-type boundary that could cause CONVERGE to crash.

Bug fix: Fixed a bug in the Poinsot-Lele [INFLOW](#) boundary condition. The previous implementation did not damp out pressure waves correctly.

Bug fix: Fixed several bugs in the NSCBC [INFLOW](#) and [OUTFLOW](#) boundary conditions that could cause anomalous oscillatory behavior at boundaries.

Bug fix: Fixed a bug in how CONVERGE sets Neumann [OUTFLOW](#) boundary conditions for turbulence variables. This could cause perturbation-level results differences and unnecessary recoveries.

Regions and Streams

Enhancement: Altered some [stream override](#) logic to reduce numerical artifacts at interfaces between streams when those streams use different solver settings or physical models.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [dependent regions](#) and parcel modeling.

Bug fix: Previously, some multi-[stream](#) simulations would crash when spray modeling and related flags in *fixed_flow.in* were activated for solid streams. These flags are now turned off for solid streams.

Bug fix: Added logic to detect when [stream_settings.in](#) has a stream with no assigned regions. This will cause CONVERGE to exit with an error. Previously, this caused CONVERGE to crash.

Bug fix: Fixed a bug that caused CONVERGE to crash on 0D cases that specify a lower heating value that is [stream-based](#).

Bug fix: Fixed a bug that caused CONVERGE to crash when calculating wall distance on a multi-[stream](#) simulation.

Grid Control

Enhancement: You can now use [sealing](#) in a simulation with an [inlaid mesh](#). The seals cannot touch or interact with the inlaid mesh.

Enhancement: You can now specify an [inlaid mesh](#) made up of more than one interior flow-through INTERFACE boundary.

Enhancement: Modified the logic for identifying cells as candidates for [proximity AMR](#) to improve performance.

Enhancement: Sped up the calculation of deforming motion for an [inlaid mesh](#).

Enhancement: It is no longer necessary to set *inlaid_mesh.in* > *inlaid_mesh_settings* > *inlaid_mesh* > *deformation* > *active* = 1 to enable deforming motion for an inlaid mesh.

CONVERGE automatically enables deforming motion when it is specified for at least one boundary in *boundary.in*, so the setting in *inlaid_mesh.in* is redundant.

Bug fix: Fixed a bug in which cells were not correctly identified as candidates for [proximity AMR](#) if they were not attached to a surface.

Bug fix: Fixed a bug that prevented a simulation from running if a reloft WALL boundary was connected to an [inlaid mesh](#).

Bug fix: Fixed a bug in how CONVERGE calculates cell volumes in cases with moving boundaries. In rare cases, this could cause CONVERGE to crash.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [inlaid mesh](#) events.

Bug fix: Fixed a bug in [AMR](#) that could cause CONVERGE to crash on a simulation with multiple streams.

Chapter 2: Release Notes

3.1.6

Bug fix: Fixed several memory leaks in grid and solver subroutines.

Bug fix: Fixed a crash that occurred during [grid generation](#) for cases with standalone INTERFACE boundaries that are not connected to other boundaries.

Bug fix: Fixed a crash that occurred rarely in cases with surface triangles that were nearly coplanar with cell faces.

Bug fix: Updated the [sealing](#) algorithm to avoid the creation of twisted polygons, which can cause sealing to fail.

Bug fix: Fixed a bug that caused CONVERGE to apply [AMR](#) when *inputs.in > simulation_control > check_grid_motion_flag = 2*. CONVERGE should not apply AMR in this scenario.

Bug fix: Fixed a bug that caused CONVERGE to crash when [seals](#) interacted with reloft WALL boundaries in the same stream. The seals and reloft WALL boundaries did not have to be near each other spatially to cause a crash.

Materials

Enhancement: When *inputs.in > tabular_fluid_properties = 1*, CONVERGE now calculates the Mach number and *cfl_mach* based on the tabulated sound speed.

Enhancement: Added new options to calculate species diffusion. If *inputs.in > property_control > species_diffusion_model = 3*, diffusion is calculated based on *schmidt_species.dat* and conductivity through a user-defined calculation in *hidden.in*. If *species_diffusion_model = 4*, *schmidt_species.dat* is used to calculate diffusion, and conductivity is calculated based on *transport.dat*.

Bug fix: Fixed a bug that caused [tabular fluid property](#) cases to crash when file names are specified for critical temperature, pressure, and acentric factor.

Bug fix: Duplicate species are now allowed in [mech.dat](#). Previously, duplicate species caused CONVERGE to exit with an error.

Bug fix: When reading [therm.dat](#), CONVERGE now exits with an error if the temperature data in the second line is formatted incorrectly.

Initialization

Bug fix: Fixed a bug that caused CONVERGE to not [map](#) Reynolds stresses correctly at initialization when running a simulation with the [RSM](#) turbulence models.

Turbulence

Enhancement: The [analytic wall treatment](#) is now available for simulations using RANS k-epsilon models. This wall treatment uses analytic functions obtained by integrating simplified momentum and energy equations near the wall.

Enhancement: For the [v2-f](#) and [zeta-f](#) turbulence models, you can now solve the *f* equation with the CONVERGE_BICGSTAB linear solver.

Enhancement: Altered the solution procedure for turbulent viscosity for RANS k-omega [models](#). This improves solution stability in near-wall cells.

Discrete Phase

Enhancement: Added logic to detect non-physical [interactions](#) between parcels and moving boundaries. These interactions will cause CONVERGE to print a warning to the log file.

Chapter 2: Release Notes

3.1.6

Enhancement: Added a check to detect case setups with [parcel film mapping](#) activated and wall films deactivated. CONVERGE will deactivate parcel film mapping and print a warning to the log file.

Enhancement: Added the FILM_LEGACY [wall film](#) model.

Bug fix: Fixed memory leaks in parcel-related and radiation-related routines.

Bug fix: Fixed several [sealing](#) bugs that could cause CONVERGE to crash on a simulation with [wall films](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation with parcel modeling and the [KH](#) or [KH-ACT](#) breakup model.

Bug fix: Fixed a bug in the [Bai-Gosman](#) film splash model that could cause parcel mass generation in the splash-slide regime.

Bug fix: Fixed several bugs that could cause CONVERGE to slowly destroy parcel mass.

Bug fix: Fixed a bug that caused CONVERGE to exit with an error when attempting to restart a simulation with [boundary injection](#).

Bug fix: Fixed a bug that caused CONVERGE to crash on a simulation with [boundary injection](#) and solid parcels.

VOF

Enhancement: For VOF simulations that use the [species density solver](#) with no front capturing scheme, CONVERGE now respects the global upwinding factor specified in *solver.in* > *Numerical_schemes* > *fv_upwind_factor* > *global*. Previously, CONVERGE overwrote the specified value and always used a first-order upwinded scheme.

Bug fix: Fixed several bugs to improve the accuracy and stability of the [species density solver](#).

Bug fix: Fixed a bug that caused CONVERGE to calculate the VOF [mixture model](#) drift velocity incorrectly for simulations with multiple gas species. As a result, CONVERGE did not correctly predict the separation of gases with different densities.

Bug fix: Fixed a bug that caused errors in mass conservation for cases that use [flux-corrected transport](#) or the VOF [mixture model](#) with a moving [inlaid mesh](#).

Bug fix: Fixed a bug that caused CONVERGE to hang on VOF simulations that use [fluid_properties.dat](#) for the gas phase.

Bug fix: Fixed a bug in the volume fraction calculation for [species-based](#) VOF simulations.

Fixed Flow

Bug fix: Fixed a bug that prevented single-stream cases with fixed flow from solving species transport during the frozen interval when *fixed_flow.in* > *solve_fixed_species_flag* = ON.

Combustion

Enhancement: You can now run the 1D flame solver using a species diffusion model in which a different Schmidt number is defined for each species. To use this option, set *one_d_solver.in* > *one_d_general_control* > *solver_control* > *species_diffusion_model* = 2 and include *gas.dat* and *schmidt_species.dat* in your case setup.

Enhancement: For simulations with [adaptive zoning](#), you can monitor the minimum, mean, and maximum species mass fractions for each species in *az_info<dump number>_<dump time>.out*. You can use this information to judge the appropriateness of the adaptive zoning bin sizes.

Chapter 2: Release Notes

3.1.6

Enhancement: You can now specify a file name for the *stretch_alpha* parameter in a simulation with [ECFM](#) or [ECFM3Z](#). [This file](#) allows for limited temporal and region-based control of this parameter.

Enhancement: For simulations with [inputs.in](#) > *property_control* > *species_diffusion_model* = 1, CONVERGE now uses a more accurate method to calculate the binary species diffusion coefficients.

Enhancement: Clarified the warning message that is printed when the number of [adaptive zoning](#) bins exceeds the maximum allowed value.

Enhancement: You can now couple the [G-Equation](#) model with a TKI table in the unburned region.

Enhancement: When estimating real gas departures for thermodynamic variables, CONVERGE now accounts for the [real_gas](#) departure of specific heat at constant volume based on the expression in Poling et al. (2001).

Enhancement: You can now specify different diffusion models for passives in [SAGE three-point PDF](#) cases.

Bug fix: Fixed a bug that would cause CONVERGE to crash when running a simulation with the [ECFM](#) or [ECFM3Z](#) combustion model when parcels were present that contained non-fuel species such as water, or with composite species.

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a 1D simulation with [one_d_solver.in](#) > *one_d_newton_control* > *discretization_control* > *sparse_solver_flag* = 1.

Bug fix: Fixed a bug that could cause CONVERGE to incorrectly apply [surface_chemistry.in](#) > *reaction_multiplier*, producing inaccurate results.

Bug fix: Fixed a bug that caused CONVERGE crash during [FGM](#) calculations when running a simulation on Windows.

Bug fix: Fixed a bug that could cause a memory leak in CONVERGE when running the [SAGE](#) solver with the G-Equation model.

Bug fix: Fixed a bug that caused CONVERGE to calculate fuel mass fractions incorrectly for spray cases that used a dual-fuel [laminar flamespeed](#) table for combustion modeling.

Bug fix: Fixed a bug in skip species that caused CONVERGE to not transport some species correctly after the [skip species](#) end time.

Bug fix: Fixed a bug in the LHV calculation in the [CEQ](#) solver that caused temperature extrapolations with the [G-Equation](#) model.

Bug fix: Fixed several memory leaks in combustion routines.

Sources

Enhancement: You can now include particulate filters in [porous media](#) sources to model particulate filtration in aftertreatment systems.

Enhancement: For sources with negative value, you can now specify a minimum solution variable value within the source region. Specify the minimum solution variable value with [source.in](#) > *source* > *max_value*.

Enhancement: Improved solver stability for case setups with [porous media](#).

Bug fix: Fixed a bug in the [thermal runaway model](#) that caused CONVERGE to print incorrect results to *passive.out* and *post*.h5*.

Bug fix: Fixed a bug in [source modeling](#) that caused CONVERGE to incorrectly calculate accumulated heat release rate when the solver recovers.

Chapter 2: Release Notes

3.1.6

Bug fix: Fixed a bug that prevented the crevice model from converging when running a simulation with [inputs.in](#) > *property_control* > *species_diffusion_model* greater than 0.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a simulation with [thermal runaway model.in](#) > *one_minus_alpha_order* set to a non-integer value.

CHT

Enhancement: In a CONVERGE + [GT-SUITE](#) co-simulation, you can now use CHT coupling between CONVERGE and the GT-SUITE 3D finite element solver to predict the surface temperature field at a WALL boundary.

Enhancement: Added a limit check to prevent CONVERGE from printing excessive CHT warnings to the [log file](#). These repeated warnings could substantially slow down a CHT simulation.

Bug fix: Fixed a bug that could cause CONVERGE to calculate the heat transfer incorrectly at a CHT [INTERFACE](#) with a moving solid stream on one side.

Bug fix: Fixed a bug that could cause CONVERGE to crash when a CHT [INTERFACE](#) was parallel to and very close to cell faces.

Bug fix: Fixed a bug that caused incorrect results in a simulation with CHT, [super-cycling](#), and a fixed mesh on a moving solid boundary.

Bug fix: Fixed a bug that caused CONVERGE to not correctly take time-dependent energy sources into account when running a simulation with [super-cycling](#).

FSI

Bug fix: Fixed a bug in the beam time-step calculation for the MMT model that caused some [FSI beam](#) cases to crash.

Bug fix: Fixed erroneous region index settings for beam events that caused some [FSI beam](#) cases to hang when run in parallel.

Bug fix: Fixed a memory leak in simulations with [FSI](#) modeling and multiple streams.

Bug fix: Fixed a bug that caused CONVERGE to incorrectly exit with an error for some valid [FSI](#) case setups.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [FSI beam](#) modeling and monitor points or monitor lines.

Bug fix: Fixed a bug that caused [seal](#) boundaries specified in *boundary.in* to not track the motion of their seal origin if the seal origin was an FSI object.

Bug fix: Fixed a bug that caused CONVERGE to write erroneous [FSI beam](#) output when run on multiple compute nodes.

Radiation

Enhancement: Added checks to detect invalid [radiation modeling](#) setups. CONVERGE will exit with an error when it detects such a setup.

Bug fix: Fixed a memory leak in [radiation modeling](#).

Aeroacoustics

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation with the [Fflowcs Williams-Hawkins](#) aeroacoustic model.

Bug fix: Previously, for multi-stream simulations with aeroacoustic modeling, CONVERGE was writing [acoustic receiver <ID> stream<ID> pressure.out](#) to the top-level output directory rather than the individual stream directory. This bug has been fixed.

Utilities

Bug fix: Fixed a bug in the [mechanism tune](#) utility that could cause 0D case types to revert incorrectly to CONSTANT_PRESSURE when running 0D and 1D cases simultaneously with the NLOpt solver.

UDF

Enhancement: Added some surface chemistry APIs.

Enhancement: Added APIs for accessing variables from thermal runaway models.

Enhancement: CONVERGE now writes a separate *global_variables.txt* file for each stream.

Enhancement: Added the *CONVERGE_get_cell_boundary_indices* API.

Enhancement: Added the *CONVERGE_source_set_value* API.

Enhancement: UDFs can now access the *bound_uplus* field variable via the FIELD macro.

Enhancement: You can now write position UDFs that are called before or after CONVERGE calculates sources. Use the *CONVERGE_BEFORE_SOURCE_MAIN* and *CONVERGE_AFTER_SOURCE_MAIN* macros for these UDFs.

Enhancement: In fixed flow simulations, UDFs can now access the *flow_is_currently_fixed* global variable via the *CONVERGE_get_int* API.

Enhancement: Added APIs for accessing flow-through INTERFACE boundaries.

Enhancement: You can now iterate over cells in source volumes during simulation setup. Previously, the *CONVERGE_source_iterator_create* API returned -1 during setup because the source cells had not yet been marked.

Enhancement: Added APIs for accessing boundary variables.

Enhancement: Added an error check for simulations that employ a user-defined turbulence model with an RNG $k-\epsilon$ model specified in *turbulence.in*. This combination of settings is not supported. CONVERGE calls user-defined $k-\epsilon$ models only when the standard $k-\epsilon$ model is specified in *turbulence.in*.

Enhancement: Added an error check for UDFs that call *CONVERGE_cloud_get_index* for spray parcel clouds. This API works only for film parcel clouds.

Enhancement: Added an error check for case setups with different values of *inputs.in > feature_control > udf_flag* in different streams. This setup is not supported. If UDFs are enabled, they must be enabled in all streams.

Enhancement: Added APIs to trigger reaction rate calculations in CONVERGE and retrieve reaction rate data.

Bug fix: Fixed a bug that caused CONVERGE to calculate species incorrectly when specifying a profile boundary condition for both species and velocity via UDF.

Bug fix: Fixed a bug that caused the *CONVERGE_get_heat_release_rate* API to return the cumulative heat release (*J*) instead of the heat release rate (*J/time*). This API now returns the heat release rate, defined as (cumulative heat release)/(duration). A new API has been added to retrieve the instantaneous heat release rate

(*CONVERGE_get_instantaneous_heat_release_rate*).

Bug fix: Fixed a bug that caused CONVERGE to pass some surface chemistry variables to UDFs incorrectly.

Bug fix: Removed obsolete function declarations and documentation for nozzle APIs that were previously removed from CONVERGE.

Bug fix: Fixed a bug that caused CONVERGE to crash when a nozzle UDF called the *CONVERGE_check_flag* or *CONVERGE_get_int* API.

Bug fix: Fixed a bug that caused CONVERGE to crash when UDFs were enabled and *hidden.in* was not in the Case Directory.

Bug fix: Fixed a memory leak in property UDFs.

Bug fix: Restored the ability to run *make install* when building a UDF library.

Bug fix: Fixed a bug in which variables defined in an onload UDF were not registered in all streams of a multi-stream simulation.

Bug fix: Fixed a bug that prevented CONVERGE from calling the *user_turbulent_statistics.c* UDF when this UDF was activated in *udf.in*.

Bug fix: Fixed a bug that caused CONVERGE to crash when running the *user_turbulent_statistics.c* UDF.

2.5 3.1.5

CONVERGE 3.1.5 is a minor release that includes enhancements and bug fixes.

Solver

Enhancement: Removed a deprecated error check that prevented CONVERGE from running a simulation [coupled with GT-SUITE](#).

Enhancement: CONVERGE now supports host names of up to 255 bytes on Linux.

Previously, CONVERGE only supported host names of up to 63 bytes.

Bug fix: Fixed a bug that caused CONVERGE to [recover](#) unnecessarily in a simulation with coupled transient and steady streams.

Bug fix: Fixed a bug that caused CONVERGE to use the wrong relaxation factor for the pressure equation for steady-state simulations using the [PISO](#) algorithm. This caused slow convergence behavior.

Bug fix: Fixed a bug that caused inaccurate momentum flux calculation when running a simulation with [central differencing](#) (*i.e.*, *solver.in > Numerical_Schemes > fv_upwind_factor > mom* less than 1.0).

Bug fix: Fixed a bug in how CONVERGE solved the total energy equation. There was an inconsistency in how the different energy modes were discretized. This bug reduced accuracy and stability in simulations that solved the total energy equation with large amounts of upwinding.

Bug fix: Fixed a bug in how CONVERGE calculated the net mass flow rate in the [steady-state monitor](#). This bug affected Case Setups that combine a periodic region with a non-periodic region, each with associated INFLOW and/or OUTFLOW boundaries (*e.g.*, the turbocharger Case Setup [described](#) in Chapter 9). CONVERGE now correctly applies a multiplicative factor to the boundaries associated with the periodic region.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a simulation with [mixing planes](#) and *solver.in > Numerical_schemes > rc_flag = GENERALIZED_RC*.

Bug fix: Fixed several bugs that caused incorrect steady-state monitor behavior after restarting a steady-state simulation with emissions modeling.

Bug fix: Fixed a bug that could cause CONVERGE to crash when starting up a simulation on Windows.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a simulation with *solver.in > Numerical_schemes > rc_flag = GENERALIZED_RC* and *inputs.in > feature_control > mrf_flag = 2*.

Bug fix: Fixed a bug that could cause CONVERGE to crash when restarting a simulation with coupled [steady-solver](#) and [transient-solver streams](#).

Input

Enhancement: You can now specify a temporal profile for side clearances in the [crevice model](#).

Enhancement: Added a warning check to detect Case Setups with *inputs.in > solver_control > species_solver = 0* and multiple [species](#) in the domain. This combination of settings is not recommended and may produce non-physical results.

Enhancement: Made [proximity_boundaries.in](#) stream-based.

Enhancement: You can specify the forward or reverse side of an INTERFACE boundary with a positive-signed or negative-signed value of the INTERFACE boundary ID in [min_vol_ratio.in](#).

Bug fix: Fixed several bugs in how CONVERGE performed warning checks for some input parameters for steady-state simulations.

Output/Post-Processing

Enhancement: Added an optional *cells > general > vol_frac_uniformity_index* variable to [uniformity index](#) output.

Enhancement: Added boundary position information metadata to [post*.h5](#).

Enhancement: In [parcel_flow.out](#), input boundaries are now reported as negative values, and outflow boundaries are reported as positive values.

Enhancement: Removed the *post_slice* output feature. [ParaView Catalyst](#) provides superior slice output, as well as other advanced functionality.

Bug fix: Fixed an initialization bug that reported incorrect flow rates to [parcel_flow.out](#).

Bug fix: Fixed a bug that caused CONVERGE to crash when *inputs.in > output_control > wall_output_flag* was specified as a file name.

Bug fix: Fixed several bugs that could cause CONVERGE to double-count mass flow across boundary/region interfaces for writing to [domain-averaged output files](#) in ELSA simulations.

Bug fix: Previously, CONVERGE did not correctly apply *inputs.in > output_control > twrite_restart_max_wct* after restarting a simulation. This parameter was only functional after CONVERGE wrote the first restart file of the new simulation. It is now active as soon as CONVERGE restarts.

Bug fix: Fixed a bug that caused CONVERGE to not write several [echo files](#) correctly.

Bug fix: Fixed a bug that caused CONVERGE to fail to write some data to [transfer.out](#).

Bug fix: Fixed a bug that prevented CONVERGE from correctly writing aeroacoustic output variables to [post*.h5](#).

Bug fix: Fixed a bug in how output files are initialized. This could cause CONVERGE to crash when performing a [latency check](#) on a parallel simulation.

Bug fix: Fixed a bug in how CONVERGE wrote *elsa_mfrac* to [post*.h5](#).

Bug fix: Fixed a bug that caused CONVERGE to write incorrect film temperatures to [post*.h5](#).

Bug fix: Fixed a bug that could cause CONVERGE to write incorrect data to [post*.h5](#).

Bug fix: Fixed a bug that caused parallel simulations on Windows to report failure to the [log file](#). This bug occurred after simulation completion and did not affect the solution.

Boundaries and Boundary Conditions

Enhancement: You can now specify a temporal or spatial profile for surface roughness in [boundary.in](#).

Enhancement: When calculating the pressure gradient on a boundary with a Neumann pressure boundary condition, CONVERGE now includes the effects of [body force](#).

Enhancement: Added a log file note that DEPENDENT boundaries can only be specified on the first [stream](#).

Enhancement: Removed an error check that prevented uncoupled flow through [INTERFACE](#) boundaries from being configured with a heat flux or Neumann temperature boundary.

Bug fix: Fixed a memory leak in the [Navier-Stokes Characteristic Boundary Condition](#) (NSCBC).

Bug fix: Fixed several bugs in data structures for [passives](#). These bugs caused passive boundary conditions to be set incorrectly.

Bug fix: Fixed a bug that could cause CONVERGE to crash after a restart for simulations with rotating [WALL](#) or [MRF](#) boundaries.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [translating](#) boundaries.

Bug fix: Fixed a bug in surface motion operation logic. Previously, if a Case Setup had triangle vertices that [moved](#) due to another stream's motion, but no moving boundaries of its own, some surface motion operations would be skipped. This caused CONVERGE to crash.

Regions and Streams

Enhancement: You can now configure *mrf.in > mrf_region > specification > method = COPY_FROM_BOUNDARY* with [INTERFACE](#) boundary IDs. A positive-valued ID indicates the forward side of the INTERFACE boundary. A negative-valued ID indicates the reverse side of the INTERFACE boundary.

Bug fix: Fixed a bug that caused CONVERGE to crash if any of the following flags in *inputs.in* were set to 1 for a simulation with at least one solid [stream](#): *property_control > tabular_fluid_prop_flag*, *property_control > real_gas_prop_flag*, *property_control > lhv_flag*, or *feature_control > vof_flag*.

Bug fix: Fixed a bug that could cause CONVERGE to crash when starting up a simulation with composite species and multiple [streams](#).

Bug fix: Fixed a bug that could cause region IDs to be assigned incorrectly in a simulation with [disconnect triangles](#) or [sealing](#) and dependent regions.

Bug fix: Fixed a bug that could cause CONVERGE to crash when defining a passive in a solid [stream](#).

Grid Control

Enhancement: You can now specify passive-based [AMR](#) in solid streams.

Enhancement: Added warning and error checks to maximum embed scale in [amr.in](#).

CONVERGE will exit with an error if the embed scale is 32 or greater and print a warning to the log file if the embed scale is 16 or greater.

Enhancement: CONVERGE can now reloft [disconnect triangles](#).

Enhancement: Patch-type reloft boundaries now support CHT.

Bug fix: Fixed a bug for cases using the steady-state monitor with [gridscale.in > gridscale > time = AUTO](#), which caused CONVERGE to apply AMR at every cycle instead of at the

frequency specified in *amr.in > amr_settings > cycle_steady*. CONVERGE now applies AMR at the *cycle_steady* frequency.

Bug fix: Fixed a bug that could cause CONVERGE to crash during setup for simulations with [inlaid mesh](#) motion.

Bug fix: Fixed a bug in grid data structures that could cause CONVERGE to crash during simulation startup.

Bug fix: Fixed a bug that could cause CONVERGE to crash when performing [cut-cell](#) operations.

Bug fix: Fixed a bug that prevented CONVERGE from breaking [seals](#) in some cases.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [sealing](#).

Bug fix: Fixed a bug in surface intersection detection that could cause CONVERGE to crash on a simulation with [sealing](#).

Bug fix: Fixed several bugs in the implementation of moving [inlaid meshes](#). These bugs could cause CONVERGE to crash, generate inaccurate results, or enter a recovery loop from which CONVERGE could not break out.

Materials

Enhancement: Made non-Newtonian [fluid](#) modeling stream-based.

Bug fix: Fixed a bug that caused CONVERGE to calculate the next time-step size incorrectly for a multi-stream simulation with multiple [fluid](#) species.

Bug fix: Fixed a bug in error checking logic that could cause CONVERGE to incorrectly exit with an error. This error reported repeated surface species definitions.

Initialization

Bug fix: When initializing a simulation from a map file that does not contain velocity information, and when a swirl profile is specified in [engine.in](#), CONVERGE should initialize velocities from the swirl profile. Previously, CONVERGE was incorrectly initializing velocities from *initialize.in*.

Discrete Phase

Enhancement: For simulations with [wall films](#), CONVERGE now solves for wall temperature with implicit film temperature coupling. This improves solver stability and may cause minor changes in local temperature and film mass.

Enhancement: Added error checking for some discrete phase modeling inputs.

Enhancement: You can no longer specify the KH-RT or LISA breakup models to simulate parcel breakup in [ELSA](#) simulations. Both of these models incorporate a mechanism-switching behavior based on a breakup length, while ELSA models breakup directly. You can run an ELSA simulation with any of the KH, KH-ACT, RT, modified KH-RT, or TAB breakup models.

Enhancement: Sped up [ELSA](#) passive initialization and Eulerian to Lagrangian conversion.

Enhancement: For a simulation with [ELSA](#), if *parcel_introduction.in > injections > injection > injection_control > injection_velocity_control > mass_per_parcel = 0*, CONVERGE will use the value of *elsa.in > stream > elsa_transition_criteria > elsa_mass_per_parcel*. Previously, this combination of settings would cause CONVERGE to crash.

Enhancement: Refactored some parcel-gas coupling routines for a modest reduction in runtime.

Chapter 2: Release Notes

3.1.5

Bug fix: Fixed a bug that caused CONVERGE to not correctly initialize [ELSA](#) passives when running with Intel MPI. CONVERGE would run, but no mass was converted between phases.

Bug fix: Fixed a bug that caused CONVERGE to crash on a simulation when: there is more than one fluid stream; one stream has discrete phase modeling but another fluid stream does not; there is a contact resistance event.

Bug fix: Previously, in simulations that use conjugate heat transfer (CHT) modeling without urea modeling, CONVERGE incorrectly used the near-wall cell temperature to calculate [film splash](#) criteria for the Kuhnke and Bai-Gosman models. CONVERGE now uses the wall temperature, as designed.

Bug fix: Fixed a bug that caused CONVERGE to calculate solid-stream temperature incorrectly on a simulation with CHT and [wall films](#).

Bug fix: Fixed a bug that caused CONVERGE to crash if a simulation included [boundary injection](#) but no nozzle-type injectors.

Bug fix: Fixed a bug that prevented CONVERGE from correctly writing a message to the [log file](#) when importing a map file with parcels.

Bug fix: Fixed a bug that caused CONVERGE to not initialize films correctly on a simulation with [stream-based](#) film settings.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [stream-based](#) injection settings.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a simulation with the [Bai-Gosman](#) film splash model and *parcels.in > liquid_parcels > parcel_list > parcel > evap_control > urea_model = NONE*.

Bug fix: Fixed several bugs that could cause CONVERGE to crash on a simulation with [wall films](#).

Bug fix: Previously, CONVERGE performed a [flash boiling](#) input check that could cause the simulation to terminate with an error. CONVERGE now prints a warning to the log file when this combination of inputs is detected, which is the correct behavior.

Bug fix: Fixed several bugs in parcel subroutines that could cause CONVERGE to incorrectly transport parcels across a [seal](#) or could cause a crash.

Bug fix: Fixed a bug that could cause CONVERGE to crash when releasing [embedding](#) on a simulation with discrete phase modeling.

Bug fix: Fixed some incorrect logic in some parcel warning checks.

VOF

Enhancement: You can now specify saturation temperature for output to [post*.h5](#).

Enhancement: Added the [long-scale interface](#) mixture model for VOF simulations.

Enhancement: Added an injector-copy feature for [VOF-spray one-way coupling](#).

Bug fix: Previously, on a simulation with [VOF-spray one-way coupling](#) and an unrelated nozzle injector, the latter injector was non-functional. This bug has been fixed.

Fixed Flow

Bug fix: Fixed a bug that caused CONVERGE to apply [erosion](#) at incorrect times when running a simulation with [fixed flow](#) and time specified in crank angle degrees.

Bug fix: Fixed a bug that caused CONVERGE to apply [fixed flow](#) at incorrect times when running a simulation with a variable RPM.

Chapter 2: Release Notes

3.1.5

Bug fix: Fixed a bug that could cause CONVERGE to crash when initiating a [fixed flow](#) interval in a simulation with discrete phase modeling and turbulent statistics collection.

Combustion

Enhancement: Previously, specifying duplicate reactions with a negative pre-exponential A factor would cause CONVERGE to exit with an error. CONVERGE now prints a warning to the [log file](#).

Enhancement: Added a check to detect missing [PFA](#) input files. If a required file is missing, CONVERGE now exits with an error.

Enhancement: Added a check to detect when CONVERGE is attempting to extrapolate outside the range of a TKI table. If this is detected, CONVERGE now prints a warning to the [log file](#).

Enhancement: CONVERGE now supports species with molecular weights above 10000. Previously, if such massive species were specified, CONVERGE would exit with an error.

Enhancement: Added an error check for 0D and 1D chemistry simulations that are missing [therm.dat](#).

Enhancement: Made additional combustion subroutines [stream-based](#).

Bug fix: Fixed a bug that would cause [ECFM](#) or [ECFM3Z](#) cases to crash when parcels contain non-fuel species such as water, or with composite species.

Bug fix: Fixed a bug that prevented CVODE error messages from being written to the [log file](#) for some chemistry cases.

Enhancement: You can now turn on detailed soot models when running a simulation with [ECFM3Z](#) and *combust.in > ecfm3z_model > post_flame_model = 2*.

Bug fix: Fixed a bug that caused CONVERGE to not print runtime to the [log file](#) when running some mechanism tuning cases.

Bug fix: Fixed a bug that prevented CONVERGE from monitoring *react_ratio* as specified in [monitor_points.in](#).

Bug fix: Fixed a bug in the implementation of the [Gulder laminar flamespeed correlation](#) that caused CONVERGE to calculate a non-zero laminar flamespeed when the equivalence ratio was zero.

Bug fix: Fixed a bug in error checking logic that would cause CONVERGE to exit with an error if a case setup included [surface chemistry](#) but did not specify SAGE for solving combustion in the domain.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a simulation that used the SAGE [three-point PDF](#) method in one stream but not another.

Bug fix: Fixed a bug that could cause CONVERGE to not call combustion subroutines when *combust.in > general_control > temporal_type = CYCLIC* and *end_time* was smaller than *start_time*.

Emissions

Bug fix: Fixed a bug that caused CONVERGE to calculate [emissions post-processing](#) outputs incorrectly when running a simulation with the SIMPLE iterative algorithm.

Sources

Enhancement: Improved the logic for crevice cell tagging. The new logic improves [crevice model](#) convergence behavior.

Chapter 2: Release Notes

3.1.5

Enhancement: Refactored routines for the local thermal non-equilibrium (LTNE) model to calculate fluid-solid coupling in [porous media](#) more efficiently.

Enhancement: In [porous media](#), you can now use `source.in > source > porosity_data > eff_conductivity_flag = 1` in conjunction with the LTNE model. When `eff_conductivity_flag = 1`, CONVERGE uses the effective thermal conductivity for both the fluid and the solid. Previously, this combination of features was not supported.

Bug fix: Fixed a bug that caused SAGE to calculate the combustion source term for the fluid energy equation incorrectly in porous media when `source.in > source > porosity_data > solid_transient_flag = LTE` or LTNE.

Bug fix: Fixed a bug that caused CONVERGE to calculate the `dt_chem` time-step limiter incorrectly for cases using combustion modeling in [porous media](#) with `source.in > source > porosity_data > solid_transient_flag = LTE` or LTNE.

Bug fix: Fixed a bug in the LTNE model that caused the solid temperature to be calculated incorrectly after grid changes.

Bug fix: Fixed a memory leak in source setup.

CHT

Bug fix: Fixed a bug in MRF error checking. CONVERGE version 3.1.4 introduced a check for [INTERFACE](#) boundaries with different MRF settings. CONVERGE would exit with an error if this was detected. Non-axisymmetric boundaries must have identical MRF settings on either side, and they must have `method = COPY_FROM_BOUNDARY`. The revised logic allows axisymmetric INTERFACE boundaries to have different MRF settings, and it does not restrict such boundaries to have `mrf.in > mrf_region > specification > method = COPY_FROM_BOUNDARY`.

Bug fix: Fixed a bug that could cause CONVERGE to incorrectly calculate the integrated heat release written to `thermo.out` on a simulation with [super-cycling](#) and a moving source.

Bug fix: Fixed a bug in how CONVERGE applies a coordinate transformation for boundary condition mapping in a simulation with [super-cycling](#).

Bug fix: Fixed a bug that caused CONVERGE to advance time inconsistently on transient simulations with [super-cycling](#).

FSI

Enhancement: CONVERGE now solves [FSI beams](#) in parallel.

Enhancement: For coupled co-simulations with [Abaqus](#), you can now set up this coupling to be implicit in time. This improves convergence and simulation stability, especially when fluids and solids are of similar density.

Bug fix: Previously, if an FSI boundary had prescribed motion before the [FSI](#) start time, this prescribed motion was not applied. Prescribed motion is now applied if specified before the FSI start time.

Bug fix: Fixed a bug in the [FSI spring](#) model that caused CONVERGE to calculate damping incorrectly.

Bug fix: Fixed a bug in the [FSI sub-spring](#) model that caused the sub-springs to be located incorrectly.

Bug fix: Fixed a bug that caused CONVERGE to over-predict buoyancy forces in simulations with [FSI](#).

Radiation

Enhancement: Redesigned some radiation solver data structures to reduce memory usage.

Enhancement: Refactored some radiation solver routines to reduce runtime.

Bug fix: Fixed a bug in the [FVM](#) radiation model that caused CONVERGE to calculate the absorption coefficient incorrectly.

Utilities

Enhancement: On Windows, CONVERGE now performs a network latency check at the beginning of a simulation. Latency information is printed to the screen and to [*mpi_latency.log*](#). This feature was previously only available on Linux.

Enhancement: The [*cleantherm*](#) utility now prints an error message instead of crashing if the *mech.dat* file is missing.

Enhancement: When running the [*mechanism tune*](#) utility with *mechanism_tune.in > solver_type = NLOPT*, you can now adjust reaction rates for pressure-dependent reactions that are marked with the keyword LOW in the reaction mechanism file. Previously, the utility tuned pressure-dependent reactions only if they were marked with the keyword PLOG.

Bug fix: Fixed several bugs in the [1D](#) PISO solver that could cause CONVERGE to crash.

Bug fix: Fixed a bug that could cause CONVERGE to not print a time limit warning when a [1D](#) case reached the limit specified by *one_d_solver.in > one_d_general_control > solver_control > case_time_limit*.

Bug fix: Fixed a bug that caused CONVERGE to calculate incorrect results when running NLopt [*mechanism tuning*](#) with Intel MPI.

Bug fix: Fixed a memory leak in mechanism tuning for [*well-stirred reactors*](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a [0D](#) or [*mechanism tuning*](#) simulation.

UDF

Enhancement: Added APIs for accessing injector rate-shape tables.

Enhancement: Added the *CONVERGE_species_index_with_phase* API to retrieve the species index based on the species name and phase.

Enhancement: Added the *CONVERGE_spray_flag_is_WALL_FILM* API.

Enhancement: UDFs can now access global variables from *udf.in*. These variables are listed in *global_variables.txt* and can be accessed with the *CONVERGE_get_int* API.

Enhancement: Added an example UDF for calculating the molecular viscosity of non-Newtonian liquids (*liquid_non_newtonian_visc.c*).

Enhancement: Added the *CONVERGE_clouds_get_local_parcel_species_index* and *CONVERGE_clouds_get_stream_parcel_species_index* APIs.

Bug fix: Fixed a bug that caused certain case setups to crash when an initialization UDF tried to access the *wall_dist* field variable.

Bug fix: Fixed a bug that caused CONVERGE to crash if a UDF did not calculate the same corrections to piston motion for the forward and reverse sides of an interface. Now, CONVERGE prints a warning and applies the correction for the forward side to both sides.

Bug fix: Fixed a bug that caused CONVERGE to incorrectly calculate cavitation rates when specified via UDF.

Chapter 2: Release Notes

3.1.5

Bug fix: Fixed a bug in which CONVERGE did not allocate memory and did not define the stream index for surface variables, causing CONVERGE to crash when a UDF tried to access those variables.

Bug fix: Fixed a bug that caused certain case setups to crash when adaptive zoning variables were defined via UDFs.

Bug fix: Fixed a bug that caused parcel-related global variables to be omitted from the *global_variables.txt* file. CONVERGE now writes parcel-related variables to *global_variables.txt*.

Bug fix: Fixed a bug that caused CONVERGE to unnecessarily calculate reaction rates twice when specifying reaction rate calculation via UDF.

Bug fix: Fixed a bug that caused CONVERGE to pass incorrect species information to the *spray_kh.c* UDF.

Bug fix: Fixed a bug that prevented UDFs from accessing thermal runaway source variables.

Bug fix: Fixed a bug that could cause CONVERGE to print warnings about recoveries to the log file when UDF time-step limiters were in use, even though no recoveries were happening.

Bug fix: Fixed a bug in the *user_passive_transport.c* example UDF that caused the UDF to access face-centered variables incorrectly.

2.6 3.1.4

CONVERGE 3.1.4 is the official release of CONVERGE 3.1. These release notes describe enhancements and bug fixes made after CONVERGE 3.1.3 was released. Please see the [Major Changes from CONVERGE 3.0 to 3.1](#) section for details about the new features and other large updates first implemented in CONVERGE 3.1.

Solver

Enhancement: Refactored some solver routines to improve performance when CONVERGE is solving the pressure equation with the [CONVERGE BiCGSTAB](#) linear solver.

Enhancement: Altered the implementation of the [steady-state monitor](#) when monitoring net mass flow rate. The new formulation ensures that the net mass flow rate is driven below the specified tolerance. The previous logic checked that the net mass flow rate was unchanging, but it did not check that it was nearly zero.

Enhancement: For transient simulations with [multiple streams](#), CONVERGE now automatically synchronizes the time-step across physically-connected streams in a way that is less restrictive of some combinations of case setup parameters.

Enhancement: Added a reduced SOR (RSOR) preconditioner option for solving with [CONVERGE BiCGSTAB on GPUs](#).

Enhancement: Refactored some routines in the [Krylov](#) enthalpy solver. This improves temperature predictions for case setups that include anisotropic thermal conductivity.

Enhancement: CONVERGE now pre-computes some radiation solver matrices, reducing computational cost when solving the radiation transport equation with the [SOR](#) linear solver.

Enhancement: For [steady-state](#) simulations, CONVERGE previously started counting the number of times AMR was applied for *solver.in > Steady_state_solver > steady_min_num_amr* after the simulation had converged on the final gridscale. CONVERGE now starts counting

the number of times AMR was applied immediately after the final gridscale is specified. This saves computational cost by not applying AMR when it is not necessary.

Enhancement: Altered the way CONVERGE applies *dt_grow* for [steady-state](#) simulations. Previously, for steady-state simulations, CONVERGE applied *dt_grow* in a manner that was less numerically robust. The difference in *dt_grow* logic between steady-state and transient simulations had not previously been documented.

Enhancement: Made several alterations to the total energy solver to improve speed and accuracy.

Enhancement: CONVERGE now includes the internal energy solution error when calculating the [PISO](#) error for solid streams.

Enhancement: Added several error checks to detect invalid [MRF](#) case setups.

Bug fix: Fixed several bugs in the total energy solver that caused some inaccuracies in simulations that use the [MUSCL flux scheme](#) or involve high Mach number flows.

Bug fix: Fixed several bugs in how CONVERGE [recovers](#) and prints time-step limiter information for simulations with multiple streams. Previously, information from the wrong stream could be used for the recovery logic or written to the log file.

Bug fix: Fixed a bug that prevented the [steady-state monitor](#) from functioning properly on simulations with multiple streams and *monitor_steady_state.in > monitor_units = SECONDS*.

Bug fix: Fixed several bugs in the solution of the momentum transport equation that could cause fluid near a boundary moving at high speed to reach physically inappropriate velocities.

Bug fix: Fixed a bug in shared memory that caused CONVERGE to not deallocate all allocated memory. This caused some MPI versions to report an error at the end of a simulation.

Bug fix: When running a simulation with *solver.in > Numerical_Schemes > strict_conserve_level = 1 or 2*, CONVERGE previously solved scalar transport with the Jacobi method at every iteration, which was not the designed behavior. CONVERGE now solves scalar transport with SOR for every iteration except for the final iteration (which is solved with the Jacobi method), which is the designed behavior.

Bug fix: Fixed a bug that caused CONVERGE to calculate inaccurate results on some simulations with VOF and more than one [stream](#).

Bug fix: Fixed a bug that caused CONVERGE to crash when writing a restart file on some simulations with [stream-based settings overrides](#) and the same [steady-state monitor](#) quantity defined on multiple streams.

Bug fix: Fixed a bug that could cause CONVERGE to hang on a combusting simulation with multiple [streams](#).

Input

Enhancement: For specifying proximity groups for [proximity AMR](#), the nomenclature for specifying the boundary IDs for INTERFACE boundaries has been altered. Use the positive-signed boundary ID to specify proximity grouping on the forward side of the INTERFACE boundary. Use the negative-signed boundary ID to apply the proximity grouping on the reverse side of the boundary.

Enhancement: CONVERGE now handles [stream-based](#) map overrides in a more general way. Previously, *map.in* was required in all stream-by-stream override directories, even though the contents of the file were identical for all copies of the file.

Enhancement: Added a passive match feature to [*stream_match_species.in*](#).

Bug fix: Fixed a bug that caused CONVERGE to exit with an error when starting a simulation with [tabular fluid properties](#) and no *gas.dat*. The latter file is not needed for a simulation with tabular fluid properties.

Bug fix: Fixed a bug that caused CONVERGE to exit with an error when starting a simulation with [tabular fluid properties](#) and solid streams when these tabulated properties were not supplied for the solid stream. Solid streams do not have tabulated fluid properties.

Bug fix: Fixed a bug that caused CONVERGE to crash when specifying a [scalar](#) in [*species.in*](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when changing [*inputs.in* > output_control > log_level](#) from 2 to 3 during a simulation.

Output/Post-Processing

Enhancement: Added a new [volume-averaged output file](#) for monitoring simulation progress, *mass_balance.out*.

Enhancement: Altered some logic in [*post*.h5*](#) output to reduce small inaccuracies that could be introduced by the motion of solid streams.

Enhancement: Altered the [log file](#) language of some sensible enthalpy solver iteration messages for greater clarity.

Enhancement: Residuals for the turbulence transport equations are now written to [*residuals.out*](#).

Enhancement: CONVERGE now writes *cell_count_ranks.out* on a [stream-by-stream](#) basis, and a global *cell_count_ranks.out* in the Case Directory. The global *cell_count_ranks.out* includes the cycle number and the cell counts. The stream-by-stream *cell_count_ranks.out* include all of this information as well as the simulation time.

Enhancement: Replaced simulation time with cycle number in *memory_usage.out*.

Enhancement: CONVERGE now writes more granular per-time-step timing data to the [log file](#), ensuring that the breakdown of different portions of the time-step sums to 100%.

Enhancement: Added an error check to detect if [*solver.in* > steady_state_solver > steady_residual_output_flag = 1](#) in a transient simulation. This combination of inputs is not allowed.

Enhancement: You can now direct CONVERGE to not write map, restart, or post files by specifying [*inputs.in* > output_control > write_map_flag, twrite_restart, or twrite_post = -1](#), respectively. This avoids the substantial file write time and disk space requirements for large simulations when these files are not desired.

Enhancement: CONVERGE now writes *cell_count_regions.out* on a [stream-by-stream](#) basis.

Enhancement: When running a steady-state simulation, CONVERGE now prints all steady-state solver information to the [log file](#) at all *log_levels*. Previously, the amount of information printed at *log_level* = 1 was insufficient to fully monitor steady-state solver behavior.

Bug fix: Fixed a bug in ParaView Catalyst that could cause CONVERGE to incorrectly exit with an error on a simulation with [*inputs.in* > simulation_control > crank_flag = 0 or 2](#).

Bug fix: Fixed a bug in how CONVERGE calculated viscosity for printing to [*thermo.out*](#). Previously, CONVERGE would also include solid streams in this calculation. Now, CONVERGE correctly only calculates over fluid streams.

Bug fix: Previously, for transient simulations that used the [steady-state monitor](#), CONVERGE did not write *transfer.out* at the end of the simulation. This has been corrected.

Chapter 2: Release Notes

3.1.4

Bug fix: Fixed a bug that caused CONVERGE to write wall shear stress files with an incorrect space in the file name, i.e., [wall stress<number>_<time>.out](#).

Bug fix: Fixed a bug that could cause CONVERGE to write a corrupt [post*.h5](#) if a surface vertex was very close to a defined region of embedding.

Bug fix: Fixed a bug that caused CONVERGE to output incorrect information to [area_avg_regions_flow.out](#) or [mass_avg_regions_flow.out](#) after restarting a simulation.

Bug fix: Fixed a bug in outputting [numerics.out](#) when running with S-A, RSM SSG, or one-equation eddy viscosity LES turbulence models.

Bug fix: Fixed a bug that could cause CONVERGE to not write a final [post*.h5](#) for a multi-stream simulation.

Bug fix: Fixed a bug in how CONVERGE outputs [echo files](#). Previously, input files were copied at the file system level to corresponding echo files. This was not the intended behavior of echo files. Now, CONVERGE correctly reads the YAML-formatted input files, then outputs YAML-formatted echo files.

Bug fix: Fixed a bug that could prevent ParaView Catalyst from outputting data when the time-step became larger than [paraview_catalyst.in > scripts > script > timing_control > twrite](#).

Bug fix: Fixed a bug that caused CONVERGE to not apply [inputs.in > output_control > twrite_restart_max_wct](#) correctly in a simulation that initialized from a restart file.

Bug fix: Fixed a bug in how CONVERGE tracked solver time when running a fixed flow simulation. Not all time was reported correctly to the [log file](#).

Boundaries and Boundary Conditions

Enhancement: Added a new correction-based [Navier-Stokes Characteristic Boundary Condition](#). The old NSCBC implementation is still available.

Enhancement: For case setups with gravity, CONVERGE now takes gravity into account when setting Neumann pressure boundary conditions on [WALL](#) boundaries.

Enhancement: Added a new type of [boundary motion](#) that interpolates between multiple surface files and vertex lists.

Enhancement: Added several error and warning checks to detect disallowed combinations of flow-through INTERFACE boundaries with [boundary motion](#).

Bug fix: Fixed a bug that could cause CONVERGE to set a solid stream velocity incorrectly if it had any boundaries that were not [WALL-type](#).

Bug fix: Fixed a bug that caused CONVERGE to incorrectly report parcel mass flow rates through flow through [INTERFACE](#) boundaries, which were then written to inter-region output files.

Bug fix: Fixed a bug in how CONVERGE calculates velocity divergence and the pressure work term in the total energy equation when [inputs.in > feature_control > mrf_flag = 2](#). This could cause CONVERGE to crash.

Bug fix: Fixed a bug in how CONVERGE calculated a gravity transformation for [PERIODIC](#) boundaries, causing inaccurate results in simulations that used these features.

Bug fix: Fixed a bug that could cause CONVERGE to crash when a boundary with [ARBITRARY](#)-type motion was translating without any rotation.

Bug fix: Fixed a bug that caused CONVERGE to restart with the wrong mass flow rate when an [INFLOW](#) boundary was configured with [boundary.in > boundary > nscbc > method = POINTSOT_LELE](#).

Bug fix: Fixed a bug that could cause CONVERGE to not calculate [backflow](#) correctly.

Chapter 2: Release Notes

3.1.4

Bug fix: Fixed a bug that could cause CONVERGE to crash when a case setup included a baffle [INTERFACE](#) boundary.

Regions and Streams

Bug fix: Fixed a bug that caused CONVERGE to calculate inaccurate results on some simulations with VOF and more than one [stream](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation that included two [streams](#) separated by an INTERFACE boundary.

Grid Control

Enhancement: [Sealing](#) has been re-enabled with a new, more flexible implementation.

Enhancement: CONVERGE now supports the deformation of [inlaid meshes](#) due to boundary motion.

Enhancement: Previously, a [cell pair](#) warning printed the xyz location of a low-quality cell pair to the log file. CONVERGE now also prints the IDs of the boundaries that are cutting the cells of the pair.

Enhancement: Altered the [cell pairing](#) detection logic to improve the quality of cell pairs.

Enhancement: Refined the logic for calling some grid operations to avoid calling them when unnecessary.

Bug fix: Fixed a bug that could cause CONVERGE to crash when an INTERFACE boundary had an [inlaid mesh](#) on both sides.

Bug fix: Fixed a memory overflow bug in grid [embedding](#) and releasing that could cause CONVERGE to crash.

Bug fix: Fixed a bug that could cause CONVERGE to crash during restart when specifying a negative value of [gridscale](#).

Bug fix: For simulations with INTERFACE boundaries, CONVERGE attempts to detect [poor-quality triangulations](#). Fixed a bug in the detection logic that could cause CONVERGE to crash.

Bug fix: Fixed a bug that could cause an [inlaid mesh](#) event to incorrectly alter the local value of a passive.

Bug fix: Fixed a bug that could cause CONVERGE to crash for simulations with CHT but no grid operations ([AMR](#), etc.) during the simulation.

Materials

Enhancement: CONVERGE now supports the specification of multiple species in [fluid properties.dat](#).

Bug fix: Fixed a bug that could cause CONVERGE to read [fluid properties.dat](#) incorrectly.

Initialization

Enhancement: Added a check and warning to indicate that case setups that use an ASCII-formatted [map file](#) do not support *map.in > map_phase_flag = 1*.

Bug fix: Fixed several bugs in [mapping](#) that could cause CONVERGE to crash.

Turbulence

Enhancement: When running a [large eddy simulation](#) with the dynamic structure model or the consistent dynamic structure model, CONVERGE now outputs the average and standard

Chapter 2: Release Notes

3.1.4

deviation of turbulent kinematic viscosity, turbulent dynamic viscosity, and the ratio of molecular to turbulent viscosity to *turbulence.out*.

Bug fix: Fixed a bug in the [enhanced wall treatment](#) that was causing the viscosity blending function to be calculated incorrectly for RANS turbulence models.

Discrete Phase

Enhancement: For simulations with radiation/spray coupling (*i.e.*, [radiation.in](#) > *radiation_spray_coupling* > *rad_spray_coupling_flag* = 1) and [discretized drop temperatures](#) (*i.e.*, *spray.in* > *evap_control* > *drop* > *drop_temp_discretization* > *active* = 1), CONVERGE accounts for the discretized drop temperature when solving radiation/spray coupling. Previously, drop temperature was considered constant when solving radiation/spray coupling.

Bug fix: Fixed several bugs that caused CONVERGE to calculate [solid parcel](#) motion incorrectly on moving inlaid meshes.

Bug fix: Fixed a bug in spray and film evaporation that could cause simulations with [ELSA](#) to enter a recovery loop from which CONVERGE could not break out.

Bug fix: Removed an incorrect error check about parcel mass removal.

Bug fix: Fixed a bug in discrete phase modeling that could cause incorrect results in simulations with PERIODIC boundaries.

Bug fix: Fixed a bug that caused CONVERGE to crash when [film evaporation](#) is on but [drop evaporation](#) is off.

Bug fix: Fixed a bug that could cause CONVERGE to crash when attempting to reloft a triangle with [film parcels](#) on it.

Bug fix: Fixed a bug that could cause CONVERGE to crash when initializing a [wall film](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash if a case setup specified a parcel species with the same name as a gas species.

Bug fix: Fixed a bug in ELSA that caused CONVERGE to calculate a very large surface area density residual when [solver.in](#) > *Numerical_Schemes* > *strict_conserve_level* = 2 and CONVERGE was performing an SOR iteration.

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation with GT-POWER coupling and without discrete phase modeling.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [ELSA](#) and multiple streams.

Bug fix: Fixed a bug that caused CONVERGE to not introduce [solid parcels](#) when a case setup included composite species.

Bug fix: Fixed a bug that could cause CONVERGE to incorrectly remove parcels in some simulations with moving geometries.

Bug fix: Fixed a bug that could cause CONVERGE to become unresponsive when running a parallel simulation with parcels.

Bug fix: For simulations with VOF-spray one-way coupling and [parcel_introduction.in](#) > *injectors* > *injector* > *injector_control* > *vof_oneway_mapping* > *transformation* = NOZZLE-DIAMETER, CONVERGE previously used the diameter of the first listed nozzle to calculate parcel radius. CONVERGE now correctly uses the diameter of each listed nozzle to calculate parcel radius for that nozzle.

VOF

Chapter 2: Release Notes

3.1.4

Enhancement: CONVERGE now properly calculates the time spent performing [FCT](#) calculations and records it to the solver section in the log file.

Bug fix: Fixed a bug in [FCT](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug in the VOF [mixture model](#) that could cause incorrect results near solid walls.

Bug fix: Fixed a bug that caused CONVERGE to not correctly account for heat conduction in the internal energy equation when solving [species-density-based](#) VOF.

Fixed Flow

Enhancement: Added an error check to confirm that, for a fixed flow simulation, the specified [fixed_flow.in](#) > *fixed_time_control* > *fixed_intervals* lie between the *start_time* and *end_time*.

Enhancement: Added an optional [fixed_flow.in](#) > *solver_control* settings block to control behavior of the fixed flow solver.

Bug fix: Fixed a bug that caused CONVERGE to calculate interface temperatures incorrectly when deactivating [fixed flow](#) on a multi-stream simulation.

Bug fix: Fixed several bugs that prevented CONVERGE from calculating species correctly during [fixed flow](#) intervals.

Combustion

Enhancement: You can now run an [ECFM](#) or [ECFM3Z](#) simulation with composite fuel species.

Enhancement: You can now specify a temporally-varying [SAGE](#) reaction multiplier.

Enhancement: Modified the Zimont turbulent flamespeed model for LES simulations for improved accuracy.

Bug fix: Fixed a memory leak in [surface chemistry](#) output.

Bug fix: Fixed a bug in adaptive zoning that could cause inaccurate results when running a combusting simulation using this feature with [SAGE](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation with the [ECFM](#) or [ECFM3Z](#) combustion model and *combust.in* > *ecfm_model* > *post_ceq_flag* = 0 or *ecfm3z_model* > *post_ceq_flag* = 0.

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation with surface chemistry.

Bug fix: Fixed a bug that could cause incorrect initiation of combustion for a simulation with the [ECFM](#) or [ECFM3Z](#) combustion model with multiple active combusting regions and different model activation times.

Bug fix: Fixed a bug that caused CONVERGE to become unresponsive when running a combusting simulation with multiple [streams](#) when combustion is not on in all streams.

Bug fix: Fixed a bug that caused CHT to calculate incorrect results when running a simulation with the [ECFM](#) or [ECFM3Z](#) combustion model or with SAGE with the three-point PDF.

Emissions

Bug fix: Fixed a bug in the [Hiroyasu-NSC](#) soot model that caused inaccurate results when coupled with the Shell+CTC combustion model.

Sources

Chapter 2: Release Notes

3.1.4

Enhancement: CONVERGE now solves for [crevice](#) motion analytically rather than iteratively, which improves performance.

Enhancement: Altered the logic that detects cells that should have [porous media](#) source terms applied. The new method produces identical results but is more computationally efficient.

Bug fix: Fixed a bug in the [crevice](#) model that could cause CONVERGE to calculate ring motion incorrectly, causing poor convergence behavior and inaccurate results.

Bug fix: Restart and map files erroneously did not contain passives required by the [thermal runaway](#) models. These passives are now included.

CHT

Enhancement: CONVERGE now supports PERMANENT-type events for [super-cycling](#) INTERFACE boundaries.

Enhancement: Added a warning check and associated warning message to detect density change in solid streams for [CHT](#) simulations.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [super-cycling](#) and MRF.

Bug fix: Fixed a bug that caused CONVERGE to crash when parcels impacted a [super-cycling](#) INTERFACE boundary.

Bug fix: Fixed several bugs that could cause CONVERGE to crash on a simulation with [super-cycling](#).

Bug fix: Fixed a bug that could cause CONVERGE to calculate incorrect solid temperatures when running a simulation with [super-cycling](#) and a rotating solid boundary.

Bug fix: Fixed a bug in transient [super-cycling](#) that caused CONVERGE to calculate the time-step size incorrectly, resulting in numerical instability.

FSI

Enhancement: Added a new FSI mapping capability controlled by the [map.in](#) > *fsi_map* settings block. Documentation on this feature is forthcoming.

Enhancement: For [FSI beams](#), you can now specify only two of area, breadth, and height, and CONVERGE will calculate the third.

Enhancement: CONVERGE can now calculate the center of mass and moment of inertia for some non-closed [FSI](#) objects. Such objects must have their open section bounded by WALL or SYMMETRY boundaries and have motion constrained so that they do not move normal to those boundaries.

Enhancement: Refined the implementation of the 3DOF [FSI](#) solver for improved accuracy.

Bug fix: Fixed a bug in the [FSI beam](#) model that could cause CONVERGE to run out of memory and crash.

Bug fix: Fixed a bug that caused CONVERGE to crash when running a cosimulation with [Abaqus](#) on Windows.

Radiation

Enhancement: Reformulated some [FVM](#) and FTnFVM solver routines to reduce runtime.

Enhancement: Refined several radiation solver routines for improved accuracy.

Enhancement: Added a new parcel variable to [post.in](#), *radiation_energy*, for net radiation absorption.

Chapter 2: Release Notes

3.1.4

Bug fix: Fixed a bug that caused the irradiation [boundary condition](#) to be inaccurate when the refractive index was not 1.0.

Bug fix: Corrected the boundary conditions of the [P1](#) radiation model. This bug could cause inaccurate results, particularly for simulations with a high extinction coefficient.

Utilities

Enhancement: You can now optionally specify species concentration as a target for [mechanism reduction](#) and [mechanism tuning](#).

Enhancement: You can now run a NLOpt [mechanism tuning](#) simulation using [well-stirred reactor](#) extinction points as constraints.

Enhancement: Rephrased some log file output for [1D](#) counterflow simulations for increased clarity.

Enhancement: Improved the robustness of the [well-stirred reactor](#) solver. Previously, this solver would sometimes fail to converge.

Bug fix: Fixed a bug that caused CONVERGE to attempt to read a species list file from an output folder rather than from the Case Directory when running the [sub-mechanism extraction](#) tool.

Bug fix: Fixed a bug that could cause a [well-stirred reactor](#) case to initialize incorrectly for some setups.

Bug fix: For [0D](#) calculations, CONVERGE previously truncated the equivalence ratio incorrectly. This bug has been fixed.

UDF

Enhancement: Added the `CONVERGE_get_array_entry_double` and `CONVERGE_get_array_entry_int` APIs for accessing global variables stored in arrays.

Enhancement: Added the `CONVERGE_check_location_within` API.

Enhancement: Added several new example UDFs.

Enhancement: Added new FSI APIs to support additional functionality.

Bug fix: Added example UDFs that were erroneously removed from previous builds of CONVERGE 3.1.

Bug fix: Fixed memory leaks that affected several UDFs. As part of this change, the `combust_rif.c` and `film_prop.c` example UDFs have been updated to deallocate memory correctly.

Bug fix: Fixed a bug that caused the `points.c` example UDF to calculate the monitor point box volume incorrectly when the box contained paired cells. This UDF now uses the `CONVERGE_mesh_cells_intersect_with_box` API to obtain the correct box volume.

Bug fix: Fixed a bug that caused CONVERGE to crash if a UDF source was applied to the solid in a simulation with super-cycling. Now, CONVERGE ignores UDF sources during super-cycling and prints a warning to indicate that this setup is not supported.

Bug fix: Fixed a bug that could cause memory access errors when a UDF uses property table APIs to look up property values.

Bug fix: Fixed a bug that prevented the compilation of Fortran source files with lines longer than 72 characters. You can now compile Fortran source files with lines of any length.

Bug fix: Fixed a bug that caused the `CONVERGE_species_index` API to retrieve the wrong index when a simulation had multiple species with the same name. Now, when calling this API for non-gas species, you must prepend the phase to the species name (e.g., specify

"liquid_H2O" instead of "H2O"). This is a preliminary method to return species indices for species that exist in multiple phases. Phase-specific APIs will be introduced in a future minor release.

2.7 3.1.3

CONVERGE 3.1.3 is a minor beta release that includes enhancements and bug fixes.

Solver

Enhancement: Altered the implementation of the [steady-state monitor](#) when monitoring net mass flow rate. The new formulation ensures that the net mass flow rate is driven below the specified tolerance. The previous logic checked that the net mass flow rate was unchanging, but it did not check that it was nearly zero.

Enhancement: You can now run simulations with [fixed flow](#) and [erosion modeling](#). The new post variable *parcel_source* allows for visualization of the drag term that is applied as a parcel source during each fixed flow stage.

Enhancement: Added a Jacobi pressure solver for simulations that solve the pressure equation on [GPUs](#).

Enhancement: Sped up the calculation of the [species transport equation](#).

Enhancement: Added a [Reduced SOR](#) pressure solver for simulations that solve the pressure equation on [GPUs](#).

Enhancement: Refined the formulation of [MUSCL](#) slope limiters to improve convergence behavior when solving the turbulence transport equations.

Enhancement: Sped up the calculation of [anisotropic conductivity](#) source terms.

Enhancement: Refined the logic used to calculate [dt_move](#). CONVERGE now runs stably with larger values of *mult_dt_move*. Refer to Chapter 5 for recommended settings.

Bug fix: Fixed a bug in calculating [dt_move](#) that could cause CONVERGE to crash in a simulation with boundaries that have very high acceleration.

Bug fix: Fixed a bug that caused CONVERGE to crash when combining [anisotropic conductivity](#) with the [Nonlinear Krylov Acceleration Method](#).

Bug fix: Fixed a bug in how CONVERGE [recovers](#) after calculating negative pressure. This could cause inaccurate results or poor convergence behavior.

Bug fix: Fixed a bug in the [total energy solver](#) for cases with *inputs.in > feature_control > mrf_flag = 2*. This could cause CONVERGE to calculate non-physically high temperatures in the domain.

Bug fix: Fixed a bug that prevented CONVERGE from reading [monitor_steady_state.in](#) after a [restart](#).

Bug fix: Fixed a bug that prevented the [steady-state monitor](#) from functioning properly on [fixed flow](#) simulations.

Bug fix: Fixed a bug that caused inaccurate results on some simulations using the $\bar{v}^2 - f$ or $\zeta - f$ [turbulence model](#). This bug also caused wall distance to be calculated incorrectly for $k-\omega$ SST and DES turbulence models.

Bug fix: Fixed a bug that could cause CONVERGE to not transport [passives](#) correctly on simulations with connected [streams](#). This could cause CONVERGE to crash.

Bug fix: Fixed a bug in how CONVERGE calculated system error after solving a [transport equation](#). This could cause solver convergence to be reported incorrectly.

Chapter 2: Release Notes

3.1.3

Bug fix: Fixed a bug that could cause CONVERGE to calculate inaccurate results on a simulation with flow-through INTERFACE boundaries and/or an inlaid mesh when combined with MRF and/or [Crank-Nicolson](#) time advancement.

Bug fix: Fixed a bug in how CONVERGE solves the [momentum transport equation](#) in simulations with porous media. This bug could cause CONVERGE to recover frequently or crash.

Bug fix: Fixed a bug in the [MUSCL](#) implementation that could cause CONVERGE to calculate inaccurate results in MRF simulations.

Input

Enhancement: Added an error check to prevent CONVERGE from attempting to run when a case setup specifies the [AMG preconditioner](#) in multi-stream cases. This combination of features is not supported.

Enhancement: Added an error check to prevent CONVERGE from attempting to run when a case setup specifies boundaries [linked](#) to FSI objects. Boundary linkage to FSI objects is not supported.

Enhancement: Added an error check to detect invalid stream IDs in [monitor_points.in](#).

Enhancement: Added an error check to detect if a [valve](#) has a lift profile generated from a boundary, but the specified ID does not correspond to the proper valve face boundary.

Bug fix: Changing an input parameter from a number to a file name, a file name to a number, or a file name to another file name is not supported. Previously, if such an input parameter change were made during a simulation, CONVERGE could crash or incorrectly report that the parameter was successfully reread. Now, CONVERGE will print a warning to the [log file](#) and proceed with the original parameter.

Bug fix: Fixed a bug that was preventing [custom time units](#) from being stream-based.

Bug fix: Fixed a bug that could cause CONVERGE to crash or become unresponsive when running with [check_inputs](#).

Bug fix: Fixed a bug in how CONVERGE calculated the critical temperature of a [composite liquid](#).

Output/Post-Processing

Enhancement: Added an in situ post-processing feature. This allows CONVERGE to couple with [ParaView Catalyst](#) to automatically generate some types of post-processed data and images while CONVERGE is running a simulation.

Enhancement: Clarified the language of a region-related [error message](#) to indicate the IDs of the regions that triggered the error.

Enhancement: Removed a [warning message](#) that was intended only for internal development usage.

Enhancement: When writing temperature extrapolation warnings to the [log file](#), CONVERGE now writes the region ID rather than the internal region index.

Enhancement: For 0D and 1D simulations, you can now optionally direct CONVERGE to write the end-of-simulation species mass fractions or mole fractions to new output files, [zero_d_end_time_species.out](#) and [one_d_end_distance_species.out](#).

Enhancement: Previously, [monitor point](#) IDs were unique within each stream, but they were not necessarily unique within an entire simulation. Monitor point IDs are now unique within a simulation.

Chapter 2: Release Notes

3.1.3

Enhancement: CONVERGE now writes the energy residual for solid streams to [*residuals.out*](#). Previously, solid stream residuals were not written.

Bug fix: Fixed a bug that could cause CONVERGE to calculate incorrect flow quantities for [*mass avg regions flow.out*](#) when mass flow at a region interface is close to zero.

Bug fix: Fixed a bug that caused CONVERGE to write an empty [*log file*](#) for simulations with a single stream.

Bug fix: Fixed a bug that caused residuals to be written incorrectly to [*residuals.out*](#) for multi-stream simulations.

Bug fix: Fixed a bug that could cause CONVERGE to delay writing [*transfer.out*](#) until all streams had run to completion.

Bug fix: Fixed a bug that prevented CONVERGE from reading a [*map_bound<ID> <TIME>.out*](#) file as a boundary condition profile.

Bug fix: Fixed a bug that caused CONVERGE to write [*wall stress files*](#) with the wrong file names.

Bug fix: Fixed a bug that could cause CONVERGE to crash during [*mapping*](#) when running a simulation with OpenMPI.

Bug fix: Corrected file writing paths written in a [*log file*](#) message.

Bug fix: Fixed a bug that caused CONVERGE to write some [*log file*](#) statements repeatedly.

Bug fix: Fixed a bug that could cause CONVERGE to not output [*vof spray.out*](#) correctly for flow-through INTERFACE boundaries.

Boundaries and Boundary Conditions

Enhancement: Added a new [*temperature wall function*](#) that accounts for wall roughness effects.

Bug fix: Fixed a bug that could cause CONVERGE to crash or become unresponsive on a simulation with [*RELOFT*](#) boundaries and shared memory.

Bug fix: Fixed a bug that caused the [*pressure boundary condition*](#) to be set incorrectly on WALL boundaries of solid streams when using the [*SIMPLE*](#) algorithm.

Regions and Streams

Bug fix: Fixed a bug in [*disconnect triangles*](#) that could cause CONVERGE to crash.

Grid Control

Enhancement: Altered some [*inlaid mesh*](#) data structures to speed up simulations that include inlaid meshes.

Bug fix: Fixed a bug that could cause CONVERGE to crash when creating an [*inlaid mesh*](#).

Bug fix: Fixed a bug that could prevent CONVERGE from releasing [*embedding*](#) on cells on a moving boundary.

Bug fix: Fixed a bug that could prevent CONVERGE from correctly identifying cells for proximity [*AMR*](#) when running a simulation in parallel.

Bug fix: Fixed a bug that could cause CONVERGE to crash when calculating [*mesh quality*](#) metrics.

Bug fix: Fixed a bug in the calculation of moving volumes that could perturb results.

Bug fix: Fixed a bug that caused CONVERGE to calculate incorrect results when an [*inlaid mesh*](#) was attached to a flow-through INTERFACE boundary.

Bug fix: Fixed a minor MPI memory leak in some grid operations.

Chapter 2: Release Notes

3.1.3

Initialization

Bug fix: Fixed a bug that could cause CONVERGE to assign the wrong [region ID](#) on simulations with AMR or fixed embedding.

Bug fix: Fixed a bug that could cause CONVERGE to crash when loading a [restart file](#) and calculating inter-region flow.

Turbulence

Bug fix: Added a missing stress work term to turbulence models that transport tke or eps. This makes the models more consistent in total energy. Affected models are the [Reynolds Stress Models](#) and the [one-equation LES](#) models.

Discrete Phase

Enhancement: [Inlaid mesh](#) events now support [solid parcels](#). Previously, solid parcels could not be combined with inlaid mesh events.

Enhancement: [Parcel consolidation](#) is now supported for liquid spray parcels. Previously, parcel consolidation was supported only for liquid film parcels.

Bug fix: Fixed a bug that could cause CONVERGE to become unresponsive when moving parcels across an [inlaid](#)-Cartesian interface.

Bug fix: Fixed a bug that could cause CONVERGE to crash on simulations with [wall films](#) and events.

Bug fix: Fixed a bug that caused CONVERGE to incorrectly calculate [solid parcel](#) velocity on boundaries.

Bug fix: Fixed a bug that could cause CONVERGE to crash when calculating parcel evaporation on a simulation with [ELSA](#).

Bug fix: Fixed a bug in [parcel collision](#) detection routines that could cause CONVERGE to generate incorrect results or crash.

Bug fix: Fixed a bug in the [discretized parcel temperature model](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug that could cause CONVERGE to crash when running an [ELSA](#) simulation with *parcel_introduction.in > injections > injection > nozzle_control > coord_sys = POLAR* or *POLAR-COPY*.

Bug fix: Fixed a memory leak in [ELSA](#).

Bug fix: Fixed a bug that could cause CONVERGE to become unresponsive when running a parallel simulation with parcels.

Bug fix: Fixed a bug that prevented CONVERGE from properly outputting the [post variable](#) *elsa_vfrac_liq_<species>*.

Bug fix: Fixed a bug that caused the film temperature to not reset after a recovery when running with the [discretized film temperature model](#).

VOF

Enhancement: A [wave relaxation zone](#) feature is now available for both [void-fraction-based](#) and [species-solution-based](#) VOF.

Enhancement: Added an error check to prevent CONVERGE from running a simulation with PERIODIC boundaries and VOF with [PLIC](#). This combination of features is not currently supported. This error check will be removed in a future minor release.

Enhancement: Added an error check to prevent CONVERGE from solving the turbulence transport equations with the SIMPLE algorithm in a simulation with [void-fraction-based VOF](#). This combination of features is not supported.

Enhancement: Refactored some VOF [flux-corrected transport](#) operations to guarantee total and species mass conservation.

Bug fix: Fixed several bugs in the VOF [mixture model](#) that reduced accuracy and solver stability.

Bug fix: Fixed a bug in VOF [flux-corrected transport](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug in [void-fraction-based](#) VOF that could cause CONVERGE to crash.

Bug fix: Fixed a bug in how CONVERGE calculates moving volumes in VOF. This bug caused a loss of accuracy.

Bug fix: The *cav_rate post.in* variable was not initialized correctly, which could cause CONVERGE to write incorrect values of this variable to *post*.h5*. This initialization bug has been fixed.

Combustion

Enhancement: Improved the performance and numerical robustness of the [0D chemistry solver](#).

Enhancement: You can now optionally specify species concentration as a target for [mechanism reduction](#) and [mechanism tuning](#).

Enhancement: Added new ECFM post-processing variables to [post.in](#).

Enhancement: Improved the performance and numerical robustness of the [1D counterflow](#) flame chemistry solver.

Enhancement: Optimized some combustion calculations to reduce runtime.

Enhancement: Previously, CONVERGE would exit with an error when running a case setup that used the [Eddy Dissipation Model](#) and included only a single reaction. CONVERGE now prints a warning to the log file and then proceeds with the simulation.

Bug fix: Fixed a bug that could cause CONVERGE to crash when restarting a simulation that uses the [ECFM](#) or [ECFM3Z](#) combustion model.

Bug fix: Fixed a bug with the [G-Equation](#) combustion model that could cause the turbulent flamespeed to be calculated incorrectly when *combust.in > st_model > active = 11*.

Bug fix: Previously, for [1D simulations](#), CONVERGE was writing a *stream_settings.in* file that contained no information useful to the user. CONVERGE requires the existence of this file at runtime, but it does not contain any information that applies to 1D simulations. CONVERGE now removes this file after reading it.

Bug fix: Fixed a bug that could cause CONVERGE to calculate the mixture fraction output inaccurately when *combust.in > mix_frac_output > mix_frac > active = 1*.

Bug fix: Fixed a bug that was preventing combusting [multi-stream](#) simulations from correctly limiting the time-steps.

Bug fix: Fixed a bug that caused G-prime to be sourced incorrectly on simulations that use the [G-Equation](#) combustion model.

Bug fix: Fixed a bug that was preventing some reaction balance error checks from working correctly.

Bug fix: Fixed a bug that could cause the [Pathway Flux Analysis](#) tool to crash.

Bug fix: Fixed a bug that could cause CONVERGE to leak memory on some combustion cases.

Chapter 2: Release Notes

3.1.3

Bug fix: Fixed a bug that was preventing the [RIF](#), [ECFM](#), and [SAGE PDF](#) combustion solvers from returning chemistry time-step limiters to the main CONVERGE solver for reporting in the log file.

Bug fix: Removed some redundant outputs from [ecfm.out](#) and [ecfm3z.out](#).

Bug fix: Fixed a typo in [well-stirred reactor](#) output file headers.

Bug fix: Fixed a bug that caused CONVERGE to incorrectly write [TFM](#) dynamic wrinkling model variables to *post*.h5*.

Bug fix: Fixed a bug that caused CONVERGE to output soot incorrectly for [well-stirred reactor](#) simulations.

Bug fix: Fixed the logic of several error checks for NLOpt [mechanism tuning](#).

Emissions

Enhancement: Variables specified in *post.in > cells > soot* that were previously available only for simulations with the [PM](#) or [PSM](#) soot model are now also available when using the [SSM](#) soot model.

Bug fix: Fixed a memory leak in the [SSM](#) soot model.

Sources

Enhancement: When directly specifying resistivity coefficients for [porous media](#), you can now optionally specify either of ALPHA or BETA to be negative. Previously, both coefficients had to be positive.

Bug fix: Fixed a bug that could cause CONVERGE to activate or deactivate a [source](#) at the wrong time after restarting.

Bug fix: Fixed a bug in [species sourcing](#) that could cause CONVERGE to calculate incorrectly high temperatures.

CHT

Enhancement: Improved [CHT](#) matching to reduce the number of neighbors across an interface. This improvement only affects interfaces of complex, moving shapes.

Enhancement: Added error checking for case setups with boundary motion and [super-cycling](#). Previously, CONVERGE crashed when attempting to run some case setups that appeared to be valid.

Bug fix: Previously, CONVERGE wrote [supercycle_point<number>.out](#) for all streams. It now writes this output file only for the stream that contains the points specified in *supercycle.in > supercycle_points*.

FSI

Enhancement: You can now use Moreau's midpoint time-stepping (MMT) scheme to solve the motion of [FSI beam](#) objects. We recommend using the MMT solver for simulations with an active contact model. When the MMT solver is in use, a resting contact model is available to prevent beam motion in a direction opposite to the direction of deflection.

Enhancement: FSI objects can now be associated with multiple [streams](#). Solid streams can surround fluid streams. FSI objects can be defined by a fluid-fluid interface. FSI output is written for all associated streams.

Enhancement: Added an error check to detect an FSI boundary with zero triangles.

CONVERGE will exit with an error if this is detected. Previously, this caused CONVERGE to crash.

Chapter 2: Release Notes

3.1.3

Enhancement: Added error checking to detect invalid FSI case setups.

Bug fix: Refined some checks in FSI to eliminate incorrect errors and warnings.

Bug fix: Fixed a bug that caused CONVERGE to crash when running an FSI simulation with *inputs.in > solver_control > species_solver = 1*.

Bug fix: Fixed a bug that could cause cosimulations with CONVERGE and [Abaqus](#) to have incorrect event timing.

Radiation

Enhancement: You can now [specify](#) spectrally-varying boundary emissivity and transmissivity.

Enhancement: Added two-way energy coupling between [solid parcels](#) and [radiative heat transfer](#).

Enhancement: Sped up some radiation modeling calculations.

Bug fix: Fixed a bug that caused CONVERGE to exit with an error when running a simulation with the [FTnFVM](#) radiation model when *num_phi* and *num_phi_pix* were not specified in *radiation.in*. The error check was removed because that model does not require these parameters.

UDF

Enhancement: Added new APIs for the ECFM3Z combustion model.

Enhancement: Added the *CONVERGE_mesh_cell_centroid* API.

Enhancement: Added the *CONVERGE_get_cells_wall_distance* API.

Enhancement: For engine cases, CONVERGE now calculates the piston work when *inputs.in > ga_control > ga_flag = 1* or *inputs.in > feature_control > udf_flag = 1*. Previously, CONVERGE calculated the piston work only when *ga_flag = 1*.

Bug fix: Fixed a bug that could cause combustion cases to crash when UDFs were activated.

Bug fix: Fixed a bug that could cause CONVERGE to crash when the *CONVERGE_get_int* API was used in a UDF.

Bug fix: Fixed a bug in the *film_velocity.c* example UDF that could cause CONVERGE to crash.

Bug fix: Fixed a bug that caused the *film_splash.c* example UDF to generate results that were inconsistent with film splash models in CONVERGE.

Bug fix: Fixed a bug that could cause CONVERGE to crash when the *CONVERGE_simulation_dt* API was used in a UDF.

Bug fix: Fixed several bugs in the registration of boundary variables that can be accessed by UDFs.

Bug fix: Fixed a bug in which the porosity was not reset after each time-step when using the *porosity_coefficients* UDF.

2.8 3.1.2

CONVERGE 3.1.2 is a minor beta release that includes enhancements and bug fixes.

Solver

Enhancement: Refactored the [electric potential solver](#) so that it takes advantage of [stream transport groups](#).

Chapter 2: Release Notes

3.1.2

Enhancement: When `inputs.in > property_control > species_diffusion_model = 0`, `solver.in > Numerical_Schemes > flux_scheme = FLUX_BLENDING`, and `solver.in > Transport > species > solver_type = SOR`, CONVERGE [solves all species simultaneously](#) to reduce the runtime for species transport.

Enhancement: Previously, when solving with the [SOR linear solver](#), CONVERGE calculated the residual exactly for every iteration. When solving the pressure, energy, and species equations, CONVERGE now estimates the residual for up to nine consecutive iterations for the pressure equation and up to three consecutive iterations for the energy and species equations. This improves solver performance.

Enhancement: Refactored [stream transport groups](#) so that transient streams are placed in different transport groups if they are separated by one or more steady-state streams. This improves solver performance.

Enhancement: Added an error check to prevent the inclusion of [scalars](#) in `species.in` when running a simulation with [void-fraction-based VOF](#). This combination of features was not supported.

Bug fix: Fixed a bug that could cause CONVERGE to crash on simulations with [super-cycling](#).

Bug fix: Fixed a bug that could cause CONVERGE to crash when running a simulation with incompressible liquids and the [CONVERGE_BICGSTAB](#) linear solver.

Bug fix: In case setups that do not contain a WALL boundary, CONVERGE previously called routines to calculate wall distance. In such case setups, CONVERGE no longer makes these unnecessary function calls.

Bug fix: Previously, on PERIODIC neighbor cells, the [anisotropic heat conduction](#) multiplier matrix was not rotated correctly. This bug has been fixed.

Bug fix: Fixed several solver memory leaks.

Bug fix: Fixed a bug in [stream transport groups](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug that could prevent the [steady-state monitor](#) from correctly ending a simulation when different streams used different time-steps.

Bug fix: Fixed a bug that could cause CONVERGE to incorrectly report the percent complete to the screen output for simulations with multiple [streams](#).

Bug fix: Fixed several bugs in [steady/transient](#) coupling that caused CONVERGE to not terminate properly and caused output files to not be written correctly.

Bug fix: Fixed a bug that caused CONVERGE to crash on a [fixed flow](#) simulation when using the CONVERGE_BICGSTAB or HYPRE_BICGSTAB linear solver to solve the energy equation.

Bug fix: Fixed a bug that caused CONVERGE to perform some [dt_move](#) calculations when they were not needed, slowing down the simulation.

Bug fix: Fixed a bug in how CONVERGE calculated [dt_move](#). This bug could cause CONVERGE to crash.

Input

Enhancement: You can now provide different [erosion.in](#) setups for different streams.

Enhancement: Refactored input file reading routines to provide more consistent error and warning behavior.

Enhancement: If CONVERGE detects an invalid [hidden.in](#) file, it will exit with an error.

Previously, CONVERGE would print a warning to the log file and screen output.

Enhancement: Rewrote input file read errors to be more verbose.

Enhancement: Disabled the re-reading of [amr.in](#). This reduces the time CONVERGE requires to perform AMR.

Enhancement: Improved error handling when reading [surface.dat](#).

Output/Post-Processing

Enhancement: Added new source output and emobility variables as optional parameters in [post.in](#).

Enhancement: Some passive variables are now available for output from solid streams to [passive.out](#). Previously, *passive.out* was not written for solid streams.

Enhancement: Fluid property generation is now [stream](#)-based.

Enhancement: Added *stream_id* as a [post](#) variable.

Enhancement: Added the CONVERGE version number to [map files](#).

Enhancement: Added momentum solver timing to the [log file](#) and [screen output](#).

Enhancement: Clarified the language of some HDF5 errors.

Enhancement: On a simulation with multiple [streams](#), CONVERGE will stop writing stream-averaged output files for streams that have run to completion.

Bug fix: Fixed a bug that could cause CONVERGE to write multiple partial stream-averaged output files of each type rather than a single file of each type.

Bug fix: Fixed a bug in the [PSM detailed soot model](#) that could cause CONVERGE to output the wrong soot size on Windows systems.

Bug fix: Fixed a bug that could cause CONVERGE to write bad data to the last line of stream-averaged output files.

Bug fix: Fixed a bug in [mixing.out](#) that caused the overall equivalence ratio to be incorrect.

Bug fix: Fixed a bug in how heat release rate was written to [thermo.out](#).

Boundaries and Boundary Conditions

Enhancement: Added AUTO and BOUND_ID options for specifying the direction of [valve motion](#).

Enhancement: Added a total pressure [OUTFLOW](#) boundary condition.

Enhancement: You can now set up a [mixing plane](#) in cylindrical coordinates.

Enhancement: Added a MASS_AVG averaging option for [mixing planes](#).

Bug fix: Fixed a bug that caused CONVERGE to deactivate [GT-SUITE coupling](#) if [boundary.in](#) > *boundary_conditions* > *boundary* > *ID* and *boundary* > *gt_id* did not match.

Bug fix: Fixed a bug that could cause CONVERGE to crash when pairing cells on [PERIODIC](#) boundaries.

Grid Control

Enhancement: Refactored some interface matching to increase simulation speed and reduce memory usage.

Enhancement: Added an [inlaid mesh](#) cell pairing feature.

Enhancement: Refactored some moving [inlaid mesh](#) routines to increase simulation speed.

Bug fix: Fixed a bug that could cause CONVERGE to become unresponsive on a simulation with [inlaid meshing](#) and spray.

Bug fix: Fixed a bug that could cause CONVERGE to not conserve parcel mass on a simulation with [inlaid meshing](#).

Bug fix: Fixed a memory leak in interface matching.

Chapter 2: Release Notes

3.1.2

Bug fix: Fixed a bug in moving [inlaid meshes](#) that could result in less effective load balancing.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with moving [inlaid meshes](#) and a boundary with DEPENDENT motion.

Bug fix: Fixed a bug that prevented CONVERGE from performing [grid scaling](#) on multi-stream simulations.

Bug fix: Fixed a bug that prevented CONVERGE from restarting a simulation with [grid scaling](#).

Bug fix: Fixed a bug that could cause [AMR](#) cells to be inappropriately released and then re-embedded.

Bug fix: Fixed a bug that could cause CONVERGE to not conserve mass on a parallel simulation with moving [inlaid meshes](#).

Bug fix: Fixed a memory leak in cell volume calculations.

Bug fix: Fixed a bug that could cause CONVERGE to crash during [mesh quality](#) metric calculations.

Bug fix: Fixed a bug that caused CONVERGE to perform location calculations when they were not needed, slowing down the simulation.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with multiple [inlaid meshes](#) in a stream.

Bug fix: Fixed a bug that could cause [AMR](#) routines to prevent a steady-state simulation from terminating promptly if no AMR was specified in the Case Setup.

Discrete Phase

Enhancement: The [Ranz-Marshall](#) heat conduction correlation is now available for solid-phase parcels. Previously, this model was available only for liquid parcels.

Enhancement: Improved the logic used to detect parcel collisions with moving walls. The previous logic could produce erroneous results when parcels and parcel walls had similar velocities.

Enhancement: Renamed the Karimi erosion model as the [Generalized E/CRC model](#) to be consistent with the literature.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation that used [ELSA](#).

Bug fix: Fixed a bug that caused CONVERGE to calculate gradients in [wall films](#) incorrectly when the film spanned a flow-through INTERFACE boundary.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [wall films](#) and sealing.

Bug fix: Fixed a bug in [bidirectional VOF-DPM-VOF](#) modeling that could cause local velocities to be calculated incorrectly when parcels transitioned to VOF.

Bug fix: Fixed a bug that could cause parcels to not be conserved during an [inlaid mesh](#) event.

Bug fix: Fixed a bug that could cause CONVERGE to crash when restarting a multi-stream simulation with parcels.

Bug fix: Fixed a bug that could cause *tot_inj_mass* in [liquid_parcel.out](#) to be written incorrectly for VOF-spray one-way coupling simulations.

Bug fix: Fixed a bug that could cause CONVERGE to crash when calculating liquid parcel [evaporation](#).

Chapter 2: Release Notes

3.1.2

Bug fix: Fixed a bug that could cause CONVERGE to calculate the velocity of [solid parcels](#) incorrectly.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a multi-stream simulation with [wall films](#).

Bug fix: Fixed a bug that caused excessive [parcel breakup](#) within porous media.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [VOF-spray one-way coupling](#).

VOF

Enhancement: Added an error check for case setups that include both the [PLIC](#) method and PERIODIC boundaries.

Bug fix: Fixed a bug that caused CONVERGE to produce incorrect results on a simulation with [PLIC](#) and flow-through INTERFACE boundaries.

Bug fix: Fixed a bug in [individual species solution method VOF](#) that could cause mass and energy to not be conserved.

Bug fix: Fixed a bug that could cause CONVERGE to incorrectly determine that a nozzle is outside the computational domain when running a simulation with [ELSA](#), causing the solver to exit with an error.

Bug fix: Fixed a bug in [ELSA](#) that could cause incorrect spray plume penetration.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [VOF](#) and a solid stream.

Combustion

Enhancement: Added soot modeling, surface chemistry, and temperature profile features to [plug flow reactors](#). Documentation on this feature is forthcoming.

Enhancement: Added an internal passive, *tfn_eoverf*, that CONVERGE can use for [adaptive zoning](#) binning.

Enhancement: Refactored some combustion calculation routines to be [stream-based](#).

Enhancement: Added sensitivity direction control to [mechanism tuning](#).

Bug fix: Fixed a bug that caused CONVERGE to handle [CVODE errors](#) incorrectly when *combust.in > solve_temp_flag = 0*.

Bug fix: Fixed a bug that caused CONVERGE to not auto-generate [stream_settings.in](#) when running a 1D premixed combustion case, causing CONVERGE to crash.

Bug fix: Fixed a bug in [1D premixed combustion](#) that caused CONVERGE to crash when running PISO with a species that had a comma in the name.

Bug fix: Fixed a bug in [FGM](#) auto-ignition cases that could cause incorrect results.

Bug fix: Fixed a bug in [adaptive zoning](#) that caused CONVERGE to run slowly.

Bug fix: Fixed a memory leak in combustion.

Bug fix: Fixed a bug in the [Annand heat transfer correlation](#) for 0D chemistry.

Bug fix: Fixed a bug in [ECFM](#) and [ECFM3Z](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug in the [0D solver](#) that wrote an incorrect message to the log file.

Bug fix: Fixed a bug in [isomer lumping](#) that could cause CONVERGE to crash.

Bug fix: Fixed a bug in [surface chemistry](#) that could cause mass to not be conserved.

Bug fix: Fixed a bug that could cause CONVERGE to crash on a simulation with [RIF](#) and liquid spray.

Bug fix: Fixed a bug that caused CONVERGE to write incorrect activation energies to [mech_check.out](#).

Chapter 2: Release Notes

3.1.2

Bug fix: Fixed a bug that caused CONVERGE to crash when running SAGE with the [three-point PDF method](#) and transporting temperature variance.

Emissions

Bug fix: Fixed a bug that caused CONVERGE to crash on a 0D case with [detailed soot modeling](#).

Sources

Bug fix: Fixed a bug that caused CONVERGE to not recognize the keyword ALL for specifying the region of a source. This bug caused CONVERGE to exit with an error.

Bug fix: Restart and map files erroneously did not contain passives required by the [thermal runaway models](#). These passives are now included.

CHT

Enhancement: Refactored some cell location routines for greatly improved performance on simulations with an inlaid mesh and [CHT1D](#).

Enhancement: Removed some unnecessary function calls to improve performance on simulations with [CHT](#).

Bug fix: Fixed a bug that caused CONVERGE to write incorrect information to [*cht1d_point<ID>.bound<ID>.out*](#).

Bug fix: Fixed a bug that caused CONVERGE to write [*cht1d_point<ID>.bound<ID>.out*](#) to the wrong location.

Bug fix: Fixed a bug that could cause CONVERGE to not read solid species correctly on a simulation with [CHT1D](#). This bug caused CONVERGE to exit with an error.

FSI

Bug fix: Fixed a bug that caused simulations with [FSI](#) and prescribed motion to incorrectly reset that motion after a restart.

Bug fix: Fixed a bug that caused CONVERGE to incorrectly calculate [FSI beam](#) initial deflection and beam length.

Radiation

Enhancement: Added an error check to prevent the specification of *RADIATION*, *RADIATION_SRC*, or *RAD_BOUND_FLUX* as non-transport [passives](#). CONVERGE now solves these variables as internal passives.

UDF

Bug fix: Deprecated the *CONVERGE_cloud_list_destroy* and *CONVERGE_cloud_list_iterator_destroy* APIs, which are no longer necessary.

Bug fix: Fixed the heat release rate output for simulations that use a UDF to calculate an explicit reaction rate multiplier.

2.9 Major Changes from CONVERGE 3.0 to 3.1

CONVERGE 3.1.1 is a beta release. Some features may not have been fully tested, and some functionality may be missing. Updated minor CONVERGE 3.1 releases are forthcoming.

Chapter 2: Release Notes

Major Changes from CONVERGE 3.0 to 3.1

We strongly recommend setting up CONVERGE 3.1 input files in CONVERGE Studio. To convert input files from an older version of CONVERGE to CONVERGE 3.1, open CONVERGE Studio 3.1 and go to *File > Import*. Then go to *File > Export* to export the 3.1 input and data files, which can be used to run a CONVERGE 3.1 simulation.

As you get started with CONVERGE 3.1, we encourage you to consult the following documents. These documents will continue to be updated, and additional 3.1 documentation is forthcoming.

- CONVERGE 3.1 Manual (see chapter 2 for a list of major changes from CONVERGE 3.0 to 3.1)
- CONVERGE Studio 3.1 Manual (see chapter 2 for a list of major changes from CONVERGE Studio 3.0 to 3.1)
- CONVERGE 3.0/3.1 Getting Started Guide
- CONVERGE Example Case Guides

Solver

- **Steady-state residuals**: For steady-state simulations, you can view residuals for the mass, momentum, energy, and species equations in *residuals.out*. When residual output is enabled, it is important to use a fully upwinded numerical scheme to ensure that the residuals fall to acceptably low levels.
- **Maximum CFL number for steady-state simulations**: You are no longer required to specify a final maximum convection CFL number (*solver.in > Steady_state_solver > steady_max_cfl_u_final*) for steady-state simulations. We recommend using the maximum convection CFL number in *inputs.in > temporal_control > max_cfl_u* for all stages, including the final stage, of a steady-state simulation.
- **Electric potential solver**: You can use the electric potential solver to predict the current and associated heat transfer for direct current (DC) applications. CONVERGE solves the electric potential only for solid streams.
- **GPU acceleration**: CONVERGE can utilize GPU resources when solving the pressure equation. Solving other equations on GPUs is forthcoming.
- **Generalized Rhie-Chow scheme**: The generalized Rhie-Chow scheme is an extension of the original implementation of the Rhie-Chow scheme in CONVERGE. The generalized Rhie-Chow scheme may improve accuracy or performance for certain applications, such as those involving porous media, body force, or buoyancy effects.
- **SIMPLE algorithm**: We have improved the performance of the SIMPLE algorithm.
- **Boundary motion time-step limit**: The time-step limit controlled by *dt_move* is no longer required for solver stability. This parameter is retained to control simulation accuracy.

Input

- **Input directory changes**: To enable different settings for different streams, the input file structure has been modified to allow for an independent subdirectory for each stream. This change adds some complexity to the inputs, which is why we strongly recommend using CONVERGE Studio to create the inputs files.

Output/Post-Processing

Chapter 2: Release Notes

Major Changes from CONVERGE 3.0 to 3.1

- **Output file changes:** CONVERGE uses a new output directory structure when writing output files. If you use post-processing scripts to process CONVERGE output files, we recommend reviewing your scripts to determine if they need to be updated in light of these changes. Key changes include the following:
 - CONVERGE creates separate subdirectories for the original run and each restart. CONVERGE no longer appends *<restart number>* to the names of *.out files from restarted simulations.
 - Within the subdirectories mentioned above, there are additional subdirectories for each stream. Region- and stream-specific *.out files are now written to the subdirectory for the corresponding stream.
- **Post file changes:** For multi-stream cases, you can choose how CONVERGE writes post files at each time: combined data for all streams into one post file, combined data for all transient streams and combined data for all steady-state streams into two separate post files, or individual post files for each stream. Currently only the individual file option is supported, but the combined options will be available soon in a future minor release.

Boundaries and Boundary Conditions

- **Sealing:** The sealing feature has been temporarily disabled in CONVERGE 3.1.1. This feature will be re-enabled in a future release.

Regions and Streams

- **Stream settings:** A new input file called *stream_settings.in* is now required for all simulations. Regions are now assigned to streams in *stream_settings.in* instead of in *initialize.in*.
- **Stream-based inputs:** For multi-stream simulations, you can use different settings, such as different solver types or reaction mechanisms, in different streams.

Grid Control

- **Geometry calculation refactoring:** Many geometry subroutines have been rewritten to reduce the runtime and system memory required.
- **Moving inlaid meshes:** Inlaid meshes now support all motion types. Previously, inlaid meshes could not move. Inlaid meshes still are not allowed to intersect other inlaid meshes.
- **Multi-stream meshes:** The simulation can use different Cartesian meshes for different streams, allowing for more accurate simulations for some case setups.

Turbulence

- **Two-equation $k-\epsilon$ LES:** CONVERGE includes the two-equation $k-\epsilon$ LES turbulence model, which is well-suited for combustion simulations that use Adaptive Mesh Refinement to resolve the flame front.

Discrete Phase

- **Generalized parcel modeling:** CONVERGE now offers solid-phase Lagrangian parcel modeling and a variety of new options for parcel introduction and interaction with other parcels and with walls. CONVERGE also allows you to define different types of liquid and solid parcels within the same simulation.

Chapter 2: Release Notes

Major Changes from CONVERGE 3.0 to 3.1

- [Erosion modeling](#): CONVERGE now offers several erosion models to predict surface recession due to the impingement of solid parcels.
- [Boundary injection](#): Parcels can be injected into the domain from a WALL or INFLOW boundary ([parcel_introduction.in](#) > [boundary_injections](#)).

VOF

- [Multi-fluid modeling](#): You can now set up a VOF simulation with an arbitrary number of gases and liquids.
- [Mixture model](#): You can calculate stratification with a mixture model.
- [Flux-corrected transport](#): You can configure VOF simulations to use FCT for enhanced interface capturing.

Combustion

- [Eddy Dissipation Model](#): You can use the Eddy Dissipation Model to determine the rate of combustion by the rate of fuel and oxygen eddies mixing (*i.e.*, eddy dissipation).
- [FGM table file](#): The Flamelet Generated Manifold (FGM) lookup table is in HDF5 format rather than the previous *.dat format. Tables that were generated prior to CONVERGE 3.1 will not be recognized. You must regenerate FGM tables using CONVERGE 3.1. The new file format contains not only the FGM data, but also the mechanism and input information necessary to recreate the table.
- [Turbulent flame stretch](#): Five new turbulent flame stretch models with Extended Coherent Flamelet Model. Previously, only two were available.
- [Initial dilution](#): For a laminar flamespeed correlation, you can specify an initial dilution amount that varies by region. Previously, you could specify only a single initial dilution.

Emissions

- [Emissions post-processing](#): You can use emissions post-processing to run a 3D simulation based on a previous run that was generated with a simple combustion model or reduced mechanism. This can be a computationally efficient way of using a detailed chemical mechanism for achieving accurate emissions predictions.
- [Sectional Soot Model](#): You can obtain detailed emission information via the Sectional Soot Model.

Sources

- [Local thermal non-equilibrium \(LTNE\) in porous media](#): The LTNE model of energy transport in porous media allows the local temperatures to be different in the fluid and solid phases. CONVERGE solves the fluid and solid energy transport equations separately to predict the fluid and solid temperatures in the porous region.
- [Thermal runaway modeling](#): You can model thermal runaway for energy sources.
- [Battery heat source modeling](#): You can model a battery heat source.
- [Source energy output file](#): You now write the heat release data for each source to `source_energy.out` when source modeling is active. Previously, only the total energy of all sources was available in `thermo.out`.

FSI

Chapter 2: Release Notes

Major Changes from CONVERGE 3.0 to 3.1

Implicit FSI: CONVERGE now incorporates an implicit FSI solver, which is much more numerically robust than the existing explicit solver, especially when the solid and fluid densities are similar.

Aeroacoustics

Aeroacoustics modeling: CONVERGE features the Ffowcs Williams-Hawkins far-field tonal model and the Curle and Proudman near-field broadband models. CONVERGE can write the Lighthill stress tensor to the *post*.h5* output files.

Utilities

Command line changes: Some command line options for the CONVERGE executable have been deprecated and replaced. This includes the command line options for all [chemistry utilities](#). To see a complete list of command line options for CONVERGE 3.1, append the `--help` command line option to any CONVERGE executable (*e.g.*, `converge-mpich --help`).

Schmidt species output: You can now calculate Schmidt species numbers when performing a 1D laminar freely propagating flamespeed calculation with Newton's method. These values are written to *schmidt_species_case<number>.dat*.

2.10 Manual Reorganization from CONVERGE 3.0 to 3.1

The CONVERGE 3.0 Manual contained a chapter dedicated to the simulation of internal combustion engines. Much of the content contained therein is also applicable to other application areas (*e.g.*, pumps and compressors). To better facilitate the simulation of these latter applications, the internal combustion engine chapter was removed, and the content was relocated as appropriate throughout the manual and into other CONVERGE documentation and training materials. This relocation is described below.

Table 2.1: Content relocation from the Chapter 20: Application: Internal Combustion Engines in the CONVERGE 3.0 Manual.

Chapter 20 Topic	New Location
Translating Wall Boundary: Piston	Chapter 9: Translating WALL: Piston
Translating Wall Boundary: Valve	Chapter 9: Translating WALL: Valves
Velocity Initialization in IC Engines	Chapter 12: Specified Initial Values
Velocity Initialization in the Piston	Chapter 12: Velocity Initialization in the Piston
Velocity Initialization in the Cylinder	Chapter 12: Velocity Initialization in the Cylinder
Swirl, Tumble, and Angular Momentum	Chapter 25: dynamic.in
Compression Ratio Calculations	CONVERGE Studio Manual Chapter 6: Compression Ratio Calculation.

Chapter 2: Release Notes

Manual Reorganization from CONVERGE 3.0 to 3.1

Chapter 20 Topic	New Location
Finite Element Analysis	Chapter 23: HTC Mapping
Energy Sources in Engine Models	Chapter 18: Energy Sources in Engine Models
Heat Release Data	Chapter 18: Heat Release Data
Spark Energy	Chapter 18: Spark Energy
Pressure Trace Data	Chapter 18: Pressure Trace Data
Synchronizing Valve Motion	Chapter 10: Synchronizing Valve Motion with Open and Close Events
Multiple Cylinder Simulations	Refer to CONVERGE Advanced Training: Multi-Cylinder Simulations, available on the Hub.
Valve Lift Profile	Chapter 9: Valve Motion Profile
Swirl and Tumble for Inclined Cylinders – <i>dynamic.in</i>	Chapter 25: <i>dynamic.in</i> .
Region-Specific Setup Considerations	Chapter 25: Region-Specific Setup Considerations for Engine Modeling
Multiple Cycle Simulations	Refer to CONVERGE Advanced Training: Multi-Cycle Simulations, available on the Hub.
Crevice Model	Chapter 25: <i>crevice.in</i>
Sealing of Boundaries for Engine Applications	Chapter 8: Sealing of Boundaries

Chapter



3

Governing Equations

3 Governing Equations

The dynamics of fluid flow are governed by the Navier-Stokes equations, which describe the conservation of [mass](#), [momentum](#), and [energy](#). Additional equations describe [turbulence transport](#) and the transport of [species](#) as well as of [passives and scalars](#). This chapter describes the governing equations used in CONVERGE and the associated input parameters.

You can specify which equations CONVERGE will solve in the [*inputs.in*](#) > *solver_control* settings block. Each conservation equation can be solved independently or in combination with the other equations. In addition, you can specify if the fluid (liquid or gas) in the simulation is compressible or incompressible (via [*inputs.in*](#) > *property_control* > *gas_compressible_flag* and *liquid_compressible_flag*) and you can select the phase (gas, liquid, or gas-liquid [volume of fluid]). CONVERGE will solve for solids if you define one or more solid species. In addition to [*inputs.in*](#), the [*species.in*](#) and [*solver.in*](#) files contain parameters related to the governing equations.

3.1 Mass and Momentum Transport

CONVERGE solves the momentum and mass conservation equations when [*inputs.in*](#) > *solver_control* > *momentum_solver* = *ON*. It is necessary to solve the mass and momentum equations together for the proper calculation of the pressure gradient in the momentum equation. The interaction between the mass and momentum equations is discussed in more detail in the [Iterative Algorithm section in Chapter 4 - Numerics](#).

The compressible equations for mass transport and momentum transport are given by

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = S \quad (3.1)$$

and

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = - \frac{\partial P}{\partial x_i} + \frac{\partial \sigma_{ij}}{\partial x_j} + S_i, \quad (3.2)$$

where the viscous stress tensor is given by

$$\sigma_{ij} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + \left(\mu' - \frac{2}{3} \mu \right) \left(\frac{\partial u_k}{\partial x_k} \delta_{ij} \right). \quad (3.3)$$

In the above equations, u is velocity, ρ is density, S is the source term, P is pressure, μ is viscosity, μ' is the dilatational viscosity (set to zero), and δ_{ij} is the Kronecker delta. Note that

Chapter 3: Governing Equations

Mass and Momentum Transport

if a turbulence model is activated, the molecular viscosity is augmented by a turbulent viscosity term. Refer to [Chapter 13 - Turbulence Modeling](#) for details.

Equations 3.1 and 3.2 both allow for source terms. For the momentum equation, the source terms can arise from, for example, gravitational acceleration, spray coupling, or mass sources. For the mass conservation equation, the source term may arise from evaporation or other submodels.

CONVERGE solves the momentum equation in [finite volume form](#). When solving the momentum equation, conserved values at the face must be expressed in terms of primitive values at the cell center. CONVERGE offers two ways to evaluate these face-centered conserved values. The first method is to calculate the cell-centered conserved value from cell-centered primitives, then project this conserved value to the face. The second method is to project primitive values to the face, and calculate face-centered conserved values from these face-valued primitives.

You can solve the momentum equation using any weighted average of these two formulations. Equation 3.4 gives the momentum equation as a combination of calculated-at-cell-center and calculated-at-face terms:

$$\begin{aligned} \frac{\partial \rho u_i u_j}{\partial x_j} &= \frac{1}{V} \sum_{nbrs} (\rho u_i u_{norm} A)_f \\ &= \frac{cons}{V} \sum_{nbrs} (\rho u_i u_{norm})_f A_f + \frac{1-cons}{V} \sum_{nbrs} \rho_f u_{i,f} u_{norm,f} A_f, \end{aligned} \quad (3.4)$$

where the *cons* term represents the fraction of the equation using the form calculated from cell-centered primitives ([solver.in](#) > *Numerical_Schemes* > *conserve*). Solving in cell-centered-evaluation form (*conserve* = 1.0) is always recommended, although the face-valued-evaluation form may be more stable for some flows.

If you employ the *step* flux limiter with a first-order accurate spatial discretization scheme, CONVERGE may [automatically adjust](#) the *cons* term in Equation 3.4.

The [inputs.in](#) file contains parameters for the momentum equation ([inputs.in](#) > *solver_control* > *momentum_solver* and [inputs.in](#) > *property_control* > *gravity*).

Multiple Reference Frame Approach

The MRF approach in CONVERGE allows you to simplify a simulation that includes moving geometry. While CONVERGE can easily simulate moving geometries, the MRF approach further reduces computational time by eliminating the need to regenerate the mesh at each time-step to accommodate the moving geometry.

With the MRF approach, the moving geometry is instead modeled as stationary. You specify a region that encompasses this portion of the geometry and CONVERGE treats this region as

Chapter 3: Governing Equations

Mass and Momentum Transport Multiple Reference Frame Approach

a different (local) reference frame. The local reference frame moves relative to the stationary (inertial) reference frame, so the geometry does not have to move to represent the fluid behavior.

For example, consider a pump impeller. With the typical moving geometry approach, the impeller is designated as a moving boundary and CONVERGE rotates the impeller the appropriate amount each time-step and regenerates the mesh around the new geometry configuration. With the MRF approach, CONVERGE considers the region around the impeller as a local reference frame fixed in the impeller blades. In this way, the geometry does not have to move in the simulation and instead, the local reference frame moves relative to the inertial frame to represent the impeller motion.

Within this local reference frame, CONVERGE solves modified continuity, momentum, and energy equations that represent the motion of the stationary geometry. There are two formulations of these equations available for the MRF approach: relative and absolute ([*inputs.in*](#) > *feature_control* > *mrf_flag* = 1 or 2, respectively). The absolute formulation performs better for certain scenarios, such as when the MRF region encompasses the entire domain.

In the continuity equation, \mathbf{u}_r is the velocity in the local reference frame. The source term in the momentum equations includes additional body force terms to account for the velocity in the local frame.

The modified continuity and momentum equations in relative formulation are

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot [\rho \mathbf{u}_r] &= 0 \\ \frac{\partial(\rho \mathbf{u}_r)}{\partial t} + \nabla \cdot [\rho \mathbf{u}_r \mathbf{u}_r + p \bar{I} - \bar{\tau}] &= -2\rho(\boldsymbol{\Omega} \times \mathbf{u}_r) - \rho \boldsymbol{\Omega} \times (\boldsymbol{\Omega} \times \mathbf{R}) - \rho \frac{d\boldsymbol{\Omega}}{dt} \times \mathbf{R} \\ \mathbf{u} &= \mathbf{u}_r + \boldsymbol{\Omega} \times \mathbf{R}, \end{aligned} \quad (3.5)$$

while these equations in absolute formulation are

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot [\rho(\mathbf{u} - \boldsymbol{\Omega} \times \mathbf{R})] &= 0 \\ \frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot [\rho \mathbf{u}(\mathbf{u} - \boldsymbol{\Omega} \times \mathbf{R}) + p \bar{I} - \bar{\tau}] &= -\rho(\boldsymbol{\Omega} \times \mathbf{u}). \end{aligned} \quad (3.6)$$

In the equations above, $\boldsymbol{\Omega}$ is the rotation vector and \mathbf{R} is the position vector with respect to the axis about which the reference frame is rotating. The final term in the momentum equation for the relative formulation accounts for a time varying rotational speed and therefore is zero when rotational speed is constant. If you wish to supply a time varying rotational speed, in the [*mrf.in*](#) > *mrf_region* > *specification* settings block, set *method* to

Chapter 3: Governing Equations

Mass and Momentum Transport Multiple Reference Frame Approach

`COPY_FROM_BOUNDARY` and supply a boundary ID from which to copy rotation information for `boundary_id`. Configure this boundary with a varying rotational speed. CONVERGE copies the rotation information (*e.g.*, in a temporally varying profile) from this specified boundary and applies it to the local reference frame.

At the interfaces between the local and the inertial reference frames, CONVERGE transforms the cell face fluxes such that they are in the appropriate reference frame.

Use this approach for simulations in which you expect a steady-state result. You can use this approach for transient simulations (*i.e.*, `inputs.in` > `solver_control` > `steady_solver = 0`), but the results may not be time-accurate. One strategy is to use the MRF approach to calculate a developed flow field to use as an initial condition for a transient simulation with moving geometry.

To activate the multiple reference frame approach, set `inputs.in` > `feature_control` > `mrf_flag = 1` or `2` and include an `mrf.in` file in your case setup.

MRF Boundary Conditions

The relative MRF formulation allows only WALL boundary conditions within a local reference frame. The absolute formulation supports WALL and INFLOW/OUTFLOW boundaries within the local reference frame. For boundaries within a local reference frame, you must supply boundary conditions in the inertial frame.

For example, for a pump impeller within a local reference frame, the frame is fixed in the impeller. Thus, the boundary motion type for the impeller WALL boundary should be moving rotating.

The shroud (the wall surrounding the impeller blades) would appear to move around the pump impeller because the frame is fixed in the impeller. Again, no geometry actually needs to move, so specify the shroud as fixed stationary with a rotation speed of Ω . CONVERGE expects the boundary condition rotation in the inertial frame, which allows you to easily convert transient, moving geometry simulations to MRF simulations.

[Chapter 9 - Boundaries and Boundary Conditions](#) describes the boundary-related setup for the MRF approach.

Output

Note that the cell-specific output (*e.g.*, velocity, pressure, etc) for all regions (including the moving local reference frames) is in the inertial reference frame in the `post*.h5` files.

3.2 Equation of State

Momentum and mass transport can be solved for compressible or incompressible flows. For compressible flows, an equation of state is required to couple density, pressure, and temperature.

Chapter 3: Governing Equations

Equation of State

CONVERGE contains several equation of state options: ideal gas, Redlich-Kwong, Redlich-Kwong-Soave, and Peng-Robinson.

The ideal gas law is given by

$$\frac{P}{\rho} = \left(\frac{R}{MW} \right) T, \quad (3.7)$$

where R is the ideal gas constant and MW is the molecular weight.

Commonly used cubic equations of state include Redlich-Kwong (RK), Redlich-Kwong-Soave (RKS), and Peng-Robinson (PR). These equations can be written in a generalized form:

$$P = \frac{RT}{v - b} - \frac{a}{v^2 + ubv + wb^2}, \quad (3.8)$$

and the following table summarizes the coefficients for the RK, RKS, and RP equations of state.

Table 3.1: Coefficients for the RK, RKS, and PR equations of state.

Equation of state	u	w	b	a
Redlich-Kwong	1	0	$\beta_{rk} v_c$	$\alpha_{rk} \frac{p_c v_c^2}{\sqrt{T_r}}$
Redlich-Kwong-Soave	1	0	$\beta_{rk} v_c$	$\alpha_{rk} p_c v_c^2 \left[1 + f_{rk}(\omega) (1 - T_r^{1/2}) \right]^2$
Peng-Robinson	2	-1	$\beta_{pr} v_c$	$\alpha_{pr} p_c v_c^2 \left[1 + f_{pr}(\omega) (1 - T_r^{1/2}) \right]^2$

Equation 3.9 gives several additional definitions needed to understand the coefficients in Table 3.1:

$$v_c = \frac{RT_c}{p_c}, \quad (3.9)$$

$$\alpha_{rk} = 0.42748, \beta_{rk} = 0.08664; f_{rk}(\omega) = 0.48 + 1.574\omega - 0.176\omega^2,$$

$$\alpha_{pr} = 0.45724, \beta_{pr} = 0.07780; f_{pr}(\omega) = 0.37464 + 1.54226\omega - 0.26992\omega^2,$$

where v_c is the critical volume, T_c is the critical temperature, P_c is the critical pressure, α represents the attractive forces between molecules, β represents the volume of the molecules,

Chapter 3: Governing Equations

Equation of State

and ω is the acentric factor. The subscripts *rk* and *pr* represent the RK/RKS and PR equations of state, respectively.

These equations of state can be generalized to a mixture. In a multi-species simulation, the cubic equations of state can be written as

$$P = \frac{RT}{v - b} - \frac{a_m}{v^2 + ub_m v + wb_m^2}, \quad (3.10)$$

where parameters a_m and b_m are calculated for a list of i species with mole fractions X_i as

$$a_m = \sum_i \left(X_i a_i^{1/2} \right)^2, \quad b_m = \sum_i X_i b_i. \quad (3.11)$$

CONVERGE calculates pseudo-critical temperatures, pressures, and acentric factors for the mixture according to:

$$T_{cm} = \begin{cases} \left(\left[\sum_i X_i \frac{T_{ci}^{1.25}}{\sqrt{P_{ci}}} \right]^2 \right)^{2/3} & \text{for RK,} \\ \left[\sum_i X_i \frac{T_{ci}}{P_{ci}} \right] & \\ \left[\sum_i X_i \frac{T_{ci}^{1.25}}{\sqrt{P_{ci}}} \right]^2 & \text{for RKS and PR,} \\ \sum_i X_i \frac{T_{ci}}{P_{ci}} & \end{cases} \quad P_{cm} = \frac{T_{cm}}{\sum_i X_i \frac{T_{ci}}{P_{ci}}}, \quad \omega_m = \sum_i X_i \omega_i. \quad (3.12)$$

The Redlich-Kwong equation performs poorly with liquid phase, the Redlich-Kwong-Soave equation performs well for hydrocarbons, and the Peng-Robinson equation performs well for the liquid density of many materials.

The [*inputs.in* > *property_control*](#) settings block contains parameters for equation of state (*eos_flag*, *real_gas_prop_flag*, *max_reduced_pres*, *crit_temp*, *crit_pres*, and *acentric_factor*).

Note that the pressure is not calculated directly from the equation of state but instead is used indirectly in the [*iterative algorithm*](#) to ensure that the equation of state is satisfied.

3.3 Energy Transport

CONVERGE solves the energy equation when [*inputs.in* > *solver_control* > *energy_solver* = TOTAL or INTERNAL](#). You can choose to solve the energy equation for incompressible or

Chapter 3: Governing Equations

Energy Transport

compressible flows. The energy equation can be solved independently or together with the momentum and mass equations. If the energy equation is solved without momentum and mass, convection will not be included in the energy equation (*i.e.*, the energy transport equation will contain only the diffusion terms).

The compressible form of the energy equation is given by

$$\frac{\partial \rho e}{\partial t} + \frac{\partial \rho e u_j}{\partial x_j} = -P \frac{\partial u_j}{\partial x_j} + \frac{\partial}{\partial x_j} \left(K \frac{\partial T}{\partial x_j} \right) + \sigma_{ij} \frac{\partial u_i}{\partial x_j} + \frac{\partial}{\partial x_j} \left(\rho \sum_m D_m h_m \frac{\partial Y_m}{\partial x_j} \right) + S, \quad (3.13)$$

where ρ is density, Y_m is the mass fraction of species m , D_m is the species mass diffusion coefficient, P is the pressure, e is the specific internal energy, K is the conductivity, and h_m is the species specific enthalpy. Note that, if a turbulence model is activated, the conductivity is replaced by the turbulent conductivity, which is given by

$$K_t = K + c_p \frac{\mu_t}{Pr_t}, \quad (3.14)$$

where c_p is the specific heat at constant pressure, μ_t is the turbulent viscosity, and Pr_t is the turbulent Prandtl number. In CONVERGE, the formulation for the turbulent Prandtl number is

$$Pr_t = \frac{c_p \mu_t}{k_t}, \quad (3.15)$$

where k_t is the turbulent conductivity.

In addition to the convection and diffusion terms, the energy equation contains four extra terms. A pressure work term, $-P \frac{\partial u_j}{\partial x_j}$, accounts for compression and expansion. For incompressible flows this term is always 0 because the field is divergence free. A viscous dissipation term, $\sigma_{ij} \frac{\partial u_i}{\partial x_j}$, accounts for kinetic energy viscously dissipating into heat. A species diffusion term, $\frac{\partial}{\partial x_j} \left(\rho \sum_m D_m h_m \frac{\partial Y_m}{\partial x_j} \right)$, accounts for energy transport due to [species diffusion](#). Finally, a source term S is added to account for user-specified energy sources and turbulent dissipation.

Chapter 3: Governing Equations

Energy Transport

For solids (in conjugate heat transfer applications), the energy transport equation retains only the transient, diffusion, and source terms:

$$\frac{\partial \rho e}{\partial t} = \frac{\partial}{\partial x_j} \left(K \frac{\partial T}{\partial x_j} \right) + S. \quad (3.16)$$

Simulating high-speed flows that include discontinuities (e.g., shock waves) may prevent energy conservation if you choose to solve for specific internal energy (given by Equation 3.13). To enforce energy conservation in these simulations, you can set [*inputs.in* > solver_control > energy_solver = 2](#) to solve for total energy instead of specific internal energy. The form of the total energy equation used in CONVERGE is

$$\begin{aligned} \frac{\partial}{\partial t} \left(\rho \left(e + \frac{1}{2} u_k^2 \right) \right) + \frac{\partial}{\partial x_j} \left(\rho u_j \left(e + \frac{1}{2} u_k^2 \right) \right) = \\ - \frac{\partial}{\partial x_j} (u_j P) + \frac{\partial}{\partial x_j} (u_i \sigma_{ij}) + \frac{\partial}{\partial x_j} \left(K \frac{\partial T}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left(\rho \sum_m D_m h_m \frac{\partial Y_m}{\partial x_j} \right) + S. \end{aligned} \quad (3.17)$$

There are some important differences between the specific internal energy equation and the total energy equation. On the left hand side of Equation 3.17, both the transient term and the convection term include the total energy. The total energy is the sum of the internal energy e

and the kinetic energy $\frac{1}{2} u_k^2$. The right hand side of Equation 3.17 includes the pressure

work term $\frac{\partial}{\partial x_j} (u_j P)$ and the viscous dissipation term $\frac{\partial}{\partial x_j} (u_i \sigma_{ij})$ written in conservative

form. In the specific internal energy equation, the pressure work term is in non-conservative form with the pressure P outside of the derivative. For the total energy equation, the diffusion, species diffusion, and source terms are the same as those in the specific internal energy equation.

Multiple Reference Frame (MRF) Approach

With the multiple reference frame (MRF) approach, CONVERGE solves a modified energy equation to account for the relative motion of the local reference frame. Equations 3.18 and 3.20 provide the energy equation solved in the local reference frame in the relative and absolute formulations, respectively.

$$\frac{\partial E_r}{\partial t} + \nabla \cdot [\mathbf{u}_r (E_r + p) + k \nabla T - \bar{\tau} \cdot \mathbf{u}_r] = 0 \quad (3.18)$$

In the above equation, E_r is relative energy, k is conductivity, and T is temperature. The formulation for relative energy is

Chapter 3: Governing Equations

Energy Transport

$$E_r = p / (\gamma - 1) + \frac{1}{2} \rho |\mathbf{u}_r|^2 - \frac{1}{2} \rho |(\boldsymbol{\Omega} \times \mathbf{R})|^2 = E - \rho \mathbf{u}_r \cdot (\boldsymbol{\Omega} \times \mathbf{R}), \quad (3.19)$$

where E is total energy and γ is the ratio of specific heats.

$$\frac{\partial E}{\partial t} + \nabla \cdot [(\mathbf{u} - \boldsymbol{\Omega} \times \mathbf{R})(E + p) + (\boldsymbol{\Omega} \times \mathbf{R})p + k\nabla T - \bar{\tau} \cdot \mathbf{u}] = 0 \quad (3.20)$$

Calculation of Diffusion Term in CONVERGE

The diffusion term of the fluid energy equation can be written as

$$\frac{\partial}{\partial x_j} \left(K \frac{\partial T}{\partial x_j} \right) = \frac{\partial}{\partial x_j} \left(\frac{K}{c_v} \frac{\partial e}{\partial x_j} \right) - \frac{\partial}{\partial x_j} \left(\sum_m e_m \frac{K}{c_v} \frac{\partial Y_m}{\partial x_j} \right), \quad (3.21)$$

where c_v is the specific heat at constant volume. To improve efficiency, CONVERGE calculates the right side of Equation 3.21 rather than calculating the temperature gradient directly.

By default, the thermal conductivity K is spatially isotropic for both fluids and solids. To define parts of the domain in which the conductivity is anisotropic, set [*inputs.in* > feature_control > aniso_cond_flag = 1](#) and include an [*aniso_cond.in*](#) file in your case setup.

The [*inputs.in*](#) file contains parameters for the energy equation (*solver_control > energy_solver* and *property_control > prandtl_turb*).

3.4 Species Transport

CONVERGE solves the species transport equation when [*inputs.in* > solver_control > species_solver = 1](#) or 2. The species transport equation solves for the mass fraction of all the species in the domain. The species mass fraction is defined as

$$Y_m = \frac{M_m}{M_{tot}} = \frac{\rho_m}{\rho_{tot}}, \quad (3.22)$$

where M_m is the mass of species m in the cell, M_{tot} is the total mass in the cell, ρ_m is the density of species m , and ρ_{tot} is the density of the cell. The species equations can be solved alone or together with any of the other transport equations. If momentum is not solved, convection will not be considered in the species transport equation (*i.e.*, the species conservation equation will contain only diffusion terms).

The compressible form of the species conservation equation is given by

Chapter 3: Governing Equations

Species Transport

$$\frac{\partial \rho_m}{\partial t} + \frac{\partial \rho_m u_j}{\partial x_j} = \frac{\partial}{\partial x_j} \left(\rho D_m \frac{\partial Y_m}{\partial x_j} \right) + S_m, \quad (3.23)$$

where

$$\rho_m = Y_m \rho \quad (3.24)$$

and where u is velocity, ρ is density, ρ_m is the species density, Y_m is mass fraction of species m , D_m is the diffusion coefficient of species m , and S_m is a source term that accounts for evaporation, chemical reactions (combustion), and other submodels.

Several models are available for calculating the diffusion coefficient D_m . When [*inputs.in*](#) > *property_control* > *species_diffusion_model* = 0, CONVERGE calculates the diffusion coefficient independently of the species, such that

$$D_m = D = \frac{V}{Sc}, \quad (3.25)$$

where Sc is the Schmidt number. This model requires a [*gas.dat*](#) file in the Case Directory.

When [*inputs.in*](#) > *property_control* > *species_diffusion_model* = 0, [*solver.in*](#) > *Numerical_Schemes* > *flux_scheme* = *FLUX_BLENDING*, and [*solver.in*](#) > *Transport* > *species* > *solver_type* = *SOR*, CONVERGE solves an identical matrix for all transported species. In this circumstance, CONVERGE solves all species simultaneously to reduce the runtime for species transport.

If *species_diffusion_model* = 1, CONVERGE calculates a mixture-averaged diffusion coefficient for each species. This model requires a [*transport.dat*](#) file (instead of a [*gas.dat*](#) file) in the Case Directory. CONVERGE calculates the local mixture-averaged diffusion coefficient from

$$D_m = \frac{1 - X_m}{\sum_{j,j \neq m} \left(\frac{X_j}{D_{mj}} \right)}, \quad (3.25)$$

where X_m is the mole fraction of species m and D_{mj} is the binary diffusion coefficient for species m and j . The binary diffusion coefficient is a function of temperature and pressure, given by

Chapter 3: Governing Equations

Species Transport

$$D_{mj} = \frac{3}{16} \frac{\sqrt{2\pi k_B^3 T^3 / m_{mj}}}{P\pi\sigma_{mj}^2 \Omega^{(1,1)*}}, \quad (3.25)$$

where k_B is Boltzmann's constant. CONVERGE calculates the reduced molecular mass m_{mj} , the reduced collision diameter σ_{mj} , and the collision integral $\Omega^{(1,1)*}$ from the diffusion parameters in [*transport.dat*](#), following the method of [Hirschfelder et al., 1954](#). For computational efficiency, CONVERGE calculates the binary diffusion coefficients and stores them in an internal reference table before running the simulation.

If *species_diffusion_model* = 2, CONVERGE determines the diffusion coefficient for each species according to the local species-weighted Schmidt number. For this model, you must include a [*gas.dat*](#) file and a [*schmidt_species.dat*](#) file in the Case Directory. CONVERGE will calculate the individual molecular species diffusion coefficients as

$$D_m = \frac{\nu}{Sc_m}. \quad (3.25)$$

For all values of *species_diffusion_model*, if a turbulence model has been activated, the molecular mass diffusion coefficient D_m in the energy transport and species transport equations is replaced by

$$D_{tot} = D_m + D_t. \quad (3.25)$$

The turbulent mass diffusion coefficient D_t is given by

$$D_t = \frac{\nu_t}{Sc_t}, \quad (3.25)$$

where Sc_t is the turbulent Schmidt number.

You can generally set *species_diffusion_model* = 0 when the values of D_m are small compared to D_t , as is often true for high-Re RANS simulations. In other situations, the effects of species-specific diffusion may be nontrivial. In particular, we recommend setting *species_diffusion_model* = 1 or 2 for laminar flame applications.

For mass conservation, CONVERGE constrains the net diffusion flux to be zero in multi-component systems as described by [Coffee and Heimerl \(1980\)](#).

Chapter 3: Governing Equations

Species Transport

Note that, even if the species are not solved, it is still necessary to initialize the species so that their properties can be evaluated. Also note that if the species are not solved, then the species must not vary in space.

The [inputs.in](#) file contains parameters for the species equation (*solver_control > species_solver*, *property_control > schmidt_turb*, and *property_control > species_diffusion_model*).

3.5 Scalar and Passive Transport

In CONVERGE, a scalar is a calculated quantity that is collocated in space and time with mass, momentum, energy, and species. A scalar can be one of two types: an active scalar (henceforth referred to simply as a "scalar") or a passive scalar (henceforth referred to as a "passive"). The only difference between a scalar and a passive is when the quantity is solved within the [time-step](#). Passives can be non-transport passives (e.g., *TEMP_SGS*, the local sub-grid scale temperature) or transport passives (e.g., *HIROY_SOOT*).

Scalars and passives have several different uses in CONVERGE. Some passives (e.g., *G_EQN_PROGRESS* in the [G-Equation](#) combustion model) may indirectly affect the other transport equations through a parameter in a source term. Some features in CONVERGE (e.g., soot models) automatically define required passives, which are called internal passives. You can also use completely neutral passives to track fluid flow. For example, in an engine case, you can add a passive to an intake port to track the fresh intake charge as it enters the combustion cylinder. Unlike the other transport equations, the solution of the passive transport equation is not activated with a flag in [inputs.in](#). Instead, CONVERGE solves the passive transport equation only when one or more passives are defined in [species.in](#).

CONVERGE versions 2.4 and earlier did not feature predefined internal passives for the models that required them. Duplicating an internal passive name in *species.in* will cause CONVERGE to exit with an error. If you want to run a Case Setup from version 2.4 or earlier in CONVERGE 3.1, we recommend [converting that Case Setup in CONVERGE Studio](#).

You can instead define any transported passive as a scalar. Because scalars are solved within an inner loop of the [time-step](#), they are more tightly coupled in time with mass, momentum, and energy. Specifically, scalars update the working temperature and species concentration at each sub-iteration, whereas passives do not. Any model variable which could alter the temperature and/or species concentration (e.g., combustion variables such as *CMEAN*, *ZMEAN*, and *ZVAR* for the Flamelet Generated Manifold Model) should be defined as a scalar in a time-accurate simulation. If these variables are instead defined as passives, this will generate inaccuracies in the time-step update. For [steady-state simulations](#) which drive the update to zero, this error may be unimportant. If a model variable cannot alter the temperature or species concentration, the solution will not depend on whether it is defined as a passive or a scalar.

The transport of passives is computationally less expensive than the transport of scalars. Therefore, you should generally define a variable as a scalar only if it could alter the temperature or species concentration. For further guidance on when to use a scalar instead of a passive, consult the CONVERGE example cases.

Chapter 3: Governing Equations

Scalar and Passive Transport

Because passives and scalars are handled identically except for the time-step, throughout this manual, typically only passives are described. Scalar behavior is identical unless otherwise noted.

The compressible form of the passive transport equation is given by

$$\frac{\partial \rho \phi}{\partial t} + \frac{\partial \rho u_i \phi}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D \frac{\partial \phi}{\partial x_i} \right) + S, \quad (3.26)$$

where u is velocity, ρ is density, D is the diffusion coefficient, S is the source term, and ϕ is the transported passive. The diffusion coefficient is given by

$$D = \frac{V}{Sc}, \quad (3.27)$$

where Sc is the Schmidt number, which can be defined for each passive individually in [*species.in*](#). If a turbulence model has been activated, the turbulent mass diffusion coefficient is given by

$$D_t = \frac{V_t}{Sc_t}, \quad (3.28)$$

where Sc_t is the turbulent Schmidt number. The source term in the passive transport equation accounts for any applicable submodels.

The [*species.in*](#) file contains parameters for the passive equation. Note that there is no [*inputs.in*](#) parameter for the solution of the passive conservation equation. CONVERGE will solve the passive conservation equation if you define passives in [*species.in*](#).

3.6 Turbulence Transport

CONVERGE solves the turbulence transport equation when [*inputs.in* > solver_control > turbulence_solver](#) = 1. CONVERGE contains several types of turbulence models, including Reynolds-Averaged Navier-Stokes, large eddy simulation, and detached eddy simulation, and you specify the desired model via [*turbulence.in* > turbulence_model](#). [Chapter 13-Turbulence Modeling](#) describes the turbulence transport equations and how to set up turbulence modeling in CONVERGE.

3.7 Electric Potential

The electric potential solver can be used to predict the current and associated heat transfer for direct current (DC) applications. When [*inputs.in*](#) > *feature_control* > *electric_potential_flag* = 1, CONVERGE solves the electric potential Φ given by

$$\frac{\partial}{\partial x_i} \left(\sigma \frac{\partial \Phi}{\partial x_i} \right) = 0, \quad (3.29)$$

where σ is the electrical conductivity. CONVERGE solves Equation 3.29 only for solid streams. The electric field E_i is given by

$$E_i = -\frac{\partial \Phi}{\partial x_i}, \quad (3.30)$$

and the current density J_i is given by Ohm's law:

$$J_i = \sigma E_i. \quad (3.31)$$

CONVERGE accounts for ohmic heat dissipation by adding a source term equal to $\sigma E_i E_i$ to the [solid energy equation](#).

When using the electric potential solver, you must supply electrical conductivity data (σ) in the fifth column of [*solid.dat*](#). By default, CONVERGE assumes that the conductivity is isotropic throughout the domain. To define parts of the domain in which the conductivity is anisotropic, set [*inputs.in*](#) > *feature_control* > *aniso_cond_flag* = 1 and include an [*aniso_cond.in*](#) file in your case setup.

If your case setup includes a [coupled solid-solid interface](#), we recommend using the [nonlinear Krylov-based acceleration \(NKA\) method](#) to solve the electric potential ([*solver.in*](#) > *Transport* > *electric* > *solver_type* = *NONLINEAR_KRYLOV*). Otherwise, we recommend using the CONVERGE implementation of BICGSTAB (*solver_type* = *CONVERGE_BICGSTAB*).

You do not need to set initial conditions for the electric potential. CONVERGE automatically initializes the electric potential to zero in solid regions.

Chapter



4

Numerics

4 Numerics

This chapter describes the numerical options available in CONVERGE and explains the relevant parameters, which are located in the [solver.in](#) and [inputs.in](#) files. This chapter describes selected parameters from these files. [Chapter 25 - Input and Data Files](#) describes these files in full.

4.1 Finite Volume

CONVERGE uses a collocated finite volume approach to numerically solve the conservation equations, a set of nonlinear coupled second-order partial differential equations. Flow quantities are calculated and stored at cell centers according to the summed fluxes through the cell faces and an internal source term, if any. Finite volume methods have the attractive property that they are highly generalizable for complex numerical schemes and non-trivial meshes, such as those resulting from [embedding](#).

Consider the simplified quasi-1D case shown in the following figure. Here, U_i is the vector of conserved variables for cell i and $F_{i-1/2}$ is the flux vector on the $i-1/2$ face (*i.e.*, between cells i and $i-1$).

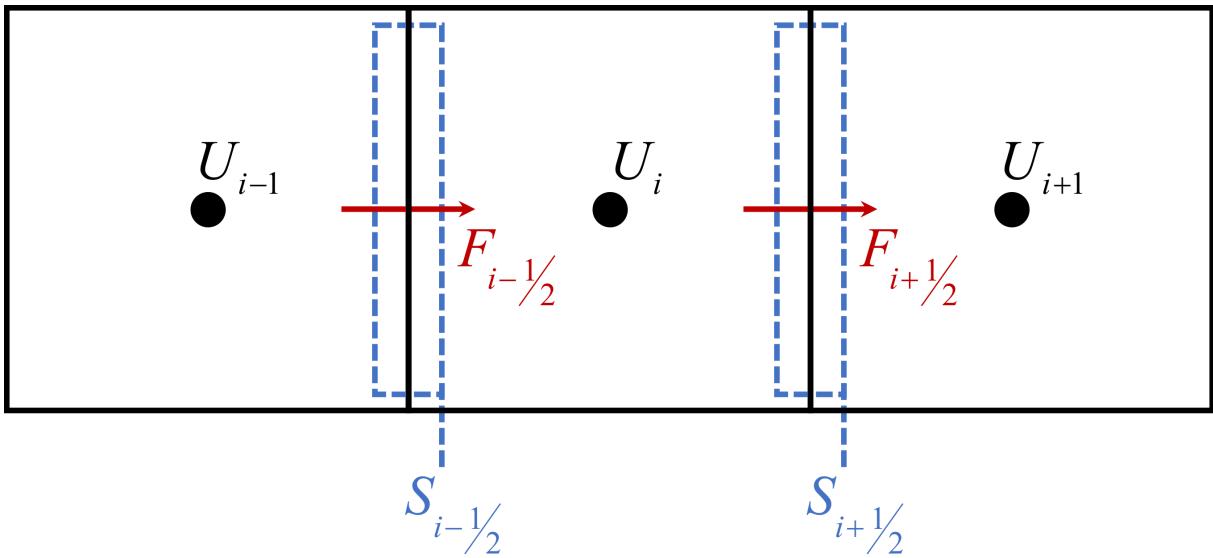


Figure 4.1: Sample three-cell, one-dimensional spatial domain.

Now consider the simple transport equation for a variable ϕ ,

$$\frac{\partial \phi}{\partial t} + \frac{\partial u\phi}{\partial x} = 0. \quad (4.1)$$

Applying the Green-Gauss theorem, we can rewrite this transport equation as

Chapter 4: Numerics

Finite Volume

$$\frac{\partial \phi}{\partial t} + \frac{1}{V} \int_S u \cdot n \phi \, dS = 0. \quad (4.2)$$

where V is the cell volume, S is the surface area, and n is the surface normal. As discussed earlier, finite volume methods solve the integral form of the conservation equations instead of the differential form. The integral form of the equation is solved by summing fluxes on the faces of the cells. This discrete equivalent to Equation 4.2 is written

$$\frac{\partial \phi}{\partial t} + \frac{1}{V} \sum_i u_{f,i} \phi_{f,i} S_i, \quad (4.3)$$

with $u_{f,i}$ as the normal velocity on face i , and $\phi_{f,i}$ representing the transported variable on face i , and S_i representing surface i .

In CONVERGE, all flow quantities are collocated and stored at the center of the cell as shown in Figure 4.1. Quantities such as the face normal and surface area are stored on the cell face. To save memory, decrease runtimes, and improve conservation, CONVERGE stores face data only once for each unique face.

To solve the integral form of Equation 4.3, the velocity and ϕ must be interpolated to the cell surface. To obtain the cell surface value, there are two local options which are numerically stable. The first option is to average the two adjacent cell values and place them on the surface, which results in a surface ϕ given as

$$\phi_{i+1/2} = \frac{1}{2} \phi_i + \frac{1}{2} \phi_{i+1}, \quad (4.4)$$

and

$$\phi_{i-1/2} = \frac{1}{2} \phi_i + \frac{1}{2} \phi_{i-1}. \quad (4.5)$$

The second option is to upwind the surface value for ϕ , which results in

Chapter 4: Numerics

Finite Volume

$$\phi_{i+1/2} = \phi_i, \quad (4.6)$$

and

$$\phi_{i-1/2} = \phi_{i-1}. \quad (4.7)$$

The parameter `Numerical_Schemes > fv_upwind_factor > global` in `solver.in` is used to set the interpolation method for most of the transport equations (`Numerical_Schemes > fv_upwind_factor > mom` and `Numerical_Schemes > fv_upwind_factor > turb` are used for the momentum and turbulence equations, respectively). An `fv_upwind_factor` of 0.5 means that the surface value will be evenly interpolated (*i.e.*, Equations 4.4 and 4.5), while a value of 1.0 means that the value will be entirely upwinded (Equations 4.7 and 4.7). A value between 0.5 and 1.0 will result in a blended scheme.

Setting the appropriate `fv_upwind_factor` to 0.5 results in a second-order accurate spatial scheme. Setting the appropriate `fv_upwind_factor` to 1.0 results in a first-order accurate spatial scheme.

4.2 Solution Procedure

Understanding the order in which the transport equations are solved is important for appropriately configuring your simulation parameters. Figure 4.2 below summarizes the order in which CONVERGE solves the transport equations.

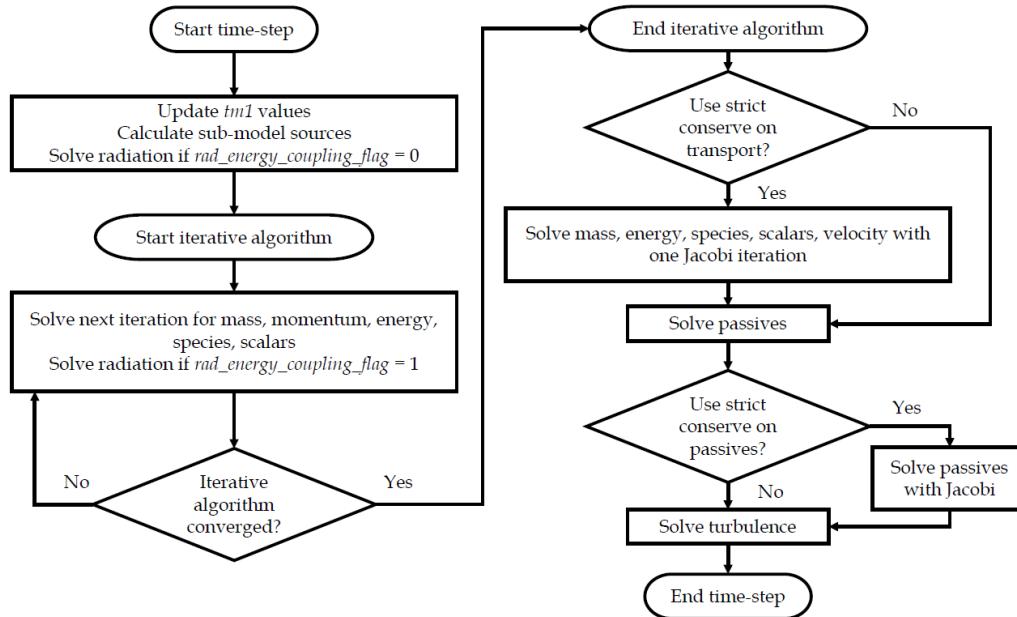


Figure 4.2: Solution order of the transport equations. In most cases, the turbulence equations are outside of the iterative algorithm loop for efficiency reasons.

At the start of each time-step, the previous values (the time-step minus 1 [$tm1$ in Figure 4.2]) are stored for all transported quantities. Next, explicit sources are calculated for each submodel that is currently activated and radiation is solved [if energy and radiation are decoupled](#). CONVERGE next enters an iterative algorithm to solve for mass, momentum, energy, species, and [scalars](#). CONVERGE performs a tolerance check at the end of each iteration, and exits the iterative algorithm if the solution has converged to within tolerance, or if the maximum number of iterations is reached.

After exiting the iterative algorithm, CONVERGE may perform an additional Jacobi iteration to enforce strict conservation. If `Numerical_Schemes > strict_conserve_level = 1` in `solver.in`, CONVERGE will perform this Jacobi iteration on the energy, density, species, and [scalar](#) transport equations. If `Numerical_Schemes > strict_conserve_level = 2`, CONVERGE performs those Jacobi iterations, as well as a Jacobi iteration on [passives](#). The Jacobi iteration guarantees that the quantity is conserved to machine zero, rather than to the tolerance set for the iterative algorithm.

After the optional Jacobi iteration, CONVERGE solves the turbulence equations and ends the time-step.

4.3 Iterative Algorithms

CONVERGE offers several iterative algorithms for the solution of mass, momentum, energy, species, and scalar transport.

PISO Algorithm

CONVERGE offers pressure-velocity coupling using a modified Pressure Implicit with Splitting of Operator (PISO) method of [Issa \(1986\)](#). The PISO algorithm as implemented in CONVERGE starts with a predictor step where the momentum equation is solved. After the predictor, a pressure equation is derived and solved, which leads to a correction, which is applied to the momentum equation. This process of correcting the momentum equation and re-solving can be repeated as many times as necessary to achieve the desired accuracy. After the momentum predictor and first corrector step have been completed, the other transport equations are solved in series.

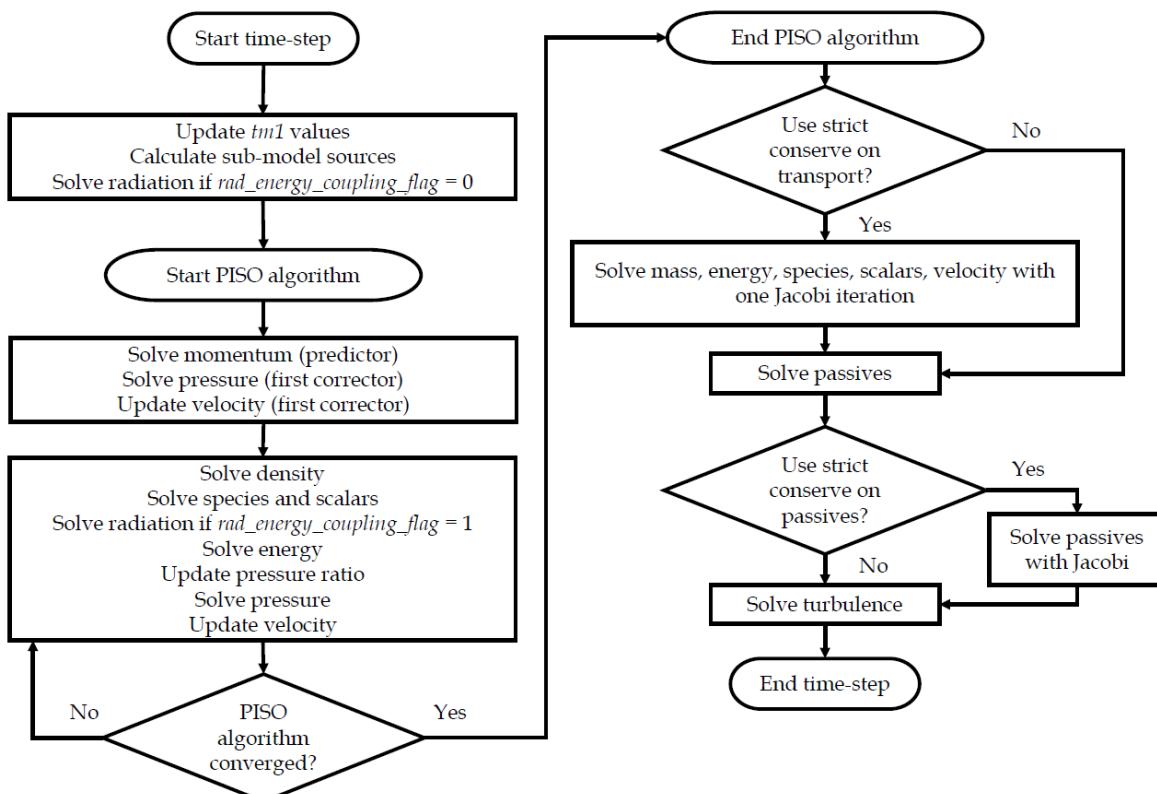


Figure 4.3: Solution order of the transport equations when using the PISO algorithm.

At the beginning of the PISO loop (*i.e.*, the first PISO iteration), CONVERGE solves for momentum and pressure, which sets the velocity for the other transport equations. After each PISO iteration, it is necessary to check for PISO loop convergence. For compressible cases, CONVERGE considers the PISO loop to be converged if

$$\Delta\rho < piso_tol, \quad (4.8)$$

where $\Delta\rho$ is the density correction error. For incompressible cases, CONVERGE considers the PISO loop to be converged if

$$\Delta P < piso_tol, \quad (4.9)$$

where ΔP is the pressure correction error. Note that, if the PISO iteration has converged but the PISO iteration number is less than *piso_itmin*, the PISO iterations will continue until the minimum number of PISO iterations has been exceeded. If the PISO iteration did not converge, CONVERGE executes an additional PISO iteration.

After the PISO loop has ended, CONVERGE may perform an additional Jacobi iteration to enforce strict conservation. If *Numerical_Schemes* > *strict_conserve_level* = 1 in *solver.in*, CONVERGE will perform this Jacobi iteration on the energy, density, species, and *scalar* transport equations. If *Numerical_Schemes* > *strict_conserve_level* = 2, CONVERGE performs those Jacobi iterations, as well as a Jacobi iteration on *passives*. The Jacobi iteration guarantees that the quantity is conserved to machine zero, rather than to the tolerance set by *PISO* > *piso_tol* in *solver.in*.

The PISO method has several attractive characteristics. With only minor variations, this method can be used for solving either compressible or incompressible flows. In addition, the predictor-corrector concept allows for a semi-implicit treatment of sources and sinks: the sources and sinks can be updated at each corrector step. PISO has the advantage of respecting the hyperbolic nature of the transport equations, while using the elliptic nature of the pressure equation to accelerate the communication of information through the domain. A detailed summary of the PISO algorithm is given below.

PISO Algorithm Implementation

The PISO algorithm starts with a predictor step where the momentum equation is solved implicitly or semi-implicitly. The momentum equation is written as

$$\frac{\rho^n u_i^*}{dt} - \frac{\rho^n u_i^n}{dt} = -\frac{\partial P^n}{\partial x_i} + H_i^* \text{ (predictor)}, \quad (4.10)$$

where H^* represents the convection, diffusion, and source terms and $*$ represents the intermediate most up-to-date field values. Each additional $*$ level denotes a higher level of temporal accuracy. The superscript n represents the previous field values. The first corrector momentum equation is written as

Chapter 4: Numerics

Iterative Algorithms PISO Algorithm

$$\frac{\rho^* u_i^{**}}{dt} - \frac{\rho^n u_i^n}{dt} = -\frac{\partial P^*}{\partial x_i} + H_i^* \quad (\text{first corrector}). \quad (4.11)$$

Subtracting Equation 4.10 from Equation 4.11 gives

$$\frac{\rho^* u_i^{**}}{dt} - \frac{\rho^n u_i^*}{dt} = -\frac{\partial}{\partial x_i} (P^* - P^n). \quad (4.12)$$

Rearranging Equation 4.12 to solve for $\rho^* u_i^{**}$ gives

$$\rho^* u_i^{**} = -dt \frac{\partial}{\partial x_i} (P^* - P^n) + \rho^n u_i^*. \quad (4.13)$$

Conservation of mass can be written as

$$\frac{\rho^* - \rho^n}{dt} = -\frac{\partial \rho^* u_i^{**}}{\partial x_i} + S, \quad (4.14)$$

where S accounts for implicit and explicit sources. Substituting Equation 4.13 into Equation 4.14 gives the following expression

$$\frac{\rho^* - \rho^n}{dt} = dt \frac{\partial^2}{\partial x_i \partial x_i} (P^* - P^n) - \frac{\partial \rho^n u_i^*}{\partial x_i} + S. \quad (4.15)$$

In order to rewrite the density on the left-hand-side in terms of pressure, an equation of state is used to relate density to pressure. Density can be rewritten as

$$\rho^* = P^* \phi^n, \quad (4.16)$$

where the pressure ratio term ϕ is defined as

Chapter 4: Numerics

Iterative Algorithms PISO Algorithm

$$\phi^n = \frac{1}{Z^n R^n T^n}, \quad (4.17)$$

where Z is the compressibility factor, R is the ideal gas constant, and T is the cell temperature. For the ideal gas law the compressibility factor is always 1.0.

Substituting Equation 4.16 into Equation 4.15 and rearranging gives

$$\frac{\partial^2}{\partial x_i \partial t} (P^* - P^n) - \frac{(P^* - P^n) \phi^n}{dt^2} = \left(\frac{\partial \rho^n u_i^*}{\partial x_i} - S \right) \frac{1}{dt}. \quad (4.18)$$

Equation 4.18 is the derived pressure equation. Once the pressure is solved, the velocity can be updated according to Equation 4.12. After the velocity is updated, the other transport equations are solved and the pressure ratio is updated accordingly. If necessary, based on the convergence criteria, the correction processes can be repeated.

If a second correction is required, the process starts by writing the first and second corrector momentum equations. These two equations are

$$\frac{\rho^* u_i^{**}}{dt} - \frac{\rho^n u_i^n}{dt} = -\frac{\partial P^*}{\partial x_i} + H_i^* \quad (\text{first corrector}) \quad (4.19)$$

and

$$\frac{\rho^{**} u_i^{***}}{dt} - \frac{\rho^n u_i^n}{dt} = -\frac{\partial P^{**}}{\partial x_i} + H_i^* \quad (\text{second corrector}). \quad (4.20)$$

Subtracting Equation 4.19 from Equation 4.20 gives

$$\frac{\rho^{**} u_i^{***}}{dt} - \frac{\rho^* u_i^{**}}{dt} = -\frac{\partial}{\partial x_i} (P^{**} - P^*). \quad (4.21)$$

Rearranging the above equation to solve for $\rho^{**} u_i^{***}$ results in

Chapter 4: Numerics

Iterative Algorithms PISO Algorithm

$$\rho^{**} u_i^{***} = -dt \frac{\partial}{\partial x_i} (P^{**} - P^*) + \rho^* u_i^{**}. \quad (4.22)$$

The conservation of mass equation for the second corrector is given as

$$\frac{\rho^{**} - \rho^n}{dt} = -\frac{\partial \rho^{**} u_i^{***}}{\partial x_i} + S. \quad (4.23)$$

Substituting Equation 4.22 into Equation 4.23 results in

$$\frac{\rho^{**} - \rho^n}{dt} = dt \frac{\partial^2}{\partial x_i \partial x_i} (P^{**} - P^*) - \frac{\partial \rho^* u_i^{**}}{\partial x_i} + S. \quad (4.24)$$

Density is now written as

$$\rho^{**} = P^{**} \phi^*, \quad (4.25)$$

where the pressure ratio is now given as

$$\phi^* = \frac{1}{Z^* R^* T^*}. \quad (4.26)$$

Substituting Equation 4.25 into Equation 4.24 and rearranging gives

$$\frac{\partial^2}{\partial x_i \partial x_i} (P^{**} - P^*) - \frac{(P^{**} - P^*) \phi^*}{dt^2} = \frac{1}{dt} \left(\frac{\partial \rho^* u_i^{**}}{\partial x_i} - S \right) + \frac{\phi^* P^*}{dt^2} - \frac{\rho^n}{dt^2}. \quad (4.27)$$

The above equation is the second corrector pressure equation. After this equation is solved, the momentum is updated according to Equation 4.22. Again, this process can be repeated, if necessary, to add another corrector. If another corrector is added, the other transport equations are re-solved before each corrector.

You can specify a tolerance value for each equation via the `*_tol` parameters ([transient](#), [steady-state](#)) in [solver.in](#). At each PISO loop, if

$$\frac{|\Psi^* - \Psi^{t-1}|}{|\Psi^*|} < \Psi_tol, \quad (4.28)$$

where Ψ^* is the current correction for the Ψ transport equation and Ψ^{t-1} is the previous value of the correction for that transport equation, then CONVERGE will not re-solve that particular transport equation (with the exception of pressure, which is always solved).

In order to limit the PISO corrections, you can enter the minimum and maximum number of PISO corrections allowed. If the maximum number of PISO iterations is exceeded and the solution has not converged, the following time-step will be reduced.

In flows in which the temperature or species vary greatly, it may aid convergence to under-relax the calculation of the pressure ratio (`PISO > omega_presrat` in [solver.in](#)). After updating the pressure ratio, the PISO convergence is checked again. CONVERGE continues this process until a PISO iteration converges or until it has completed twice the maximum number of PISO iterations (`PISO > piso_itmax` in [solver.in](#)). CONVERGE cuts the time-step if `piso_itmax` is exceeded. CONVERGE recovers if it reaches twice `piso_itmax`.

SIMPLE Algorithm

CONVERGE offers pressure-velocity coupling using a modified Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) method of [Patankar \(1980\)](#). The modified SIMPLE algorithm differs from the [PISO algorithm](#) described above in that the momentum equation is solved within the iterative algorithm, rather than as a predictor step. At each iteration, SIMPLE solves the momentum equation, then a derived pressure equation. As with PISO, after the pressure solution is used to update the velocity, the other transport equations are solved in series.

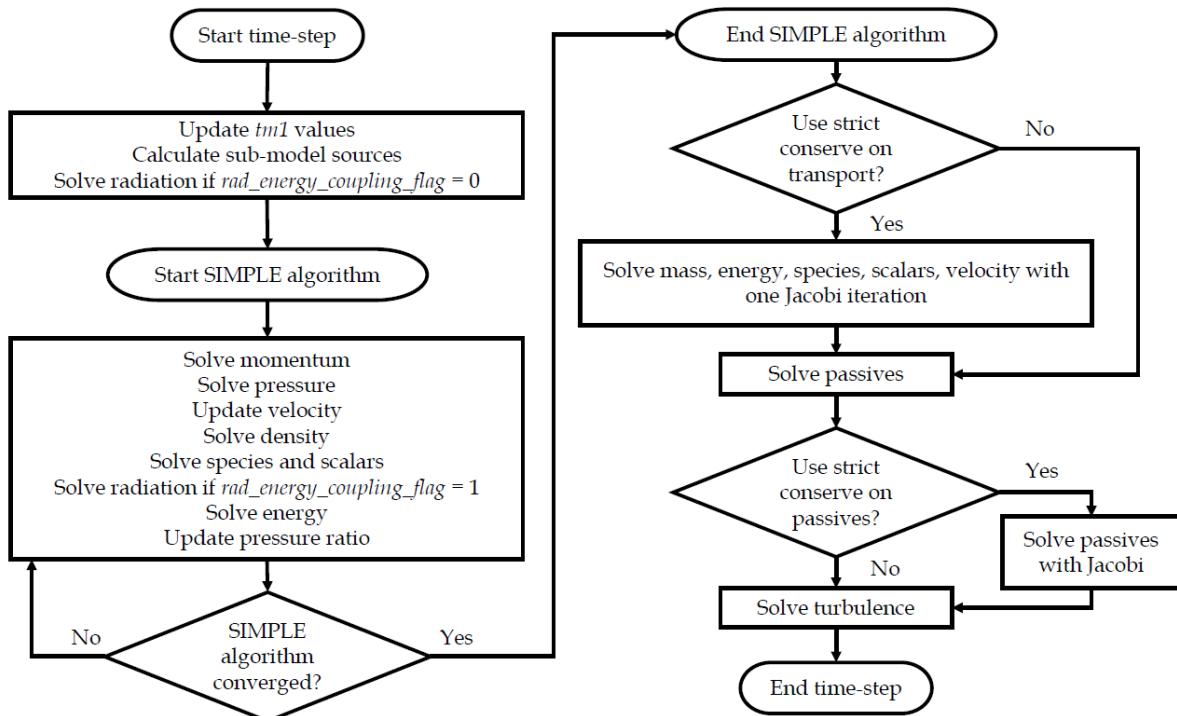


Figure 4.4: Solution order of the transport equations when using the SIMPLE algorithm.

At the beginning of each SIMPLE iteration, CONVERGE solves for momentum and pressure, which sets the velocity for the other transport equations. After each SIMPLE iteration, it is necessary to check for SIMPLE loop convergence. For compressible cases, CONVERGE considers the SIMPLE loop to be converged if

$$\Delta\rho < simple_tol, \quad (4.29)$$

where $\Delta\rho$ is the density correction error. For incompressible cases, CONVERGE considers the SIMPLE loop to be converged if

$$\Delta P < simple_tol, \quad (4.30)$$

where ΔP is the pressure correction error. Note that, if the SIMPLE iteration has converged but the SIMPLE iteration number is less than $simple_itmin$, the SIMPLE iterations will continue until the minimum number of SIMPLE iterations has been exceeded. If the SIMPLE iteration did not converge, CONVERGE executes an additional SIMPLE iteration.

After the SIMPLE loop has ended, CONVERGE may perform an additional Jacobi iteration to enforce strict conservation. If *Numerical_Schemes > strict_conserve_level = 1* in *solver.in*, CONVERGE will perform this Jacobi iteration on the energy, density, species, and *scalar* transport equations. If *Numerical_Schemes > strict_conserve_level = 2*, CONVERGE performs those Jacobi iterations, as well as a Jacobi iteration on *passives*. The Jacobi iteration guarantees that the quantity is conserved to machine zero, rather than to the tolerance set by *SIMPLE > simple_tol* in *solver.in*.

The SIMPLE method shares many of the attractive characteristics of PISO. With only minor variations, this method can be used for solving either compressible or incompressible flows. In addition, the predictor-corrector concept allows for a semi-implicit treatment of sources and sinks: the sources and sinks can be updated at each corrector step. SIMPLE has the advantage of respecting the hyperbolic nature of the transport equations, while using the elliptic nature of the pressure equation to accelerate the communication of information through the domain.

The practical difference between SIMPLE and PISO is that SIMPLE more strongly couples the momentum equation with the rest of the transport equations. While this is more numerically expensive than the PISO approach, it is more robust in some flows, particularly those which may be locally dominated by viscosity. SIMPLE is appropriate for well-resolved boundary layers, and may be more stable in *transient* simulations with a large time-step or in *steady-state* simulations with a large pseudo time-step.

SIMPLE Algorithm Implementation

The modified SIMPLE algorithm solves the same equations as PISO, but they are solved in a modified sequence. Each SIMPLE iteration follows the same sequence, without a predictor before the first iteration. At each iteration, CONVERGE performs the following steps.

1. Solve the momentum equation, Equation 4.10, to compute the intermediate velocity field.
2. Solve for the pressure correction, Equation 4.18, and update the pressure field, Equation 4.26.
3. Update the density, Equation 4.13.
4. Correct the velocities and face fluxes using the pressure correct, Equation 4.20.
5. Solve density and the other transport equations.
6. Update the pressure ratio, Equation 4.12.
7. Check for convergence, as described above.

You can optionally direct CONVERGE to calculate the turbulence transport equations within the SIMPLE loop (*SIMPLE > couple_turb_flag = 1* in *solver.in*). In this case, the turbulence equations are not re-calculated at the end of the time-step as is shown in Figure 4.4.

You can specify a tolerance value for each equation via the *_tol parameters (*transient*, *steady-state*) in *solver.in*. At each SIMPLE loop, if

$$\frac{|\Psi^* - \Psi^{t-1}|}{|\Psi^*|} < \Psi_tol, \quad (4.31)$$

where Ψ^* is the current correction for the Ψ transport equation and Ψ^{t-1} is the previous value of the correction for that transport equation, then CONVERGE will not re-solve that particular transport equation (with the exception of pressure, which is always solved).

In order to limit the SIMPLE corrections, you can enter the minimum and maximum number of SIMPLE corrections allowed. If the maximum number of SIMPLE iterations is exceeded and the solution has not converged, the following time-step will be reduced.

In flows in which the temperature or species vary greatly, it may aid convergence to under-relax the calculation of the pressure ratio (*SIMPLE > omega_presrat* in [solver.in](#)). After updating the pressure ratio, the SIMPLE convergence is checked again. CONVERGE continues this process until a SIMPLE iteration converges or until it has completed twice the maximum number of SIMPLE iterations (*SIMPLE > simple_itmax* in [solver.in](#)). CONVERGE cuts the time-step if *simple_itmax* is exceeded. CONVERGE recovers if it reaches twice *simple_itmax*.

Pressure-Based PISO and SIMPLE

For compressible simulations, in addition to the density-based formulations described above, CONVERGE offers pressure-based formulations of the PISO and SIMPLE algorithms. This formulation trades reduced accuracy for reduced computational cost by not explicitly solving the density transport equation. Instead, density is calculated based on pressure, temperature, and the equation of state. The reduction in accuracy will be greater at higher Mach numbers (greater than about 0.3), and the reduction in computational time will be greater for simulations with a small number of gas species.

For incompressible simulations, CONVERGE always solves the pressure-based formulation of PISO or SIMPLE. Because there is no density change in an incompressible simulation, there is no associated reduction in accuracy.

4.4 Rhee-Chow Algorithm

All transported quantities in CONVERGE are colocated at the center of the cell. The use of colocated quantities can result in a decoupling of the pressure and velocity. This decoupling can produce fluctuations in the pressure and velocity solution that appear in a checkerboard pattern. In the past, many CFD codes used a staggered grid approach to eliminate the checkerboarding (*i.e.*, velocity located at the cell face and pressure at the cell center). However, the Rhee-Chow interpolation scheme ([Rhee and Chow, 1983](#)) can be used to maintain colocated variables and eliminate the undesirable checkerboarding.

The Rhee-Chow scheme reduces the checkerboarding effect in the pressure and velocity, but it also adds a small error to the solution, similar to the error introduced in the staggered

approach. Despite this small error, we recommended running most simulations with either the legacy Rhie-Chow scheme ([solver.in](#) > *Numerical_schemes* > *rc_flag* = *LEGACY_RC*) or the generalized Rhie-Chow scheme ([solver.in](#) > *Numerical_schemes* > *rc_flag* = *GENERALIZED_RC*) activated. While the two options produce similar results for many simulations, the generalized Rhie-Chow scheme may improve accuracy or performance for certain applications, such as those involving porous media, body force, or buoyancy effects. Note that CONVERGE automatically sets *rc_flag* = *GENERALIZED_RC* in streams that contain at least one [porous media source](#).

Legacy Rhie-Chow Scheme

The legacy Rhie-Chow scheme is the original implementation of the Rhie-Chow algorithm in CONVERGE. This algorithm works by approximating the beneficial effects of the staggered grid approach but leaving the variables colocated. For a colocated set of variables on a uniform grid, a velocity at a cell face would be written as

$$u_{i+1/2} = \frac{u_i + u_{i+1}}{2}, \quad (4.32)$$

where i represents the cell of interest and $i+1$ represents the cell to the right. A staggered update to the cell face for pressure would be

$$u_{i+1/2}^* = u_{i+1/2} - \frac{dt}{\rho} \left(\frac{P_{i+1} - P_i}{dx} \right), \quad (4.33)$$

while a colocated update would be

$$u_i^* = u_i - \frac{dt}{\rho} \left(\frac{P_{i+1} - P_{i-1}}{2dx} \right), \quad (4.34)$$

where $*$ indicates a velocity that has been updated for pressure. For a face velocity using a colocated scheme, the update would be

$$u_{i+1/2}^* = \frac{u_i}{2} - \frac{dt}{2\rho} \left(\frac{P_{i+1} - P_{i-1}}{2dx} \right) + \frac{u_{i+1}}{2} - \frac{dt}{2\rho} \left(\frac{P_{i+2} - P_i}{2dx} \right), \quad (4.35)$$

or, equivalently,

Chapter 4: Numerics

Rhie-Chow Algorithm Legacy Rhie-Chow Scheme

$$u_{i+1/2}^* = \frac{u_i + u_{i+1}}{2} - \frac{dt}{2\rho} \left(\frac{P_{i+1} - P_{i-1}}{2dx} + \frac{P_{i+2} - P_i}{2dx} \right). \quad (4.36)$$

For the Rhie-Chow scheme, a pressure gradient term is added to cancel the colocated pressure update (Equation 4.36) and a pressure term is added. This additional pressure term is equivalent to the staggered pressure update (Equation 4.33). This process results in

$$u_{i+1/2}^* = \frac{u_i^* + u_{i+1}^*}{2} - \frac{dt}{\rho} \left(\frac{P_{i+1} - P_i}{dx} \right) + \frac{dt}{2\rho} \left(\frac{P_{i+1} - P_{i-1}}{2dx} + \frac{P_{i+2} - P_i}{2dx} \right), \quad (4.37)$$

where the last two terms represent the legacy Rhie-Chow correction. The expression from Equation 4.37 is valid for a uniform grid. The equivalent expression for a non-uniform grid is given by

$$u_f^* = u_i^* (1 - \beta) + u_{i+1}^* \beta - \frac{dt}{\rho} \frac{\partial P}{\partial x} \Big|_f + \frac{dt}{\rho} \left[(1 - \beta) \frac{\partial P}{\partial x} \Big|_i + \beta \frac{\partial P}{\partial x} \Big|_{i+1} \right], \quad (4.38)$$

where the factor β accounts for non-uniform grid spacing ($0 < \beta < 1$).

Equations 4.37 and 4.38 reflect the legacy Rhie-Chow correction that is applied by the [transient solver](#). The [steady-state solver](#) modifies these expressions by replacing dt with $\min(dt, 5.0e-6 \text{ s})$. For time-steps larger than $5.0e-6 \text{ s}$, this makes the face velocities independent of the time-step and thereby prevents the residuals from fluctuating with the time-step.

To apply the legacy Rhie-Chow correction, set [`solver.in`](#) > *Numerical_schemes* > *rc_flag* = *LEGACY_RC*.

Generalized Rhie-Chow Scheme

To obtain a more general form of the Rhie-Chow correction that can account for flow resistance, body force, and other effects ([Zhang et al., 2014](#)), we start from a shorthand description of the pressure update to the Cartesian components of velocity, given by

$$u^* = D_u \left(H_u - \frac{\partial P}{\partial x} \right), v^* = D_v \left(H_v - \frac{\partial P}{\partial y} \right), w^* = D_w \left(H_w - \frac{\partial P}{\partial z} \right), \quad (4.39)$$

Chapter 4: Numerics

Rhie-Chow Algorithm Generalized Rhie-Chow Scheme

where H represents all of the terms on the right-hand side of the momentum equation except for the pressure gradient, and D is the inverse of the diagonal part of the coefficient matrix.

Taking the x component as an example, a staggered update to the velocity at the cell face would be

$$u_f^* = D_{u,f} \left(H_{u,f} - \frac{\partial P}{\partial x} \Big|_f \right), \quad (4.40)$$

while a colocated update at the cell center would be

$$u_i^* = D_{u,i} \left(H_{u,i} - \frac{\partial P}{\partial x} \Big|_i \right). \quad (4.41)$$

The face velocity update for a uniform grid in the colocated scheme is then

$$\begin{aligned} u_f^* &= \frac{u_i^* + u_{i+1}^*}{2} \\ &= \frac{1}{2} \left(D_{u,i} H_{u,i} + D_{u,i+1} H_{u,i+1} \right) - \frac{1}{2} \left(D_{u,i} \frac{\partial P}{\partial x} \Big|_i + D_{u,i+1} \frac{\partial P}{\partial x} \Big|_{i+1} \right). \end{aligned} \quad (4.42)$$

Subtracting Equation 4.42 from Equation 4.40 gives the generalized Rhie-Chow correction:

$$\begin{aligned} rc_u &= -\frac{1}{2} \left(D_{u,i} + D_{u,i+1} \right) \frac{\partial P}{\partial x} \Big|_f + \frac{1}{2} \left(D_{u,i} \frac{\partial P}{\partial x} \Big|_i + D_{u,i+1} \frac{\partial P}{\partial x} \Big|_{i+1} \right) \\ &= -D_{u,f} \frac{\partial P}{\partial x} \Big|_f + \frac{1}{2} \left(D_{u,i} \frac{\partial P}{\partial x} \Big|_i + D_{u,i+1} \frac{\partial P}{\partial x} \Big|_{i+1} \right), \end{aligned} \quad (4.43)$$

where we have applied the following assumptions:

$$D_{u,f} H_{u,f} = \frac{1}{2} \left(D_{u,i} H_{u,i} + D_{u,i+1} H_{u,i+1} \right) \text{ and } D_{u,f} = \frac{1}{2} \left(D_{u,i} + D_{u,i+1} \right). \quad (4.44)$$

Similar expressions can be derived for rc_v and rc_w . The full correction to the normal velocity at the cell face is given by the dot product with the face normal $n_f = [n_x, n_y, n_z]$:

Chapter 4: Numerics

Rhie-Chow Algorithm Generalized Rhie-Chow Scheme

$$\begin{aligned}
 rc &= [rc_u, rc_v, rc_w] \cdot [n_x, n_y, n_z] \\
 &= -\left(D_{u,f}n_x^2 + D_{v,f}n_y^2 + D_{w,f}n_z^2\right) \frac{P_{i+1} - P_i}{dr} + \frac{1}{2} \left(D_{u,i} \frac{\partial P}{\partial x} \Big|_i + D_{u,i+1} \frac{\partial P}{\partial x} \Big|_{i+1} \right) n_x \\
 &\quad + \frac{1}{2} \left(D_{v,i} \frac{\partial P}{\partial y} \Big|_i + D_{v,i+1} \frac{\partial P}{\partial y} \Big|_{i+1} \right) n_y + \frac{1}{2} \left(D_{w,i} \frac{\partial P}{\partial z} \Big|_i + D_{w,i+1} \frac{\partial P}{\partial z} \Big|_{i+1} \right) n_z,
 \end{aligned} \tag{4.45}$$

where we have used the relationship

$$\nabla P \Big|_f = \frac{P_{i+1} - P_i}{dr} n_f \tag{4.46}$$

to simplify the first term.

The generalized Rhie-Chow correction in Equation 4.45 is valid for a uniform grid. The equivalent expression for a non-uniform grid is given by

$$\begin{aligned}
 rc &= -\left(D_{u,f}n_x^2 + D_{v,f}n_y^2 + D_{w,f}n_z^2\right) \frac{P_{i+1} - P_i}{dr} \\
 &\quad + \left[(1-\beta)D_{u,i} \frac{\partial P}{\partial x} \Big|_i + \beta D_{u,i+1} \frac{\partial P}{\partial x} \Big|_{i+1} \right] n_x \\
 &\quad + \left[(1-\beta)D_{v,i} \frac{\partial P}{\partial y} \Big|_i + \beta D_{v,i+1} \frac{\partial P}{\partial y} \Big|_{i+1} \right] n_y \\
 &\quad + \left[(1-\beta)D_{w,i} \frac{\partial P}{\partial z} \Big|_i + \beta D_{w,i+1} \frac{\partial P}{\partial z} \Big|_{i+1} \right] n_z,
 \end{aligned} \tag{4.47}$$

where the factor β accounts for non-uniform grid spacing ($0 < \beta < 1$).

When calculating the components of D that appear in Equation 4.39, the [steady-state solver](#) replaces the time-step dt with $\min(dt, 5.0e-6 \text{ s})$. For time-steps larger than $5.0e-6 \text{ s}$, this makes the face velocities independent of the time-step and thereby prevents the residuals from fluctuating with the time-step.

To apply the generalized Rhie-Chow correction, set [`solver.in`](#) > `Numerical_schemes` > `rc_flag = GENERALIZED_RC`.

4.5 Non-Local Convective Flux Schemes

In addition to the local interpolants described above, it is possible to formulate convective flux schemes that are not purely local (*i.e.*, the flux at the $i-1/2$ face is a function of cells beyond i and $i-1$). Such schemes offer numerical behavior that is potentially more desirable than purely local methods, although their construction requires the inclusion of limiter functions to ensure numerical stability.

MUSCL Scheme

The MUSCL (Monotonic Upstream-Centered Scheme for Conservation Laws) scheme of van Leer ([van Leer, 1979](#)) provides second-order spatial accuracy for the convection term. To do so, MUSCL calculates the value of a scalar ϕ at a cell face using a blend of second-order upwind and reconstructed central difference spatial discretization schemes.

For a scalar quantity ϕ , CONVERGE approximates the value at a cell face (ϕ_f) with a linear reconstruction method. This method calculates ϕ_f using a Taylor Series expansion from the center of cell i to its face. Values for ϕ_f on either side of the cell face are given as

$$\begin{aligned}\phi_{f,i-1} &= \phi_{i-1} + \nabla \phi_{i-1} \cdot d_{i-1} \\ \phi_{f,i} &= \phi_i + \nabla \phi_i \cdot d_i\end{aligned}\quad (4.48)$$

where d_i is the vector from the center of cell i to the cell face. The first expression reconstructs ϕ_f from the center of cell $i-1$ while the second expression reconstructs ϕ_f from the center of cell i . In this example, the flow moves from cell $i-1$ to cell i , so cell $i-1$ is upwind of cell i . For numerical consistency, each variable is reconstructed to the face independently.

Next, the reconstructed upwind scheme (denoted by subscript ru) uses upwinding to represent ϕ_f . We can approximate the value of ϕ at a cell face using the reconstructed quantity from the upwind cell as

$$\phi_{f,ru} = \phi_{f,i-1}. \quad (4.49)$$

The other half of the MUSCL scheme is to take a central difference approximation of the two reconstructions listed in Equation 4.48. This method averages the two adjacent cell quantities as

Chapter 4: Numerics

Non-Local Convective Flux Schemes	MUSCL Scheme
---	--------------

$$\phi_{f,rcd} = \frac{1}{2}(\phi_{f,i-1} + \phi_{f,i}). \quad (4.50)$$

To reach higher-order accuracy, the MUSCL scheme blends the second-order upwind discretization (Equation 4.49) and the central difference discretization (Equation 4.50) as

$$\phi_{f,muscl} = (1 - \beta)\phi_{f,ru} + \beta\phi_{f,rcd}, \quad (4.51)$$

where β is a blending factor.

Limiter Functions

When you activate the MUSCL scheme in CONVERGE, you must choose a limiter function. Limiter functions improve numerical stability by limiting either the slope (gradient) of the solution in a cell or by limiting the flux through a face. CONVERGE offers the following options:

- *MUSCL_CVG* (recommended) employs a 3D gradient-based slope limiter. You can choose between two types of slope limiters: *minmod* and *venkatak*.
- *MUSCL_1* employs a 1D flux limiter in each individual direction. You can choose from several types of flux limiters.
- *MUSCL_2* employs *MUSCL_1* with step limiter and additional limiting factors to help stabilize the simulation.

To activate the MUSCL scheme for a desired equation or set of equations, set *Numerical_Schemes > flux_scheme* to *MUSCL_CVG*, *MUSCL_1*, or *MUSCL_2* for the appropriate equations in [*solver.in*](#). (*Numerical_Schemes > flux_scheme > mom* corresponds to the momentum equation, *Numerical_Schemes > flux_scheme > global* corresponds to all transport equations except momentum and turbulence, and *Numerical_Schemes > flux_scheme > turb* corresponds to the turbulence equations.)

You can specify different blending factors for each set of equations. Setting *Numerical_Schemes > muscl_blend_factor > ** to 0.0 results in pure second-order upwind MUSCL for those equations.

Slope Limiters

We can express ϕ_f with the application of a slope limiter as

Chapter 4: Numerics

Non-Local Convective Flux Schemes Limiter Functions

$$\phi_{f,u} = \phi_u + \eta_u \nabla \phi_u \cdot d_u, \quad (4.52)$$

where η_u is the slope limiter for cell u .

CONVERGE calculates η_u using either the minmod method of [Barth and Jespersen \(1989\)](#) or the method of [Venkatakrishnan \(1993\)](#). The minmod method ensures that the gradient $\nabla \phi$ is smooth and preserves monotonicity by preventing ϕ from creating new local maxima or minima. However, this method causes the reconstructed function ϕ_f to be non-differentiable, which can produce oscillations that prevent convergence, particularly for steady-state cases. The method of [Venkatakrishnan \(1993\)](#) avoids this problem by replacing the minmod function with a smooth function.

MUSCL_2 employs the slope limiter of [Venkatakrishnan \(1993\)](#). For MUSCL_CVG, you can use the *Numerical_Schemes > flux_limiter* parameter in [*solver.in*](#) to specify which slope limiter you want to use for each set of equations. Note that this parameter is typically used to specify a flux limiter, but in this special case, it is used to specify a slope limiter. Enter *minmod* to use the minmod slope limiter or *venkatak* to use the slope limiter of [Venkatakrishnan \(1993\)](#).

Flux Limiters

A flux limiter is a function that restricts fluxes (represented by f) to meaningful values near discontinuities in the domain. Near these discontinuities, a flux limiter switches to a first-order spatial discretization to avoid spurious oscillations in the solution. In the rest of the domain, however, the flux limiter function employs a higher-order spatial discretization to improve solution accuracy. The formulation for a flux-limited quantity is

$$f_{HO,limited} = f_{1st,up} + \psi(r) (f_{HO} - f_{1st,up}), \quad (4.53)$$

where $f_{HO,limited}$ is the limited higher-order flux, $f_{1st,up}$ is the first-order flux, $\psi(r)$ is the limiter function, and f_{HO} is the higher-order flux. Note that this equation implies if the flux is already evaluated as first-order upwind (*i.e.*, *Numerical_Schemes > fv_upwind_factor > * = 1.0* or *Numerical_Schemes > upwind_all_dir_flag = 1* in [*solver.in*](#)), the flux is already $f_{1st,up}$ and limiters have no effect. In Equation 4.53 above, r is the input to the flux limiter and is a function of the gradients on either side of a cell face. The equation for r is

Chapter 4: Numerics

Non-Local Convective Flux Schemes Limiter Functions

$$r = \frac{\phi_i - \phi_{i-1}}{\phi_{i+1} - \phi_i}, \quad (4.54)$$

where ϕ is some scalar quantity.

CONVERGE includes many flux limiter functions, which are listed along with their formulations in the following table. *MUSCL_2* employs the *step* flux limiter. For *MUSCL_1*, you can choose any flux limiter in the table. Set *Numerical_Schemes > flux_limiter > mom*, *Numerical_Schemes > flux_limiter > global*, or *Numerical_Schemes > flux_limiter > turb* in *solver.in* to the appropriate parameter name from column 1.

Note that when you choose the *step* flux limiter, use *Numerical_Schemes > monotone_tolerance* in *solver.in* to control how CONVERGE preserves monotonicity. CONVERGE will switch to a first-order upwind spatial discretization when the ratio of gradients on either side of a cell face exceeds the value you specify for *monotone_tolerance*. CONVERGE can list whether the solver used the specified numerical scheme or first-order upwind scheme if you include *cells > general > monotone_upwind* and *cells > general > prod_monotone_upwind* in *post.in*. CONVERGE compares the gradients of density on either either side of a cell face as

$$\text{Sign}\left[\frac{\rho_{i+1} - \rho_i}{x_{i+1} - x_i}\right] \neq \text{Sign}\left[\frac{\rho_i - \rho_{i-1}}{x_i - x_{i-1}}\right], \quad (4.55)$$

where i indicates the cell, $i+1$ is the cell to the right, $i-1$ is the cell to the left, ρ is the density, and x is the spatial location. If either the density or velocity field becomes non-monotonic, it is likely that there is a discontinuity in the field. In this event, first-order upwind is the most accurate spatial discretization scheme. If your simulation is free of shocks or other large discontinuities, a relatively larger value of *Numerical_Schemes > monotone_tolerance* will produce a more accurate solution with less dissipation.

If you choose the *step* flux limiter in conjunction with a first-order upwind spatial scheme (*Numerical_Schemes > fv_upwind_factor > global* and *Numerical_Schemes > fv_upwind_factor > mom* are 1.0 in *solver.in*), adjusting *Numerical_Schemes > monotone_tolerance* may affect your solution. With this configuration, if CONVERGE detects non-monotonic behavior in a cell (such that the ratio of gradients exceeds *monotone_tolerance*), *Numerical_Schemes > conserve* becomes 0 locally, effectively solving the momentum equation with *cell-center-evaluation form* to preserve stability. If the ratio of gradients for the cell is within the limit given by *monotone_tolerance*, CONVERGE uses the value of *conserve* you supply in *solver.in*.

For guidance on which flux limiter to use, consult the CONVERGE example cases. Your choice of flux limiter will rarely have a major effect on the solution. More often, the effect

Chapter 4: Numerics

Non-Local Convective Flux Schemes Limiter Functions

will be subtle and case-dependent. If you have had success with a specific flux limiter on a specific case setup, we do not make a strong recommendation to change.

Table 4.1: Flux limiters in CONVERGE.

Parameter name	Flux limiter	Formulation
<i>step</i>	Step	$ \psi_i - \psi_{i-1}) - (\psi_{i+1} - \psi_i) < \text{monotone_tolerance}$
<i>charm</i>	CHARM (Zhou, 1995)	$\psi_{cm}(r) = \begin{cases} \frac{r(3r+1)}{(r+1)^2} & r > 0 \\ 0 & r \leq 0 \end{cases}$
<i>hcus</i>	HCUS (Waterson and Deconinck, 1995)	$\psi_{hc}(r) = \frac{1.5(r+ r)}{(r+2)}$
<i>hquick</i>	HQUICK (Waterson and Deconinck, 1995)	$\psi_{hq}(r) = \frac{2(r+ r)}{(r+3)}$
<i>koren</i>	Koren (Koren, 1993)	$\psi_{kn}(r) = \max[0, \min(2r, (2+r)/3, 2)]$
<i>minmod</i>	minmod (Roe, 1986)	$\psi_{mm}(r) = \max[0, \min(1, r)]$
<i>mc</i>	monotonized central (van Leer, 1977)	$\psi_{mc}(r) = \max[0, \min(2r, 0.5(1+r), 2)]$
<i>ospre</i>	ospre (Waterson and Deconinck, 1995)	$\psi_{op}(r) = \frac{1.5(r^2+r)}{(r^2+r+1)}$
<i>smart</i>	smart (Gaskell and Lau, 1988)	$\psi_{sm}(r) = \max[0, \min(2r, (0.25+0.75r), 4)]$
<i>superbee</i>	superbee (Roe, 1986)	$\psi_{sb}(r) = \max[0, \min(2r, 1), \min(r, 2)]$
<i>umist</i>	UMIST (Lien and Leschziner, 1994)	$\psi_{um}(r) = \max\left[0, \min\left(2r, (0.25+0.75r), (0.75+0.25r), 2\right)\right]$
<i>user</i>	User-defined function (UDF)	Calculated by the <i>flux_limiter</i> UDF (refer to the CONVERGE 3.1 UDF Manual for more information).
<i>vanalbada1</i>	van Albada 1 (van Albada et al., 1983)	$\psi_{val}(r) = \frac{r^2+r}{r^2+1}$

Chapter 4: Numerics

Non-Local Convective Flux Schemes Limiter Functions

Parameter name	Flux limiter	Formulation
<i>vanalbada2</i>	van Albada 2 (Kermani, 2003)	$\psi_{va2}(r) = \frac{2r}{r^2 + 1}$
<i>vanleer</i>	van Leer (van Leer, 1974)	$\psi_{vl}(r) = \frac{r + r }{1 + r }$

4.6 Iterative Linear Solvers

Before running a CONVERGE simulation, you need to select a linear solver for each governing equation. CONVERGE offers the [pointwise successive over-relaxation \(SOR\) algorithm](#) and two implementations of the [biconjugate gradient stabilized \(BiCGSTAB\) method](#), although BiCGSTAB is not available for some of the governing equations. Refer to the description of [solver.in](#) to see which solvers are available for each governing equation.

SOR iterations are much less computationally expensive than BiCGSTAB iterations. In general, you should use the SOR algorithm for simulations in which the number of pressure iterations per time-step is, on average, fewer than 15. This algorithm is ideal for cases with compressible flows and small time-steps, such as engine simulations. The BiCGSTAB methods are more appropriate for simulations which are slow to converge, such as volume of fluid (VOF) cases.

After discretizing the governing equations, we can write each equation as a linear system

$$Ax = b. \quad (4.56)$$

The solvers in CONVERGE are iterative, *i.e.*, each solver starts with an initial guess x_0 and finds subsequent values that increasingly approach the given value of x . This process can be described as the limit given by

$$\lim_{n \rightarrow \infty} x_n = x. \quad (4.57)$$

After each iteration, CONVERGE calculates the residual

Chapter 4: Numerics

Iterative Linear Solvers

$$r_n = \frac{\|Ax_n - b\|_2}{\|b\|_2} \quad (4.58)$$

and checks if

$$r_n \leq * _tol \quad (4.59)$$

and

$$r_n \leq \frac{r_0}{tol_scale}, \quad (4.60)$$

where $* _tol$ (e.g., $Transport > mom > tol$) is an equation-specific tolerance in [solver.in](#) and the tolerance scale (e.g., $PISO > tol_scale$) is a scaling parameter. If either of Equations 4.59 and 4.60 is satisfied, or if $Transport > * > itmax$ has been reached, the solver stops. If not, the solver will perform another iteration.

The following subsections ([SOR Algorithm](#), [CONVERGE BiCGSTAB Method](#), [HYPRE BiCGSTAB Method](#), and [Nonlinear Krylov Acceleration Method](#)) contain solver-specific information.

SOR Algorithm

The SOR algorithm is an iterative scheme with a relaxation factor, ω , for accelerating convergence. Given a square system of

$$Ax = b, \quad (4.61)$$

the system can be rewritten as

$$x_i^{k+1} = (1 - \omega)x_i^k + \frac{\omega}{a_{ii}} \left(b_i - \sum_{j < i} a_{ij}x_j^{k+1} - \sum_{j > i} a_{ij}x_j^k \right), \quad (4.62)$$

where $i = 1, 2, \dots, n$.

When solving systems of equations using iterative techniques, it may be necessary to under-relax the solution to aid in the convergence. Under-relaxing means that the next iteration is set to a value based on the previous value and a scaled corrector term, as

$$\phi^{**} = \phi^* + \omega_{under_relaxation} (\Delta\phi), \quad (4.63)$$

where ϕ^{**} is the new iteration value, ϕ^* is the old iteration value, $\omega_{under_relaxation}$ is the under-relaxation parameter (usually less than 1.0), and $\Delta\phi$ is the calculated change in the iteration value.

For the SOR algorithm, CONVERGE calculates an optimal under-relaxation parameter (ω_{opt}) for each transport equation in order to speed up convergence and ensure stability. The values that you specify for the under-relaxation parameters in [solver.in](#) are the maximum under-relaxation values for the simulation. That is, if the value CONVERGE calculates for ω_{opt} exceeds ω_{input} , CONVERGE will use ω_{input} . If ω_{opt} does not exceed ω_{input} , CONVERGE will use ω_{opt} to ensure stability. At each time-step, CONVERGE determines the optimal value of the under-relaxation parameter for each transport equation in each cell via

$$\omega_{opt} = \frac{|a_{ii}|}{\sum_{\forall j, i \neq j} |a_{ij}|}, \quad (4.64)$$

where a_{ij} is the value of entry (i, j) in the coefficient matrix of the discretized transport equation. The result is a field of ω_{opt} values that CONVERGE uses in conjunction with Equation 4.63 to determine the new iteration values. Note that Equation 4.64 does not apply to pressure under-relaxation. In CONVERGE, there is no dynamic relaxation for pressure, and you can enter a relaxation factor for pressure using *Transport > pres > omega* in [solver.in](#).

The under-relaxation parameters are given in *Transport > * > omega* in [solver.in](#), where * is *mom*, *pres*, etc. Each of these parameters can be a fixed value or can [vary temporally](#). For a temporally varying parameter, specify a file name instead of a value and include that file in your case setup.

For the pressure, energy, and species equations, the SOR solver estimates the residual at some iterations, which helps reduce runtimes by eliminating an MPI reduction operation per iteration. If r_1 and r_2 are the residuals calculated at iterations 1 and 2, respectively, the estimated residual at iteration 3 satisfies

$$\frac{r_1}{r_2} = \frac{r_2}{r_3}, \quad (4.65)$$

which gives

$$r_3 = \frac{r_2^2}{r_1}. \quad (4.66)$$

After the first two iterations, CONVERGE begins to estimate the residual from Equation 4.66 for up to $N-1$ consecutive iterations, recalculating the exact residual from Equation 4.58 every N iterations ($N = 10$ for the pressure equation and $N = 4$ for the energy and species equations). If the estimated residual for a given iteration meets the convergence criteria, or if the SOR solver has reached the maximum number of iterations ([solver.in > Transport > \[variable\] > itmax](#)), CONVERGE calculates the residual from Equation 4.58 for that iteration. In the [log file](#), CONVERGE prints "not calculated" for the residual value for each iteration at which the residual is estimated from Equation 4.66.

CONVERGE BiCGSTAB Method

CONVERGE offers a native implementation of the biconjugate gradient stabilized (BiCGSTAB) method, a member of the conjugate gradient family of linear solvers. As originally formulated ([Hestenes and Stiefel, 1952](#)), the conjugate gradient (CG) method can be thought of as an iterative direct solution method. CG was proven to exactly solve $Ax = b$ in no more than N iterations, where N is the dimension of the system and A is both symmetric and positive-definite.

Conjugate gradient has some conceptual similarities to, but is more sophisticated than, the simpler gradient descent method (also known as the method of steepest descent). Both methods consider the gradient in the working solution x^* at each iteration. By construction, the residuals (*i.e.*, the change in the solution between each iteration) in CG are all mutually orthogonal in N -space, guaranteeing convergence in N iterations. CG is equal or superior to gradient descent in all cases, and the superiority is particularly pronounced for stiff, ill-conditioned systems (*i.e.*, large condition numbers).

In practice, the CG method is often implemented as an iterative method, and modern formulations are generalized and improved. The biconjugate gradient method, or BiCG ([Fletcher, 1976](#)), does not require that A be symmetric. The biconjugent gradient stabilized method ([Van der Vorst, 1992](#)) is a further improvement which converges faster and more smoothly than BiCG. It is this final formulation which is available in CONVERGE.

Chapter 4: Numerics

Iterative Linear Solvers CONVERGE BiCGSTAB Method

To solve $Ax = b$, BiCGSTAB starts with an initial guess x_0 (in CONVERGE, this is the value of the solution vector at the last time-step). BiCGSTAB calculates the initial residual r_0 according to

$$r_0 = b - Ax. \quad (4.67)$$

Next, the algorithm chooses a vector r_0' such that

$$r_0 \cdot r_0' \neq 0. \quad (4.68)$$

Theoretically, r_0' can be any vector with a non-zero inner product with r_0 , but in practice CONVERGE takes the common approach of setting $r_0' = r_0$.

BiCGSTAB then iterates on the index j , with an intermediate break to check for convergence to within a specified criterion. Note that j is not a summation index. In sequence, BiCGSTAB calculates a_j , s_j and ω_j as

$$\alpha_j = \frac{r_j \cdot r_j'}{(Ap_j) \cdot r_0'}, \quad (4.69)$$

$$s_j = r_j - \alpha_j Ap_j, \quad (4.70)$$

and

$$\omega_j = \frac{(As_j) \cdot s_j}{(As_j) \cdot (As_j)}. \quad (4.71)$$

Next, BiCGSTAB calculates the next iterated solution x_{j+1} ,

Chapter 4: Numerics

Iterative Linear Solvers CONVERGE BiCGSTAB Method

$$x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j, \quad (4.72)$$

and calculates the new residual r_{j+1} ,

$$r_{j+1} = s_j - \omega_j A s_j. \quad (4.73)$$

If the new residual has a magnitude less than the convergence criterion ε_0 ,

$$\|r_{j+1}\| \leq \varepsilon_0, \quad (4.74)$$

then the BiCGSTAB iterations end, and x_{j+1} is taken as the solution for the next time-step. Otherwise, the new residual has not converged sufficiently, and BiCGSTAB successively calculates β_j and p_{j+1} ,

$$\beta_j = \left(\frac{\alpha_j}{\omega_j} \right) \times \frac{(r_{j+1} \cdot r'_0)}{(r_j \cdot r'_0)} \quad (4.75)$$

and

$$p_{j+1} = r_{j+1} + \beta_j (p_j - \omega_j A p_j), \quad (4.76)$$

and BiCGSTAB proceeds to the next iteration.

Hybrid CONVERGE BiCGSTAB

For calculating wall distance, CONVERGE offers a hybrid iterative linear solver option. If you set *Transport > wall_dist > solver_type = HYBRID* in [*solver.in*](#), CONVERGE uses the CONVERGE BiCGSTAB solver for the first time-step and uses the SOR solver for subsequent time-steps.

Preconditioners

When using the CONVERGE BiCGSTAB solver (*i.e.*, when `Transport > * > solver_type = CONVERGE_BICGSTAB` in [solver.in](#)), you can apply a [preconditioner](#) in order to improve the speed of convergence for certain equations. Preconditioners modify the linear system in order to reduce the condition number of the system.

GPU Acceleration

A GPU implementation of the CONVERGE BiCGSTAB solver is available for the pressure equation. Using massively parallel GPU resources can speed up certain types of simulations, particularly those with a large number of computational cells and a large scaling factor for the convergence tolerance ([solver.in](#) > `PISO > tol_scale` or [solver.in](#) > `SIMPLE > tol_scale`). To use the GPU-enabled version of CONVERGE BiCGSTAB, set [inputs.in](#) > `accelerator_control > use_gpu_flag = 1` and [solver.in](#) > `Transport > pres > solver_type = CONVERGE_BICGSTAB_GPU`.

HYPRE BiCGSTAB Method

In addition to the implementation described above, CONVERGE offers the High Performance Preconditioners (HYPRE) implementation of the BiCGSTAB method. HYPRE is a linear solver library developed by Lawrence Livermore National Laboratory. Refer to the [HYPRE 2.9 Reference](#) for additional information on this solver.

Note that the HYPRE BiCGSTAB solver is not available for a particular governing equation if you activate the [MUSCL scheme](#) for that equation. HYPRE BiCGSTAB is not available for turbulence transport equations, the radiation equation, or for wall distance calculations.

Preconditioners

When using the HYPRE BiCGSTAB solver (*i.e.*, when `Transport > * > solver_type = HYPRE_BICGSTAB` in [solver.in](#)), you can apply a [preconditioner](#) in order to improve the speed of convergence for certain equations. Preconditioners modify the linear system in order to reduce the condition number of the system.

Preconditioners

When using the [CONVERGE](#) or [HYPRE BiCGSTAB](#) solvers (*i.e.*, when `Transport > * > solver_type = HYPRE_BICGSTAB` or `CONVERGE_BICGSTAB` in [solver.in](#)), you can apply a preconditioner in order to improve the rate of convergence for certain equations. If you specify a preconditioner, instead of solving the linear system $Ax = b$, CONVERGE will solve the system

$$M^{-1}Ax = M^{-1}b, \quad (4.77)$$

where we use a preconditioner to form the matrix M^{-1} such that the new system has a reduced condition number.

Preconditioners are themselves computationally expensive, and you should generally select a preconditioner only if BiCGSTAB requires a large number of iterations to converge.

CONVERGE offers four general types of preconditioner: Euclid/ILU, algebraic multigrid (AMG), the SOR algorithm described above, and a modified symmetric SOR (SSOR). The availability of each preconditioner depends on which equation is being solved and which BiCGSTAB solver is in use. Refer to the description of [solver.in](#) to see which preconditioners are available for each equation.

The ILU and AMG preconditioners are formulated differently for the CONVERGE and HYPRE solvers. The CONVERGE formulations are given below. Refer to the [HYPRE 2.9 Reference Manual](#) for additional information on the HYPRE formulations.

ILU Preconditioner

The ILU preconditioner implemented in CONVERGE is the incomplete LU factorization with no fill-in (ILU0), which forms the matrix M as the product of a sparse lower triangular matrix L and a sparse upper triangular matrix U :

$$M = LU. \quad (4.78)$$

The matrices L and U are computed such that the residual R , given by

$$R = LU - A, \quad (4.79)$$

has zeroes in locations corresponding to the non-zero entries of A ([Saad, 2003](#)).

AMG Preconditioner

The AMG preconditioner is a type of multigrid preconditioner which is primarily designed for elliptic equations. Conceptually, a multigrid method solves a global system on a series of representations (*e.g.*, grids) of various resolution. Information propagates throughout the domain on the coarser grids, and the local structure of the solution is calculated on the finer grids. Geometric multigrid approaches solve a hierarchical set of grids, and are thus limited to completely structured domains. CONVERGE uses an algebraic multigrid (AMG) approach, which aggregates cells on the fluid solver grid into increasingly larger sets to perform the coarsening operation.

A multigrid preconditioner follows a cycle of restriction (coarsening the domain and re-solving) and interpolation (refining the domain and re-solving). The most basic V-cycle performs an iteration on the finest grid (or aggregated representation), restricts the system, and iterates again. It successively restricts and iterates to the coarsest grid, then successively interpolates step by step to the original grid. More complex cycles, such as the W-cycle and the K-cycle, move between levels in a more complex pattern.

The AMG preconditioner can perform very well in some cases, but performance is strongly affected by the number of successive coarsening levels, the number of cycles, and the cycle shape. These parameters are highly case-dependent. Contact the Convergent Science Applications team for assistance.

SOR Preconditioner

The SOR preconditioner performs several iterations of the [SOR iterative linear solver](#), then solves with the selected BiCGSTAB method. To form the matrix M , the matrix A is first split into upper, lower, and diagonal matrices,

$$A = L_A + D_A + U_A, \quad (4.80)$$

where the diagonal matrix D_A has all non-diagonal elements set to zero, the lower matrix L_A has all elements on and above the diagonal set to zero, and the upper matrix U_A has all elements on and below the diagonal set to zero. Effectively, SOR calculates M as

$$M = \frac{1}{\omega} (L_A + \omega D_A). \quad (4.81)$$

CONVERGE uses an under-relaxation factor ω of 1.3 and performs three iterations. The SOR preconditioner is relatively inexpensive and performs well in many applications.

SSOR Preconditioner

The modified symmetric SOR (SSOR) preconditioner uses the same decomposition of A as the SOR preconditioner, which is shown above in Equation 4.80. SSOR calculates the matrix M as

$$M = (I + \omega L_A D_A^{-1}) \frac{1}{\omega(2-\omega)} D_A (I + \omega D_A^{-1} U_A), \quad (4.82)$$

where I is the identity matrix. CONVERGE uses an under-relaxation factor ω of 1.3 and performs three iterations, where each iteration consists of one forward Gauss-Seidel iteration and one backward Gauss-Seidel iteration.

GPU-Enabled Solvers

CONVERGE offers GPU implementations of several iterative linear solvers, as detailed below. Using massively parallel GPU resources can speed up certain types of simulations. The cases that benefit most from GPU acceleration are those with a large number of computational cells and a large scaling factor for the convergence tolerance ([solver.in](#) > *PISO* > *tol_scale* or *SIMPLE* > *tol_scale*).

Jacobi Method

A GPU implementation of the Jacobi method is available for the pressure equation. To use this option, set [inputs.in](#) > *accelerator_control* > *use_gpu_flag* = 1 and [solver.in](#) > *Transport* > *pres* > *solver_type* = CONVERGE_JACOBI_GPU.

Reduced SOR Method

The reduced SOR (RSOR) method is a modified version of the [SOR](#) method that involves only matrix vector multiplication ([Xu and Hao, 2020](#)). A GPU implementation of RSOR is available for the pressure equation. To use this option, set [inputs.in](#) > *accelerator_control* > *use_gpu_flag* = 1 and [solver.in](#) > *Transport* > *pres* > *solver_type* = CONVERGE_RSOR_GPU.

CONVERGE BiCGSTAB Method

A GPU-enabled version of the [CONVERGE BiCGSTAB](#) method is available for the pressure equation. To use this option, set [inputs.in](#) > *accelerator_control* > *use_gpu_flag* = 1 and [solver.in](#) > *Transport* > *pres* > *solver_type* = CONVERGE_BICGSTAB_GPU.

When the CONVERGE BiCGSTAB method requires a large number of iterations, you can use the RSOR method as a [preconditioner](#) to improve the rate of convergence ([solver.in](#) > *Transport* > *pres* > *preconditioner* = RSOR_GPU).

AmgX Linear Solver Library

CONVERGE offers integrated support for NVIDIA's [AmgX linear solver library](#), which provides open-source GPU implementations of distributed algebraic multi-grid (AMG) methods and Krylov solvers, along with numerous preconditioners. The AmgX library is based on the CUDA development platform and is available only for machines with CUDA-enabled GPUs. Note that there are two kinds of AMG, classical and aggregation, and CONVERGE supports only aggregation AMG.

For optimal performance, the ratio of processing cores to GPUs on each compute node should be 1:1. If the ratio is higher, meaning that multiple MPI processes will be mapped to each GPU, we recommend enabling CUDA's [Multi-Process Service \(MPS\)](#) to ensure efficient utilization of GPU resources.

The AmgX library is available for the pressure equation and requires setup in [solver.in](#) and [inputs.in](#). In [solver.in](#), set *transport* > *pres* > *solver_type* = AMGX. When *solver_type* is set to this

value, CONVERGE ignores the other parameters in the *transport > pres* settings block. In [*inputs.in*](#), set *accelerator_control > use_gpu_flag = 1* and enter the name of the AmgX configuration (*.json) file in *accelerator_control > amgx_config_filename*. Include this configuration file in your case setup. A number of AmgX configuration files are available for selection in CONVERGE Studio (*Case Setup > Simulation Parameters > Run parameters > GPU/Acceleration*). For general-purpose use, we recommend that you use these files without modification, but you can edit them if necessary. Refer to AmgX documentation for information about the configuration options.

Some applications may run more efficiently if you use pinned host (CPU) memory, which increases the bandwidth for CPU-GPU data transfers. To enable this option, set [*inputs.in > accelerator_control > enable_pin_mem = 1*](#). This option is required for overlapping CUDA kernel execution and data transfers in different CUDA streams.

Nonlinear Krylov Acceleration Method

The nonlinear Krylov-based acceleration (NKA) scheme uses the generalized minimal residual (GMRES) method to accelerate an existing fixed-point iteration scheme. The NKA scheme is available only for the [electric potential equation](#) and helps to speed up convergence for simulations that involve spatial and temporal changes in boundary conditions.

The general form for a nonlinear system of equations can be written as

$$A(x)x = b(x), \quad (4.83)$$

where the matrix A and the vector b are both functions of the solution vector x . In a fixed-point iteration scheme, the solution at iteration $k+1$ is given by

$$x^{k+1} = x^k - f(x^k), \quad (4.84)$$

where the update $f(x)$ is calculated from the solution from the previous iteration. The form of $f(x)$ depends on the choice of fixed-point iteration scheme. The CONVERGE implementation of NKA, described below, uses the [SOR algorithm](#) as the fixed-point scheme.

The iterative procedure for the NKA solver can be summarized as follows:

Chapter 4: Numerics

Iterative Linear Solvers Nonlinear Krylov Acceleration Method

```

for k = 1, 2, ..., n
     $x_{k+1}^*$  = SORP( $x_k$ )
     $f_k$  =  $x_{k+1}^*$  -  $x_k$ 
     $v_{k+1}$  = NKA( $f_k$ )
     $x_{k+1}$  =  $x_k$  +  $v_{k+1}$ 
end

```

(4.85)

SOR^P indicates P iterations of SOR, which is used as a preconditioner to calculate an initial update f . The NKA solver takes f as an input to calculate the accelerated update v , given by

$$v^{k+1} = \sum_{i=k-M+1}^k z_i^k v^i + f(x^k) - \sum_{i=k-M+1}^k z_i^k w^i. \quad (4.86)$$

To evaluate this expression, CONVERGE stores the subspace vectors v and w for the previous M iterations ($M \ll n$), where

$$v^i = x^{i-1} - x^i \quad (4.87)$$

and

$$w^i = f(x^{i-1}) - f(x^i). \quad (4.88)$$

The coefficients z_i^k are the components of the vector z^k that solves the minimization problem

$$z^k = \arg \min_{y \in \mathbb{R}^M} \left\| f(x^k) - \sum_{i=k-M+1}^k y_i w^i \right\|_2. \quad (4.89)$$

The NKA solver uses the solution history of the last M iterations to accelerate the convergence of the fixed-point scheme (in this case, SOR). The value of M must be large enough to accelerate convergence, but not so large that it reduces the computational efficiency of the solver. In CONVERGE, M is set to 10 and P (the number of SOR iterations) is set to 20.

Chapter 4: Numerics

Iterative Linear Solvers Nonlinear Krylov Acceleration Method

To use the NKA solver, set `solver.in > Transport > electric_potential > solver_type = NONLINEAR_KRYLOV`.

Chapter



5

Time Control Methods

5 Time Control Methods

This chapter describes CONVERGE's [transient](#) and [steady-state](#) solvers, [time-step control](#), as well as two specific time-control methods: [super-cycling](#) and the [fixed flow method](#).

5.1 Solvers

There are two types of solvers available in CONVERGE: [transient](#) and [steady-state](#). The following subsections describe the different solvers and include recommendations for parameters in [solver.in](#) and [inputs.in](#). CONVERGE Studio can automatically set recommended parameters for different types of cases (*e.g.*, compressible gas transient case, incompressible gas steady-state case).

Transient Solver

In order to solve a transient case, the governing equations described earlier are approximated using numerical techniques. CONVERGE internally generates a grid, which is then used to discretize the governing equations. By discretizing the domain via a grid, the governing equations can be approximated by a system of algebraic equations. If solving implicitly, an iterative (multi-step) technique will be required for the system of algebraic equations. To activate the transient solver, set [inputs.in > solver_control > steady_solver = 0](#). If [inputs.in > simulation_control > crank_flag = 1 or 2](#), CONVERGE requires [steady_solver = 0](#).

CONVERGE offers three methods for time advancement of a transient case. Select first-order explicit Euler by setting [solver.in > Numerical_Schemes > implicit_fraction = 0](#). Select first-order implicit Euler by setting [implicit_fraction = 1](#). Implicit Euler is more computationally costly per time-step than explicit Euler, but it is stable at much larger time-steps. Select second-order Crank-Nicolson by setting [implicit_fraction = 0.5](#). Crank-Nicolson is more accurate than implicit Euler, but it is not stable at such large time-steps. Crank-Nicolson is available for only the momentum equation. If you solve the momentum equation with Crank-Nicolson, CONVERGE will solve the other equations with implicit Euler.

If solving with first-order explicit Euler time advancement, only a single iteration will be required to solve the algebraic equations.

When solving the equations with first-order implicit Euler or Crank-Nicolson time advancement, a convergence criterion (or tolerance) is required. A solution is considered converged when the iteration error in the solution is at or below the user-specified convergence criterion. In CONVERGE, the iteration error is related to the change in the solution field from each iteration, $\Delta\phi$, which is given by

$$\text{error} = \frac{\Delta\phi}{\text{normalization}}. \quad (5.1)$$

Chapter 5: Time Control Methods

Solvers Transient Solver

The following table summarizes the tolerance parameter and normalization for each equation.

Chapter 5: Time Control Methods

Solvers Transient Solver

Table 5.1: Tolerance parameters and normalizations for a transient simulation.

Equation	Parameter name	Typical value (inputs.in > solver_control > steady_solver = 0)	Normalization
Momentum	<i>Transport > mom > tol</i>	1.0e-4	Velocity or 1.0, whichever is larger.
Pressure	<i>Transport > pres > tol</i>	1.0e-8	Pressure or 1.0, whichever is larger.
Density	<i>Transport > density > tol</i>	1.0e-4	Density or 1.0, whichever is larger.
Energy	<i>Transport > energy > tol</i>	1.0e-4	Internal energy or 1.0, whichever is larger.
Species	<i>Transport > species > tol</i>	1.0e-4	1.0
Passive	<i>Transport > passive > tol</i>	1.0e-4	Passive or 1.0, whichever is larger.
Radiation	<i>Transport > radiation > tol</i>	1.0e-8	Cell intensity.
Tke	<i>Transport > tke > tol</i>	1.0e-3	Cell tke.
Eps	<i>Transport > eps > tol</i>	1.0e-3	Cell eps.
Omega	<i>Transport > omega > tol</i>	1.0e-3	Cell omega.
Velocity variance scale (v^2)	<i>Transport > v2 > tol</i>	1.0e-3	Cell velocity variance scale.
Elliptic relaxation function (f)	<i>Transport > f > tol</i>	1.0e-3	Cell elliptic relaxation function.
Velocity scale ratio (ζ)	<i>Transport > zeta > tol</i>	1.0e-3	Cell velocity scale ratio.
Wall distance	<i>Transport > wall_dist > tol</i>	1.0e-6	Cell wall distance.
Electric potential	<i>Transport > electric_potential > tol</i>	1.0e-6	Cell electric potential.

Each tolerance parameter can be a fixed value or can [vary temporally](#). For a temporally varying parameter, specify a file name instead of a value in [inputs.in](#) and include that file in your case setup.

Table 5.2 below lists typical PISO parameter values for a transient simulation.

Chapter 5: Time Control Methods

Solvers Transient Solver

Table 5.2: PISO parameters (all in [solver.in](#)).

Parameter name	Typical value (inputs.in > solver_control > steady_solver = 0)
Numerical_Schemes > strict_conserve_level	For engine cases: 1 for transient cases with inputs.in > temporal_control > max_cfl_u > 2.5, 2 for transient cases with max_cfl_u ≤ 2.5.
PISO > tol_scale	20
PISO > piso_tol	1e-3
PISO > piso_itmin	2
PISO > piso_itmax	9
PISO > omega_presrat	0.7 - 1.0
Transport > pres > itmax	200

Table 5.3 below lists typical SIMPLE parameter values for a transient simulation.

Table 5.3: SIMPLE parameters (all in [solver.in](#)).

Parameter name	Typical value (inputs.in > solver_control > steady_solver = 0)
Numerical_Schemes > strict_conserve_level	For engine cases: 1 for transient cases with inputs.in > temporal_control > max_cfl_u > 2.5, 2 for transient cases with max_cfl_u ≤ 2.5.
SIMPLE > tol_scale	20
SIMPLE > simple_tol	1e-3
SIMPLE > simple_itmin	2
SIMPLE > simple_itmax	9
SIMPLE > omega_presrat	0.7 - 1.0
SIMPLE > omega_simple	0.5
Transport > pres > itmax	200

Table 5.4 below lists typical CFL parameter values for a transient simulation.

Table 5.4: CFL number parameters (all in [inputs.in](#)).

Parameter name	Typical value (inputs.in > solver_control > steady_solver = 0)
temporal_control > max_cfl_u	0.5 - 3.0 (in VOF cases, 0.03 to 0.10).

Chapter 5: Time Control Methods

Solvers Transient Solver

Parameter name	Typical value (inputs.in > solver_control > steady_solver = 0)
<i>temporal_control</i>	0.5 - 2.5 > <i>max_cfl_nu</i>
<i>temporal_control</i>	0.5 - 100.0 > <i>max_cfl_mach</i>

Table 5.5 below lists typical time-step control parameter values for a transient simulation.

Table 5.5: Selected time-step control parameters in [inputs.in](#). Other time-step control parameters are simulation-dependent.

Parameter name	Typical value (inputs.in > solver_control > steady_solver = 0)
<i>temporal_control</i> > <i>mult_dt_parcel</i>	0.5 - 1.5
<i>temporal_control</i> > <i>mult_dt_coll_mesh</i>	0.5 - 1.5 (Similar to <i>mult_dt_parcel</i> but the collision mesh is also used to determine the maximum dt.)
<i>temporal_control</i> > <i>mult_dt_chem</i>	0.1 - 1.0
<i>temporal_control</i> > <i>mult_dt_source</i>	0.1 - 1.0
<i>temporal_control</i> > <i>mult_dt_move</i>	2.0

Steady-State Solver

The steady-state solver in CONVERGE allows faster solutions to steady-state problems in which an accurate time history of transient behavior is not desired. While it is possible to solve steady-state problems using the transient solver, this method is not efficient and not recommended. This section explains notable differences between the steady-state and transient solvers. To activate the steady-state solver, set [inputs.in](#) > solver_control > steady_solver = 1.

In a transient case, the general transport equation is

$$\frac{\partial \rho\phi}{\partial t} + \frac{\partial \rho u_i \phi}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D \frac{\partial \phi}{\partial x_i} \right) + S, \quad (5.2)$$

where ϕ is any transported quantity. For a steady-state case, by definition, the solution does not change with time. Thus the transport equation for a steady-state case contains no time derivative term, as follows:

$$\frac{\partial \rho u_i \phi}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D \frac{\partial \phi}{\partial x_i} \right) + S. \quad (5.3)$$

Although the transport equation does not include the time derivative term, CONVERGE uses a pseudo time-step to aid in numerically solving Equation 5.3. Because the steady-state solver uses a pseudo time-step, this solver works in a conceptually similar manner to [the transient solver](#). The steady-state solver, however, is not required to be time-accurate while proceeding in pseudo time to a steady solution, so the rate of convergence is significantly improved when compared to the transient solver. The maximum convection CFL number (described previously, [inputs.in > temporal_control > max_cfl_u](#)) dictates the size of the pseudo time-step for each cycle. A larger convection CFL number decreases the number of cycles required by the momentum solver to attain a steady-state value. Larger local pseudo time-steps permit the momentum information to traverse the domain much more quickly. The steady-state solver also uses looser solution tolerances ([solver.in > Steady_state_solver > steady_piso_tol_init](#) and [solver.in > Steady_state_solver > steady_tol_scale_init](#)) than the transient solver (e.g., [solver.in > PISO > piso_tol](#) and [PISO > tol_scale](#)).

Because the steady-state solver does not include time terms, specify time-based inputs in cycles instead of seconds (remember that a cycle is the completion of the [solution procedure algorithm](#)). For example, replace the start time ([inputs.in > simulation_control > start_time](#)) with the starting cycle number and end time ([inputs.in > simulation_control > end_time](#)) with the ending cycle number. Parameters that directly control the pseudo time-step (e.g., [temporal_control > dt_start](#), [temporal_control > dt_min](#), [temporal_control > dt_max](#)) are still in seconds. Typically, to reach steady-state convergence, CONVERGE will perform hundreds of cycles.

We generally recommend using the [pressure-based](#) formulation of the [SIMPLE algorithm](#) for steady-state simulations. These and other recommended settings are summarized below in Table 5.6.

Steady-State Solver Procedure

Steady-state CFD simulations typically reach a statistically stationary state after a brief initial transient phase. Generally, the fidelity of the final steady-state solution is independent of the initial transients that are present, so the grid used for the initial transient phase can have a lower resolution than the grid used for the final solution.

The steady-state solver works in conjunction with [grid scaling](#) to reach the final steady-state solution as efficiently as possible. The simulation begins with a coarse grid (the initial grid scaling value specified in [gridscale.in](#)), which allows the flow to propagate quickly through the domain. CONVERGE continuously monitors the solution variables that you designate in the [steady-state monitor](#) until the mean and standard deviation of each variable drop to an acceptably steady level. At this point, CONVERGE considers the solution locally converged and proceeds to the next grid scaling value. This process is repeated until the final grid scaling value is applied.

Chapter 5: Time Control Methods

Solvers Steady-State Solver

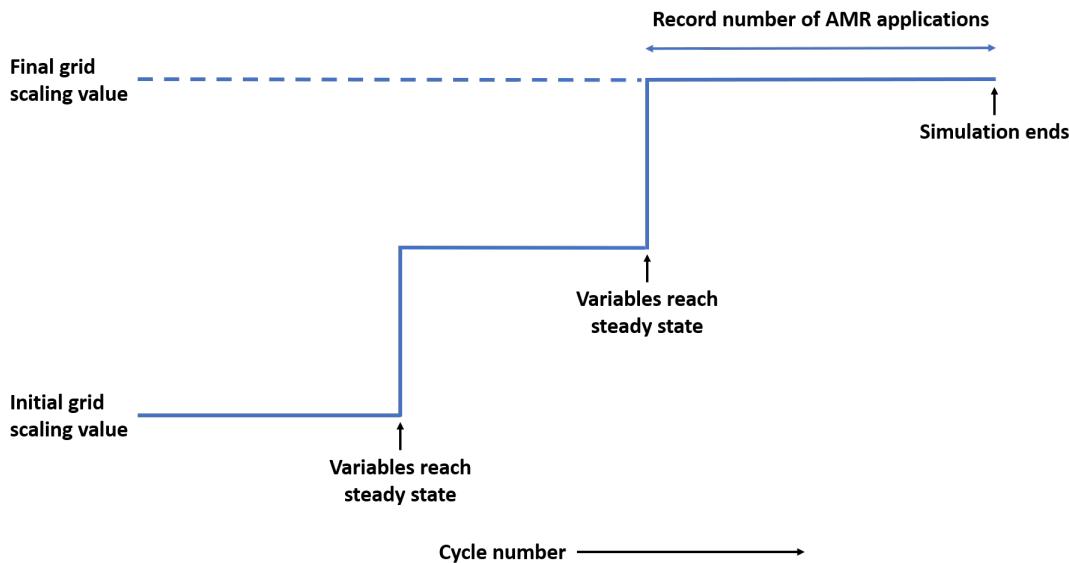


Figure 5.1: Steady-state solver procedure with `steady_max_cfl_u_final = NOT_USED` (recommended).

To maintain stability, CONVERGE begins each grid scaling stage with a maximum convection CFL number of 1 and doubles this value at the frequency you specify ([solver.in](#) > *Steady_state_solver* > *steady_tol_update_freq*) until reaching [inputs.in](#) > *temporal_control* > *max_cfl_u*. If the iterative algorithm or any transport equations fail to converge, the maximum CFL number is halved. After the number of cycles given by *steady_tol_update_freq* pass, the CFL number doubling procedure resumes. Additionally, CONVERGE attempts to modify relaxation parameters in [solver.in](#) (e.g., *PISO* > *omega_presrat*, *Transport* > *tke* > *omega*, *Transport* > *eps* > *omega*) to ensure a smooth and stable path toward the steady-state solution.

At the final grid scaling stage, if [solver.in](#) > *Steady_state_solver* > *steady_switch_solver_flag* = 1, CONVERGE will switch the pressure solver between BiCGSTAB and SOR to determine which method is faster and use the faster solver for the remainder of the simulation. If there is a recovery within 100 cycles of the pressure solver switch, CONVERGE will revert the pressure solver to its original setting.

You can optionally define a final maximum convection CFL number in [solver.in](#) > *Steady_state_solver* > *steady_max_cfl_u_final*, such that *steady_max_cfl_u_final* < *max_cfl_u*. If you use this option, CONVERGE reduces the maximum convection CFL number to *steady_max_cfl_u_final* when the solution variables first reach a steady state at the final grid scaling value, as shown below in Figure 5.2. Additionally, CONVERGE tightens the solution tolerances to the values specified in the [solver.in](#) > *SIMPLE* or [solver.in](#) > *PISO* settings block. We do not recommend using this option because it extends the simulation runtime and usually does not improve the accuracy of the solution.

Chapter 5: Time Control Methods

Solvers Steady-State Solver

By default, `steady_max_cfl_u_final = NOT_USED`, which corresponds to the recommended procedure shown above in Figure 5.1. With this default setting, the maximum convection CFL number from `inputs.in > temporal_control > max_cfl_u` is used for all stages of the simulation including the final grid scaling stage, as are the steady-state solution tolerances from `solver.in > Steady_state_solver > steady_piso_tol_init` and `Steady_state_solver > steady_tol_scale_init`. (Note that `steady_piso_tol_init` applies to all steady-state simulations, regardless of whether the SIMPLE or PISO algorithm is in use.)

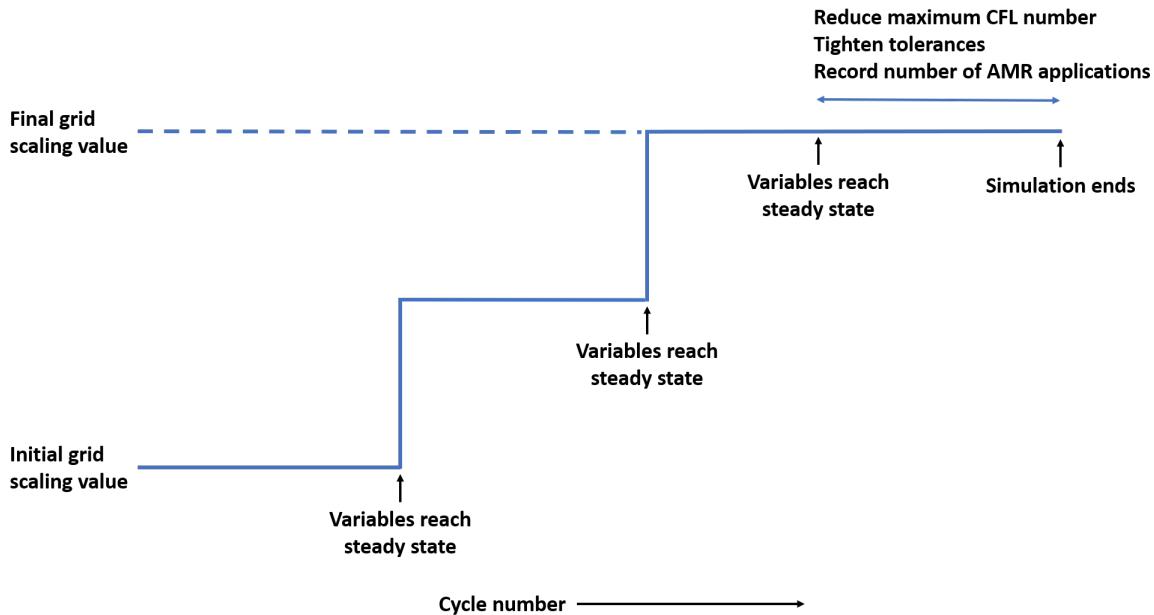


Figure 5.2: Steady-state solver procedure with `steady_max_cfl_u_final` set to a value lower than `max_cfl_u` (not recommended).

Adaptive Mesh Refinement

To control [Adaptive Mesh Refinement](#) (AMR) in conjunction with the steady-state solver, specify `GRIDSCALE`, `PERMANENT`, or `SEQUENTIAL` for the temporal type in `amr.in`. If you select `GRIDSCALE`, CONVERGE activates AMR when the simulation reaches the grid scaling value specified in `amr.in > amr_group > amr_[type] > starting_gridscale`. In this case, CONVERGE ignores the time to end AMR (`amr_[type] > end_time`) and chooses to deactivate AMR based on other criteria.

The solver records the number of times AMR is applied during the final grid scaling stage. If `steady_max_cfl_u_final = NOT_USED`, this starts at the beginning of the final grid scaling stage (see Figure 5.1). If `steady_max_cfl_u_final` is set to a number, this starts when the solution variables first reach a steady state at the final grid scaling stage (see Figure 5.2). The number of applications of AMR determines one of the stopping criteria for the steady-state solver, as explained below.

Stopping Criteria for the Steady-State Solver

The steady-state solver will stop before `inputs.in > simulation_control > end_time` if all of the following criteria are satisfied:

1. The solution variables reach a steady state at the final grid scaling value, the final solver tolerances (if applicable), and the final maximum convection CFL number (if applicable).
2. If AMR is active, the number of AMR applications during the final grid scaling stage (calculated as described above) equals or exceeds the value specified in `solver.in > Steady_state_solver > steady_min_num_amr`.
3. If AMR is active, the change in total cell count from one AMR application to the next is less than 10% of the total cell count.

Steady-State Residuals

For a given variable ϕ , the discretized transport equation can be written as

$$a_p \phi_p = \sum_{nb} a_{nb} \phi_{nb} + b, \quad (5.4)$$

where the sum is over the neighboring cells of cell p . The residual of ϕ at any given cycle is

$$R^\phi = \sum_p \left| -a_p \phi_p + \sum_{nb} a_{nb} \phi_{nb} + b \right|. \quad (5.5)$$

To track the progress of residuals during a steady-state simulation, set `solver.in > Steady_state_solver > steady_residual_output_flag = 1`. When `steady_residual_output_flag = 1`, CONVERGE writes the relative residuals for the mass, momentum, energy, and species equations to `residuals.out`. The relative residual RR^ϕ is defined as

$$RR^\phi = \frac{R^\phi}{R_{m50}^\phi}, \quad (5.6)$$

where R_{m50}^ϕ is the maximum residual for the first 50 iterations (for momentum, R_{m50}^ϕ is the maximum residual for the first 50 iterations for all three momentum equations).

Generally, the residuals in a steady-state simulation should approach zero as the simulation progresses. However, changes to the grid can cause brief increases in the residuals, after which the residuals continue to fall, as shown below in Figure 5.3. In this example, the residual for the x component of momentum has a large spike at $ncyc = 500$ corresponding to a grid scaling event, followed by a series of smaller spikes that correspond to applications of AMR.

Chapter 5: Time Control Methods

Solvers Steady-State Solver

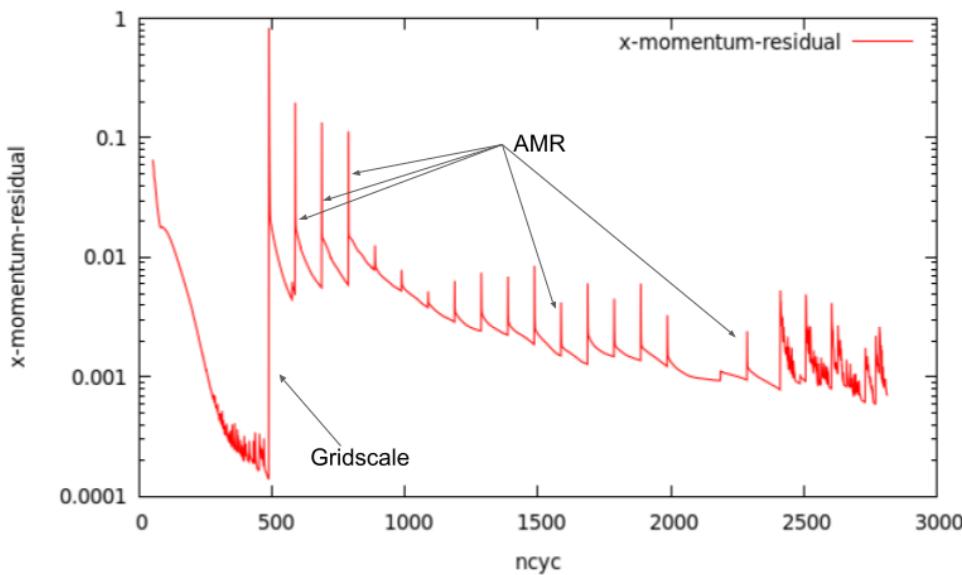


Figure 5.3: An example of relative residuals for the x component of momentum, plotted as a function of the number of cycles ncyc.

It is important to use a fully upwinded numerical scheme to ensure that the residuals fall to an acceptably low level. The required settings are noted with an asterisk in Table 5.6 below. If you do not use these settings, it is possible that the residuals will not approach zero at the final grid scaling value even when the steady-state convergence criteria have been satisfied.

Recommended Inputs

No single set of input parameters can be used for all cases, but this section provides some guidelines that can be used when setting up a new steady-state case. Table 5.6 summarizes our general solver recommendations.

Table 5.6: Recommended solver-related parameter values for steady-state cases.

Parameter name	File name	Typical value
<code>ns_solver_scheme</code>	<code>solver.in</code>	<code>SIMPLE</code>
<code>ns_solver_type</code>	<code>solver.in</code>	<code>PRESSURE_BASED</code>
<code>SIMPLE > simple_itmin</code> or <code>PISO > piso_itmin</code>	<code>solver.in</code>	2
<code>SIMPLE > simple_itmax</code> or <code>PISO > piso_itmax</code>	<code>solver.in</code>	20
<code>Steady_state_solver ></code> <code>steady_auto_flag</code>	<code>solver.in</code>	0

Chapter 5: Time Control Methods

Solvers Steady-State Solver

Parameter name	File name	Typical value
<i>Steady_state_solver > max_cfl_u_final</i>	<i>solver.in</i>	<i>NOT_USED</i>
<i>Steady_state_solver > steady_residual_output_flag</i>	<i>solver.in</i>	1
<i>Steady_state_solver > steady_piso_tol_init</i>	<i>solver.in</i>	10 times the value for a transient simulation (<i>solver.in > SIMPLE > simple_tol</i> or <i>PISO > piso_tol</i>)
<i>Steady_state_solver > steady_tol_scale_init</i>	<i>solver.in</i>	0.5 times the value for a transient simulation (<i>solver.in > SIMPLE > tol_scale</i> or <i>PISO > tol_scale</i>)
<i>Numerical_Schemes > flux_scheme > mom*</i>	<i>solver.in</i>	FLUX_BLENDING
<i>Numerical_Schemes > flux_scheme > global*</i>	<i>solver.in</i>	FLUX_BLENDING
<i>Numerical_Schemes > flux_scheme > turb*</i>	<i>solver.in</i>	FLUX_BLENDING
<i>Numerical_Schemes > fv_upwind_factor > global*</i>	<i>solver.in</i>	1.0
<i>Numerical_Schemes > fv_upwind_factor > mom*</i>	<i>solver.in</i>	1.0
<i>Numerical_Schemes > fv_upwind_factor > turb*</i>	<i>solver.in</i>	1.0
<i>Numerical_Schemes > conserve*</i>	<i>solver.in</i>	0
<i>Numerical_Schemes > strict_conserve_flag*</i>	<i>solver.in</i>	0
<i>temporal_control > dt_start</i>	<i>inputs.in</i>	This value should be set such that the CFL numbers are equal to or less than the maximum CFL numbers provided in the input.
<i>temporal_control > dt_max</i>	<i>inputs.in</i>	Typically this is a large number so that the pseudo time-step is not limited by <i>dt_max</i> .
<i>temporal_control > max_cfl_u</i>	<i>inputs.in</i>	The maximum convection CFL number. A smaller <i>cfl_u</i> may be more stable, but may take longer to converge. 20.0 - 40.0: For non-reacting flows, 10.0 - 20.0: For reacting flows.
<i>temporal_control > max_cfl_mach</i>	<i>inputs.in</i>	50.0 times <i>inputs.in > temporal_control > max_cfl_u</i> . (A smaller value may be more stable, but may take longer to converge.)
<i>temporal_control > max_cfl_nu</i>	<i>inputs.in</i>	5.0 - 10.0 times <i>inputs.in > temporal_control > max_cfl_u</i> . (A smaller value may be more stable, but may take longer to converge.)

Chapter 5: Time Control Methods

Solvers Steady-State Solver

Parameter name	File name	Typical value
<i>property_control > max_visc</i>	<i>inputs.in</i>	10.0
<i>grid_control > grid_scale</i>	<i>inputs.in</i>	<p>Integer or file name (e.g., gridscale.in). If you specify a file name (recommended), you must include that file in your case setup. Typically when running a steady-state simulation, start with a coarse mesh (gridscale.in > gridscale > value = -2 or -1) before refining the mesh (<i>value = 0</i>). Values in gridscale.in must be integers that increase monotonically throughout the simulation.</p> <p>We recommend using automatic grid scaling via gridscale.in > gridscale > time = AUTO.</p>
<i>temporal_control > mult_dt_parcel</i>	<i>inputs.in</i>	10.0 - 20.0
<i>temporal_control > mult_dt_chem</i>	<i>inputs.in</i>	0.1 - 1.0
<i>temporal_control > mult_dt_move</i>	<i>inputs.in</i>	2.0
<i>source > mult_dt_source</i>	<i>source.in</i>	0.1 - 1.0

*Required to drive residuals toward zero.

Table 5.7 below provides recommended minimum and maximum iteration numbers for the various transport equations solved during a steady-state simulation.

Table 5.7: Recommended iteration limits (in [solver.in](#)) for transport equations solved during a steady-state simulation.

Transport equation	Minimum iterations	Maximum iterations
Momentum	<i>Transport > mom > itmin = 2</i>	<i>Transport > mom > itmax = 500</i>
Pressure	<i>Transport > pres > itmin = 2</i>	<i>Transport > pres > itmax = 300</i>
Density	<i>Transport > density > itmin = 2</i>	<i>Transport > density > itmax = 20</i>
Energy	<i>Transport > energy > itmin = 2</i>	<i>Transport > energy > itmax = 20</i>
Species	<i>Transport > species > itmin = 2</i>	<i>Transport > species > itmax = 20</i>
Passive	<i>Transport > passive > itmin = 2</i>	<i>Transport > passive > itmax = 20</i>
Turbulent Kinetic Energy (tke)	<i>Transport > tke > itmin = 2</i>	<i>Transport > tke > itmax = 50</i>
Turbulent Dissipation (eps)	<i>Transport > eps > itmin = 2</i>	<i>Transport > eps > itmax = 50</i>
Specific Dissipation Rate (omega)	<i>Transport > omega > itmin = 2</i>	<i>Transport > omega > itmax = 50</i>

Chapter 5: Time Control Methods

Solvers Steady-State Solver

Transport equation	Minimum iterations	Maximum iterations
Velocity variance scale (v^2)	<i>Transport > v2 > itmin = 2</i>	<i>Transport > v2 > itmax = 50</i>
Elliptic relaxation function (f)	<i>Transport > f > itmin = 2</i>	<i>Transport > f > itmax = 100</i>
Velocity scale ratio (ζ)	<i>Transport > zeta > itmin = 2</i>	<i>Transport > zeta > itmax = 50</i>
Radiation	<i>Transport > radiation > itmin = 2</i>	<i>Transport > radiation > itmax = 2000</i>
Electric potential	<i>Transport > electric_potential > itmin = 2</i>	<i>Transport > electric_potential > itmax = 5000</i>

5.2 Time-Step Control

CONVERGE requires that you specify some method of time-step control. Very small time-steps are numerically stable, but they are computationally expensive. Larger time-steps allow CONVERGE to generate a solution more quickly, but larger time-steps will reduce accuracy, and too large a time-step will be unstable.

CONVERGE allows you to specify one of two methods for setting the time-step. You can specify a fixed time-step or a variable time-step that is calculated internally. For a fixed time-step, set `inputs.in > temporal_control > time_flag = 0` and set the time-step (`inputs.in > temporal_control > dt_fixed`) to the desired value. For a variable time-step, set `time_flag = 1` and provide additional inputs to guide the selection of the time-step. When choosing variable time-step control, the first time-step of the simulation is the value of `inputs.in > temporal_control > dt_start`. Subsequently, CONVERGE performs a number of checks and calculations to determine the size of the next time-step, and takes the largest time-step that satisfies all of the applicable limiters shown below in Table 5.8. Some limiters are applied only if certain solver features are activated (e.g., `dt_parcel` is applied only in a simulation with parcel modeling).

Table 5.8: Time-step limiters in CONVERGE.

Time-step limiter	Formulation	Comments
<code>dt_grow</code>	$dt \leq 1.25 \cdot dt_{prev}$	Every cycle, CONVERGE attempts to increase the time-step by 25% of the previous time-step dt_{prev} . No associated inputs.
<code>dt_min</code>	$dt \geq dt_{min}$	Set <code>inputs.in > temporal_control > dt_min</code> .
<code>dt_max</code>	$dt \leq dt_{max}$	Set <code>inputs.in > temporal_control > dt_max</code> .

Chapter 5: Time Control Methods

Time-Step Control

Time-step limiter	Formulation	Comments
<i>dt_cfl</i>	$\max_{cfl} u \geq u \frac{\Delta t}{\Delta x}$	Convection CFL. For a Cartesian cell, CONVERGE calculates the grid length Δx as $L=V/S$, where V is the volume of the cell and S is the face area projected along the x , y , or z axis (whichever is most restrictive). For an inlaid cell, the grid length is calculated by examining vectors from the cell center to the face center such that $\Delta x=2L$, and $\Delta x^2 = (\text{sum over each face})(V/S * \text{cell center to face center distance})$. Set inputs.in > temporal_control > max_cfl_u .
<i>dt_cflv</i>	$\max_{cfl} nu \geq v \frac{\Delta t}{\Delta x^2}$	Viscosity CFL. The grid length $\Delta x^2 = (\text{sum over all faces})(V/S)^2$. Set inputs.in > temporal_control > max_cfl_nu .
<i>dt_cflk</i>	$\max_{cfl} nu \geq \frac{v}{Pr} \frac{\Delta t}{\Delta x^2}$	Conduction CFL. The grid length $\Delta x^2 = (\text{sum over all faces})(V/S)^2$. Set inputs.in > temporal_control > max_cfl_nu . Note that Pr is the calculated cell value and not inputs.in > property_control > prandtl_turb .
<i>dt_cfld</i>	$\max_{cfl} nu \geq \frac{v}{Sc} \frac{\Delta t}{\Delta x^2}$	Mass diffusion CFL. The grid length $\Delta x^2 = (\text{sum over all faces})(V/S)^2$. Set inputs.in > temporal_control > max_cfl_nu . Note that Sc is the calculated cell value and not inputs.in > property_control > schmidt_turb .
<i>dt_mach</i>	$\max_{cfl} mach \geq c \frac{\Delta t}{\Delta x}$	Mach CFL based on the speed of sound c . The value of Δx is calculated for each cell face and the most restrictive value is applied to the cell. Set inputs.in > temporal_control > max_cfl_mach .
<i>dt_parcel</i>	$dt \leq \frac{\Delta x \cdot \text{mult_dt_parcel}}{\text{parcel_velocity}}$	Parcel time-step limit. Set inputs.in > temporal_control > mult_dt_parcel .

Chapter 5: Time Control Methods

Time-Step Control

Time-step limiter	Formulation	Comments
<i>dt_move</i>	$dt \leq \frac{\min_vol_ratio \cdot \Delta x \cdot \text{mult_}dt\text{_move}}{\text{surf_speed}}$	Surface motion time-step limit. Set inputs.in > temporal_control > mult_dt_move and inputs.in > grid_control > min_vol_ratio .
<i>dt_event</i>	N/A	Event time-step limit. Guarantees an event will occur at exactly the specified time. No associated inputs.
<i>dt_inject</i>	N/A	Injection time-step limit. Guarantees an injection will start and end at exactly the specified times. No associated inputs.
<i>dt_coll_mesh</i>	N/A	Collision mesh time-step limiter. The recommended setting for inputs.in > temporal_control > mult_dt_coll_mesh deactivates this legacy feature.
<i>dt_evap</i>	N/A	Drop evaporation time-step limiter. The recommended setting for inputs.in > temporal_control > mult_dt_evap deactivates this legacy feature.
<i>dt_chem</i>	$dt \leq \frac{dt_{\text{prev}} \cdot \text{mult_}dt\text{_chem}}{\max\left[\frac{\Delta T}{T}\right]}$	Chemistry time-step limit. Limits <i>dt</i> based on the current temperature and the change in temperature this time-step. Set inputs.in > temporal_control > mult_dt_chem .
<i>dt_src</i>	$dt \leq \frac{dt_{\text{prev}} \cdot \text{mult_}dt\text{_src}}{\max\left[\frac{\Delta src}{src}\right]}$	Source time-step limit. Limits <i>dt</i> based on the current source level <i>src</i> and the change in <i>src</i> this time-step. Set inputs.in > temporal_control > mult_dt_src .
<i>Cosimulation</i>	N/A	The time-step is being limited by another solver (e.g., Abaqus) in a cosimulation . No associated inputs.
<i>dt_piso</i> or <i>dt_simple</i>	$dt \leq 0.75 \cdot dt_{\text{prev}}$	Iterative solver time-step limit. Reduces the time-step if the number of PISO or SIMPLE iterations was larger than solver.in > PISO > piso_itmax or solver.in > SIMPLE > simple_itmax and

Chapter 5: Time Control Methods

Time-Step Control

Time-step limiter	Formulation	Comments
		smaller than twice the value of <i>piso_itmax</i> or <i>simple_itmax</i> . Set <i>solver.in</i> > <i>Transport</i> > * > <i>itmax</i> for each transport equation *.
<i>*, recovering</i>	N/A	See below.

The speed of sound in a cell is calculated based on whether the cell contains a compressible liquid or gas. The speed of sound in a compressible liquid is given by

$$c = \sqrt{\frac{B}{\rho}}, \quad (5.7)$$

where B is the bulk modulus (in Pa) and ρ is the density of the liquid (in kg/m^3). The speed of sound in a gas is given by

$$c = \sqrt{\frac{\gamma RT}{MW}}, \quad (5.8)$$

where γ is the ratio of the specific heat of a gas at a constant pressure to the specific heat of a gas at constant volume (C_p/C_v), T is the temperature of the gas (in K), R is the ideal gas constant (in $\text{J}/\text{mol}\cdot\text{K}$), and MW is the molecular weight of the gas (in kg/mol). The speed of sound in a two-phase region depends on the flow regime. For this reason, we recommend using only values from cells with one phase.

Refer to the *inputs.in* > *temporal_control* section of [Chapter 25 - Input and Data Files](#) for recommended settings of the time-step limiter parameters. Some solver settings require strict time-step limits based on the Courant-Friedrichs-Lowy (CFL) numbers. Under-relaxation may be required if *inputs.in* > *temporal_control* > *max_cfl_u* is greater than 1.0 or if *inputs.in* > *temporal_control* > *max_cfl_v* is greater than 0.5 when CONVERGE is run in an implicit mode. A *max_cfl_u* greater than 1.0 or a *max_cfl_v* number greater than 0.5 may not be stable when CONVERGE is run in an explicit mode.

Note that each CFL number parameter can be a fixed value or can [vary temporally](#). For a temporally varying parameter, specify a file name instead of a value in *inputs.in* and include that file in your case setup.

Recoveries

If the iterative algorithm does not converge the transport equations within twice the value of [*solver.in*](#) > *PISO* > *piso_itmax* or [*solver.in*](#) > *SIMPLE* > *simple_itmax*, or if CONVERGE detects a problem that will cause the simulation to crash (*e.g.*, negative density in a cell), it will reduce the time-step by a factor of two and attempt to re-solve the iteration. This is called a recovery. It is not uncommon to have a few recoveries in a simulation, especially during start-up. If repeated recoveries force an unreasonably small time-step (*i.e.*, approaching *dt_min* or stuck at *dt_recover*), or if you notice repeated patterns in recoveries, you should examine your case setup.

5.3 Super-Cycling

In some simulations, the time-scale necessary to resolve solid heat transfer is usually much greater than that for fluid heat transfer. Super-cycling in CONVERGE is an acceleration method that solves time-dependent conjugate heat transfer problems to a steady state in a solid in fewer cycles.

To activate super-cycling, set [*inputs.in*](#) > *feature_control* > *cht_supercycle_flag* = 1 in and include the [*supercycle.in*](#) file in your case setup.

Super-cycling iterates between fully-coupled transient and steady-state solvers via the following sequence. All of the parameters are located in [*supercycle.in*](#) unless otherwise specified.

1. CONVERGE solves the fluid and solid equations together using the transient solver (but does not store the solid heat transfer data) from the start of the simulation until *time_control* > *supercycle_start_time* in order to develop the fluid flow field.
2. At *time_control* > *supercycle_start_time*, CONVERGE begins storing values for a heat transfer coefficient (HTC) and near-wall temperature for each cell at the solid/fluid interface, one value for each time-step.
3. CONVERGE continues to solve both the fluid and solid equations and stores HTCs and near-wall fluid cell temperatures for a time equal to the *time_control* > *supercycle_stage_interval*.
4. At time = *supercycle_start_time* + *supercycle_stage_interval*, CONVERGE freezes the fluid solver and calculates the time-averaged HTC and temperature for each cell at the solid/fluid interface based on the values stored in step 3. CONVERGE performs the solid heat transfer calculation at this time using the time-averaged HTC and near-wall fluid cell temperatures as boundary conditions at the INTERFACE. There are two solid heat transfer calculation methods that CONVERGE can use: steady-state or transient.
 - a. Perform steady-state solid heat transfer calculations by setting *time_control* > *supercycle_length* to a negative value. CONVERGE solves the solid heat transfer until the tolerance defined in *numerics_control* > *supercycle_energy_tol* is achieved. This solid energy equation calculation will not appear to take any time on an output file plot of Solid Temperature vs. Time (*e.g.*, the solid temperature will appear to immediately

Chapter 5: Time Control Methods

Super-Cycling

jump).

- b. Perform transient solid heat transfer calculations by setting `time_control > supercycle_length` to a positive value. CONVERGE solves the solid heat transfer for the length of time set by `supercycle_length`. Transient solid heat transfer calculations are not commonly used. Contact Convergent Science to discuss a transient solid solver configuration.

After performing the solid energy transfer calculations, the temperature of the solid represents the steady-state solid temperature.

5. CONVERGE starts the fluid solver again after calculating the steady-state solid temperature, and solves the fluid and solid equations for another period of time equal to `time_control > supercycle_stage_interval`. The updated solid temperature causes the fluid temperature to rise dramatically. The fluid temperature rise then levels off when the difference between the solid and fluid temperatures is no longer great enough to increase the fluid temperature at the given flow rate. CONVERGE also stores a new heat transfer coefficient and temperature for each solid/fluid interface cell, at each time-step. CONVERGE uses HTC and temperature values from only the current super-cycle stage when averaging these values.
6. At time = `supercycle_start_time + (2 * supercycle_stage_interval)`, CONVERGE again pauses the fluid solver and recalculates the time-averaged heat transfer coefficient and temperature for each cell at the solid/fluid interface based on the values stored in step 5. With the fluid solver paused, CONVERGE again performs the solid heat transfer calculations as described in step 4.
7. CONVERGE repeats this super-cycling process until simulation reaches `inputs.in > simulation_control > end_time`.

Contact Resistance with Super-Cycling

In some simulations, there are areas through which fluid flows for only part of the simulation (e.g., the gap between a valve and a valve seat in an internal combustion engine). To model solid-solid contact resistance between these temporarily separated regions, configure a contact region via [contact resistance events](#) in [events.in](#). When the two solids are specified as closed, CONVERGE does not allow fluid flow between the two regions and performs solid-solid heat transfer calculations. To improve the speed of temperature convergence in a contact region when super-cycling is active, CONVERGE copies the temperature and heat transfer coefficient from each solid boundary and applies these properties as boundary conditions to the solid with which the first solid is in contact.

In contact resistance regions, CONVERGE calculates the effective contact resistance based on the nominal contact resistance R specified in [events.in](#) and the distance between contacting surfaces dx . The thermal conductivity k is calculated as

$$\frac{k}{dx} = \frac{1}{R}. \quad (5.9)$$

Treatment of Moving Solid Boundaries During Super-Cycling

In some boundary configurations, a solid boundary slides into and out of a stationary solid boundary, which results in changing boundary conditions for the stationary surface. Examples of this phenomenon in an internal combustion engine include a valve sliding within a valve guide or a solid piston in contact with the solid liner.

For the piston—liner contact scenario, the piston moves up and down within the liner. Therefore, at different times during the simulation, portions of the liner may be exposed to the fluid in the cylinder, the solid piston surface, or the air in the crankcase. In CONVERGE, the boundary conditions for these scenarios are a fluid-solid interface, a solid-solid interface, and a solid wall with a convective boundary condition, respectively. Figure 5.4 below shows these boundary conditions.

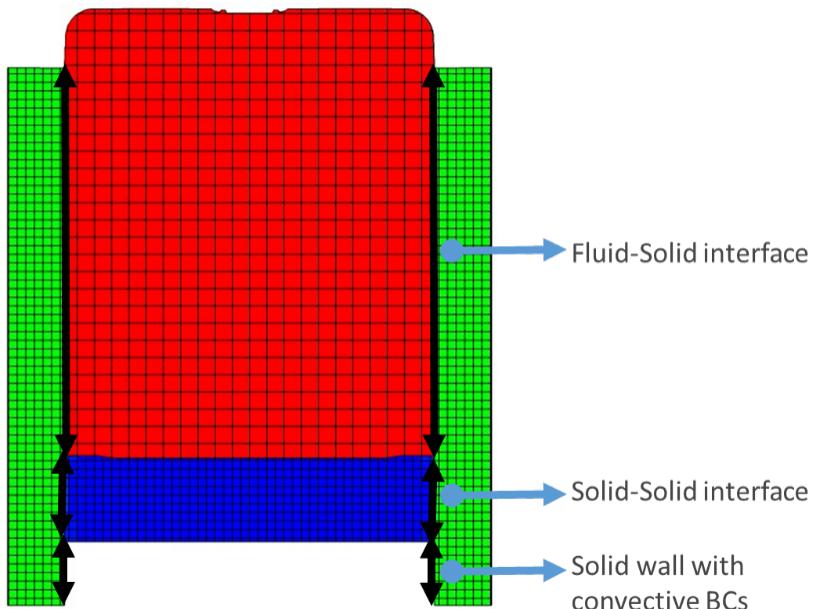


Figure 5.4: Boundary condition types for a moving solid piston within a solid liner.

Discrete portions of the liner (*i.e.*, cells that are adjacent to the combustion chamber) experience each of these three boundary conditions for a certain amount of time during the engine cycle. To accurately calculate heat transfer information, CONVERGE weights each of the three boundary conditions according to the fraction of an engine cycle that a discrete portion of the liner experiences each boundary condition. Table 5.9 below provides an example of the weights for a cell on the liner surface.

Chapter 5: Time Control Methods

Super-Cycling

Table 5.9: Boundary conditions and respective weights.

Boundary condition for a point on the liner surface	Time percentage that the point is exposed to the boundary condition over an entire engine cycle (weight)
Fluid-solid interface	40%
Solid-solid interface	10%
Fixed convection	50%

At each super-cycle stage, the solid-solid interface (*e.g.*, piston-liner interface) is temporarily decoupled. At each iteration of the steady-state solid solver, CONVERGE calculates an effective heat transfer coefficient and temperature for the stationary solid boundary (*e.g.*, the liner) based on the three boundary conditions and their weights. Then, CONVERGE computes a new piston contact surface temperature based on these effective variables. This procedure helps improve the accuracy of the heat transfer calculation and the speed of convergence.

When super-cycling is active, CONVERGE automatically creates boundary groups for boundaries attached to stretching fluid-solid interfaces. In the previous example, one boundary group includes the solid liner interface, the solid piston interface and bottom of the liner. If there are solid coupled boundaries that are supposed to be included in the boundary group, you must set the velocity boundary condition as *MOVING Translating* or *FIXED Translating*. Otherwise, CONVERGE cannot find the correct boundaries.

Log File

In the log file, CONVERGE creates a record of the boundary grouping. Figure 5.5 below shows a sample log file excerpt for the boundary grouping created for the piston-liner example.

```
there are 1 bound_group created
boundary ids in bound_group 0
grouped boundary 29
boundary in contact 40
depend boundary 7
```

Figure 5.5: A sample log file excerpt for moving solid boundaries during super-cycling.

In the above sample, *bound_group* refers to the number of boundary groups matching the criteria for this feature. On the next line, the log file lists the number of boundaries in the group. The boundary ID of the stationary solid follows *grouped boundary*. The boundary ID of the solid boundary in contact with the stationary solid boundary follows *boundary in contact*. Finally, the boundary ID for *depend boundary* corresponds to the boundary used to obtain position information for the grouped boundaries.

Limitations

The overall length of the grouped boundaries must be constant in time. The moving boundaries (*e.g.*, the piston) must move in the same direction with the same speed. Additionally, the volume of these boundaries must be constant (they cannot deform).

Chapter 5: Time Control Methods

Super-Cycling

For an engine case, you must simulate at least an entire engine cycle for correct averaging. Otherwise, this feature will not produce reasonable results.

Super-Cycle Stages

To expedite the solution of a conjugate heat transfer simulation, CONVERGE allows you to update the averaged data in the storage routines via [*supercycle.in*](#) > *time_control* > *supercycle_num_stages*.

For example, for a four-stroke engine running at 2,000 RPM, you could set *supercycle_num_stages* = 12. Then you could set [*supercycle.in*](#) > *time_control* > *supercycle_stage_interval* = 60 crank angle degrees. CONVERGE will average 12 blocks of data, each 60 crank angle degrees long, in the first super-cycle. In the second super-cycle, CONVERGE discards the earliest stage from the first super-cycle during the averaging process and uses the latest stage generated in the storage routine of the second super-cycle to update the 12 blocks of 60 crank angle degrees-data. Figure 5.6 illustrates the update process of super-cycle stage data.

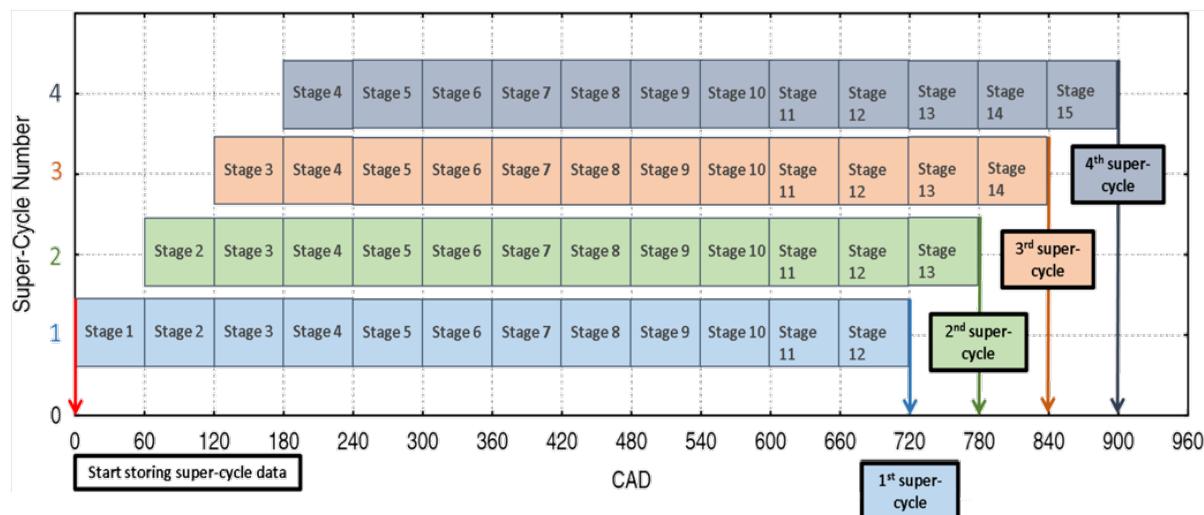


Figure 5.6: Stages in the super-cycle process.

Super-Cycle Output

You can prescribe monitor points to obtain temperature data from specific locations in the solid domain. Define the monitor points via [*supercycle.in*](#) > *supercycle_points*. CONVERGE writes the solid temperature data to the *supercycle_point<number>.out* file(s), where *<number>* represents the number of the monitor point.

If you want to monitor more than just solid temperatures, use [*monitor_points.in*](#) or create a user-defined function (UDF) for monitor points. For information about using a UDF for monitor points, refer to the CONVERGE 3.1 UDF Manual.

Cylinder Duplication

In some multi-cylinder conjugate heat transfer (CHT) engine simulations, the goal is to predict the heat transfer between each of the cylinders and the solid cylinder head. Modeling combustion in all cylinders with a method like SAGE, however, is often prohibitively expensive. The heat transfer mapping for multi-cylinder CHT feature offers a faster alternative. Since the combustion behaves in a similar manner in all cylinders, CONVERGE models combustion in a primary cylinder and then maps the results to one or more copy cylinders.

This feature is available only with [*super-cycling \(inputs.in\) > feature_control > cht_supercycle_flag = 1*](#). At each super-cycle, CONVERGE cycle-averages the wall temperatures and heat transfer coefficients (HTC) from relevant boundaries in the primary cylinder and maps this information to the corresponding boundaries in the copies. Thus, the timing of the mapping for copy cylinders is entirely dependent on the super-cycling configuration for the primary cylinder.

CONVERGE can map information on fluid-solid INTERFACE and solid WALL type boundaries between a primary cylinder and copy cylinders. For mapped solid WALL boundaries, the temperature boundary condition for the copy boundary must be Neumann.

One method to approximate combustion in the copy cylinders when the full cylinder geometry exists is to specify a heat release source that mimics the effects of combustion within the cylinder. In some cases, surface triangles for the cylinder may not exist for copy cylinders. In this case, no attempt is made to model in-cylinder combustion. The cylinder head boundary is a solid WALL and thus requires a Neumann temperature boundary condition. This requirement is because CONVERGE cannot map the data from the primary cylinder fluid-solid INTERFACE to the copy cylinder solid WALL if the solid WALL temperature boundary condition is Dirichlet.

Configuration

First set up [*inputs.in*](#) as desired. Set [*supercycle.in > supercycle_surface_map_flag = 1*](#) to activate multi-cylinder heat transfer duplication.

Next set up multi-cylinder heat transfer mapping via [*supercycle_surface_map.in*](#). For each primary cylinder (a cylinder with full combustion from which data will be mapped), you can have as many copy cylinders (or solid WALL boundaries used to represent combustion) as desired. In this file you provide transformation information from the copy to the primary. That is, you will enter the transformation that would move the copy cylinder to the primary cylinder. In practice, the cylinders may not overlap perfectly (due to small differences in the surface geometry), but, as long as they match with a tolerance, the mapping will be successful. CONVERGE uses the transformation to determine which boundaries are linked between the primary and copy cylinders.

For any of the transformation types below, you can set entries to 0, which means the transformation will not occur. If you specify non-zero values, CONVERGE will perform the transformation in accordance with your specifications.

If you set the elements of `supercycle_surface_map.in` > `primaries` > `primary` > `copies` > `copy` > `transformation` > `mirror_plane` to non-zero values, CONVERGE will use a mirror transformation. That is, reflecting the copy cylinder across the given plane will result in the copy overlapping the main exactly. Enter the a , b , c , and d coefficients of the equation that describes this mirror plane ($ax + by + cz + d = 0$).

To specify a translation, enter the x , y , and z translation amounts necessary to translate the copy cylinder onto the primary cylinder.

To specify a rotation, enter a value for `supercycle_surface_map.in` > `primaries` > `primary` > `copies` > `copy` > `transformation` > `rotation` > `rot_angle` (the angle with which to rotate the copy cylinder in the counter-clockwise direction). Also supply the x , y , and z coordinates of the rotation origin for `orig_xyz`. Enter the x , y , and z components of the direction vector about which to rotate the copy cylinder for `vector_xyz`.

Forced Pairs

In each `supercycle_surface_map.in` > `primaries` > `primary` > `copies` > `copy` settings block, you can optionally supply forced pairs. Forced pairs are used if CONVERGE cannot determine the copy boundary on which to map the data (possibly due to imperfect geometries) based on the given transformation information. The two specified boundary IDs become forced pairs and CONVERGE copies the information from primary to copy, regardless of the transformation.

Log File Output

When the heat transfer mapping for multi-cylinder CHT feature is active, CONVERGE provides information about the primary-copy pairing during the initialization of the simulation (*i.e.*, before the first time-step). This information will appear either [on the screen](#) or [in the log file](#), depending on how you execute CONVERGE. The first section shows how the area, geometric center, surface normal vector, and moment of inertia of each copy boundary compare to those for the primary boundary. The second section lists the boundary IDs of successful boundary pairings. The third section lists the boundary IDs of any forced pairs (if present).

5.4 Fixed Flow Method

During a simulation, some parts of a computational domain may progress at a very different time-scale than other parts. Rather than solve the entire domain at the smaller time-scale at a high computational cost, CONVERGE offers a fixed flow time-control approach to temporally decouple the solution of the two phases. We recommend that you activate the fixed flow feature only when one phase is in a quasi-steady state with respect to the other. For example, in an exhaust aftertreatment simulation with a urea-water solution spray, the momentum of the spray parcels is very small compared to the momentum of the gas phase, and therefore the spray parcels have only a weak effect on the gas-phase flow.

You can activate fixed flow in conjunction with [conjugate heat transfer \(CHT\) modeling](#).

Chapter 5: Time Control Methods

Fixed Flow Method

This section describes the theory behind the fixed flow approach, and subsequent subsections describe [setup considerations](#) and [applications for which fixed flow may be useful](#).

Consider a case in which all of the boundaries are typically non-moving and the boundary conditions are constant in time, such as an exhaust aftertreatment case. When a spray is first injected, spray parcels drag and evaporate by interacting with the gas phase. The gas-phase flow variables (velocity, pressure, density, temperature, and species concentrations) initially change rapidly but quickly reach a quasi-steady state that remains as long as the spray continues.

Consider the conservation of mass for the gas phase:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = S, \quad (5.10)$$

where S represents the mass source term from the spray. We can observe the quasi-steady state for mass conservation when the local density at a point in the flow field is at equilibrium as

$$\frac{\partial \rho}{\partial t} = S - \frac{\partial \rho u_i}{\partial x_i} \approx 0, \quad (5.11)$$

which shows that the quasi-steady state accounts for the source term from the spray.

Similarly, the equations for conservation of momentum, energy, and species reach a quasi-steady state when values of velocity, temperature, and species mass fractions, respectively, reach a local equilibrium. As long as the boundary conditions and spray state do not change, the gas-phase conservation equations do not need to be solved because the flow field is no longer changing significantly from time-step to time-step.

When fixed flow is active for a particular stream, the Eulerian conservation equations are not solved for that stream. The most recent values for the flow state are used for all calculations. Thus, CONVERGE simulates a quasi-steady state for that stream and can solve equations for the other streams on a much larger time-scale.

For a gas stream, by default, the most recently solved values of velocity, pressure, temperature, density, and species concentration in each cell are maintained until the fixed flow interval ends. After the end of the fixed flow interval, CONVERGE will once again solve the full set of conservation equations for that stream. You can optionally choose to continue solving for species concentrations during the fixed flow interval.

In solid streams, only the equation for the conservation of energy is solved to obtain the temperature. When a solid stream is fixed, the most recently solved value of temperature in each cell are maintained until the fixed flow interval ends. After the end of the fixed flow

Chapter 5: Time Control Methods

Fixed Flow Method

interval, CONVERGE will once again solve all the energy conservation equations for that stream.

Fixed Flow Setup

To activate the fixed flow method, set [*inputs.in*](#) > *feature_control* > *fixed_flow_flag* = 1 and include a [*fixed_flow.in*](#) file in your case setup. You can specify the streams and the time intervals during which fixed flow is active. Outside of these intervals, CONVERGE will solve the full set of conservation equations in all streams.

You can set the temporal control ([*fixed_flow.in*](#) > *fixed_time_control* > *temporal_type*) for fixed flow intervals to either *SEQUENTIAL* or *CYCLIC*. For a pulsating spray, verify that the [*fixed_flow.in*](#) > *fixed_time_control* > *cyclic_period* is equal to the cyclic period of the spray. We recommend that you set the fixed flow intervals so that sufficient time is available to update the flow field at the beginning and at the end of the spray pulse. More information on estimating fixed flow intervals for pulsating sprays is [described below](#).

The fixed flow feature is applied on a stream-by-stream basis via [*fixed_flow.in*](#) > *fixed_stream_ids*. We recommend activating fixed flow simultaneously for all gas regions. Conveniently, often all of the gas regions belong to the same stream.

If you set [*fixed_flow.in*](#) > *solve_fixed_species_flag* = ON, CONVERGE will continue solving the species transport equations during fixed flow intervals. You can customize the solver settings that are used during fixed flow intervals in the [*fixed_flow.in*](#) > *solver_control* settings block.

If fixed flow is activated in conjunction with [conjugate heat transfer modeling](#), you can apply fixed flow to the solid stream. However, the fixed flow method is not typically applied to solids. For example, if the solid phase does not rapidly reach a quasi-flow state with respect to the fluid phase, applying fixed flow to the solid stream would not be useful.

Refer to [Chapter 25 - Input and Data Files](#) for a detailed description of the parameters in [*fixed_flow.in*](#).

Treatment of Parcels

The fixed flow feature is typically activated in cases with [discrete phase modeling](#) of liquid parcels. You can change how CONVERGE treats spray and film parcels while fixed flow is activated via [*fixed_flow.in*](#) > *fixed_spray_film_flag*.

If *fixed_spray_film_flag* = *TRACK_PARCELS*, fixed flow does not apply to spray and film parcels in the domain. Spray injection, parcel trajectory, evaporation, and wall interaction are calculated and updated according the models specified in [*parcels.in*](#). The film continues to move and evaporate according to the physical models activated in [*parcels.in*](#).

If *fixed_spray_film_flag* = *FREEZE_PARCELS*, spray and film parcels remain in the domain, but parcel properties do not change while fixed flow is active.

Chapter 5: Time Control Methods

Fixed Flow Method Fixed Flow Setup

If `fixed_spray_film_flag = REMOVE_PARCELS`, all spray and film parcels are removed from the domain during fixed flow intervals. The parcels are replaced in the domain at the end of the fixed flow interval in the same location with the same properties as prior to the fixed flow interval. Parcel properties do not change during the fixed flow interval. We do not recommend this option.

Estimating Fixed Time Intervals for a Pulsating Spray

You must specify the time interval(s) during which fixed flow is activated. If the computational domain contains a pulsating spray, it is important to ensure that the flow field has sufficient time to reach a quasi-steady state in the presence of the spray before the field is rendered static.

You can estimate the time required for the flow field to adjust to the spray from the residence time of the gas in the domain of interest.

The general expression for mean residence time of a flow in a channel is given by [Spalding \(1958\)](#) as

$$t_{\text{residence}} = \frac{V}{v}, \quad (5.12)$$

where V is the volume of the channel and v is the volumetric flow rate within it. For a straight channel with constant cross-sectional area, this simplifies to

$$t_{\text{residence}} = \frac{L}{U_{\text{inlet}}}. \quad (5.13)$$

When a prescribed mass flow rate m is applied at the inlet, the residence time becomes

$$t_{\text{residence}} = \frac{L\rho A_{\text{inlet}}}{m_{\text{inlet}}}. \quad (5.14)$$

You can estimate flow residence times similarly for alternate boundary conditions.

For complex geometries, you can simulate the domain without activating fixed flow and examine the local velocities to estimate the residence time.

Example: Estimating a Fixed Flow Interval

Consider a simple channel with $L = 0.45 \text{ m}$ and $U = 30 \text{ m/s}$. The residence time is $L/U = 0.015 \text{ s}$. The residence time is the minimum time the flow field needs to respond to the change in spray state. We expect the flow to have reached a quasi-steady state after each spray state change (*i.e.*, the start and end of injection) at slightly more than the residence time (say, 0.02 s in this case).

Chapter 5: Time Control Methods

Fixed Flow Method Fixed Flow Setup

Table 5.10 below lists the parameters used to estimate fixed flow intervals for a pulsating spray. The first two parameters ([fixed_flow.in > fixed_time_control > temporal_type](#) and [fixed_flow.in > fixed_time_control > cyclic_period](#)) must match the values specified for the appropriate injector in [parcels.in](#).

Table 5.10: Parameters used to estimate [fixed_flow.in > fixed_time_control > fixed_intervals](#).

Parameter	File name	Example value
fixed_time_control > temporal_type	fixed_flow.in	CYCLIC
fixed_time_control > cyclic_period	fixed_flow.in	0.5 s
injections > injection > injection_control > start_time	parcel_introduction.in	0.0 s
injections > injection > injection_control > end_time	parcel_introduction.in	0.2 s

In this case, since spray starts at 0.0 seconds, flow should be solved until 0.0 + 0.02 s. Fixed flow can then be activated until the spray state changes, which in this case is when spray ends at 0.2 s. Thus the first fixed flow interval should be specified as [0.02 to 0.2] s.

Flow should be solved after a spray state change. In this case, after the spray ends at 0.2 s, flow should be solved for 0.2 + 0.02 s, and then fixed flow can be activated until the end of the cycle (here, 0.5 s). The second fixed flow interval should be specified as [0.022 to 0.5] seconds.

In each cycle, CONVERGE will solve now all the conservation equations only between 0 to 0.02 s and 0.02 to 0.022 s. The fixed intervals should be set to [0.02 to 0.2] and [0.022 to 0.5] seconds.

Fixed Flow Applications

In urea-based selective catalytic reduction (SCR) systems, urea-derived deposits often form near injectors, on mixers and pipes, and on the SCR catalyst face. In simulations to predict these deposits, the locations of spray impingement, film accumulation, movement, and decomposition are of primary interest. The wall temperature is important in determining spray-wall interaction and film evaporation rates. Due to the much longer thermal time-scale in metal walls compared to gas, a quasi-steady state is not reached in the metal nearly as quickly as in the gas. Thus, fixed flow is typically activated for the gas phase (via the corresponding stream) and if necessary, in conjunction with [conjugate heat transfer modeling](#).

The fixed flow approach was validated for a canonical SCR test case by [Maciejewski et al. \(2019\)](#). Testing and validation work in other applications is ongoing.

Chapter



6

Optimization Techniques

6 Optimization Techniques

This chapter describes CONVERGE's [CONVERGE Genetic Optimization \(CONGO\) utility](#), which allows you to run a [genetic algorithm \(GA\)](#) optimization or a [design of experiments \(DoE\)](#).

6.1 CONGO Optimization and Model Interrogation

The [CONVERGE Genetic Optimization \(CONGO\) utility](#) allows you to run a [genetic algorithm \(GA\)](#) optimization or a [design of experiments \(DoE\)](#) model interrogation study for a CONVERGE simulation on the LINUX platform. This utility sets up CONVERGE case directories, starts the CONVERGE simulation for each case, collects results, and calculates a merit based on criteria you specify.

CONGO will run the GA for the number of generations you define. For a DoE, it will run it for the number of runs you specify. The workflow required to set up a CONGO GA or DoE case is described in the [Running CONGO](#) section.

You can use CONGO also for non-CONVERGE optimization and model interrogation studies, provided that the analysis tool coupled with CONGO generates output consistent with the format required by CONGO.

After you start a CONGO GA, it runs successive (or concurrent) generations without requiring action on your part. CONGO automatically parses parameter values into CONVERGE input files in run directories. Additionally, CONGO can automatically run design of experiments (DOE) using the same setup method as running a GA. Using the [gatodoe.in](#) file, you can also have CONGO set up a few selected cases from a GA to run as a DoE automatically.

You can modify a routine for CONGO to allow you to define parameterized features, such as piston bowl shape or fuel injection rate-shape. You can also implement complex merit function formulations via a user-defined function (written in C).

Genetic Algorithm

A micro genetic algorithm (GA) can be used to optimize a model for a given set of input parameters and a defined model output. For the purposes of this chapter, a model is defined as a combination of surface geometry parameters (*e.g.*, piston bowl shape or stroke distance) and other CONVERGE input parameters (*e.g.*, injection rate or valve lift position files). The model variations generated by the GA are controlled by a merit function, which includes the parameters to optimize, such as engine fuel consumption, and can include parameters to impose constraints upon, such as engine emissions.

A GA takes a survival of the fittest approach to optimize a design.

A set of input parameters comprises an individual. A set of individuals comprises a population. You will define the size of the population. Each generation consists of a

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Genetic Algorithm

population, which consists of the best (or fittest) individual from the previous generation and new individuals (children) created from individuals in the previous generation (parents).

The figure below illustrates this process. Individuals (A) comprise a population for each generation. Individuals are created for subsequent generations from the parents using a tournament methodology. The tournament shuffles the individuals into pairs (B), selects individuals from the pairs by comparing the merit values (C), and pairs the winners of the tournaments into new parents (D). Children (E) are created from the parents. The tournament continues until enough children are created for the next generation (F). The individual in the generation with the highest merit (best-so-far) is automatically passed to the next generation.

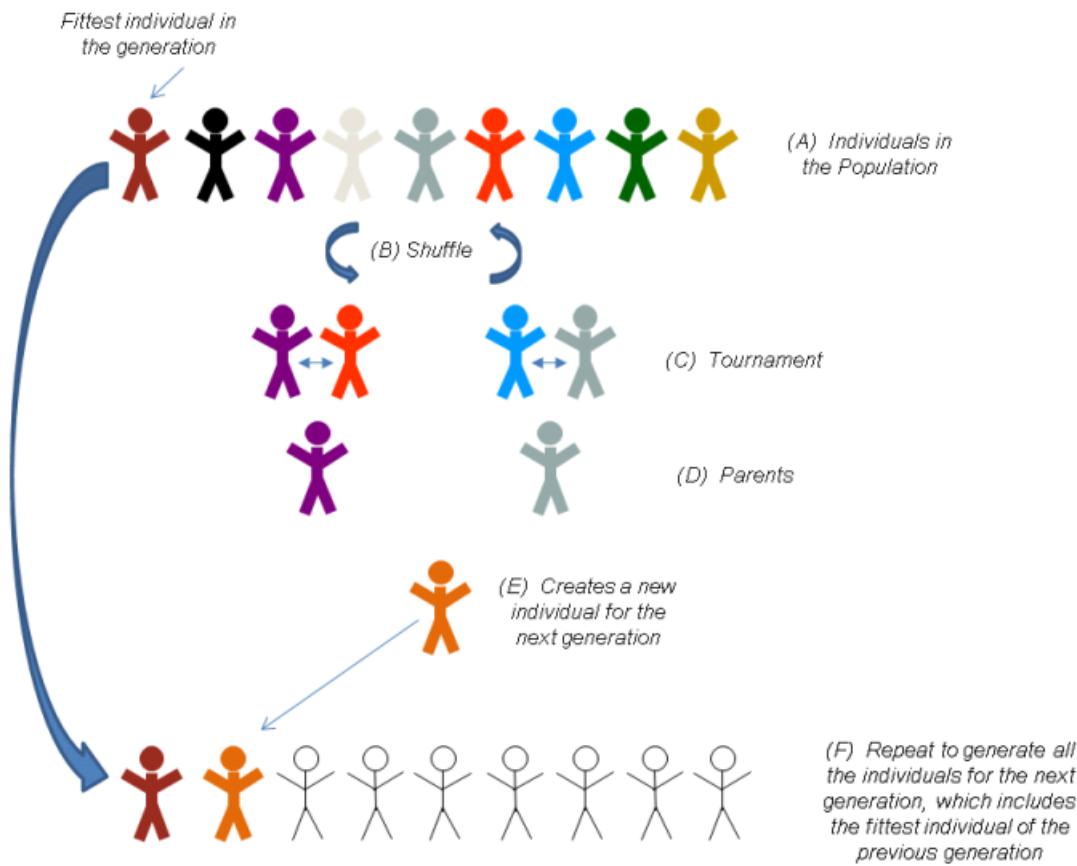


Figure 6.1: Micro genetic algorithm.

As the GA proceeds through generations, the individuals become more and more similar to the fittest individual. The GA applies a convergence test to determine when the parameters of the individuals in the population have converged within a user-defined criteria. Once a population has converged, the individual with the highest merit is retained, and a new set of

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Genetic Algorithm

individuals is generated with randomly assigned parameters. Generating a new population randomly after a population converges enables the GA to evaluate the entire parameter space instead of identifying a local maximum as an overall optimum.

Depending on characteristics of the model and the number of parameters being optimized, the GA will converge on the optimum model configuration within a certain number of generations.

For example, in a typical diesel engine case, where fuel consumption and emissions will be optimized, the GA may converge in approximately 50-100 generations for typical input parameters (e.g., start of injection). This number depends on the complexity of the model response to the range of the input parameters and the number of generations that the GA uses to converge on the optimum input parameters.

Parameter Representation and Crossover Evaluation

Parameters in the GA are represented by an n -bit binary number (typically $n = 30$). Each individual consists of the binary parameters in sequence. This parameter sequence is the DNA of the individual.

For instance, if three parameters are represented with eight bits each, the DNA of an individual would be as follows:

Param1	Param2	Param3
01110101	01011011	11001000

The spaces between parameters are added for clarity.

The CONVERGE GA can use real number convergence checks (recommended) or a DNA convergence check (binary evaluation). A real number convergence check means that the DNA is converted to real parameter values before checking for parameter convergence. The conversion from DNA to real numbers is carried out by converting the binary value back to a decimal number. Define a convergence criteria fraction via [congo.in > p_conv](#). A typical value for p_{conv} is 0.97.

A crossover algorithm creates individuals for the next generation from parents from the previous generation. A child inherits matching DNA from the parents and differences in the DNA of the two parents cause a random bit to be evaluated for the child. For example, two parents with only three differing bits would create a child as shown below, where a, b, c will be determined randomly as 1 or 0. Note that the middle parameter (Param 2) is converged for these parents. Although there are only 6 possibilities for the DNA makeup of the child, the real number values of the child's parameters will be significantly different among the 6 possibilities.

	Param1	Param2	Param3
parent 1:	011 1 0101	01011011	110010 00
parent 2:	011 0 0101	01011011	110010 11
child:	011 a 0101	01011011	110010 bc

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Genetic Algorithm

As the GA progresses through generations and the population approaches a local maximum optimization, individuals in the population will become similar. If the GA is allowed to continue generating new populations, the individuals in the population will become identical.

The GA has some pre-defined optimization parameters based on commonly adjusted model inputs. You can also create user-defined optimization parameters based on more complex inputs in CONVERGE, such as a geometry object (e.g., a piston bowl shape) or a temporally or spatially varying parameter, such as a fuel injection rate or valve lift profile.

Design of Experiments

A design of experiments (DoE) is a model interrogation technique, which is different than a genetic algorithm, which is an *optimization* technique. For a DoE, you generate a table containing a pre-determined number of models, or runs. Each run consists of a combination of different values of selected model parameters.

In a DoE, there are no generations, parents, children, DNA, or a tournament structure as in a GA. A DoE simply runs different combinations of model parameters and compares the resulting merit. The merit function CONGO uses for a DoE is the same as for a GA, but is calculated only for reference and is not used by CONGO. CONGO generates the merit function results simply as an output.

You can use CONGO to specify which individual values for each model parameter in the DoE, as opposed to a range of values in a GA. CONGO allows you to specify the total numbers of runs in a DoE and each of these runs can be run concurrently, provided you have the hardware and licenses to accommodate this number of runs.

Do not mistake a DoE for an optimization technique; it simply interrogates the specific models and evaluates the merit of each. After the CONGO DoE is complete, you can refer to the [output.out](#) and [perform.out](#) for the merit calculated for each model.

A DoE may be a good option if you have limited time, computing resources, and you have a reasonably good understanding of how the variance of different model parameters (and the interplay thereof) will affect the performance of the model.

Running CONGO

CONGO is a utility for use by advanced CONVERGE users who are comfortable setting up CONVERGE simulations with a user-defined function (UDF). Refer to the CONVERGE 3.1 UDF Manual for more information regarding running CONVERGE with a UDF.

CONVERGE Studio contains a module that allows you to set up the [CONGO input files](#) required to configure a GA or DoE case, but you should still be comfortable with running CONVERGE with a UDF prior to attempting a GA or DoE with CONGO.

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

The following figure shows the overall organization and content of a CONGO Case Directory. The following sub-sections describe each sub-section of the CONGO Case Directory and provide an overview of how each of these subsections interacts with CONGO to produce meaningful GA or DoE conclusions.

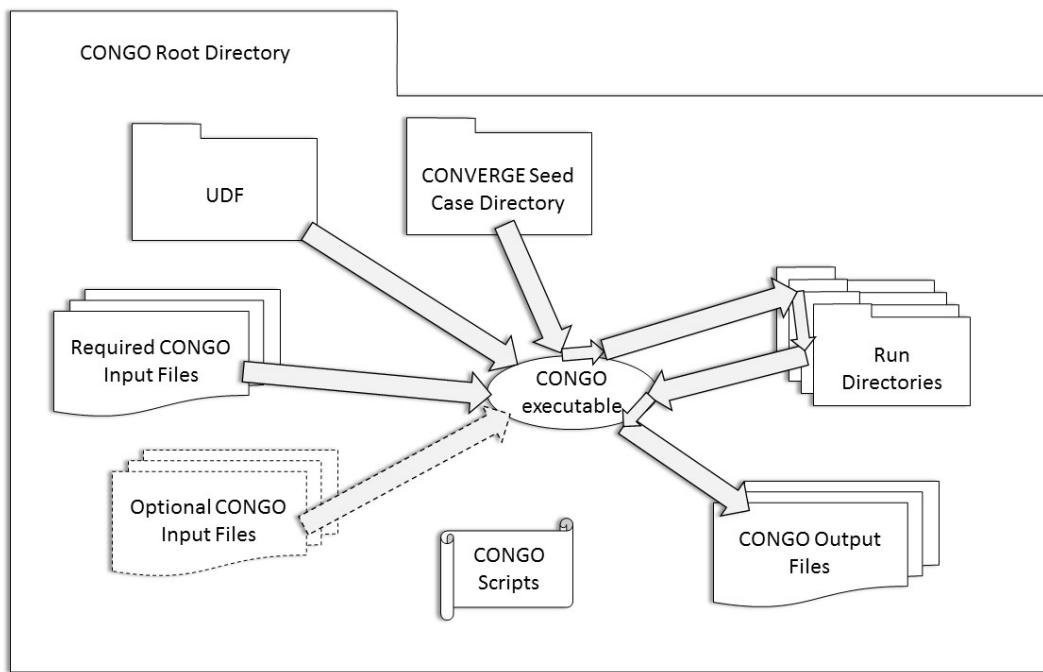


Figure 6.2: CONGO communication structure.

CONGO Input Files

The [congo.in](#), [case.in](#), [execute.in](#), and [merit.in](#) files are mandatory for a GA simulation. The [congo.in](#) file defines the parameters for the GA. The [case.in](#) file defines the parameter ranges for the experiment model inputs and parsing for the CONVERGE input files. The [execute.in](#) file defines how CONVERGE will be executed for a GA simulation. The [merit.in](#) file defines the merit function in terms of CONVERGE performance parameters.

You can use the CONGO module in CONVERGE Studio to import, configure, and export input files for a CONGO run with CONVERGE. The CONGO module tracks and coordinates all of your genetic algorithm input files so you can easily design a genetic algorithm case.

The [udi.in](#) and [gatodoe.in](#) files are optional for GA simulations. These files are described in full in [Chapter 25 - Input and Data Files](#). The [udi.in](#) file allows you to enter a user-defined individual (UDI) into the GA experiment at any generation. The [gatodoe.in](#) file can be used to automatically set up a set of cases from the GA experiment.

Merit Function

A properly defined merit function is vital to the efficiency and success of a GA. By definition, the GA works to maximize the merit through progressive generations. A poorly defined merit function will cause an undesirable optimization.

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

When defining the merit function, you can define response variables. Response variables can be performance terms (maximizing, minimizing, or error) and constraint terms (maximum and minimum). Performance terms are weighted and multiplied by 100 in the default merit calculation. Constraint and error terms are then subtracted to decrease the merit score. It is recommended to formulate the merit function so that an optimized design would have a merit of 100 and the merit score of a baseline design (or starting point) would be near 0. Negative merit scores are permissible as are scores greater than 100.

Default Merit Function

You can use the default merit function in CONGO by defining response variables in [merit.in](#). Response variables consist of *performance variables (PV)*, *error* or *normalized error* of performance variables, and *constraint variables (CV)*. The default merit function sums contributions from all types of response variables:

$$\begin{aligned} \text{Merit} = & \\ 100 \times & [MinimizePV + MaximizePV + Error\ of\ PV + Normalized\ Error\ of\ PV \\ & + MinimumCV + MaximumCV], \end{aligned} \quad (6.1)$$

where

$$MinimizePV = weight \times \frac{Value_{target}}{Value_{calculated}}, \quad (6.2)$$

$$MaximizePV = weight \times \frac{Value_{calculated}}{Value_{target}}, \quad (6.3)$$

$$\begin{aligned} Error\ of\ PV = & \frac{1}{Number\ of\ PVs} \\ & - \left(\frac{weight}{100} \right) \times abs \left(Value_{calculated} - Value_{target} \right), \end{aligned} \quad (6.4)$$

$$\begin{aligned} Normalized\ Error\ of\ PV = & \frac{1}{Number\ of\ PVs} \\ & - \left(\frac{weight}{100} \right) \times abs \left(\frac{Value_{calculated} - Value_{target}}{Value_{target}} \right), \end{aligned} \quad (6.5)$$

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

$$MinimumCV = -weight \times \left[\left(\frac{Value_{target}}{Value_{calculated}} \right)^{power} - 1 \right], \text{ and} \quad (6.6)$$

$$MaximumCV = -weight \times \left[\left(\frac{Value_{calculated}}{Value_{target}} \right)^{power} - 1 \right]. \quad (6.7)$$

MinimizePV and *MaximizePV* are performance variables of the type *minimize* and *maximize*, respectively. You can define multiple *minimize* and *maximize* performance variables. As the name suggests, the lower the calculated value of the *minimize* performance variable, the more it contributes to a higher merit score. Likewise, the higher the calculated *maximize* performance variable, the more it contributes to merit. You must assign a weight to each *minimize* and *maximize* PV. CONGO uses this weight to increase or decrease the contribution of this performance variable relative to other response variables.

You can use the *Error of PV* or *Normalized Error of PV* term to lower the merit score when a performance variable differs from the target value you define.

MinimumCV and *MaximumCV* are constraint variables of the type *minimum* and *maximum*, respectively. You can use constraint variables to control the negative impact on merit when specified variables exceed (*maximum*) or fall short of (*minimum*) the target *value* you specify. The *weight* CONGO uses in these subtractions from the merit score is as follows:

$$\begin{aligned} &\text{if } (Value_{calculated} > Value_{target}), \text{weight} = \text{weighting_greater}, \\ &\text{if } (Value_{calculated} < Value_{target}), \text{weight} = \text{weighting_lessthan}. \end{aligned} \quad (6.8)$$

The *weighting_greater* value applies to *maximum* constraint variables and the *weighting_lessthan* applies to *minimum* constraint variables.

Simple Example Merit Function

The result of interest—or level of optimization—of any one individual model is defined by a merit function. The GA uses this merit function to determine which individual model iteration is the fittest of its generation. The merit function should be normalized at a value, typically 1 or 100, and should be a continuous function.

A simple example of a merit function for a diesel engine is one that gives a model high merit for lower levels of Indicated Specific Fuel Consumption (ISFC). The merit function could then be defined with a target value as

$$merit = \frac{ISFC_{target}}{ISFC_{calculated}}. \quad (6.9)$$

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

As the ISFC calculated for the model is reduced below the target level, the merit function becomes greater than 1.

Similarly, a more complicated merit function could include penalties when emission levels (such as NOx and particulate matter) exceed a constraint value. This merit function could be written as

$$merit = \frac{ISFC_{target}}{ISFC_{calculated}} - \alpha \left(\frac{NOx_{calculated} - NOx_{constraint}}{NOx_{constraint}} \right) - \beta \left(\frac{PM_{calculated} - PM_{constraint}}{PM_{constraint}} \right), \quad (6.10)$$

where α or β is set to 1 when the value of NOx or PM emissions, respectively, exceeds the constraint. Either α or β is set to 0 when the value of NOx or PM emissions, respectively, is less than the constraint. This function is identical to the ISFC merit function above when these emission values are less than the constraints, but the merit score is penalized when the emissions exceed constraints. This function is written to be continuous when the emissions levels exceed the constraints.

Multi-Mode Merit Function

You can optimize or interrogate possible designs based on each possible model's performance at multiple operating conditions, or modes. For instance, you can make CONGO evaluate an engine design based on the results at idle, power, and speed load conditions, which you could set up as a three-mode CONGO case. Using the inputs in *case.in*, you can define the number of modes to evaluate, as well as the directories containing the CONVERGE input files needed to simulate the model at different operating conditions.

The performance of the model at a particular mode is called *fitness*. Using *case.in*, you can specify the weight of each mode fitness. CONGO will use these *fitness* calculations and weights to generate an overall merit for the multi-mode GA or DoE as follows:

$$merit = \frac{1}{\frac{1}{weight_1 \times fitness_1} + \frac{1}{weight_2 \times fitness_2} + \dots + \frac{1}{weight_n \times fitness_n}}. \quad (6.11)$$

User-Defined Merit Function

If you need a merit function to more specifically address the needs of your case, you can create a user library to calculate the merit function (written in C coding language). This user library (*libcongo.so*) works with CONGO the same way a UDF works with CONVERGE.

CONGO/CONVERGE Communication

Within the CONGO root directory, you must save a CONVERGE seed directory. The name of this directory must be consistent with that specified as the *dir_name* in *case.in*. You must create the CONVERGE seed directory the same way you would a normal CONVERGE Case

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

Directory. Then, in the appropriate input file (e.g., *parcel_introduction.in*, as shown below) in the CONVERGE seed directory, you must replace the value of each model parameter (e.g., *start_inject*) to be manipulated by CONGO with the appropriate marker (e.g., *GA_START_INJ*). These markers you place in the CONVERGE input files must be consistent with the marker names you specify in *case.in*.

CONGO automatically parses CONVERGE input files. You must set up the input files correctly with markers in order to facilitate the search-and-replace for variables of interest. CONGO parses these files by finding and replacing markers in the input files. For example, you need to redefine the *start_inject* variable in *parcel_introduction.in*:

```
injections:  
  - injection:  
    injection_control:  
      start_time: -25.0
```

as

```
injectors:  
  - injector:  
    injection_control:  
      start_time: GA_START_INJ
```

in *parcel_introduction.in* in the seed directory.

You must create a unique name for each marker. Verify that the name of one marker does not appear within the name of another marker. For instance, if you name two markers *GA_START_INJ* and *GA_START_INJ_EMBEDDING*, CONGO will identify the string *GA_START_INJ* and parse in a value in the *GA_START_INJ_EMBEDDING* marker. This process would lead to the undesirable result of a value that resembles *-25_EMBEDDING*, which will cause an error.

Additionally, you must also define two markers in *inputs.in* to identify the individual and generation for CONVERGE. Prepare the *inputs.in* file as shown in the figure below.

```
..  
.  
ga_control:  
  ga_flag: 1  
  ga_individual: GA_INDIVIDUAL  
  ga_generation: GA_GENERATION  
.  
.
```

Figure 6.3: An excerpt of *inputs.in* with relevant file parsing parameters.

Place the CONVERGE input files in a directory (*input_files*) for use by CONGO during the GA. CONGO copies all files from this directory into each run directory before parsing. This input file directory should also contain an appropriate CONVERGE executable, or a link to the correct path thereto. Refer to the CONVERGE 3.0/3.1 Getting Started Guide (<https://convergecfd.com/support/getting-started-guide>) for more information.

CONVERGE Output for GA

Configure the CONVERGE simulations to create a file containing performance variables for the GA using a user-defined function (UDF). Designate one consistent output file name for both CONVERGE and CONGO, such as *ga_output.#-#*.

Customize the *ga.c* example UDF to create the variable outputs of interest. Compile the UDF and save the library (*libconverge_udf.so*) in the directory with CONVERGE. Refer to the CONVERGE 3.1 UDF Manual for information about modifying and compiling UDFs.

You need to enable several flags in the CONVERGE input files *inputs.in* and *udf.in* in order to run a GA or DoE that uses a UDF. These flags are shown in Figures 6.4 and 6.5 below.

```
.  
.ga_control:  
    ga_flag:          1  
    ga_individual:   GA_INDIVIDUAL  
    ga_generation:  GA_GENERATION  
.  
.feature_control:  
    udf_flag:        1  
.
```

Figure 6.4: An excerpt of *inputs.in* with relevant GA flags.

```
.  
.user_event_flag:          0  
user_ga_merit_flag:        1  
user_piston_position_flag: 0  
.  
.
```

Figure 6.5: An excerpt of *udf.in* with relevant GA flags.

CONGO Output Files

CONGO generates output in two places: in the [individual run directories](#) of the CONVERGE simulations, and in the [main CONGO folder](#). CONGO uses the data written to the individual run directories to create the output files in the main CONGO folder.

Running CONVERGE with a UDF

We recommend that you become familiar with running CONVERGE with a user-defined function (UDF) before attempting to run CONGO. Refer to the CONVERGE 3.1 UDF Manual for more information.

To run CONVERGE with a UDF, set *inputs.in* > *feature_control* > *udf_flag* = 1 and activate the relevant flag(s) in *udf.in*. Additionally, you need to set the *LD_LIBRARY_PATH* environment variable to point to the UDF directory, which contains the *libconverge_udf.so* library. This library path should be relative to the run directories for the GA, since CONVERGE is executed from within the individual run directories. For example, the UDF directory should be in the base directory of the GA and you should set *\$LD_LIBRARY_PATH* = *../udf*

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

To set the library environment variable using the bash shell command-line interpreter, type:

```
export LD_LIBRARY_PATH=../udf
```

To set this variable using the c shell command line interpreter, type:

```
setenv LD_LIBRARY_PATH ..\udf
```

libcongo.so

You can configure a *libcongo.so* to work with CONGO as a library for generating parameterized geometry (such as a piston bowl profile) or a rate-shape (such as a fuel injection profile). You can also use *libcongo.so* to program complex merit functions. CONGO looks for *libcongo.so* file in the working directory.

Run Directories

CONGO uses the CONVERGE seed directory to create individual run directories for a GA or DoE. The input files in these individual run directories will contain all of the input files in the seed directory, but instead of the markers designated for the model parameters CONGO will vary, CONGO has populated the values for each run (for a DoE) of the first generation (for a GA) according to the *value* (DoE) and *min* and *max* (ranges in GA) you specify for these parameters in [*case.in*](#).

Before running the full CONGO case, first perform a test run. A test run is useful for ensuring CONGO is properly configured before expending significant computational resources on the actual CONGO run. To perform test run, setup CONGO as you normally would, but set the *test_run_flag* = 1 in [*execute.in*](#). Then run CONGO as described in the [Running and Restarting CONGO](#). This test run will create run folders for each run (of the first generation, for a GA), containing the input files with model parameters populated by CONGO. These folders will contain sample, unpopulated CONGO output files, *ga_output.<run#>-<gen#>* and *param.<run#>-<gen#>*.

When you are satisfied with the CONGO setup, you can then execute the full CONGO run, which will create CONVERGE input files and launch the runs in each directory. For a GA, it will create these output files for each generation of each run. CONGO uses the output from the CONVERGE output files in the individual run directories to calculate the merit of each run, which it then uses to create the CONGO written to the CONGO root directory (e.g., *congo_max.out*, *congo_micro.out*, *output.out*).

Scripts

CONGO contains several utilities, which are described below.

- *clean.sh* will remove all files from a previous CONGO run. Use *clean.sh* if a run has been stopped and you want to start the GA from the beginning. You must remove several files before restarting CONGO, so this utility can speed up the process of starting a new run of CONGO.
- *clean_restart.sh* will remove files to allow a restart of the GA, but it also preserves the results of generations that are already complete.

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

- `all_kill.sh` will kill all CONVERGE runs currently in progress using the `mkill` and `pykill` commands. Contact Convergent Science, Inc. to determine the availability and compatibility of the `all_kill.sh` script. You will need to edit this script based on the details of your CONVERGE installation.
- `status.sh` will display the current status of all of the GA runs in progress.
- `monitor_runs.sh` will automatically monitor CONVERGE runs for crashes. If a crashed run is detected, this script will create an output file for the crashed case. You must supply a template for the output file (named `replace.outdata`) in the GA directory. A flag will appear on the first line to indicate that the run crashed, followed by performance variables that will produce a very low merit score for this individual. See a list of runs that crashed by viewing the `replace.log` file. The `replace.log` is updated every time a crashed run is detected. You can view the very low merit given to crashed runs by opening the `perform.out` file in a text editor.

Running and Restarting CONGO

Starting CONGO

After defining the experiment in [`case.in`](#) and configuring the other required input files, you can start a CONGO GA by typing the following into the command line:

```
./congo
```

To start CONGO in the background and record the log in a file, type:

```
./congo > logfile &
```

Stopping CONGO

To end a CONGO run, kill the CONGO process. You must also manually kill any CONVERGE runs in progress and any scripts that are being run by CONGO (*i.e.*, `monitor_runs.sh`).

Restarting CONGO

After a GA simulation has started, CONGO automatically creates the restart file `restart.in`. CONGO updates this file automatically after each generation. If CONGO is stopped before results are collected for the current generation of CONVERGE simulations, a recollect option is available. Set `recollect_data_flag` in [`execute.in`](#) to 1 before this initial CONGO run.

Once the CONVERGE simulations are complete, you can re-run CONGO with the data recollection option activated. With the data recollection option activated, CONGO will collect the results of the current generation, create the next generation, and update the `restart.in` file. When the data recollection option is on, CONGO will not start the simulations for the next generation; use this option only to collect information from the last generation of previously executed simulations. After the data recollection step is complete, you can restart CONGO with [`execute.in`](#) > `recollect_data_flag = 0` and with [`congo.in`](#) > `restart_flag = 1`.

Chapter 6: Optimization Techniques

CONGO Optimization and Model Interrogation Running CONGO

While a GA is in progress, you may wish to alter the merit function slightly. To do so, kill the CONGO process, alter the merit function as needed (either in [merit.in](#) or in the user library used to calculate the merit function), and then restart CONGO. Because the formulation of the merit function has changed, the merit of the best-so-far individual will be different and you must update it in the *restart.in* file. You can change the elite (best-so-far) merit in *restart.in* file to correspond to the new merit function before a restart. Then, the GA will continue with a different calculation, and CONGO will assess the best-so-far individual based on the new merit value. Note that the merit in the max (*congo_max.out*) and merit (*congo_micro.out*) and bestcases (*bestcases.out*) files will not be changed, so evaluate the merit listed in these files accordingly.

The *restart.in* file contains all the DNA of every individual in the current generation of the GA, followed by the merit and DNA of the best-so-far individual. To find and modify the merit of the best-so-far individual, search the *restart.in* file for the best-so-far merit and change it as needed.

CONGO updates the *restart.in* file every generation and therefore always contains data for only a single generation. The [congo_dna.out](#) file also contains the restart information for every generation of a GA, which allows the GA to be restarted at any generation if necessary. For instance, you may find during a GA that the definition of the merit was not ideal initially and the best-so-far individual at an earlier point in the GA was actually deemed to be stronger after you correct the merit function. You can use the [congo_dna.out](#) file to restart the GA at the earlier generation with a modified merit function. To do so, find the generation containing the desired best-so-far individual in [congo_dna.out](#) and copy the contents from this generation to the *restart.in* file. Update the best-so-far merit in *restart.in* according to the revised merit function before restarting.

The format of the restart information (contained for each generation in the [congo_dna.out](#) file) is:

```
generation#
best individual #
dna of each individual in the population preceded by the individual number
merit of best-so-far individual
dna of best-so-far individual
eight numbers defining the current random number sequence
```

Chapter



7

Parallel Processing

7 Parallel Processing

This chapter summarizes the parallel processing capability available in CONVERGE, which allows simulations to be run on a multi-processor shared memory machine or across a cluster of distributed memory machines. The parallelization is written using the Message Passing Interface (MPI) libraries (MPI version 1.2). CONVERGE performs all parallelization automatically based on parameters in [inputs.in](#).

CONVERGE parallelizes the solution of the mass, momentum, and energy transport equations independently from the [SAGE detailed chemical kinetics solver](#). The former requires substantial communication between processors and therefore good load balancing depends on the specifics of the setup. The detailed chemistry solution does not require inter-processor communication and parallelizes without regard to these specifics.

Note that you do not explicitly provide the number of processors as an input to CONVERGE. This information is part of the job submission process that is dependent on which MPI libraries are being used. CONVERGE performs a function call after starting to request the number of processes from the MPI libraries.

In parallel CONVERGE simulations, a single process performs all read and write operations for ASCII files. This process broadcasts and receives data from the rest of the processes using MPI.

By default, all processes perform read and write operations for binary (map, restart, and post) files. You can use [inputs.in](#) > `mpi_control` > `io_hints_filename` to adjust the I/O settings for binary files.

By default, CONVERGE uses shared memory to store one copy of common information (*e.g.*, the surface triangulation) on each node, rather than one copy for each rank. You can disable shared memory by setting [inputs.in](#) > `shared_memory_flag` = 0. CONVERGE will then store an identical copy of this information on every rank. We do not recommend disabling shared memory for any general use.

7.1 Parallelization for the Transport Equations

To parallelize the solution of the transport equations, CONVERGE partitions the grid into as many computational blocks as there are processors, while adhering to the load balance criteria. The partitioning and load balancing procedure makes use of the ParMETIS software package ([Karypis, 2011](#)).

The load balancing procedure ([Som et al., 2013](#)) moves cells between processors to achieve two goals:

1. Ensure that each processor's partition has approximately the same number of cells.
2. Minimize the amount of information that will need to be passed between processors by minimizing the block interface area (*i.e.*, group neighboring cells together on the same processor as much as possible).

These two items are evaluated simultaneously, with weighting factors applied to achieve a suitable balance between them. Placing too much importance on one goal can yield poor performance for the other. The figure below shows an example of a load-balanced eighteen-processor case modeling flow in an internal combustion engine.

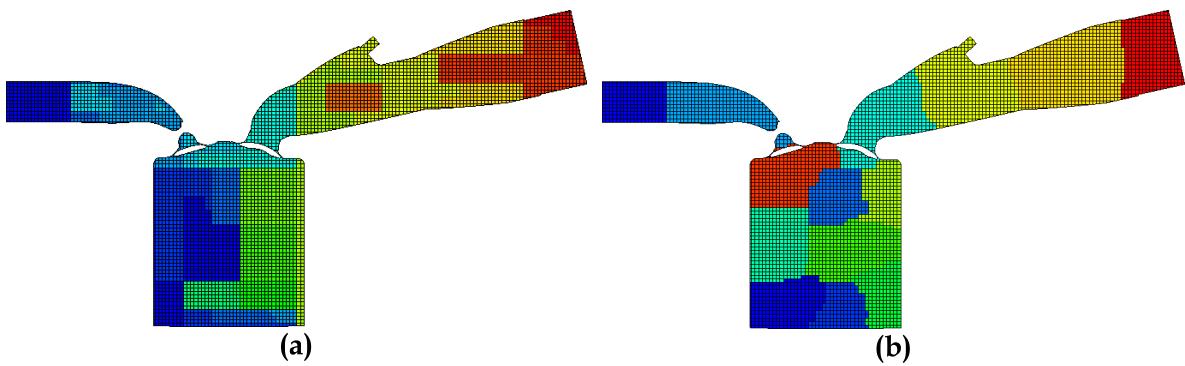


Figure 7.1: An example of automatic parallel domain decomposition of an internal combustion engine: (a) the initial domain decomposition; (b) the decomposition after load balancing by ParMETIS. Each has approximately the same number of cells per processor, but the load-balanced decomposition will have much lower inter-block communication.

The load balance algorithm assumes that your computer is homogeneous and that each processor used in the simulation has the same speed. As a result, the overall job speed of a perfectly balanced case will be limited by the speed of the slowest processor.

The load balancing algorithm in CONVERGE places the highest priority on lowering the cell count on the processors with the highest cell count. If there is a single processor with a cell count much higher than the average, the load balancing algorithm will not consider the relative sizes of the remaining computational regions. However, it will still attempt to make those regions compact (*i.e.*, have a low surface-area-to-volume ratio).

ParMETIS interprets compactness by evaluating the graph of the computational mesh. Each cell face has a graph edge running through it, connecting the two cell centers on either side of the face. ParMETIS assigns a weight to each edge, and the compactness criterion is calculated as a minimization of the overall sum of cut edge weights. To control the partitioning of paired cells, CONVERGE assigns a very high weight to the graph edge connecting the parent cell to the partial children. Because of this weight, ParMETIS will not split paired cells across multiple processors.

Neglecting communication cost, one processor with a cell count much higher than the average is much worse than having one processor with a cell count much lower than the average. For example, let us consider two scenarios with four processors and their cell counts as shown in the following table.

Chapter 7: Parallel Processing

Parallelization for the Transport Equations

Table 7.1: Cell count in four processors during a load balance cycle.

Processor number	Scenario 1	Scenario 2
Processor 1	10,000 cells	10,000 cells
Processor 2	3,000 cells	8,000 cells
Processor 3	3,000 cells	0 cells
Processor 4	3,000 cells	1,000 cells

Although it appears that Scenario 1 has much better load balance, both will give the same speed because Processor 1 is the bottleneck in both scenarios.

Aside from the load balance operation performed when a simulation begins, CONVERGE performs a load balancing operation based on several other criteria. CONVERGE will load balance when the grid moves, when embedding is set, or when the cell count changes by more than 20% in a single time-step. It will automatically perform a load balance if AMR adds so many cells to a block that the imbalance factor exceeds a threshold value ([inputs.in > simulation_control > imbalance_factor](#)), which is checked every time-step. This imbalance factor is calculated as

$$IF = 100 \frac{N_{rank,max}}{(N_{cell} / N_{ranks})}, \quad (7.1)$$

where $N_{rank,max}$ is the maximum cell count on a single rank, N_{cell} is the total number of cells, and N_{ranks} is the number of ranks. Each load balance takes approximately as long as one simulation time-step.

In order to solve the transport equations in a given cell, CONVERGE needs information from adjacent cells. In some cases, the adjacent cell will be on another processor, and that information will need to be communicated across the network. This communication can be slow relative to a processor clock, and for this reason ParMETIS attempts to minimize the interface area between computational regions. If you run on more processors, you inevitably increase this interface area and increase the amount of network traffic. Because of this increase, the computation time does not scale linearly with the number of processors, and in fact there is a crossover point at which running on more processors can actually slow the simulation.

You should run your calculation on enough processors to have some free system memory on each processor but not so many that your processors are spending all of their time communicating with each other. The optimal number of processors depends on your specific system and on the sort of calculation you are performing. For a no-chemistry simulation, if you have a fast interconnect, approximately 3,000 cells per processor is a reasonable target.

Chapter 7: Parallel Processing

Parallelization for the Transport Equations

For the same case, if you have a slower interconnect, approximately 15,000 cells per processor is reasonable.

7.2 Parallelization for the SAGE Solver

The ParMETIS load balance used to load balance the transport equations generally would not evenly divide the detailed chemistry solver's computational load among the processors. If the [SAGE detailed chemical kinetics solver](#) is used, the SAGE portion of the simulation is load balanced independently of the fluid cells. To reduce the computational expense, SAGE solves only cells that meet fuel and temperature criteria. Frequently, only a portion of the domain actually needs chemistry to be solved.

For a SAGE simulation that does not include [adaptive zoning](#), CONVERGE uses stiffness-based load balancing for the SAGE solver. For stiffness-based load balancing, at each time-step the code calculates the number of SAGE cells on each processor. The processors then intercommunicate to determine how to best share the load for that time-step. Cells are weighted by the computational cost of the SAGE calculation on the previous time-step, and these weighted cell values are distributed evenly. Keep in mind that when solving chemistry, only relevant node information is passed between processors for the parallel chemistry (the node itself is not reassigned in the process). The SAGE computations are local, and the solver does not impose strict communications requirements.

If a SAGE simulation includes [adaptive zoning](#), the chemistry cells are grouped into zones and CONVERGE solves chemistry once per zone rather than once per cell. For SAGE simulations that include adaptive zoning, stiffness-based load balancing (described in the previous paragraph) is optional. Specify `combust.in > adaptive_zone_model > stiffness_load_balance_flag = 1` to activate this feature. If you do not invoke stiffness-based load balancing, CONVERGE simply distributes the combustion zones amongst all available processors.

The SAGE solver is often more computationally expensive than the solution of the transport equations. However, it scales much better. The cells-per-processor guidelines given above do not apply if you are running SAGE. You can run efficiently on as many processors as you have available. CONVERGE has performed detailed chemistry simulations on over 4,000 processors.

This difference in scaling between non-reacting simulations and simulations using SAGE means that some cases are best run partly on a small number of processors and partly on a large number. This split is a good application of CONVERGE's [mapping capability](#).

7.3 Hardware Considerations

Parallelization inevitably brings potential hardware issues. First and foremost, you should do everything possible to maximize bandwidth, and minimize network bottlenecks and latency.

Chapter 7: Parallel Processing

Hardware Considerations

Typically, a system with InfiniBand or Omni-Path interconnects will also have full Ethernet connectivity. You should always use the fastest interconnect available on your system. On Linux systems, CONVERGE calculates a representative latency at the beginning of each simulation, and prints this information at the top of the screen output and in *mpi_latency.log*. CONVERGE does not perform this check on Windows systems.

Node-to-node latencies are sensitive to exact hardware configuration and setup, and they will be system-dependent. We have seen latencies ranging from about $2\ \mu s$ to $25\ \mu s$ for high-speed interconnects like InifiniBand and Omni-Path. If you see a latency greater than about $100\ \mu s$, this is indicative of a slow interconnect. CONVERGE may scale poorly in this case, and you should check your configuration details if your system has a fast interconnect available.

You can also print CONVERGE interconnect latency by running the *check_system* utility from the command line. Go to an empty Case Directory and enter, e.g.,

```
mpixec -np 16 converge-ompi -u check_system
```

and CONVERGE will perform the latency check and write *mpi_latency.log*, then exit. Running the latency check in this fashion does not require CONVERGE to check out a license.

When running over Ethernet interconnects (regular network cables) it is important to ensure that all machines are plugged into the same physical switch. Running on multiple switches results in a severe network bottleneck. Even a single machine on a different switch will be very detrimental to the overall speed of the calculation. If at all possible, avoid running a parallel computation over multiple switches.

The load balancer assumes that each processor used in the simulation has the same speed. If a simulation is run on processors of several speeds, the overall calculation time will be driven by the speed of the slowest machine.

A single transient hardware issue on a single processor can cause an entire job to terminate. When running a large calculation, it is good practice to increase the frequency of the restart file writes by reducing [*inputs.in* > *output_control* > *twrite_restart*](#).

You can use graphics processing units (GPUs) to accelerate certain types of simulations, as described in the [Iterative Linear Solvers](#) section. GPU support is available only for the pressure solver.

Chapter



8

Surface Preparation

8 Surface Preparation

This chapter describes several surface preparation-related items.

We recommend preparing your surface geometry in CONVERGE Studio, and the CONVERGE Studio 3.1 Manual provides information about the surface preparation options in CONVERGE Studio.

8.1 Surface Geometry

In a CONVERGE simulation, the geometry must be represented as a closed triangulated surface. CAD packages typically export geometry information in the STL file format, which does not intrinsically represent a closed surface (because STL files do not contain connectivity information for the surface triangles). If you have an STL file, you can import it into CONVERGE Studio, edit the geometry as needed, and export the geometry as a [properly formatted .dat file](#) (e.g., *surface.dat*). Alternatively, you can use the [make surface](#) utility to generate a surface geometry file for a simple surface.

The triangulated surface must be closed (*i.e.*, no gaps between the triangles) in order for CONVERGE to process the surface properly. If the surface is not closed, CONVERGE will not run. Use CONVERGE Studio to check the surface for closure and triangle orientation before exporting the surface definition file and running a simulation.

The computational expense of trimming the Cartesian fluid cells by the surface depends on the number of triangles constituting the surface. This trimming process may not be critical for a stationary surface; however, for a moving surface, where CONVERGE regenerates the grid at each time-step, trimming the Cartesian fluid cells can be somewhat costly. Therefore, for a case with moving geometry, ensure that only necessary triangles are included in the surface. CONVERGE does not limit the number of triangles that can cut a single fluid cell.

8.2 Surface Defects

This section and the subsequent subsections describe the criteria that need to be met by a surface geometry that will be used in a CONVERGE simulation. It is essential to ensure that your surface is properly configured before running a CONVERGE simulation, and you should never assume that an STL file generated by a CAD program does not contain any of these problems. We recommend using CONVERGE Studio to identify and eliminate problems in the surface geometry.

While the surface geometry needs to accurately represent the shape of the object(s) to be simulated, CONVERGE does not place restrictions on the size or skewness of the triangles that constitute the surface. Skewed triangles will not affect the numerical stability of the fluid mesh.

All surface triangles must have an inside/outside orientation. In other words, CONVERGE does not allow baffle triangles that point inside on both sides of the triangle. Any solid

immersed within the fluid geometry must have a finite thickness so that the normal vectors of the triangles constituting the solid point to the fluid (inside) on one side and to the solid (outside) on the other.

The subsections that follow describe the types of surface defects (open edges, nonmanifold edges, intersecting triangles, overlapping triangles, and normal orientation inconsistencies) that must be eliminated before running a CONVERGE simulation.

Open Edges

In a closed triangulated surface, each triangle shares each of its three edges with exactly one other triangle (except in some [conjugate heat transfer](#) simulations). If an edge is part of only one triangle, that edge is called an open edge.

The figure below shows examples of open edges. In (a), a series of open edges looks like a hole. In (b), open edges make the surface appear incomplete. CONVERGE will not run if there are open edges in the surface.

You can often repair open edges by compressing vertices or by adding triangles to fill any gaps. You can simulate multiple independent, closed regions in one simulation, for example, simulating flow over a sphere immersed in a box. Both the box and sphere must be closed shapes, but the two surfaces do not need to be connected.

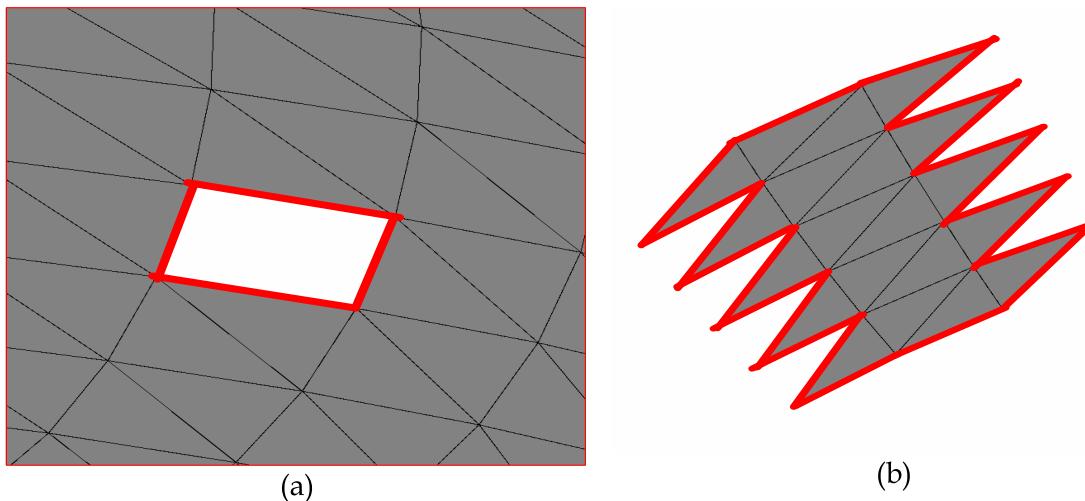


Figure 8.1: Two examples of open edges. The edges highlighted in red indicate edges that are part of only one triangle, resulting in a surface that is not closed.

Nonmanifold Edges

With the exception of special conjugate heat transfer cases, every edge in the geometry needs to be shared by exactly two triangles. If more than two triangles share a common edge, thus forming a T-junction, this edge is a nonmanifold edge.

Chapter 8: Surface Preparation

Surface Defects Nonmanifold Edges

It is impossible to have distinct inside and outside directions when a surface contains nonmanifold edges. Figure 8.2 shows a two-dimensional example of nonmanifold edges with line segments instead of triangles. Note that in Figure 8.2 (a) the geometry has a distinct inside and outside (*i.e.*, all normal vectors point inside). In Figure 8.2 (b), however, the presence of the red segment precludes any proper orientation of the geometry (*i.e.*, there is no proper inside/outside orientation of the red segment that will be consistent with the inside/outside orientation of the rest of the geometry). In this two-dimensional example, the points where the red segment intersects the original closed geometry create T-junctions each consisting of a single point, which corresponds to a T-junction edge in a three-dimensional case.

Figure 8.3 shows a three-dimensional representation of nonmanifold edges and the triangles that form them.

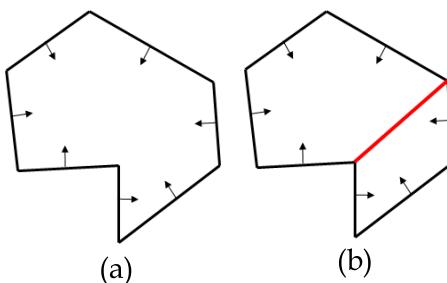


Figure 8.2: 2D example of the multiple edge sharing problems. (a) A properly defined 2D geometry with a single inside/outside orientation. Each segment shares an intersection point with exactly one other segment. (b) The original surface plus an additional segment that makes it impossible to define a proper inside/outside orientation. This new geometry has two improper 3-way intersections.

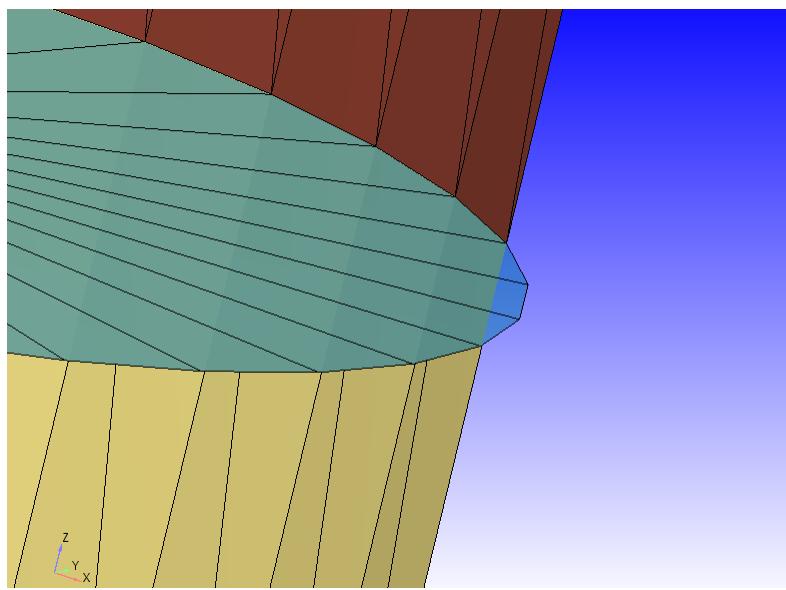


Figure 8.3: A cut-away, three-dimensional view of nonmanifold (or T-junction) triangles as they appear in CONVERGE Studio. The blue-green triangles have been rendered semi-transparent for clarity. Some brown and tan triangles have been hidden to show the T-junction.

Intersecting Triangles

CONVERGE cannot use a surface containing triangles that intersect because an intersection would violate the concept of a consistent inside/outside orientation of the surface. This type of problem can be difficult to detect in a surface. Figure 8.4 shows a two-dimensional example. Figure 8.5 shows how CONVERGE Studio graphically represents parallel triangles that have been moved too close together and thus are considered intersecting.

At a minimum, visually inspect the surface to ensure that there are no surface intersections. For example, when modeling valves, verify that the valves are not closed too far, so that the valve angle overlaps the valve seat, as shown in Figure 8.5.

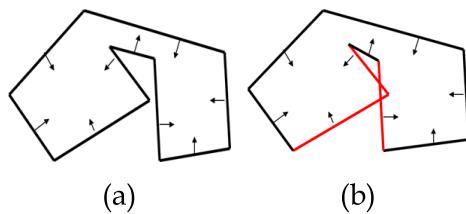


Figure 8.4: (a) All points in space are either inside the surface or outside the surface. (b) Intersecting surfaces cause some volumes to violate the above criteria.

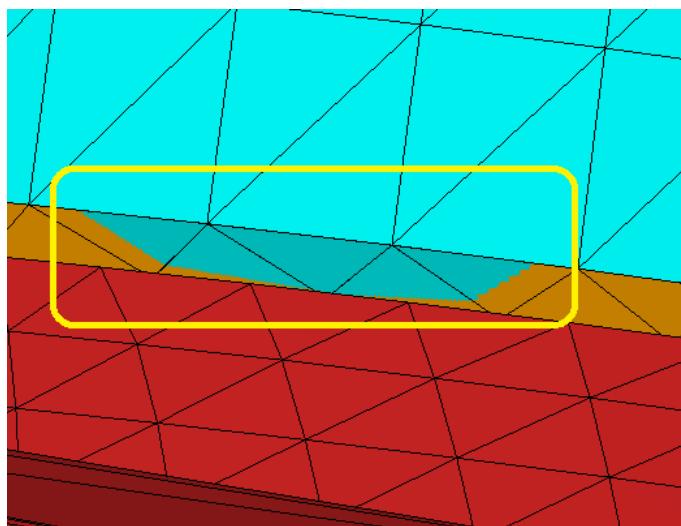


Figure 8.5: CONVERGE Studio depicts parallel triangles that are too close with inconsistent coloration. In CONVERGE Studio, go to Geometry > Transform > Translate to move these intersecting triangles.

Overlapping and Sliver Triangles

Overlapping and sliver (distorted aspect ratio) triangles are some of the most common problems found in triangulated surfaces generated from CAD packages, and they cannot be easily seen. These problem triangles do not result in open edges or nonmanifold edges because the connectivity forms a manifold volume.

The figures below illustrate the concept of overlapping triangles and sliver triangles. Figures 8.6 (a) and (b) show two acceptable two-dimensional (*i.e.*, all vertices are co-planar) triangle configurations. The triangles in the example are AEC, ABE, ACB, and BCD, with all of their normal vectors pointing out of the page. Both surfaces are considered properly triangulated (with respect to overlapping triangles) and could be part of acceptable surface files in CONVERGE.

Chapter 8: Surface Preparation

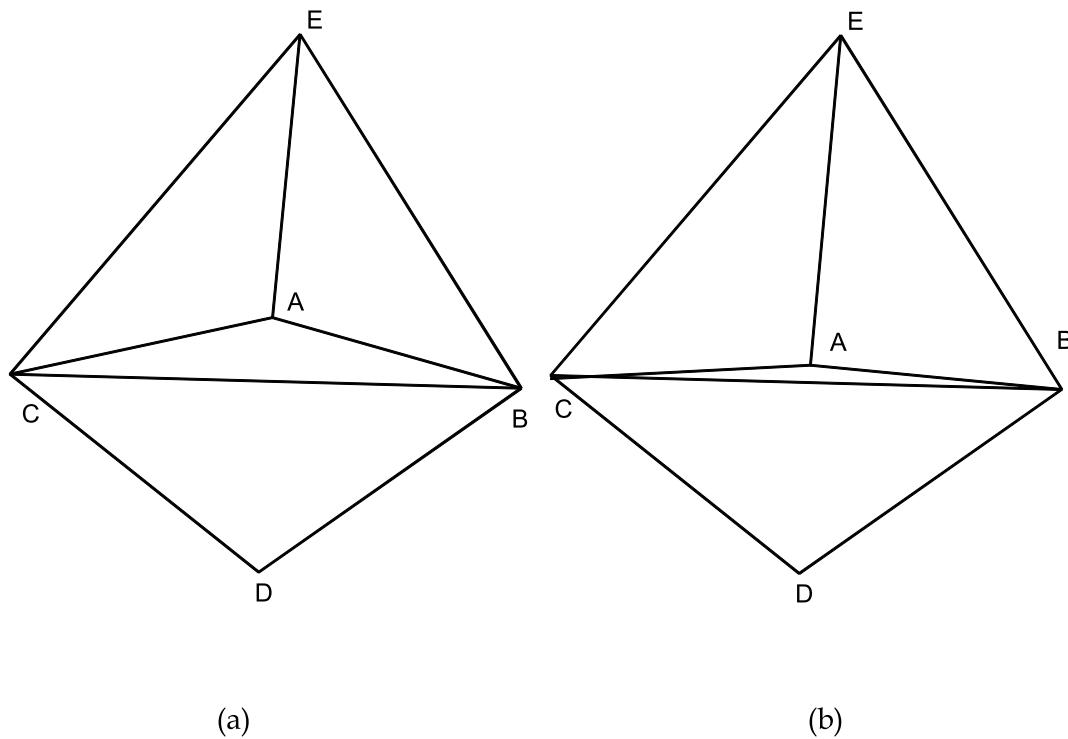


Figure 8.6: Four triangles formed by coplanar vertices A, B, C, D, and E. Both (a) and (b) are acceptable surface triangulations.

If vertex A is located on segment BC, as shown in Figure 8.7 (a), then vertices A, B, and C are collinear and thus triangle ABC has zero area. This zero-area (distorted aspect ratio) triangle will have a random normal vector and will cause errors in CONVERGE.

Figure 8.7 (b) shows the same set of triangles with vertex A located below segment BC. In this case, triangle ABC has flipped and now its normal vector points *into* the page instead of *out of* the page. In essence, the entirety of triangle ACB and small portions of triangles AEC, ABE, and BCD are all occupying the same space, generating a triangle intersection, which CONVERGE does not allow.

Chapter 8: Surface Preparation

Surface Defects

Overlapping and Sliver Triangles

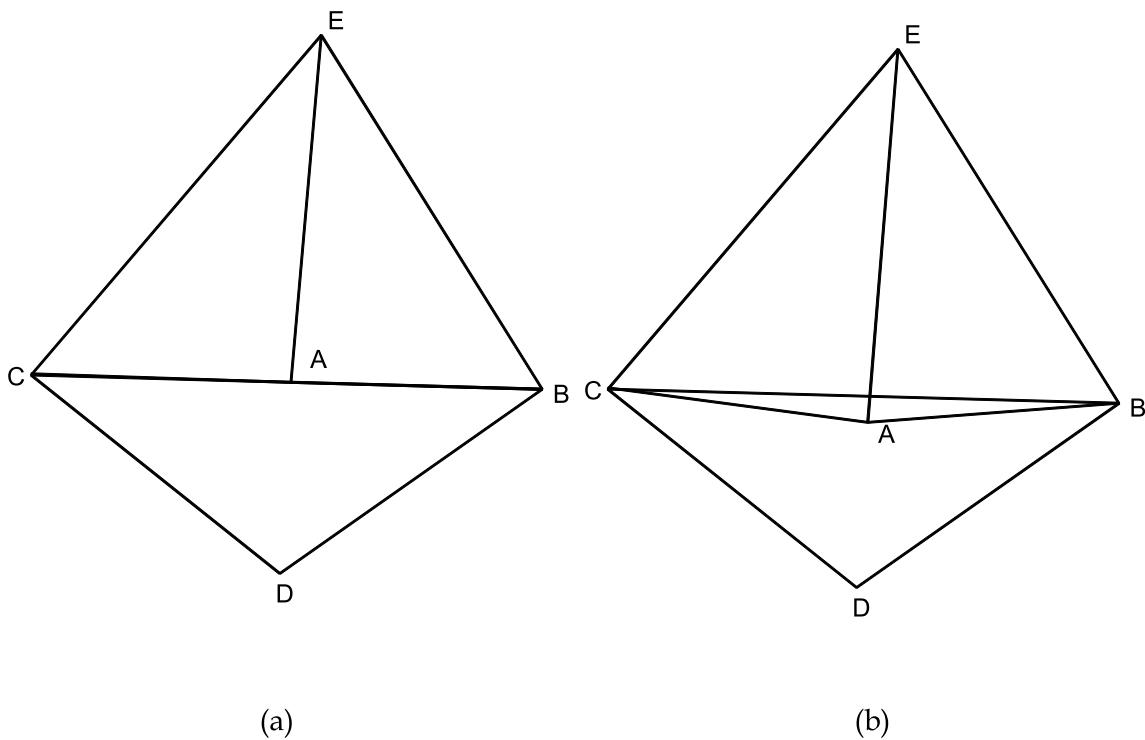


Figure 8.7: The same triangles as in Figure 8.6, but in case (a), vertex A is collinear with segment BC, resulting in a zero-volume (distorted aspect ratio) triangle ACB, which will have an undefined normal vector. For case (b), triangle ACB is flipped and has a normal vector pointing opposite those of the other three triangles. Neither example is permitted in the CONVERGE surface geometry file.

In Figure 8.6 (a) and (b), all segments between triangles have edge angles of 180° . In Figure 8.7 (a), the edge angle associated with segments AB, BC, and AC is 0 degrees. An edge angle of less than 60 degrees usually indicates a problem area in the surface. However, many geometries will have sharp features consisting of triangles that will be highlighted by a 60 -degree feature angle. Examine areas with a highlighted feature angle to determine if the geometry actually has a sharp feature in the surface, or if the geometry is flat and the triangulation of the surface has resulted in an inappropriate situation such as in Figure 8.7 (b). You can repair these triangles by moving (translating) vertex A, stitching A to B or C, or retriangulating the system as shown below in Figure 8.8. The retriangulation creates triangles ABE, AEC, ACD, and ADB.

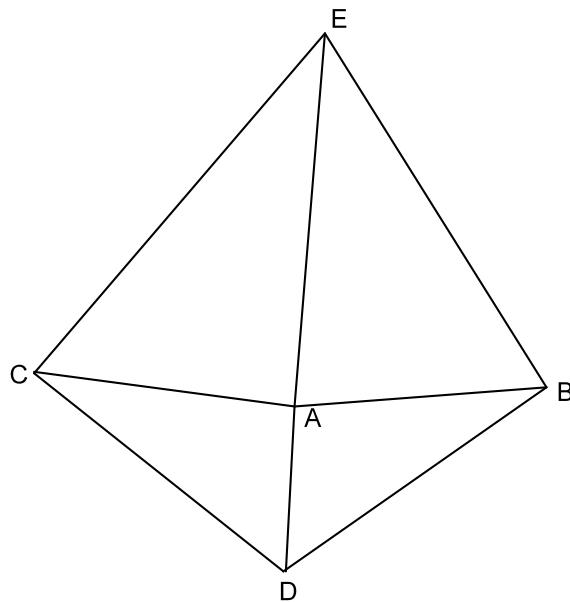


Figure 8.8: A retriangulation of Figure 8.7 (b) yields an acceptable triangulation of the surface.

Normal Orientation

CONVERGE requires the surface to have a properly defined inside (solved fluid side) and outside (non-fluid side). The normal vectors of all triangles (as defined by the right-hand rule) must point toward the fluid.

One extra normal orientation step is required when a surface consists of multiple independent surfaces, as in the case of an object completely immersed within a fluid domain, such as a sphere immersed in (but not connected to) a box. For the immersed sphere, the normal vectors of the triangles must point away from the center of the sphere, since the solved fluid portion of the domain is outside of the sphere. The normal vectors of the box triangles must point inward, toward the sphere.

8.3 Periodic Surfaces

To use the periodic surfaces option, CONVERGE requires that the two surfaces have exactly the same perimeter geometry and must both be designated as [PERIODIC](#) boundaries in [boundary.in](#).

On the interior of curved surfaces, vertices do not have to match exactly on both sides of the periodic surfaces (*i.e.*, there is a one-to-one correspondence of vertices on both faces such that the geometry on the matched periodic surfaces is exactly the same), but configuring the geometry in this manner may aid in complex cases. For engine sector cases, using the [make_surface](#) utility is the best way to ensure these surfaces are identical.

8.4 Boundary Identification

Once the surface has been checked for problems and corrected, the last step is to flag the triangles to their appropriate boundaries. The actual description of the boundary conditions is not part of the surface preparation (although you can prepare the boundary input file using CONVERGE Studio), but is instead in the [boundary.in](#) file. All that is required in the surface preparation is to flag the triangles according to the corresponding boundary id in the associated [boundary.in](#) file. As described previously in the [Surface Geometry](#) section, each triangle in the surface geometry file is defined by three vertices and a boundary ID.

It is often useful to divide up a single boundary type into multiple boundaries to allow for more control and flexibility when specifying grid refinement. For example, even though a valve could be represented as a single boundary type, additional embedding near the valve angle may improve the precision of the results in this area. To achieve increased precision, flag the valve angle as a separate boundary type from the rest of the valve. Then, when defining embedded regions, you can dictate additional layers of embedding on this portion of the valve.

The boundary IDs used do not need to be sequential and there is no limit on the number of boundary IDs that can be specified. Any boundary IDs assigned to triangles in the surface geometry file that do not have a corresponding entry in the [boundary.in](#) file will cause the code to abort and give an error.

8.5 Moving Boundaries

[Moving boundaries](#) are flagged in the same manner as stationary boundaries. Before executing a simulation with moving boundaries you must ensure that your geometry is prepared for motion. The following subsections describe potential problems associated with boundary motion.

Surface Defects Caused by Motion

Three of the surface defects described earlier ([surface intersection](#), [overlapping triangles](#), and [sliver \(zero-area\) triangles](#)) can occur when boundaries are moving during a simulation. The boundary locations as they appear in the exported surface geometry file represent the actual geometry at only one instant in time. During the simulation, CONVERGE will move the vertices on moving boundaries according to the motion defined in the [boundary.in](#) file. Therefore, you must ensure that no surface defects will occur throughout the path to be taken by the moving surfaces.

To avoid sliver or overlapping triangles, verify that no stationary vertices will be in the path of a moving surface. This problem often occurs on boundaries adjacent to both a moving surface and a stationary surface.

For example, in an engine, you must define the cylinder liner boundary as a ring of vertices around the periphery of the head connected by a single row of triangles to an identical,

Chapter 8: Surface Preparation

Moving Boundaries Surface Defects Caused by Motion

parallel ring of vertices on the periphery of the piston face. Figure 8.9 below shows an example of a single-row cylinder liner.

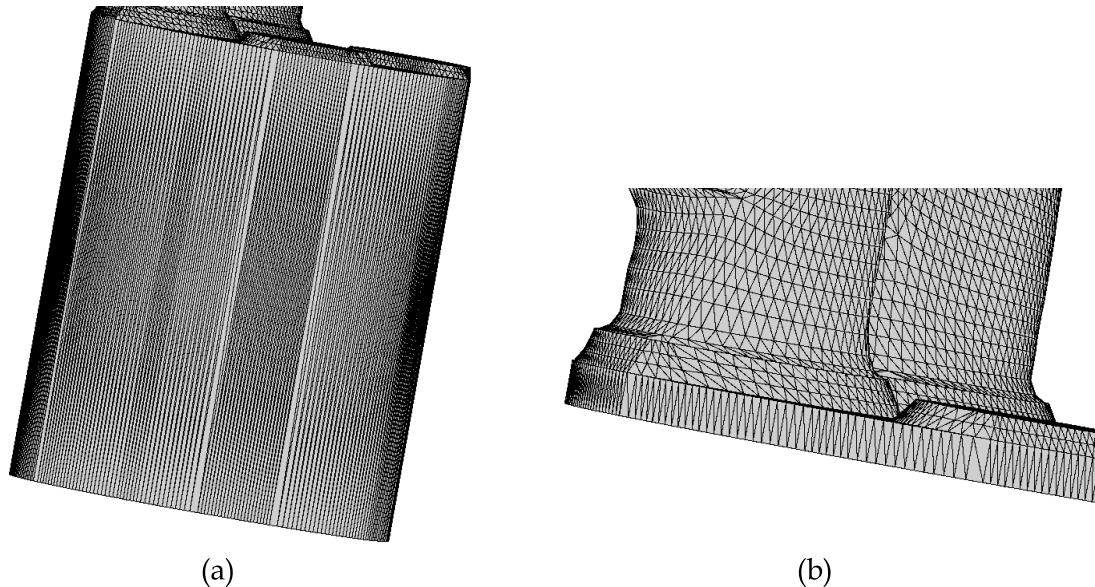


Figure 8.9: An example of a properly configured cylinder liner at (a) bottom dead center and (b) top dead center. For both positions, the liner surface consists of a single row of triangles.

Figure 8.10 below shows a triangle configuration that will cause an error during boundary motion.

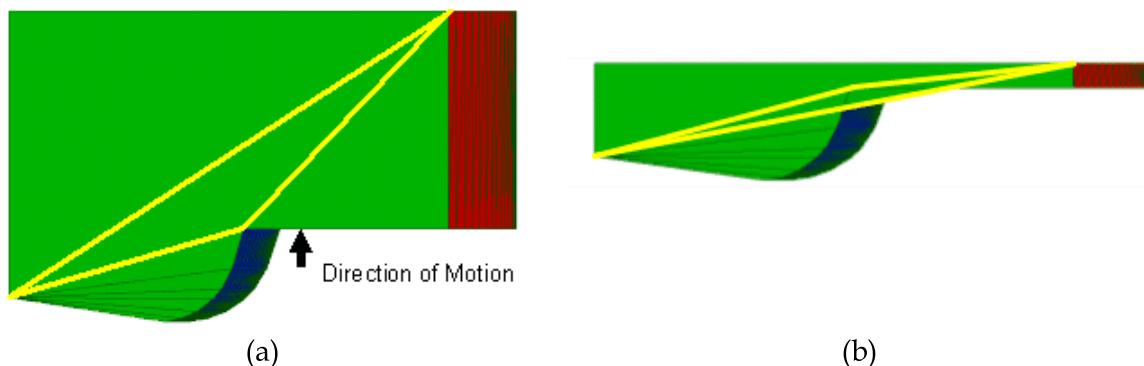


Figure 8.10: An example of a moving piston engine sector case with improper triangulation. The problem triangle is outlined in yellow. The surface in (a) is valid, but, when the piston nears top dead center, the triangle flips as shown in (b), causing a surface defect.

Figure 8.11 below shows an acceptable surface triangulation. Notice that no triangles flip as the piston nears top dead center.

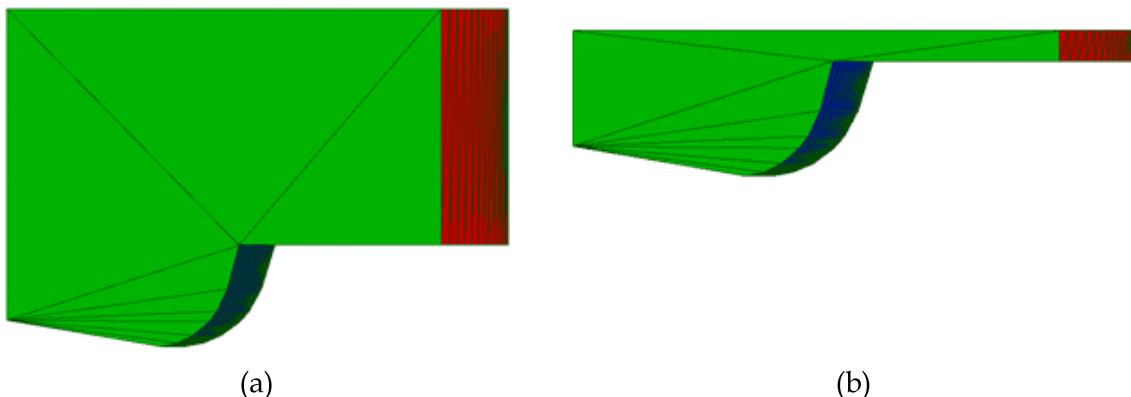


Figure 8.11: A modified and allowable triangulation of the piston cylinder sector case. The surface triangulation is valid both in (a) and (b).

You must also be aware of intersecting triangles. At no time during the simulation can triangles overlap or intersect. At any location where, in the actual object, two surfaces come in contact with each other, you should prepare the surface geometry with a minimum separation between these two portions of the surface.

A good example of potentially overlapping triangles is the seating of valves in an engine. In the actual cylinder, the valves close tight against their seats. In the CFD simulation, the triangles on the valve boundary must not contact the triangles on the seat boundary, since this contact would cause intersecting triangles. There is no minimum distance required between two triangles. As long as the triangles do not intersect, the geometry is valid.

Typically, the reason for bringing two surfaces together is to close off flow between two regions. Since CONVERGE does not allow triangles to overlap or intersect, you cannot close off flow by bringing two regions together. Instead, use [OPEN and CLOSE events](#) to allow or prohibit flow between adjacent regions.

Surface Alignment Issues Caused by Motion

Boundary motion results in three different types of triangle motion: moving, non-moving, and intermediate. A triangle's motion type depends on how each vertex in the triangle moves (or does not move).

Vertices are shared by multiple triangles. If any of the triangles attached to a vertex are moving, the vertex also moves. Moving triangles consist of vertices that are all moving along the same vector. Non-moving triangles consist of vertices that are all stationary.

A triangle with intermediate motion is one that is composed of both moving and non-moving vertices. This type of triangle is flagged to a non-moving boundary, but it shares one or two vertices with a triangle on a moving boundary.

Inspect these intermediate triangles carefully when preparing the surface. The normal vector on an intermediate triangle must not change as the boundaries move during the simulation.

To keep the normal vector constant during boundary motion, the moving vertices of the intermediate triangle must move within the plane of the triangle. That is, the direction of motion of the moving vertices in an intermediate triangle must be perpendicular to the normal vector of the triangles.

An example of intermediate triangles are on a cylinder liner in an engine. The piston moves in only the z direction, and the head is stationary. Figure 8.-2 in the previous section shows how the triangles on a cylinder liner deform (stretch and shrink) during the piston motion (the triangles on the cylinder liner move with intermediate-type motion). To keep the normal vector constant during the geometry motion, the vertices on the periphery of the head and the piston (the vertices that make up the intermediate triangles on the cylinder liner) must be aligned in the x and y directions. Ensuring proper vertex alignment will prevent the triangles from becoming skewed as the piston moves in the z direction and will minimize the potential loss of region volume.

This type of surface defect can result in two problems. First, if the vertex misalignment is slight, mass may not be perfectly conserved. Second, if the misalignment is severe (i.e., if new fluid cells will be created due to the non-planar effects of the intermediate triangles), it could lead to problems initializing the new cells. CONVERGE will check alignment when a simulation starts and will write a warning if alignment exceeds a predetermined tolerance.

8.6 Sealing of Boundaries

You can use boundary sealing to remove gaps between close objects and control the flow from one part of the geometry to another. Unlike [open and close events](#), which control the flow between different regions, sealing can be used to control the flow between different parts of the same region (in addition to the flow between regions). Sealing also allows you to specify the seal location directly and to remove parts of the geometry that are not needed to describe the computational domain.

For example, one common use for boundary sealing is to prevent flow between the combustion chamber and the intake and exhaust ports in a two-stroke engine. In an actual two-stroke engine, the body of the piston is positioned tightly against the cylinder wall, thereby preventing flow through the ports attached to the cylinder wall when the piston nears top dead center. In CONVERGE, however, two parts of the geometry cannot intersect with each other (except in the special case of sealing triangles), so you must reduce the diameter of the cylinder. As shown in Figure 8.12, this creates an unrealistic crevice volume between the cylinder liner and the piston skirt. By placing seals in the locations indicated by the red arrows, you can prevent flow from the cylinder to the intake and exhaust ports and exclude the crevice volume from flow and volume calculations. The computational domain after seal formation is shown in Figure 8.13.

Chapter 8: Surface Preparation

Sealing of Boundaries

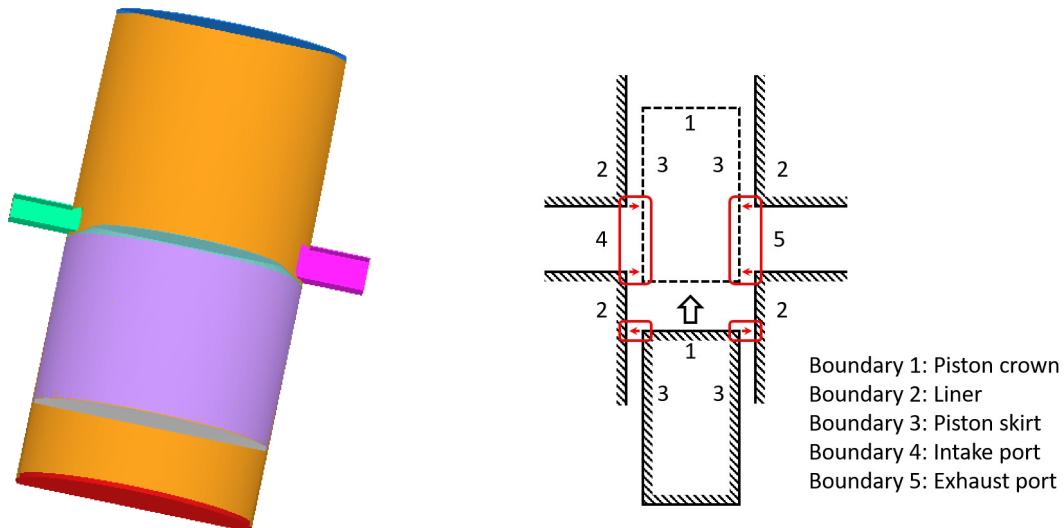


Figure 8.12: Full two-stroke engine geometry (left) and schematic showing seal locations (right). In the schematic, the solid lines for the piston crown (boundary 1) and the piston skirt (boundary 3) represent the piston at bottom dead center. The dashed lines represent the piston as it is compressing the air-fuel mixture.

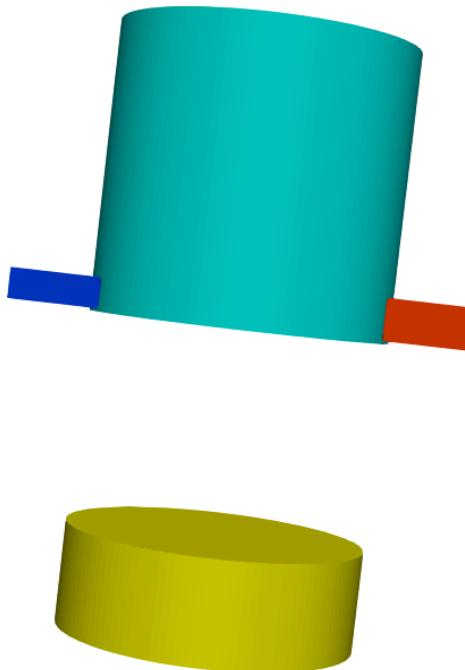


Figure 8.13: Computational domain for two-stroke engine after seal formation.

Sealing Methods

Seals are a special type of [INTERFACE](#) boundary with rules for handling surface intersections. They are infinitesimally thin double-sided walls. Unlike other boundaries, seals are allowed to intersect with other surfaces and to have open edges. You can create seals in two ways: with the seal configuration panel in the *Geometry* dock of CONVERGE Studio or with seal INTERFACE boundaries in [boundary.in](#).

When you use the seal configuration panel, the seal information becomes part of the [surface geometry file](#) that you export from CONVERGE Studio before running the simulation. At runtime, CONVERGE will attempt to extrude sealing triangles based on your setup. These triangles are assigned to auto-generated boundaries. This type of seal is quick to set up and is compatible with sealing cases from versions prior to CONVERGE 3.1. However, when you use this method, CONVERGE might fail to create seals for some configurations.

When you specify seal INTERFACE boundaries in [boundary.in](#), you create the sealing triangles directly in CONVERGE Studio and assign them to an INTERFACE boundary for which the forward and reverse sides have *type = SEAL*. This method supports a wider range of sealing configurations and gives you more control over seal placement and geometry removal. However, it is not compatible with sealing cases from versions prior to CONVERGE 3.1.

Both methods result in sealing triangles that extend from the seal origin to one or more seal-to boundaries, forming a closed loop. For example, if we consider the bottom seal in Figure 8.12 (the seal between the piston and the liner), the piston crown is the seal origin and the liner is the seal-to boundary. Sealing triangles inherit their motion and boundary conditions from the seal origin.

The table below provides a detailed comparison between the two sealing methods. If your case setup requires multiple seals, you can use different methods for different seals, which allows you to mix and match these two approaches based on ease of configuration.

Table 8.1: Comparison of CONVERGE sealing methods.

Seal configuration panel	Seal INTERFACE boundaries
Sealing information stored in surface geometry file (e.g., <i>surface.dat</i>)	Sealing information stored in boundary.in
CONVERGE creates sealing triangles at runtime and assigns them to auto-generated boundaries	User creates sealing triangles and assigns them to user-specified INTERFACE boundaries
Compatible with sealing cases from versions prior to CONVERGE 3.1	Not compatible with sealing cases from versions prior to CONVERGE 3.1
All seals have the sealing tolerance specified in inputs.in > <i>grid_control</i> > <i>seal_tol</i>	Sealing tolerance is not required
Loops that are not in a plane might create overlapping triangles	Overlapping triangles can be identified and corrected

Chapter 8: Surface Preparation

Sealing of Boundaries Sealing Methods

Seal configuration panel	Seal INTERFACE boundaries
Geometry removal is tied to seal direction	User specifies which sides of the seal to keep (forward, reverse, both, or neither)
Can seal to a single boundary or a list of boundaries	Can seal to a single boundary, a list of boundaries, or any boundary
Cannot be used to create patches	Can be used to create patches

When using the seal configuration panel, you select the edges from which the sealing triangles will be extruded, the direction in which they will be extruded, and the seal-to boundary or boundaries. The direction can be parallel or orthogonal to the normal vector of the face adjacent to the selected edge, or the average of the parallel and orthogonal directions. CONVERGE creates the sealing triangles and assigns them to auto-generated boundaries, one for each unique boundary to which sealing triangles are connected at the time of their creation. The new boundary has the parent boundary's name with "(SEAL COPY)" appended to the end. The parent boundary is the seal origin for that set of sealing triangles. Since seals are double-sided walls, CONVERGE often assigns each side of the seal to a different boundary as shown in Figure 8.14.

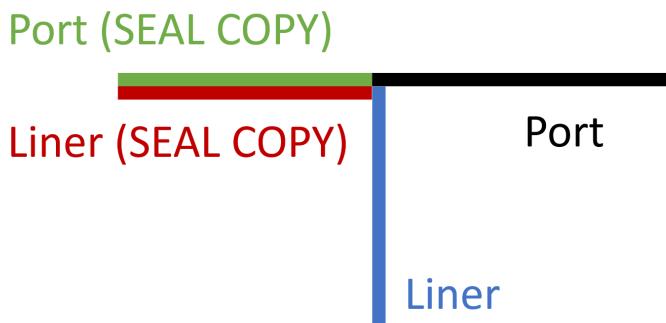


Figure 8.14: A schematic that shows new boundaries and their names. Note that the two sides of the seal overlap in reality, but they are shown separately for illustration purposes.

When specifying a seal INTERFACE boundary, you create the sealing boundary manually, as shown in Figure 8.15. Intersections and open edges are allowed for seal interfaces. You specify a seal origin and a list of seal-to boundaries for the forward and reverse sides. You also specify which side(s) of the seal to keep. At runtime, CONVERGE resolves the intersections and determines which parts of the geometry to remove.

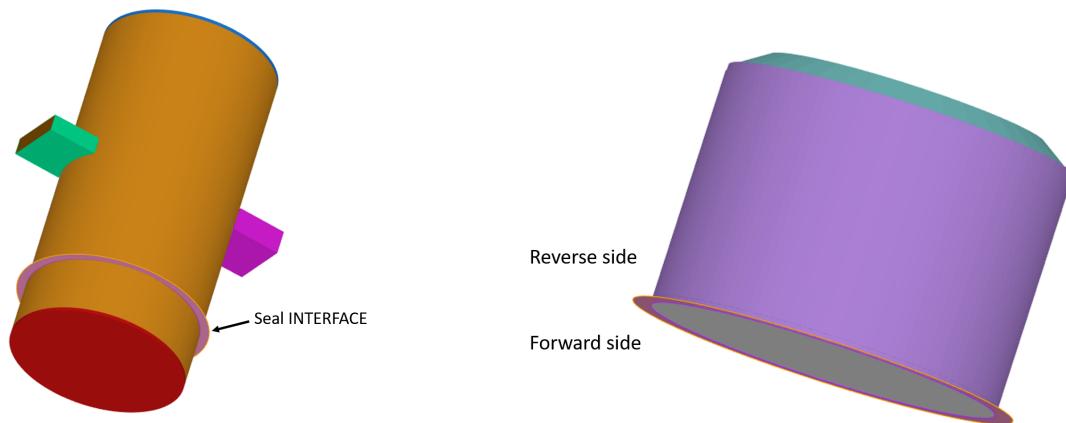


Figure 8.15: Seal INTERFACE boundary for a two-stroke engine. The left image shows the full geometry, in which the sealing triangles intersect with the liner. The right image shows only the piston boundaries and the seal interface.

Sealing Logic

At the start of a simulation, CONVERGE reads the [surface geometry file](#) and creates sealing triangles for seals created in the seal configuration panel. At each time-step, CONVERGE updates the locations of sealing triangles based on the motion of their seal origin and invokes the sealing algorithm described below.

Step 1: Remove seals that intersect with boundaries other than seal-to boundaries.

CONVERGE begins by resolving all intersections between seals and other boundaries (including other seals). Wherever a seal touches a surface, it becomes connected to that surface. Next, CONVERGE removes seals that are connected to a boundary that is not on the seal's list of seal-to boundaries. Although these seals are removed, the cuts they made on the intersecting surfaces remain. The locations of these cuts can be useful for diagnosing seals that did not form as expected.

If the simulation includes any seals created in the seal configuration panel, all seals will treat all other seals as seal-to boundaries. By contrast, if all seals are specified as INTERFACE boundaries, seal A will treat seal B as a seal-to boundary only if seal B is attached to one of seal A's specified seal-to boundaries.

Step 2: Remove seals with open edges (if necessary).

CONVERGE identifies seal boundaries with open edges. These seals are removed unless (a) they were created in the seal configuration panel with the direction specified as *Average*, or (b) they were specified as INTERFACE boundaries with [*boundary.in* > *boundary_conditions* > *boundary* > *partial_seal* = YES.](#)

Step 3: Remove seals specified to keep neither side.

CONVERGE removes seals specified as INTERFACE boundaries with [*boundary.in* > *boundary_conditions* > *boundary* > *keep_flag* = NEITHER.](#)

Step 4: Remove seals that do not have a path to a seal source.

CONVERGE removes any sealing triangles that do not have a path to a seal source through other sealing triangles of the same surface. A seal source is any non-sealing triangle that is connected to a sealing triangle before the sealing algorithm is invoked. Seals can thus seal to jagged surfaces and can seal around or into obstructions of any shape, but they cannot seal through multiple surface layers. This allows a hollow object to penetrate a seal without the interior of the object being sealed. For example, a needle can penetrate the top of a medicine vial (the seal surface), and the flow path through the needle will be left open because the interior walls of the needle are not connected to the medicine vial surface via other seals from the same surface.

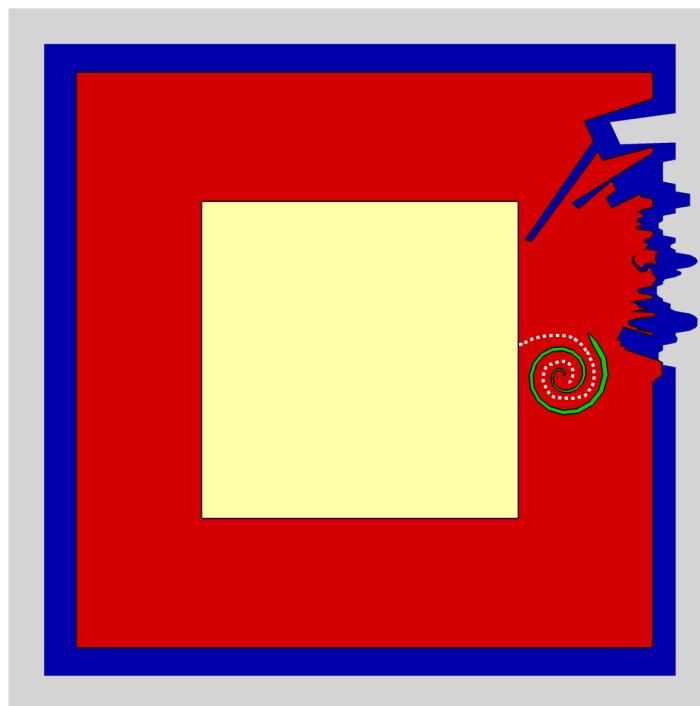


Figure 8.16: This figure illustrates seals that form to surfaces of arbitrary complexity but are bound by several rules. The yellow box in the center is composed of surface triangles. It is the seal source. The red, blue, and gray surfaces all represent portions of the seal boundary. The red portion of the seal is the intact seal, and it has sealed around a concave obstruction. None of the red sealing triangles have open edges, and all have a path through other sealing triangles back to the seal source (such as the dotted line shown in white). The blue portion of the seal depicts the portion of the seal that formed between a pair of jagged surfaces. The blue sealing triangles do not have any open edges, but they do not have an uninterrupted path to the seal source, so they will be removed. The outer edge of the seal is represented in gray. This portion of the seal does not have a path to the seal source, and it has open edges and it will be removed. Note that if any surface of the seal comes in contact with a surface that is not in its list of seal-to boundaries, the seal will break.

Step 5: Remove seals specified to keep one side.

CONVERGE removes all of the triangles connected to seals that are specified to keep only one side. This includes (a) seals created in the seal configuration panel with the direction specified as *Parallel* or *Orthogonal*, and (b) seals specified as INTERFACE boundaries with [*boundary.in*](#) > *boundary_conditions* > *boundary* > *keep_flag* = *FORWARD* or *REVERSE*. CONVERGE removes the specified side of the seal as well as the geometry to which it is connected. To prevent accidental deletion of portions of the geometry, you must carefully select the seal-to-boundaries. You also must ensure that the removed area is bounded by either a single seal or a set of seals.

Sealing Considerations and Recommendations

If a seal boundary is connected to a region/region boundary, any disconnect triangles dividing those regions at that connection will be made permanent open events. Additionally, the use of boundaries with [*boundary.in*](#) > *boundary* > *region* = *DEPENDENT* is allowed only in the first stream listed in [*stream_settings.in*](#). Boundaries cannot dynamically change the stream to which they belong.

In a simulation with an [*inlaid mesh*](#), seals cannot touch or interact with the inlaid mesh.

NEITHER seals (*i.e.*, seal INTERFACE boundaries with [*boundary.in*](#) > *boundary_conditions* > *boundary* > *keep_flag* = *NEITHER*) are useful only for seals that interact with other seals. They form a barrier visible only to other seals and can be used to control the placement of seal edges. For example, if you patch a port opening with a *NEITHER* seal, other seals that strike the *NEITHER* seal are cut at the *NEITHER* seal boundary. CONVERGE then removes the *NEITHER* seal triangles, leaving the port open to the flow. Although the other seals now have open edges, CONVERGE does not flag them for removal, even if [*boundary.in*](#) > *boundary_conditions* > *boundary* > *partial_seal* = *NO*.

We do not recommend setting *partial_seal* = *YES* for a seal with *keep_flag* = *FORWARD*, *REVERSE*, or *NEITHER*. This will cause the geometry on both sides of the seal to be removed if the seal boundary has any open edges. Similarly, we do not recommend allowing *NEITHER* seals to interact with *FORWARD* or *REVERSE* seals. This may produce open edges and lead to the removal of geometries connected to both sides of the seal.

Although the above seal-seal interactions are not recommended, several other seal-seal interactions can be essential to obtain the desired seal behavior. Figure 8.17 shows the interaction of seals near a port opening, such as might be observed when the tip of a Wankel rotor is at a port. For simplicity, in this example, all seals are specified to keep both sides unless stated otherwise. Consider the following seal configurations:

1. Seal into the port with partial seals disabled. The seal will break along the whole edge.
2. Seal into the port with partial seals enabled. The open edge of the seal will remain within the port. (Partial seals are always enabled for seals created in the seal configuration panel with the direction specified as *Average*).
3. Seal into the port with an additional *NEITHER* seal from the port to the rotor. Open edges will be produced where the seals cross, creating a divot in the red seal at the port.

Chapter 8: Surface Preparation

Sealing of Boundaries Sealing Considerations and Recommendations

4. Seal into the port with an additional *NEITHER* seal placed over the port opening. The open edge of the red seal will remain at the port opening.
5. Seal into the port with a flow-through INTERFACE boundary over the port opening. The portion of the seal to the right of the port opening, which remained in configuration (2), here will be removed because it does not have a path to the seal source (the rotor).

In this example, configurations (4) and (5) produce the desired seal behavior.

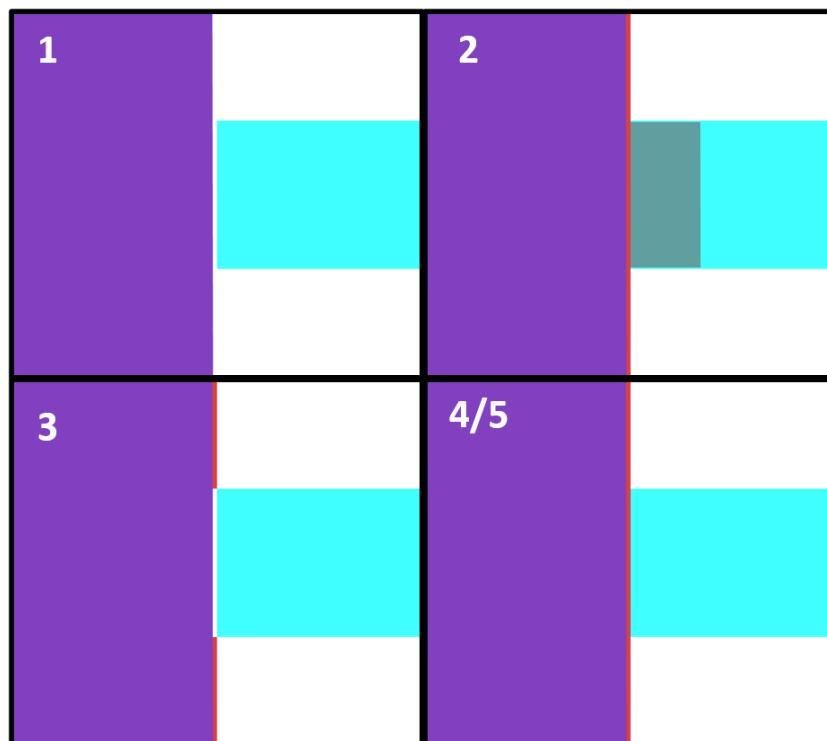


Figure 8.17: Side view of sealing for a Wankel rotor (purple) near a port (cyan). A number of seal configurations can be produced depending on seal settings and placement. The seal (red) boundary is missing in configuration (1) because it had an open edge and was removed, while it remains in configuration (2) where open edges have been allowed. In configuration (3), a *NEITHER* seal from the port toward the rotor was used to create a divot in the red seal. In configuration (4), a *NEITHER* seal marks the opening of the port but is removed by the time sealing is complete. In configuration (5), a flow-through INTERFACE marks the entrance to the port.

Chapter



9

Boundaries and Boundary Conditions

9 Boundaries and Boundary Conditions

This chapter describes boundaries and boundary conditions in CONVERGE. You must also set a boundary condition for each partial differential conservation equation at each boundary. In general, boundary conditions can be Dirichlet (*i.e.*, specified value), given by

$$\phi = f, \quad (9.1)$$

or Neumann (*i.e.*, specified value of the first derivative), given by

$$\frac{\partial \phi}{\partial x} = f, \quad (9.2)$$

where ϕ is a solved quantity (*e.g.*, pressure, energy, velocity, or species) and f is the specified value or specified derivative on the boundary. In CONVERGE, f is generally set to zero. In some cases, you can specify a boundary conditions (*e.g.*, slip or law-of-the-wall) that is a special case of a Dirichlet or Neumann boundary condition. CONVERGE also includes other options, such as the [Navier-Stokes characteristic boundary conditions](#).

CONVERGE reads boundary-related information from the [*boundary.in*](#) file, which is described in detail in [Chapter 25 - Input and Data Files](#).

9.1 INFLOW and OUTFLOW

This section describes boundary conditions for INFLOW and OUTFLOW boundaries. You can specify boundary conditions for [velocity](#), [pressure](#), [temperature](#), [species](#), [passive](#), [turbulent kinetic energy](#), [turbulent dissipation](#), and [specific dissipation rate](#).

INFLOW and OUTFLOW boundaries are similar to one another, but there are some boundary conditions and options available for only INFLOW or OUTFLOW boundaries. At OUTFLOW boundaries, you must also set boundary conditions for [backflow](#) (*i.e.*, flow into the domain).

Many INFLOW/OUTFLOW boundary conditions can vary [temporally](#) and/or [spatially](#).

CONVERGE also includes the [Navier-Stokes characteristic boundary conditions](#).

Certain combinations of boundary conditions may not be stable and should be avoided. For example, an INFLOW Neumann pressure boundary condition and an OUTFLOW Neumann pressure boundary condition may result in the drifting of the mean domain pressure. CONVERGE will warn of potential instabilities at runtime.

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW

To set up an INFLOW or OUTFLOW boundary, set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *type* = INFLOW or OUTFLOW and configure the subsequent parameters in the [*boundary.in*](#) > *boundary_conditions* > *boundary* settings block.

Velocity Boundary Conditions

[Dirichlet](#), [mass flow](#), [average velocity](#), and [Neumann](#) velocity boundary conditions are available to solve the momentum equation at INFLOW and OUTFLOW boundaries. Two additional boundary conditions—[Normal Neumann](#) and [pump](#)—are available for INFLOW boundaries. The [fluctuating INFLOW option](#), which superimposes fluctuations on the INFLOW velocity profile, is available for all boundary conditions.

Some velocity and [pressure](#) boundary condition combinations are not allowed in CONVERGE. If the boundary conditions you specify are not allowed, CONVERGE will not run.

Set up a velocity boundary condition via [*boundary.in*](#) > *boundary_conditions* > *boundary* > *velocity* > *type* and [*boundary.in*](#) > *boundary_conditions* > *boundary* > *velocity* > *value*. For a [temporally](#) and/or [spatially](#) varying velocity boundary condition (not available for all conditions), specify a file name for *value* and include that file in your case setup.

Dirichlet

You can specify a Dirichlet velocity boundary condition for an INFLOW or OUTFLOW boundary. You can set both the INFLOW and OUTFLOW velocity boundary conditions to Dirichlet only when the flow is supersonic.

Mass Flow

The mass flow velocity boundary condition is a special case of the Dirichlet velocity boundary condition. For a mass flow boundary condition, the velocity at the boundary is calculated from

$$u_i = \frac{\text{mass_flow}}{\rho_{\text{ave}} A} n_i, \quad (9.3)$$

where *mass_flow* is the mass flow into or out of the boundary, ρ_{ave} is the average density at the boundary, *A* is the total area of the INFLOW or OUTFLOW boundary, and n_i is the velocity normal. By default, CONVERGE uses the outward pointing surface normal as the velocity normal. For an INFLOW boundary, a positive *mass_flow* indicates flow into the domain and a negative *mass_flow* indicates flow out of the domain. For an OUTFLOW boundary, a positive *mass_flow* indicates flow out of the domain and a negative *mass_flow* indicates flow into the domain. At an OUTFLOW boundary, CONVERGE uniformly scales the velocities interior to the domain to achieve a fully developed velocity profile.

For a steady-state simulation, CONVERGE updates the OUTFLOW pressure via a two-step process, first calculating the pressure differential via the Bernoulli equation

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Velocity Boundary Conditions

$$\Delta p = \left(A V_{\text{calc}} - A V_{\text{spec}} \right) \frac{\rho_{\text{avg}}}{2}, \quad (9.4)$$

where $A V_{\text{calc}}$ is the calculated average velocity and $A V_{\text{spec}}$ is the target average velocity (calculated via the specified mass flow rate).

CONVERGE then updates the OUTFLOW pressure as

$$p_{\text{new}} = p_{\text{old}} + \alpha \Delta p, \quad (9.5)$$

where α is an under-relaxation factor.

Note that if reverse flow occurs through an OUTFLOW boundary with a mass flow boundary condition, CONVERGE sets the reverse flow value to zero (*i.e.*, no backflow can occur).

For an INFLOW boundary with the mass flow velocity condition, you may optionally specify a velocity direction vector. CONVERGE will first calculate the velocity normal to the boundary necessary to satisfy the specified mass flow rate. It will then calculate the velocity tangential to the boundary necessary to satisfy the velocity direction you have specified. You do not need to normalize this direction vector. By default, CONVERGE accepts the velocity direction vector in Cartesian coordinates. To specify a velocity vector for an INFLOW boundary with a mass flow velocity condition in a 3D simulation, provide the x, y, and z components of the vector after the mass inflow rate. In a 2D simulation, provide the x and y components, followed by a zero. You can instead supply the velocity direction vector in cylindrical coordinates. Refer to the [boundary.in](#) section of Chapter 25 for more details.

The velocity direction vector feature is not available for an OUTFLOW boundary with the mass flow velocity condition.

Average Velocity

For the average velocity boundary condition, CONVERGE assumes the average velocity is a scalar normal to the INFLOW or OUTFLOW boundary. The average velocity follows the same sign convention as mass flow (described above). CONVERGE scales the velocities at the cell centers adjacent to the boundary to produce a fully developed profile so that the average velocity at the boundary is equal to the specified average velocity.

Note that if reverse flow occurs through an OUTFLOW with an average velocity boundary condition, CONVERGE sets the reverse flow value to zero (*i.e.*, no backflow can occur).

Neumann

You can specify a zero-gradient Neumann velocity boundary condition for an INFLOW or OUTFLOW boundary. If you set a Neumann velocity boundary condition and a Dirichlet

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Velocity Boundary Conditions

pressure boundary condition, CONVERGE will adjust the magnitude and direction of the velocity based on the pressure gradient in the vicinity.

Normal Neumann

The Normal Neumann velocity boundary condition constrains the direction of the velocity to the normal of the INFLOW boundary. This boundary condition is allowed only for INFLOW boundaries set up with a Dirichlet pressure boundary condition. The Normal Neumann boundary condition is not available for OUTFLOW boundaries.

Pump

For this boundary condition, which is available only for INFLOW boundaries, you must include an input file (*e.g.*, [pump massflow.in](#)) that specifies pressure (in Pa) and mass flow (in kg/s) data.

Fluctuating INFLOW Boundary Option

The fluctuating INFLOW boundary option generates turbulent fluctuations that are superimposed on the INFLOW velocity profile. You can generate these fluctuations using the [digital filter method](#) or the [Fourier method](#). We recommend using the Fourier method because it is less expensive and it scales more favorably.

The digital filter method employs the two-point correlation function described by [Klein et al. \(2003\)](#). Starting from a white noise signal r_m such that $\overline{r_m} = 0$ and $\overline{r_m r_m} = 1$, a filtered signal is given by

$$u_m = \sum_{n=-N}^N b_n r_{m+n}, \quad (9.6)$$

where b_n are the one-dimensional filter coefficients and N is connected to the support of the filter. The autocorrelation function for white noise is $\overline{r_m r_n} = 0$ for $m \neq n$, so

$$\frac{\overline{u_m u_{m+k}}}{\overline{u_m u_m}} = \frac{\sum_{j=-N+k}^N b_j b_{j-k}}{\sum_{j=-N}^N b_j^2}, \quad (9.7)$$

gives the autocorrelation function for the filtered signal in terms of the one-dimensional filter coefficients. Three-dimensional filter coefficients b_{ijk} are obtained by convoluting three one-dimensional filters such that $b_{ijk} = b_i \bullet b_j \bullet b_k$.

For a turbulent signal,

$$\lim_{L \rightarrow \infty} R_{uu}(L) = 0, \quad (9.8)$$

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Velocity Boundary Conditions

where R_{uu} is the autocorrelation function and L is the user-specified length scale. The autocorrelation function for late stage homogenous turbulence takes the form

$$R_{uu}(r, 0, 0) = \exp\left(-\frac{\pi r^2}{4L^2}\right), \quad (9.9)$$

where $L = L(t) = \sqrt{2\pi\nu(t-t_0)}$. For late stage homogeneous turbulent cases, assuming the velocity of one eddy is not correlated to the velocity of any other eddy, L approximately corresponds to the size of a turbulent eddy. Small values for L result in small, high-frequency eddies. Larger values for L lead to larger, lower-frequency eddies.

The Fourier method models the fluctuations as a sum of random Fourier modes ([Davidson and Billson, 2006](#)). The inflow velocity is written as

$$u_i = u_{i, \text{set}} + |u'| f_i', \quad (9.10)$$

where $u_{i, \text{set}}$ is the freestream velocity profile, $|u'|$ is a user-specified magnitude, and

$$f_i' = 2 \sum_{n=1}^N \hat{u}^n \cos(k_j^n x_j + \psi^n) \sigma_i^n \quad (9.11)$$

is a sum over N Fourier modes, where k_j^n is the wave number of the n -th mode in direction j , ψ^n is the phase, and σ_i^n is the direction. The wave number, phase, and direction are generated randomly, while the amplitude \hat{u}^n is derived from the modeled turbulent spectrum.

To activate the fluctuating INFLOW boundary option, include the `boundary.in > boundary_conditions > boundary > fluctuation` settings block. In this block you will specify which method to use, along with a fluctuation intensity, a fluctuation length scale (in meters), and a fluctuation direction.

The fluctuation intensity determines the magnitude of the fluctuations as a fraction of the freestream velocity. This is a dimensionless parameter and can have any positive value. Physically realistic values range from 0.02 to 0.1. However, if you specify a [spatially varying](#) fluctuation intensity, you must provide a file with resolved turbulent kinetic energy (tke) values in m^2/s^2 , rather than dimensionless intensity values. CONVERGE uses the resolved tke values to determine the spatial profile of velocity fluctuations.

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Velocity Boundary Conditions

For the digital filter method, the fluctuation length scale corresponds to the length scale L from the autocorrelation function. For the Fourier method, the fluctuation length scale is the shortest allowed wavelength for the random Fourier modes.

For both methods, you can choose to apply fluctuations in all directions or only in the direction normal to the INFLOW boundary.

Note that both methods require the use of a random number generator to generate fluctuations. CONVERGE uses a pseudo-random number generator that is initialized from a user-specified seed value ([inputs.in](#) > *simulation_control* > *random_seed*). If you run multiple simulations on the same computer hardware with the same case setup and the same value of *random_seed*, CONVERGE will generate identical fluctuations for those simulations. If you want to apply different fluctuations to each simulation, you must use different values of *random_seed* for each simulation.

Pressure Boundary Conditions

[Dirichlet](#) and [Neumann](#) boundary conditions are available to solve the pressure equation at INFLOW and OUTFLOW boundaries. An additional boundary condition—[transonic](#)—is available for OUTFLOW boundaries.

Set up a pressure boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *pressure* > *type* and [boundary.in](#) > *boundary_conditions* > *boundary* > *pressure* > *value*. For a [temporally](#) and/or [spatially](#) varying pressure boundary condition (not available for all conditions), specify a file name for *value*.

Dirichlet

You can specify a Dirichlet pressure boundary condition for an INFLOW or OUTFLOW boundary. To specify a static pressure, set [boundary.in](#) > *boundary_conditions* > *boundary* > *pressure* > *type* = *DIRICHLET* and specify a static pressure for [boundary.in](#) > *boundary_conditions* > *boundary* > *pressure* > *value*. To specify a total (stagnation) pressure, set *pressure* > *type* = *DIRICHLET_TOTAL* and specify a total pressure for *pressure* > *value*.

If you specify a total pressure, CONVERGE calculates the corresponding static pressure to set as the boundary condition. For incompressible flows, the static pressure is given by

$$P_{\text{static}} = P_{\text{total}} - \frac{\rho u_i^2}{2} - \rho gh, \quad (9.12)$$

where g is the gravity optionally specified in [inputs.in](#) > *property_control* and h is the height above the gravity reference coordinate. For compressible flows, the relationship between the total and static pressures is

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Pressure Boundary Conditions

$$P_{\text{static}} = P_{\text{total}} \left(1.0 + \frac{\gamma - 1}{2} \frac{u_i^2}{\gamma RT} \right)^{-\frac{\gamma}{\gamma - 1}} - \rho gh, \quad (9.13)$$

where γ is the ratio of specific heats and R is the ideal gas constant.

For subsonic INFLOW boundaries with the Dirichlet pressure boundary condition, we recommend specifying total pressure and setting a [Neumann velocity boundary condition](#). For supersonic INFLOW boundaries, we recommend specifying a static Dirichlet pressure boundary condition and setting a [Dirichlet velocity boundary condition](#).

For subsonic OUTFLOW boundaries with the Dirichlet pressure boundary condition, transient pressure waves will reflect back into the domain if there is a pressure mismatch between the cell and the boundary. Depending on your simulation, these may or may not represent real flow physics. If your OUTFLOW boundary represents a downstream plenum (e.g., an exhaust manifold), reflected pressure waves are physically appropriate, although they will be phase-shifted depending on the location of the boundary. If your OUTFLOW boundary represents free space (e.g., downwind of an airfoil), reflected pressure waves may not represent the appropriate flow physics. CONVERGE offers two methods to dissipate these reflected pressure waves.

For OUTFLOW boundaries with the Dirichlet pressure boundary condition, you can optionally specify the weighting factor *presdist*, which must be a positive real value. Specifying *presdist* dampens reflecting pressure waves in the domain and can improve convergence rates for steady-state simulations. If you provide *presdist*, CONVERGE interpolates out of the domain to obtain the static pressure at the OUTFLOW boundary as

$$P_{\text{static}} = \frac{P_{\text{bound}} + \text{presdist} \cdot P_{\text{fluid}}}{1 + \text{presdist}}, \quad (9.14)$$

where P_{bound} is the Dirichlet boundary condition specified in [boundary.in](#) and P_{fluid} is the boundary cell pressure. This interpolation is normal to the outflow boundary. Numerically, specifying *presdist* is equivalent to setting a hybrid Dirichlet/Neumann boundary condition at the OUTFLOW boundary. Specify this quantity via [boundary.in > boundary_conditions > boundary > pressure > presdist](#). Grid embedding, grid scaling, and AMR have no effect on the behavior of *presdist*.

We recommend specifying *presdist* only if your solution is being polluted by unrealistic reflected pressure waves. We do not recommend it for general use. Contact the Applications team for additional guidance on this parameter.

Alternately, you can specify a sponge layer for Dirichlet pressure boundary conditions at OUTFLOW boundaries. The sponge layer, which is based on the formulation of [Bodony \(2006\)](#), is a zone adjacent to an OUTFLOW boundary with a length that you specify. Within

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Pressure Boundary Conditions

this zone, CONVERGE attempts to detect and damp pressure waves. To activate the sponge layer, include the [*boundary.in*](#) > *boundary_conditions* > *boundary* > *sponge* settings block. In this block you must specify the center of the sponge layer, the direction of growth of the sponge layer, and the length of the sponge layer. CONVERGE creates a box-shaped sponge layer from the center point in the direction you specified with thickness equal to the distance value.

We recommend using a sponge layer only if your solution is being polluted by unrealistic reflected pressure waves. We do not recommend it for general use. Contact the Convergent Science Applications team for further guidance on this feature.

Neumann

You can specify a zero-gradient Neumann pressure boundary condition for an INFLOW or OUTFLOW boundary. For subsonic INFLOW boundaries with the Neumann pressure boundary condition, we recommend a [Dirichlet velocity boundary condition](#).

To set up a non-zero-gradient Neumann pressure boundary condition, enter the keyword *Neumann* followed by the value of the pressure gradient. This option is available for INFLOW but not OUTFLOW boundaries. By convention, a positive value of the pressure gradient corresponds to pressure decreasing from the boundary into the domain.

Transonic

The transonic pressure boundary condition is suitable for cases in which the flow accelerates from subsonic to supersonic at the outlet. The transonic pressure boundary condition is a weighted average of the Dirichlet and Neumann pressure boundary conditions. You can specify the minimum and maximum Mach numbers (M_{min} and M_{max} in Equation 9.15 below) between which the flow is considered transonic. Equation 9.15 describes the formulation for pressure in each regime:

$$p_{bound} = \begin{cases} p_{static} & M < M_{min} \\ \left(\frac{M_{max} - M}{M_{max}} \right) p_{static} + \left(\frac{M}{M_{max}} \right) p_{node} & M_{min} \leq M \leq M_{max} \\ p_{node} & M > M_{max} \end{cases}, \quad (9.15)$$

where p_{static} is the calculated Dirichlet pressure and p_{node} is the static pressure at the cell center of a cell adjacent to the OUTFLOW boundary. CONVERGE calculates p_{static} as

$$p_{static} = \left(\frac{1}{1+x} \right) p_{specified} + \left(\frac{x}{1+x} \right) p_{node}, \quad (9.16)$$

where $p_{specified}$ is the static pressure that you enter and x is the distance to the ambient pressure.

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Pressure Boundary Conditions

Typically you specify minimum and maximum Mach numbers when using a transonic pressure boundary condition. If you do not specify minimum and maximum Mach numbers ([boundary.in](#) > `boundary_conditions` > `boundary` > `pressure` > `min_mach` and `max_mach`), CONVERGE will use 0.9 and 1.1, respectively.

Temperature Boundary Conditions

[Dirichlet](#) and [Neumann](#) temperature boundary conditions are available to solve the energy equation at INFLOW and OUTFLOW boundaries. We recommend Dirichlet for INFLOW boundaries and Neumann for OUTFLOW boundaries.

For OUTFLOW boundaries, CONVERGE Studio allows only a Neumann temperature boundary condition. CONVERGE, however, will run with a Dirichlet temperature boundary condition.

For OUTFLOW boundaries, you must also specify a [backflow temperature boundary condition](#).

Set up a temperature boundary condition via [boundary.in](#) > `boundary_conditions` > `boundary` > `temperature` > `type` and [boundary.in](#) > `boundary_conditions` > `boundary` > `temperature` > `value`. For a [temporally](#) and/or [spatially](#) varying temperature boundary condition (not available for all conditions), specify a file name for `value`.

Dirichlet

You can specify a Dirichlet temperature boundary condition.

If you specify a total pressure for the [pressure boundary condition](#), CONVERGE will assume the specified temperature is the total temperature and calculate the static temperature as

$$T_{\text{static}} = T_{\text{total}} \left(1.0 + \frac{\gamma - 1}{2} \frac{u_i^2}{\gamma RT} \right)^{-1}, \quad (9.17)$$

where γ is the ratio of specific heats and R is the ideal gas constant.

Neumann

You can specify a zero-gradient Neumann temperature boundary condition.

Species Boundary Conditions

[Dirichlet](#) and [Neumann](#) boundary conditions are available to solve the species equation at INFLOW and OUTFLOW boundaries. We recommend Dirichlet for INFLOW boundaries and Neumann for OUTFLOW boundaries.

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Species Boundary Conditions

For OUTFLOW boundaries, CONVERGE Studio allows only a Neumann species boundary condition. CONVERGE, however, will run with a Dirichlet species boundary condition.

For OUTFLOW boundaries, you must also specify a [backflow species boundary condition](#).

Set up a species boundary condition via `boundary.in > boundary_conditions > boundary > species > type` and `boundary.in > boundary_conditions > boundary > species > value`. For a [temporally](#) and/or [spatially](#) varying species boundary condition (not available for all conditions), specify a file name for `value`.

Dirichlet

You can specify a Dirichlet species boundary condition. The species mass fractions that you specify should add up to 1.0. Note that CONVERGE will not normalize the values at runtime if the sum is not 1.0.

Neumann

You can specify a zero-gradient Neumann species boundary condition.

Passive Boundary Conditions

[Dirichlet](#) and [Neumann](#) boundary conditions are available to solve the passive equation at INFLOW and OUTFLOW boundaries. We recommend Dirichlet for INFLOW boundaries and Neumann for OUTFLOW boundaries.

For OUTFLOW boundaries, CONVERGE Studio allows only a Neumann passive boundary condition. CONVERGE, however, will run with a Dirichlet passive boundary condition.

For OUTFLOW boundaries, you must also specify a [backflow passive boundary condition](#).

Set up a passive boundary condition via `boundary.in > boundary_conditions > boundary > passive > type` and `boundary.in > boundary_conditions > boundary > passive > value`. For a [temporally](#) and/or [spatially](#) varying passive boundary condition (not available for all conditions), specify a file name for `value`.

Dirichlet

You can specify a Dirichlet species boundary condition.

Neumann

You can specify a zero-gradient Neumann passive boundary condition.

Turbulent Kinetic Energy Boundary Conditions

You must specify a boundary condition for the turbulent kinetic energy (tke) equation when your simulation includes certain [turbulence models](#).

Dirichlet, Neumann, and intensity boundary conditions are available to solve the tke equation at INFLOW and OUTFLOW boundaries.

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Turbulent Kinetic Energy Boundary Conditions

For OUTFLOW boundaries, you must also specify a [backflow turbulent kinetic energy boundary condition](#).

If your simulation includes a [RANS k- \$\omega\$ turbulence model](#), you can approximate the far-field tke at INFLOW or OUTFLOW boundaries as

$$\frac{10^{-5}U_{\infty}^2}{Re_L} < k_{far\ field} < \frac{0.1U_{\infty}^2}{Re_L} . \quad (9.18)$$

Set up a tke boundary condition via `boundary.in > boundary_conditions > boundary > turbulence > tke > type` and `boundary.in > boundary_conditions > boundary > turbulence > tke > value`. For a [temporally](#) and/or [spatially](#) varying tke boundary condition (not available for all conditions), specify a file name for `value`.

Dirichlet

You can specify a Dirichlet tke boundary condition for an INFLOW or OUTFLOW boundary.

Neumann

You can specify a zero-gradient Neumann tke boundary condition for an INFLOW or OUTFLOW boundary.

Intensity

The intensity tke boundary condition is a special case of the Dirichlet tke boundary condition. For the intensity tke boundary condition, the boundary turbulent kinetic energy, k , is given by

$$k = \frac{3}{2} u_i^2 I^2 , \quad (9.19)$$

where I is the turbulence intensity that you specify in `boundary_conditions > boundary > turbulence > tke > value`. This value is typically between 0.01 and 0.1.

Turbulent Dissipation Boundary Conditions

You must specify a boundary condition for the turbulent dissipation rate (eps) equation when your simulation includes certain [turbulence models](#).

Dirichlet, Neumann, and length scale boundary conditions are available to solve the eps equation at INFLOW and OUTFLOW boundaries. We recommend Neumann for OUTFLOW boundaries.

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Turbulent Dissipation Boundary Conditions

For OUTFLOW boundaries, CONVERGE Studio allows only a Neumann eps boundary condition. CONVERGE, however, will run with a Dirichlet eps boundary condition.

For OUTFLOW boundaries, you must also specify a [backflow turbulent dissipation boundary condition](#).

Set up an eps boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *turbulence* > *eps* > *type* and [boundary.in](#) > *boundary_conditions* > *boundary* > *turbulence* > *eps* > *value*. For a [temporally](#) and/or [spatially](#) varying eps boundary condition (not available for all conditions), specify a file name for *value*.

Dirichlet

You can specify a Dirichlet eps boundary condition for an INFLOW or OUTFLOW boundary.

Neumann

You can specify a zero-gradient Neumann eps boundary condition for an INFLOW or OUTFLOW boundary.

Length Scale

The length scale eps boundary condition is a special case of the Dirichlet eps boundary condition. For the length scale boundary condition, the turbulent dissipation at the boundary is given by

$$\varepsilon = \frac{c_\mu^{3/4} k^{3/2}}{l_e}, \quad (9.20)$$

where c_μ is a model constant, k is the turbulent kinetic energy, and l_e is the length scale (in meters) that you specify via *boundary_conditions* > *boundary* > *turbulence* > *eps* > *value*. You can estimate the length scale from a physical dimension in the domain (e.g., you can set the length scale to a fraction of the intake port diameter).

Specific Dissipation Rate Boundary Conditions

You must specify a boundary condition for the specific dissipation rate (ω) equation when your simulation includes certain [turbulence models](#).

Dirichlet, Neumann, and length scale boundary conditions are available to solve the ω equation at INFLOW and OUTFLOW boundaries.

For OUTFLOW boundaries, you must also specify a [backflow specific dissipation rate boundary condition](#).

If your simulation includes a [RANS k- \$\omega\$ turbulence model](#), you can approximate the far-field ω at INFLOW or OUTFLOW boundaries as

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Specific Dissipation Rate Boundary Conditions

$$\frac{U_\infty}{L} < \omega_{far_field} < 10 \frac{U_\infty}{L}, \quad (9.21)$$

where L is the approximate length of the domain.

Set up an omega boundary condition via [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [turbulence](#) > [omega](#) > [type](#) and [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [turbulence](#) > [omega](#) > [value](#). For a [temporally](#) and/or [spatially](#) varying omega boundary condition (not available for all conditions), specify a file name for [value](#).

Dirichlet

You can specify a Dirichlet omega boundary condition for an INFLOW or OUTFLOW boundary.

Neumann

You can specify a zero-gradient Neumann omega boundary condition for an INFLOW or OUTFLOW boundary.

Length Scale

The length scale omega boundary condition is a special case of the Dirichlet omega boundary condition. For the length scale boundary condition, the specific dissipation at the boundary is given by

$$\omega = \frac{c_\mu^{-1/4} k^{1/2}}{le}, \quad (9.22)$$

where ω is the specific dissipation, c_μ is a model constant, k is the turbulent kinetic energy, and le is the length scale (in *meters*) that you specify via [boundary_conditions](#) > [boundary](#) > [turbulence](#) > [omega](#) > [value](#). You can estimate the length scale from a physical dimension in the domain (e.g., you can set the length scale to a fraction of the intake port diameter).

OUTFLOW Backflow Boundary Conditions

For an OUTFLOW boundary, you must define backflow boundary conditions, which will apply if flow comes into the OUTFLOW boundary. You can define backflow boundary conditions for the [temperature](#), [species](#), [passive](#), [turbulent kinetic energy \(tke\)](#), [turbulent dissipation \(eps\)](#), and [specific dissipation rate \(omega\)](#) equations. Specifying realistic backflow values helps to ensure convergence.

When reverse flow occurs at an OUTFLOW boundary, CONVERGE does not immediately initiate backflow. For example, suppose O₂ is flowing out of an OUTFLOW boundary for which N₂ is specified as the backflow species. If backflow occurs, CONVERGE does not have a way to ensure that the backflow species is all N₂ and thus backflow does not begin immediately. If reverse flow occurs for longer than $2.5 * \text{cell size} / \text{flow velocity}$, CONVERGE will begin flow of N₂ into the OUTFLOW boundary.

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW OUTFLOW Backflow Boundary Conditions

Backflow parameters are in the [*boundary.in*](#) > *boundary_conditions* > *boundary* > *backflow* settings block.

Only certain backflow boundary conditions are available for each parameter, as listed below in the following table.

Table 9.1: Backflow boundary conditions.

Parameter	Location in <i>boundary.in</i>	Boundary condition options
Temperature	<i>boundary_conditions</i> > <i>boundary</i> > <i>backflow</i> > <i>temperature</i>	Dirichlet (strongly recommended), Neumann
Species	<i>boundary_conditions</i> > <i>boundary</i> > <i>backflow</i> > <i>species</i>	Dirichlet (strongly recommended), Neumann
Passive	<i>boundary_conditions</i> > <i>boundary</i> > <i>backflow</i> > <i>passive</i>	Dirichlet (strongly recommended), Neumann
Turbulent kinetic energy	<i>boundary_conditions</i> > <i>boundary</i> > <i>backflow</i> > <i>turbulence</i> > <i>tke</i>	Dirichlet, intensity
Turbulent dissipation	<i>boundary_conditions</i> > <i>boundary</i> > <i>backflow</i> > <i>turbulence</i> > <i>eps</i>	Dirichlet, length scale
Specific dissipation rate	<i>boundary_conditions</i> > <i>boundary</i> > <i>backflow</i> > <i>turbulence</i> > <i>omega</i>	Dirichlet, Neumann, length scale

Navier-Stokes Characteristic Boundary Conditions

CONVERGE features non-reflecting subsonic [INFLOW](#) and subsonic [OUTFLOW](#) boundaries derived from an eigenanalysis of the Euler equations. These boundary conditions are designed to reduce the reflection of acoustic (*i.e.*, pressure) disturbances from INFLOW and OUTFLOW boundaries, which are important in some applications. The Navier-Stokes characteristic boundary condition (NSCBC) formulation in CONVERGE is based on the works of [Thompson \(1987\)](#), [Thompson \(1990\)](#), and [Poinsot and Lele \(1992\)](#), invoking a local associated one-dimensional inviscid (LODI) relation to neglect viscous and transverse terms. These boundary conditions consider multi-component and real gas effects, following the approach of [Okong'o and Bellan \(2002\)](#).

Consider discretizing the Euler equations in an analogous fashion to that presented in [Chapter 4 - Numerics](#). The finite volume formulation of the Euler equations in 1D can be written as

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Navier-Stokes Characteristic Boundary Conditions

$$\frac{\partial U}{\partial t} + A \frac{\partial U}{\partial x} = 0 \text{ and } A = \frac{\partial F}{\partial U}, \quad (9.23)$$

where U is the vector of conserved variables and F is the flux vector across a face. We perform an eigenanalysis and diagonalize A as

$$A = S^{-1} \Lambda S$$

and

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}, \quad (9.24)$$

where the columns of S^{-1} are the eigenvectors of A . It can be shown that in the 1D case, the eigenvalues of the system are

$$\begin{aligned} \lambda_1 &= u - c \\ \lambda_2 &= u \\ \lambda_3 &= u + c, \end{aligned} \quad (9.25)$$

where c is the speed of sound. For a subsonic INFLOW boundary in the positive x direction, λ_1 represents a pressure wave attempting to exit the domain. Similarly, for a subsonic OUTFLOW boundary in the positive x direction, λ_1 represents a pressure wave attempting to enter from outside the domain of the simulation.

[MacCormick \(2014\)](#) presents the derivation of the general three dimensional analog, with three eigenvalue matrices of similar form. Taking the x coordinate as being perpendicular to the boundary, the diagonal eigenvalue matrix takes the values

$$\begin{aligned} \lambda_1 &= u - c \\ \lambda_2 &= u \\ \lambda_3 &= u \\ \lambda_4 &= u \\ \lambda_5 &= u + c, \end{aligned} \quad (9.26)$$

with corresponding eigenvector matrices.

CONVERGE offers two implementations of the NSCBC scheme: the [Poinsot-Lele method](#) and the [correction-based method](#).

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Navier-Stokes Characteristic Boundary Conditions

Poinsot-Lele Method

The NSCBC formulation of [Poinsot and Lele \(1992\)](#) solves the following set of equations on boundaries

$$\begin{aligned}\frac{\partial \rho}{\partial t} + d_1 + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} &= 0 \\ \frac{\partial \rho u}{\partial t} + u d_1 + \rho d_3 \frac{\partial \rho u v}{\partial y} + \frac{\partial \rho u w}{\partial z} &= 0 \\ \frac{\partial \rho v}{\partial t} + v d_1 + \rho d_4 \frac{\partial \rho v v}{\partial y} + \frac{\partial \rho v w}{\partial z} + \frac{\partial p}{\partial y} &= 0 \\ \frac{\partial \rho w}{\partial t} + w d_1 + \rho d_5 \frac{\partial \rho v w}{\partial y} + \frac{\partial \rho w w}{\partial z} + \frac{\partial p}{\partial z} &= 0 \\ \frac{\partial \rho E}{\partial t} + \left(E + \frac{p}{\rho} \right) d_1 + C_p T d_2 + \rho (u d_3 + v d_4 + w d_5) &= 0,\end{aligned}\tag{9.27}$$

where

$$\begin{aligned}d_1 &= \frac{1}{c^2} \left(L_2 + \frac{1}{2} [L_5 + L_1] \right) \\ d_2 &= -\frac{L_2}{c^2} \\ d_3 &= \frac{1}{2\rho c} [L_5 - L_1] \\ d_4 &= L_3 \\ d_5 &= L_4\end{aligned}\tag{9.28}$$

and

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Navier-Stokes Characteristic Boundary Conditions

$$\begin{aligned} L_1 &= \lambda_1 \left(\frac{\partial p}{\partial x} - \rho c \frac{\partial u}{\partial x} \right) \\ L_2 &= \lambda_2 \left(c^2 \frac{\partial \rho}{\partial x} - \frac{\partial p}{\partial x} \right) \\ L_3 &= \lambda_3 \frac{\partial v}{\partial x} \\ L_4 &= \lambda_4 \frac{\partial w}{\partial x} \\ L_5 &= \lambda_5 \left(\frac{\partial p}{\partial x} + \rho c \frac{\partial u}{\partial x} \right). \end{aligned} \tag{9.29}$$

The values L_1 through L_5 are the amplitudes of the propagating waves associated with the corresponding eigenvalues. To eliminate reflections at a subsonic INFLOW or OUTFLOW, setting $L_1 = 0$ would eliminate the spurious reflected pressure waves. However, for a general multidimensional Navier-Stokes flow, there is no general method to enforce this.

Correction-Based Method

The correction-based NSCBC method uses the LODI formulation of propagating waves to calculate corrections to boundary values of density, pressure, and velocity within each PISO iteration. The boundary values for energy are then reconstructed from these corrected variables. The correction-based method is available only when [`solver.in`](#) > `NS_solver_scheme = PISO`.

Defining the vector $U = (\rho, p, u, v, w)$, we can write the time evolution of U in terms of the residual R ,

$$\frac{\partial U}{\partial t} = -R. \tag{9.30}$$

The predicted residual at iteration n is given by

$$R^P = -\frac{U^{n+1,P} - U^n}{dt}, \tag{9.31}$$

where $U^{n+1,P}$ denotes the predicted (uncorrected) solution at the end of the PISO iteration. The correction-based method seeks to remove the incoming waves contained in this solution and replace them with the "corrected" incoming waves corresponding to the desired boundary condition. The residual is decomposed as $R = ML$, where L is the vector of wave amplitudes defined in Equation 9.29 and the matrix M is defined by the system of equations ([Poinsot and Lele, 1992](#))

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Navier-Stokes Characteristic Boundary Conditions

$$\begin{aligned}\frac{\partial \rho}{\partial t} + \frac{1}{c^2} \left[L_2 + \frac{1}{2} (L_5 + L_1) \right] &= 0 \\ \frac{\partial p}{\partial t} + \frac{1}{2} (L_5 + L_1) &= 0 \\ \frac{\partial u}{\partial t} + \frac{1}{2\rho c} (L_5 - L_1) &= 0 \\ \frac{\partial v}{\partial t} + L_3 &= 0 \\ \frac{\partial w}{\partial t} + L_4 &= 0.\end{aligned}\tag{9.32}$$

CONVERGE calculates the corrected solution at the boundary according to

$$U^{n+1,C} = U^n - dt \left(R^P - R_{BC}^{in,P} + R_{BC}^{in,C} \right),\tag{9.33}$$

where

$$R_{BC}^{in,P} = M L^{in}\tag{9.34}$$

and

$$R_{BC}^{in,C} = M L^{in,C}.\tag{9.35}$$

At an INFLOW boundary, only L_1 corresponds to an outgoing wave, so $L^{in} = (0, L_2, L_3, L_4, L_5)$ and $L^{in,C} = (0, L_2^C, L_3^C, L_4^C, L_5^C)$. At an OUTFLOW boundary, L_1 is the only incoming wave, so $L^{in} = (L_1, 0, 0, 0, 0)$ and $L^{in,C} = (L_1^C, 0, 0, 0, 0)$. CONVERGE calculates the L_i^C values from the boundary conditions specified in [boundary.in](#), as detailed below.

Subsonic INFLOW

To set up a subsonic INFLOW boundary, specify velocity as mass flow or Dirichlet and pressure as Neumann. Set [boundary.in](#) > *boundary* > *nscbc* > *method* = *POINSOT_LELE* or *CORRECTION_BASED*. If *method* = *POINSOT_LELE*, temperature can be either Dirichlet or Neumann. If *method* = *CORRECTION_BASED*, temperature must be Dirichlet.

For the Poinsot-Lele method, density is solved through the continuity equation, and, from boundary temperature and density, pressure is solved through the equation of state. CONVERGE sets

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Navier-Stokes Characteristic Boundary Conditions

$$\begin{aligned} L_1 &= \lambda_1 \left(\frac{\partial p}{\partial x} - \rho c \frac{\partial u}{\partial x} \right) \\ L_2 &= L_1 (\gamma - 1) \\ L_3 &= 0 \\ L_4 &= 0 \\ L_5 &= L_1. \end{aligned} \tag{9.29}$$

For the correction-based method, CONVERGE calculates the corrected wave amplitudes L_i^c from

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\frac{1}{2\rho c} L_5^c = -K(u - u_\infty) \\ \frac{\partial v}{\partial t} &= -L_3^c = -K(v - v_\infty) \\ \frac{\partial w}{\partial t} &= -L_4^c = -K(w - w_\infty) \\ \frac{\partial T}{\partial t} &= -\frac{T}{\rho c^2} L_2^c = -K(T - T_\infty), \end{aligned} \tag{9.36}$$

where the far-field velocity ($u_\infty, v_\infty, w_\infty$) and far-field temperature T_∞ correspond to the Dirichlet boundary conditions in [boundary.in](#). These expressions assume that the inlet velocity and temperature relax to the far-field values with a relaxation constant K , given by

$$K = \sigma (1 - M^2) \frac{c}{L}, \tag{9.37}$$

where M is the maximum Mach number in the domain. Specify the tuning parameter σ and the length scale L in [boundary.in](#) > boundary > nscbc > sigma and [boundary.in](#) > boundary > nscbc > lengthscale, respectively. When *lengthscale* = AUTO, CONVERGE calculates L as the maximum extent of the domain in the direction normal to the boundary.

Subsonic OUTFLOW

To set up a subsonic OUTFLOW, specify pressure as Dirichlet and set [boundary.in](#) > boundary > nscbc > method = POINSOT_LELE or CORRECTION_BASED.

For the Poinsot-Lele method, CONVERGE calculates L_1 in Equations 9.27-9.29 according to

$$L_1 = K(p - p_\infty), \tag{9.38}$$

Chapter 9: Boundaries and Boundary Conditions

INFLOW and OUTFLOW Navier-Stokes Characteristic Boundary Conditions

where K is given by Equation 9.37 and p_∞ is the far-field (Dirichlet) pressure.

For the correction-based method, CONVERGE calculates L_1^C in Equation 9.35 from

$$\frac{\partial p}{\partial t} = -\frac{1}{2} L_1^C = -K(p - p_\infty). \quad (9.39)$$

For both methods, specify the tuning parameter σ and the length scale L in [boundary.in > boundary > nsabc > sigma](#) and [boundary.in > boundary > nsabc > lengthscale](#), respectively.

9.2 WALL

This section describes boundary conditions for WALL boundaries. You can specify boundary conditions for [velocity](#), [pressure](#), [temperature](#), [species](#), [passive](#), [turbulent kinetic energy](#), [turbulent dissipation](#), and [specific dissipation rate](#).

You can configure a [sliding](#) WALL to apply non-zero velocity boundary conditions to the fluid at WALL boundaries.

The surface motion of a WALL boundary can be fixed (*i.e.*, that surface does not move), [moving](#), or moving with a [reloft](#) option.

Many boundary conditions can vary [temporally](#) and/or [spatially](#).

To set up a WALL boundary, set [boundary.in > boundary_conditions > boundary > type = WALL](#) and configure the subsequent parameters in the [boundary.in > boundary_conditions > boundary](#) settings block.

Sliding WALL Boundary Conditions

A sliding WALL boundary applies a non-zero velocity boundary condition to the fluid while keeping the surface geometry fixed (*i.e.*, the WALL surface triangles do not move during the simulation). The velocity at a sliding WALL boundary is parallel to the boundary. Sliding WALL boundaries may be useful to simulate rotating drums or conveyor belts.

You can specify a [rotating](#), [translating](#), or [tangential](#) sliding WALL boundary.

Rotating

A rotating sliding WALL boundary means that the surface triangles themselves will not move during the simulation, but the fluid adjacent to the boundary will be affected by a rotating velocity boundary condition. The right-hand rule is used to determine the direction of rotation about the axis.

To set up a rotating sliding WALL boundary, set [boundary.in > boundary_conditions > boundary > motion = ROTATING](#) and [boundary.in > boundary_conditions > boundary > geometry_motion = FIXED](#). Specify the rotation rate (either a constant value or a file name for

Chapter 9: Boundaries and Boundary Conditions

WALL Sliding WALL Boundary Conditions

a [temporally varying profile](#)) via `boundary.in > boundary_conditions > boundary > velocity > rotation_speed`.

Translating

A translating sliding WALL boundary means that the surface triangles themselves will not move during the simulation, but the fluid adjacent to the boundary will be affected by a translating velocity boundary condition. CONVERGE determines the direction of the axis via the right-hand rule and the direction of translation by computing the cross product of the boundary normal and the rotation axis.

To set up a translating sliding WALL boundary, set `boundary.in > boundary_conditions > boundary > motion = TRANSLATING` and `boundary.in > boundary_conditions > boundary > geometry_motion = FIXED`. Specify the velocity (either a constant value or a file name for a [temporally varying profile](#)) via `boundary.in > boundary_conditions > boundary > velocity > value`.

Tangential

A tangential sliding WALL boundary is a special case of a translating sliding WALL boundary. To set up a tangential sliding WALL boundary, set `boundary.in > boundary_conditions > boundary > motion = STATIONARY` and `boundary.in > boundary_conditions > boundary > geometry_motion = FIXED`. Set `boundary.in > boundary_conditions > boundary > velocity > type = Tangential`. Specify the axis of rotation and the velocity magnitude (in m/s) via `boundary.in > boundary_conditions > boundary > velocity > translate_axis` and `translate_speed`, respectively. For a tangential sliding WALL boundary, the velocity magnitude must be a constant value.

Moving WALL Boundaries

CONVERGE allows you to set up moving WALL boundaries. To set up a moving WALL boundary, set `boundary.in > boundary_conditions > boundary > geometry_motion = MOVING`.

Several types of WALL motion—[translating](#), [rotating](#), [simultaneous rotating and translating](#), [arbitrary](#), [fluid-structure interaction](#) (FSI), [Abaqus](#), [surface list](#), [user](#), and [linked](#)—are available. Some motion types (e.g., arbitrary or temporally varying rotating) require you to include motion files. We recommend creating these files in CONVERGE Studio.

Translating

To set up a translating WALL boundary, set `boundary.in > boundary_conditions > boundary > motion = TRANSLATING` and `boundary.in > boundary_conditions > boundary > geometry_motion = MOVING`. Specify the velocity (either a constant value or a file name for a [temporally varying profile](#)) via `boundary.in > boundary_conditions > boundary > velocity > value`.

For constant velocity translating motion, the [surface geometry file](#) will represent the wall locations at 0 seconds or crank angle degrees (CAD), depending on the value of `inputs.in > simulation control > crank_flag`. If the simulation does not start at 0, CONVERGE will adjust the translating boundaries as though they had moved from 0 seconds or CAD to the simulation start time that is specified in `inputs.in`. If the simulation start time is negative,

Chapter 9: Boundaries and Boundary Conditions

WALL Moving WALL Boundaries

CONVERGE will move the WALL backward, so that when the simulation time is 0 the boundary returns to the location specified in the surface file. Because the grid must encompass the entire geometry throughout the simulation and the grid size is based on the size of the geometry in the surface file, you must configure the surface file so that each boundary is at its maximum extent at 0 *seconds* or 0 CAD. Then you can start the simulation at a negative time.

For a translating WALL boundary with a temporally varying velocity, motion involves two steps. First, CONVERGE applies the velocity as the boundary condition when solving the flow in the cells on the moving boundary. Next CONVERGE changes the geometry to account for the WALL motion. Because CONVERGE determines the velocity by differentiating the distance, the velocity applied as a boundary in the first step and distance moved in the second step must be consistent to prevent distance errors from happening. To ensure this consistency, CONVERGE divides the distance the WALL has to move in the second step by the time-step size and applies the result as the velocity in the first step.

Note that CONVERGE does not allow surfaces to intersect and will terminate the internal grid generation if they do. You can verify that translating WALL boundaries do not cross other WALL boundaries with the *Translate* option in CONVERGE Studio.

There are several special cases of temporally varying translating WALL motion files, including [pistons](#) and [valves](#).

Rotating

To set up a rotating WALL boundary, set [boundary.in](#) > *boundary_conditions* > *boundary* > *motion* = ROTATING and [boundary.in](#) > *boundary_conditions* > *boundary* > *geometry_motion* = MOVING.

Specify the center of rotation and the vector around which the wall rotates via [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *origin* and [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *rotation_axis*, respectively. Specify the rate of rotation (either a constant value or a file name for a [temporally varying profile](#)) via [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *rotation_speed*. CONVERGE uses the right-hand rule to determine the direction of the rotation about the axis.

Rotating and Translating

The rotating and translating WALL motion type allows you to specify simultaneous rotating and translating motion.

To set up a rotating and translating WALL boundary, set [boundary.in](#) > *boundary_conditions* > *boundary* > *motion* = ROTATING_AND_TRANSLATING and [boundary.in](#) > *boundary_conditions* > *boundary* > *geometry_motion* = MOVING.

Specify the translational velocity via [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value*. Specify the center of rotation and the vector around which the wall rotates via [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *origin* and [boundary.in](#) >

Chapter 9: Boundaries and Boundary Conditions

WALL Moving WALL Boundaries

boundary_conditions > boundary > velocity > rotation_axis, respectively. Specify the rate of rotation via [boundary.in > boundary_conditions > boundary > velocity > rotation_speed](#). CONVERGE uses the right-hand rule to determine the direction of the rotation about the axis.

Arbitrary

The arbitrary WALL motion type allows you to specify the position and orientation of a moving WALL boundary at different times.

To set up a WALL boundary with arbitrary motion, set [boundary.in > boundary_conditions > boundary > motion = ARBITRARY](#) and [boundary.in > boundary_conditions > boundary > geometry_motion = MOVING](#). Specify a file name for the arbitrary motion profile via [boundary.in > boundary_conditions > boundary > velocity > value](#).

Dependent

The dependent WALL motion type allows you to specify that the motion of a moving WALL boundary depends on the motion of an adjacent boundary (*e.g.*, lateral piston motion). When the adjacent boundary moves, the shared vertices move with it and the triangles that make up the dependent boundary deform (stretch or compress). WALL boundaries with dependent motion must not contain internal triangles. An internal vertex (*i.e.*, one not shared with any other boundary) would remain stationary and could cause undesirable deformations to the geometry.

Figure 9.1 illustrates dependent WALL motion.

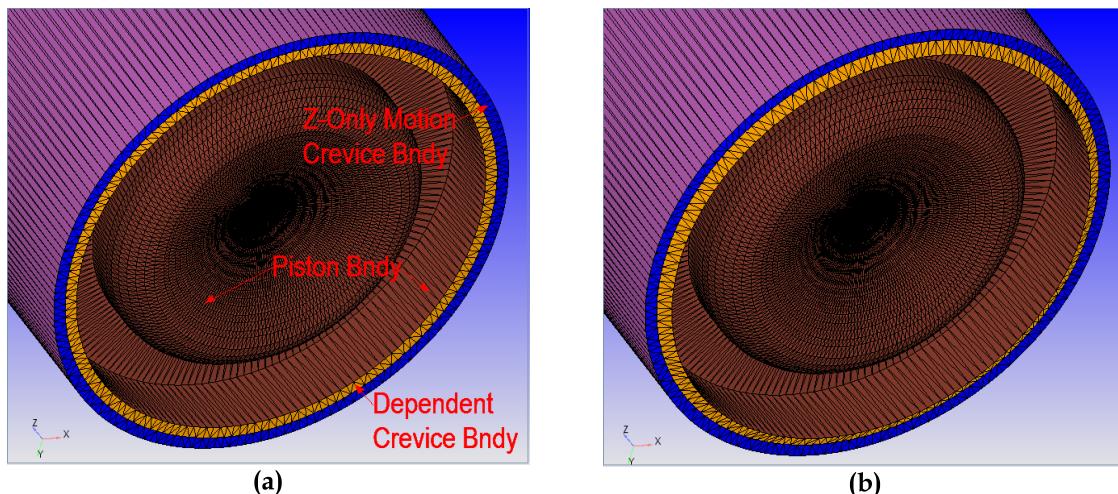


Figure 9.1: The underside of a piston (a) before and (b) after lateral movement. The orange boundary is a WALL boundary with dependent motion. The blue ring of triangles moves with the piston along the z axis but does not move laterally. When the piston moves, the inner vertices of the orange boundary move laterally with the piston, but the outer vertices remain stationary in the xy plane. The triangles stretch or compress to accommodate the lateral piston movement.

Chapter 9: Boundaries and Boundary Conditions

WALL Moving WALL Boundaries

To set up a WALL boundary with dependent motion, set [boundary.in](#) > `boundary_conditions > boundary > motion = DEPENDENT` and [boundary.in](#) > `boundary_conditions > boundary > geometry_motion = MOVING`. CONVERGE ignores any velocity boundary conditions specified in [boundary.in](#) for WALL boundaries with dependent motion.

Fluid-Structure Interaction

The fluid-structure interaction (FSI) WALL motion type allows you to specify that the motion of moving WALL is determined by [fluid-structure interactions](#). The WALL boundary must be part of an FSI object in [fsi.in](#).

To set up a WALL boundary with FSI-determined motion, set [boundary.in](#) > `boundary_conditions > boundary > motion = FSI` and [boundary.in](#) > `boundary_conditions > boundary > geometry_motion = MOVING`. CONVERGE ignores any velocity boundary conditions specified in [boundary.in](#) for WALL boundaries with FSI-determined motion.

Abaqus

Abaqus boundaries are defined as walls with a special motion type. To set up a WALL boundary with will be subject to Abaqus analysis, set [boundary.in](#) > `boundary_conditions > boundary > type = WALL`, [boundary.in](#) > `boundary_conditions > boundary > motion = ABAQUS`, and [boundary.in](#) > `boundary_conditions > boundary > geometry_motion = MOVING`.

Surface List

CONVERGE can calculate boundary motion from a list of surface geometry files that represent the surface geometry at different times. To use this option, set `inputs.in > surface_filename = surface_list.in` and include the [surface_list.in](#) file in your case setup. For boundaries whose motion is determined from the list of surface geometry files, set [boundary.in](#) > `boundary_conditions > boundary > motion = SURFACE_LIST` and [boundary.in](#) > `boundary_conditions > boundary > geometry_motion = MOVING`. For any boundaries for which [boundary.in](#) > `boundary_conditions > boundary > motion ≠ SURFACE_LIST`, CONVERGE will use the base surface geometry file specified in [surface_list.in](#) > `base_surface_file`.

User-Defined

The user-defined WALL motion type allows you to specify the motion of a moving WALL via a [user-defined function](#) (UDF).

To set up a WALL boundary with user-defined motion, set [boundary.in](#) > `boundary_conditions > boundary > motion = USER` and [boundary.in](#) > `boundary_conditions > boundary > geometry_motion = MOVING`. CONVERGE ignores any velocity boundary conditions specified in [boundary.in](#) for WALL boundaries with user-defined motion. Refer to the CONVERGE 3.1 UDF Manual for more information about motion UDFs.

Translating WALL: Piston

Pistons are an example of translating WALL boundaries. Figure 9.2 below shows the wrist pin offset sign convention for engine applications in CONVERGE. A non-zero wrist pin offset occurs when the axes of the piston and the crankshaft are different. When the wrist

Chapter 9: Boundaries and Boundary Conditions

WALL Moving WALL Boundaries

pin offset is positive, the piston reaches TDC before 0 CAD. When the wrist pin offset is negative, the piston reaches TDC after 0 CAD.

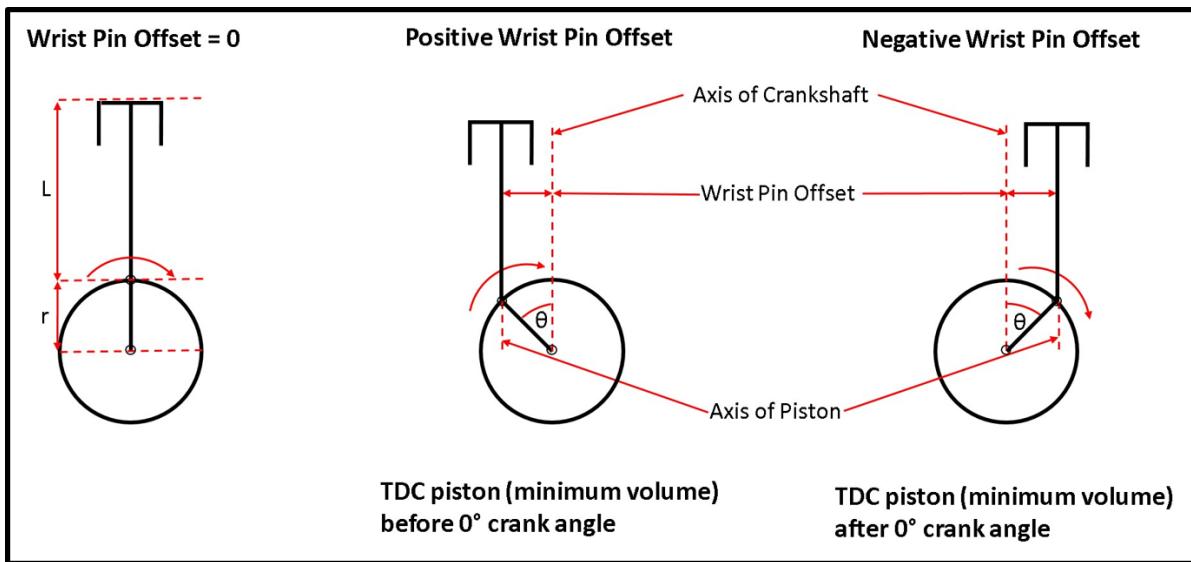


Figure 9.2: The geometry of the moving piston for engine applications and the sign convention for wrist pin offset.

Since the connecting rod length, L , remains constant, the piston will reach a lower TDC when the wrist pin offset is non-zero, as shown below in 9.3.

Chapter 9: Boundaries and Boundary Conditions

WALL Moving WALL Boundaries

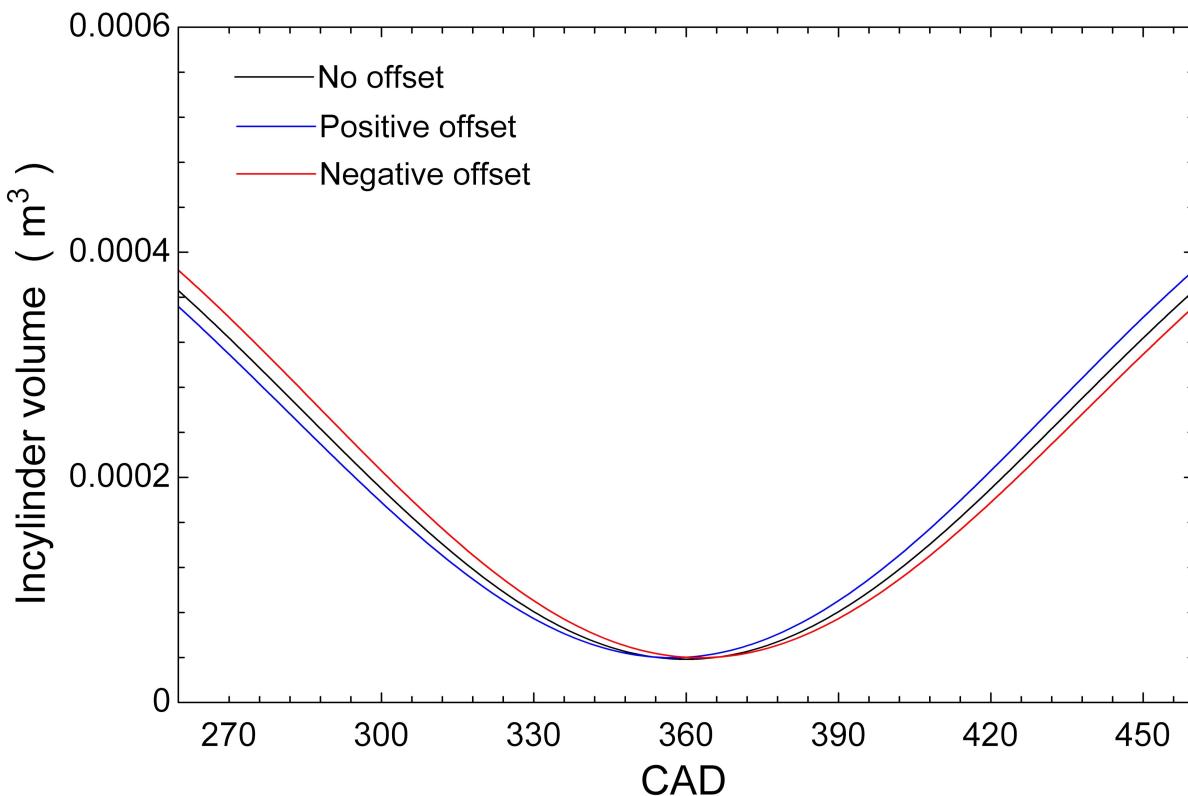


Figure 9.3: In-cylinder volume versus crank angle for no offset, positive offset, and negative

You can calculate the piston location in the z direction by

$$z_{\text{piston}} = 2a - l \left[1 - \cos \left\{ \sin^{-1} \left(\frac{a \sin \theta + \text{wrist_pin_offset}}{l} \right) \right\} \right] - a(1 - \cos \theta), \quad (9.40)$$

where a represents the crank radius (in *meters*), l is the connecting rod length (in *meters*), and θ is the crank angle (in *radians*).

To direct CONVERGE to generate a position table for the piston in a single-cylinder engine, set [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value* = *AUTO_GENERATE* for the piston boundary. Ensure that the piston is at bottom dead center in your [surface geometry file](#). CONVERGE determines the piston position table from parameters in [engine.in](#), using a crank-slider motion with options for the wrist pin offset. CONVERGE calculates the piston profile every 0.1 *CAD*, determines the smallest value for piston position from those generated in the piston position table, and then subtracts the smallest value from the calculated piston position in order to set the BDC position.

To direct CONVERGE to generate a position table for the piston in a non-engine application, set [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value* = *MOTION_CONFIG* and invoke a CRANK_SLIDER object in [motion_sets.in](#). In this case, CONVERGE determines the

Chapter 9: Boundaries and Boundary Conditions

WALL Moving WALL Boundaries

piston position table from the parameters specified in the piston motion file. CONVERGE calculates the piston profile every 0.1 CAD. Note that CONVERGE measures piston displacement relative to its original position in the [surface geometry file](#).

If you have a position table for the piston, save this file to the Case Directory and specify this file name in [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value*. The piston position table must specify the position of the piston as a function of time or crank angle degree.

To direct CONVERGE to write a piston position output file, set [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *echo_profile* = 1 for the piston boundary. The output file is named [piston_profile<boundary ID>.out](#), where the boundary ID is the value in [boundary.in](#) that corresponds to the piston. Note that CONVERGE measures piston displacement relative to its original position in the [surface geometry file](#).

For multi-cylinder engines, you must specify the phase lag angle (ϕ) and the inclination angle (θ) relative to the crank axis for each piston. Specify these parameters via [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *phase_lag_angle* and [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *inclination_angle*. Note that the inclination angle must follows the right-hand rule of rotation with respect to the rotation axis specified via [boundary.in](#) > *rotation_axis*. The phase lag angle in different cylinders represents the firing order. For example, for a four-cylinder engine with firing order 1-3-4-2, the phase lag angles would be 0.0 for the first cylinder, +180.0 for the third cylinder, +360.0 for the fourth cylinder, and +540.0 for the second cylinder.

Translating WALL: Valves

Valves are an example of of [translating WALL boundaries](#). You can specify valve motion via a valve lift profile. CONVERGE can automatically create OPEN and CLOSE [events](#) between regions separated by valves based on the valve lift profile you provide. To invoke a valve motion profile, specify the valve motion file name (e.g., *intake_lift.in*) for [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value*.

Note that when you specify the valve motion direction in a V engine or other simulation in which the cylinder is inclined relative to the z axis, CONVERGE rotates the valve motion direction vector (or angle) specified for that piston's velocity boundary condition in [boundary.in](#). If you have already measured the valve motion direction with the cylinder aligned with the z axis, you do not need to calculate the new direction vector (or angle) of the valve in the V engine configuration. To change the surface geometry from a straight engine (piston moves along the z axis) to a V engine (pistons are inclined relative to the z axis), specify [boundary.in](#) > *rotation_axis*.

Reloft WALL Boundaries

If the triangles on a WALL boundary will become degenerate due to the motion of neighboring boundaries, CONVERGE can automatically reloft that boundary. To use this option, set [boundary.in](#) > *boundary_conditions* > *boundary* > *geometry_motion* = RELOFT.

Chapter 9: Boundaries and Boundary Conditions

WALL Reloft WALL Boundaries

CONVERGE will calculate the quality of the triangulation at each time-step and retriangulate the boundary if necessary.

The reloft option is available for WALL boundaries that meet all of the criteria listed below. For WALL boundaries that are part of an interface (*i.e.*, the forward and reverse sides of a [conjugate heat transfer \(CHT\) interface](#)), either both sides or neither side must be marked **RELOFT**.

- All triangles on the boundary have the same normal direction. While this is not a strict requirement, it is a strong recommendation. If a reloft boundary is non-planar, the new triangulation will not identically represent the previous surface geometry.
- There are no internal vertices on the boundary (*i.e.*, all vertices are shared with at least one other boundary).
- The vertices on the boundary form two concentric loops (as shown below in Figure 9.4) or one loop (as shown below in Figure 9.5).
- The boundary motion ([boundary.in > boundary_conditions > boundary > motion](#)) is **STATIONARY** or **DEPENDENT**.
- If any rotational motion is imposed on the boundary by a neighboring boundary, the rotational axis is the same for both boundaries and is parallel to the normal direction of the triangles on the reloft boundary.

The reloft option is also available for [TWO_D](#) boundaries. Only the first three criteria apply to TWO_D reloft boundaries.

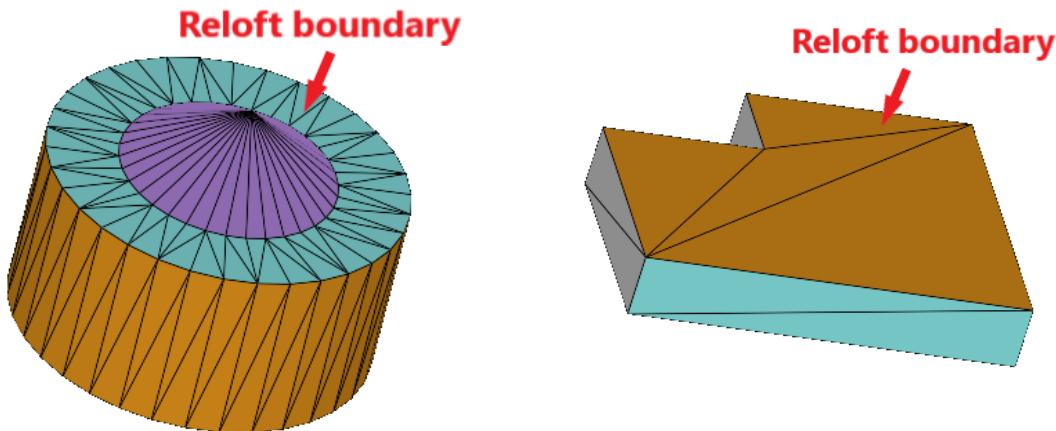


Figure 9.4: Example of a reloft boundary with two concentric loops. In this geometry, the outer cylinder rotates while the inner disc (purple) remains stationary.

Figure 9.5: Example of a reloft boundary with a single loop. In this geometry, all boundaries are stationary except the front (green) boundary, which translates toward the opposite wall.

Chapter 9: Boundaries and Boundary Conditions

WALL Reloft WALL Boundaries

For a stationary reloft boundary, CONVERGE will use the specified [velocity](#) boundary condition. Thus the velocity boundary condition is not explicitly linked to the triangle motion (skewing and relofting due to relative motion of the connected boundaries). This configuration is appropriate for relofted boundaries whose connected motion does not introduce motion in the direction normal to its triangles (*e.g.*, a seal between two rotating parts), but could lead to mass conservation issues for other configurations. Figures 9.4 and 9.5 above are both stationary reloft boundaries.

If a reloft boundary has translational motion that is normal to its triangles, we recommend setting the boundary motion to *DEPENDENT* to conserve mass. In this case, CONVERGE calculates the velocity boundary condition of the relofted surface based on the actual triangle motion caused by the two connected boundaries, guaranteeing mass conservation for any case.

Rather than configuring the reloft option for individual boundaries, you can set `inputs.in > grid_control > auto_reloft_flag = 1`. CONVERGE will then try to identify WALL boundaries that meet the above criteria and have triangles that will become degenerate due to connected motion. Only boundaries of the type shown in Figure 9.4, with two concentric loops, can be automatically identified as reloft boundaries. CONVERGE reports any boundaries identified as such during the simulation setup. Boundaries with a single loop and TWO_D boundaries will not be identified.

Linked WALL Boundaries

You can specify a linked WALL boundary that inherits motion from a parent WALL boundary. While linked boundaries are typically [moving](#) boundaries, they can inherit stationary motion. The linked boundary inherits data from `boundary.in > boundary > geometry_motion` and `boundary.in > boundary > velocity` from the boundary that it is linked to. For example, if a linked boundary is translating, vertices within that boundary will move with the velocity inherited from the parent boundary.

Note that if `boundary.in > boundary > motion = FSI` for the parent boundary, the linked boundary will be included in the force calculation of the [fluid-structure interaction](#) (FSI) model only if the linked boundary is also listed in `fsi.in > *_objects > object > boundary_id` for the FSI object associated with the parent boundary. If the linked boundary is not listed in `fsi.in`, it can still inherit motion from the parent boundary, but it will not be included in the FSI force calculation.

You cannot link a boundary to itself, to another linked boundary (*i.e.*, you cannot set up a chain of linked boundaries), or to a non-WALL boundary. Multiple linked boundaries can inherit motion from a single WALL boundary.

To set up a linked boundary, set `boundary.in > boundary > motion = LINKED`. Then, set `boundary.in > boundary > velocity > linked_id` to the boundary ID of the parent boundary.

Chapter 9: Boundaries and Boundary Conditions

WALL Velocity Boundary Conditions

Velocity Boundary Conditions

[Dirichlet](#) (*i.e.*, no-slip), [slip](#), and [law-of-the-wall](#) are available to solve the momentum equation at WALL boundaries.

This section describes how to set up boundary conditions for fixed stationary WALL boundaries (*i.e.*, non-[moving](#), non-[sliding](#) WALL boundaries).

For WALL boundaries associated with solid streams ([stream_settings.in](#) > *stream_list* > *stream* > *solid_flag* = 1), only the Dirichlet boundary condition is allowed.

If you set a law-of-the-wall [temperature boundary condition](#), you must set the velocity boundary condition to law-of-the-wall.

Set up a velocity boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *type* and [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value*. For a [temporally](#) and/or [spatially](#) varying velocity boundary condition (not available for all conditions), specify a file name for *value* and include that file in your case setup.

Dirichlet

You can specify a Dirichlet velocity boundary condition for [sliding](#) and [moving](#) WALL boundaries. This boundary condition is also known as a no-slip boundary condition.

Slip

A slip boundary condition is a special case of the Neumann boundary condition. For this boundary condition, CONVERGE sets the components of the stress tensor that are tangential to the wall to 0.0. You can specify a slip velocity boundary condition for [sliding](#) and [moving](#) WALL boundaries.

Law-of-the-Wall

For high Reynolds number turbulent flows, it may not be possible to resolve the viscous sublayer of the flow ([Warsi, 1993](#)). The law-of-the-wall boundary condition is suitable for these cases, in which the turbulent boundary layer resolution is not sufficient (*i.e.*, the viscous sublayer is not resolved). The law-of-the-wall is a logarithmic curve fit of a turbulent boundary layer. In practice, the law-of-the-wall profile is used to determine the tangential components of the stress tensor at the wall.

For simulations with an [LES turbulence model](#), CONVERGE includes two options for the velocity law-of-the-wall boundary condition: the standard law-of-the-wall model and the Werner and Wengle wall model ([Werner and Wengle, 1991](#)). Select the desired option via [tubulence.in](#) > *LES_models* > *les_wall_model*. The standard law-of-the-wall model requires iterations. The Werner and Wengle model is more complicated than the standard law-of-the-wall model, but it does not require iterations. We recommend using the Werner and Wengle model when your simulation includes an LES turbulence model.

Chapter 9: Boundaries and Boundary Conditions

WALL Velocity Boundary Conditions

For simulations with a [k- \$\epsilon\$ turbulence model](#), CONVERGE includes one option for the velocity law-of-the-wall boundary condition: the Launder and Spalding ([Launder and Spalding, 1974](#)) wall model.

For the standard law-of-the-wall model, the shear speed is defined as

$$u_\tau = \frac{|U_{i,\text{tang}}|}{\left(\frac{1}{\kappa} \ln \left(E \frac{\rho u_\tau y}{\mu} \right) \right)} \quad (9.41)$$

for y^+ greater than 11.05 and as

$$u_\tau^2 = \frac{\mu U_{i,\text{tang}}}{\rho y} \quad (9.42)$$

for y^+ less than 11.05. The above equation is solved iteratively. The tangential components of the wall stress are given as

$$\sigma_{ij} = \left[\left(\frac{U_{i,\text{tang}}}{|U_{i,\text{tang}}|} \rho u_\tau^2 \right) n_j \right]. \quad (9.43)$$

For the Werner and Wengle model, if the magnitude of the tangential velocity satisfies

$$|U_{i,\text{tang}}| \leq \frac{\mu}{2\rho y} A^{\frac{2}{1-B}}, \quad (9.44)$$

then the shear speed is given as

$$u_\tau = \sqrt{\frac{\mu |U_{i,\text{tang}}|}{\rho y}}, \quad (9.45)$$

where A is 8.3 and B is 1/7. If the inequality is not satisfied then the shear speed is given by

Chapter 9: Boundaries and Boundary Conditions

WALL Velocity Boundary Conditions

$$u_\tau = \sqrt{\left[\frac{1-B}{2} A^{\frac{1+B}{1-B}} \left(\frac{\mu}{\rho y} \right)^{1+B} + \frac{1+B}{A} \left(\frac{\mu}{\rho y} \right)^B |U_{i,\text{tang}}| \right]^{\frac{2}{1+B}}}. \quad (9.46)$$

Once the shear speed is calculated, Equation 9.2 is used to determine the stress tensor.

The Launder and Spalding model is given as

$$u^* = \begin{cases} \frac{1}{\kappa} \ln(Ey^+) & y^+ > 11.2 \\ y^+ & y^+ \leq 11.2, \end{cases} \quad (9.47)$$

where E is 9.8,

$$u^* = \frac{U_{i,\text{tang}} \rho u_\tau n_j}{\sigma_{ij}}, \quad (9.48)$$

and

$$y^+ = \frac{y \rho u_\tau}{\mu}. \quad (9.49)$$

The shear speed is given as

$$u_\tau = c_\mu^{1/4} k^{1/2}, \quad (9.50)$$

where κ is the von Karman constant (typically 0.42 but can be changed via [turbulence.in](#) > Wall_modeling > law_kappa), k is the turbulent kinetic energy, and c_μ is the model constant from the k - ϵ model. The tangential relative velocity is given as

$$U_{i,\text{tang}} = U_i - (U_j n_j) n_i. \quad (9.51)$$

Rearranging Equation 9.2, an expression for the law-of-the-wall stress tensor can be derived as

Chapter 9: Boundaries and Boundary Conditions

WALL Velocity Boundary Conditions

$$\sigma_{ij} = \frac{U_{i,\text{tang}} \kappa \rho u_r n_j}{\ln(Ey^+)} . \quad (9.52)$$

If you specify law-of-the-wall, then you must specify an absolute wall roughness for each wall boundary. If you specify a wall roughness, the law-of-the-wall is modified as shown in [Cebeci and Cousteix \(2005\)](#) to account for the user-specified roughness. The law-of-the-wall for mean velocity modified for roughness has the form

$$u^* = \frac{1}{\kappa} \ln(Ey^+) - \beta, \quad (9.53)$$

where

$$\beta = \frac{1}{\kappa} \ln f_r. \quad (9.54)$$

The roughness function, f_r , quantifies the shift of the intercept due to roughness effects, and β depends on the type (uniform sand, rivets, etc.) and size of the roughness. For uniform roughness elements, β has been found to be well-correlated with the non-dimensional roughness height as

$$k^+ = \frac{\rho K_s u^*}{\mu}, \quad (9.55)$$

where K_s is the physical roughness height. There are three distinct roughness regimes and the values of β for each regime are given by the following set of equations.

For the hydrodynamically smooth regime ($k^+ \leq 2.25$),

$$\beta = 0. \quad (9.56)$$

For the transitional regime ($2.25 < k^+ \leq 90$),

$$\beta = \frac{1}{\kappa} \ln \left[\frac{k^+ - 2.25}{87.75} + r_s k^+ \right] \sin \left\{ 0.4258 (\ln k^+ - 0.811) \right\}, \quad (9.57)$$

where r_s is a roughness constant that depends on the type of roughness.

Chapter 9: Boundaries and Boundary Conditions

WALL Velocity Boundary Conditions

In the fully rough regime ($k^+ > 90$),

$$\beta = \frac{1}{\kappa} \ln \left(1 + r_s k^+ \right). \quad (9.58)$$

Depending on the regime of roughness, the corresponding Equations 9.3, 9.4, and 9.5 are used for the modified law-of-the-wall in Equation 9.0. The results of these equations are then used to evaluate the shear stress at the wall and the wall functions for the mean temperature and turbulent quantities.

To model the wall roughness effects, you must specify two roughness parameters ([boundary.in](#) > *boundary_conditions* > *boundary* > *roughness*): the absolute roughness (K_s , the roughness height, given in *meters*) and the roughness constant (r_s , unitless). The default value for absolute roughness is zero, which corresponds to a perfectly smooth wall. If the wall is not smooth, you can use a sand-grain absolute roughness value. These values are available in many fluid mechanics textbooks. For example, the absolute roughness is 0.00026 *m* for cast iron, 0.0009 *m* for riveted steel, and 0.000045 *m* for commercial steel. You can specify the absolute roughness as a constant across the entire surface or as a [spatially](#) or [temporally](#) varying profile.

Choosing a proper roughness constant depends mainly on the type of roughness. The default roughness constant is 0.5. When 0.5 is used with $k-\varepsilon$ turbulence models, it reproduces Nikuradse's resistance data for pipes with tightly-packed, uniform sand-grain roughness. For non-uniform sand-grains, ribs, or wire-mesh roughness, a larger value (0.5 to 1.0) is more appropriate.

Note that wall-adjacent cells that are smaller than the absolute roughness will not improve the accuracy of the simulation results near the wall.

Pressure Boundary Conditions

You can specify a zero-gradient Neumann pressure boundary condition for a WALL boundary.

Set up a pressure boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *pressure* > *type*.

Temperature Boundary Conditions

[Dirichlet](#), [Neumann](#), [law-of-the-wall](#), [convection](#), [radiation convection](#), and [flux](#) temperature boundary conditions are available to solve the energy equation at WALL boundaries. (An additional temperature boundary condition is available for [INTERFACE](#) boundaries, which are a special type of WALL boundaries). For WALL boundaries associated with solid streams ([stream_settings.in](#) > *stream_list* > *stream* > *solid_flag* = 1), only the Dirichlet boundary condition is allowed.

Chapter 9: Boundaries and Boundary Conditions

WALL Temperature Boundary Conditions

Set up a temperature boundary condition via [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [temperature](#) > [type](#) and [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [temperature](#) > [value](#). For a [temporally](#) and/or [spatially](#) varying temperature boundary condition (not available for all conditions), specify a file name for [value](#).

Dirichlet

The Dirichlet boundary condition is suitable for cases in which the viscous sublayer is resolved. If the near-wall resolution is inadequate (*i.e.*, if the viscous sublayer is not resolved), we recommend [law-of-the-wall](#) instead.

For the Dirichlet boundary condition, heat transfer at the wall occurs by conduction and is given by

$$\dot{q} = k \frac{\partial T}{\partial x_i} = k \frac{(T - T_w)}{(x - x_w)} n_i. \quad (9.59)$$

Neumann (Adiabatic)

CONVERGE allows a zero-gradient Neumann boundary condition. For this boundary condition, there is no heat transfer between the wall and the fluid (*i.e.*, the boundary is adiabatic).

Law-of-the-Wall

The law-of-the-wall temperature boundary condition is suitable for cases in which the turbulent boundary layer resolution is not sufficient (*i.e.*, the viscous sublayer is not resolved). When using the law-of-the-wall option, you must also configure the heat transfer model and the wall distance calculation scheme, both of which are described below.

Heat Transfer Models

For the temperature law-of-the-wall boundary condition, CONVERGE includes several heat transfer models: O'Rourke and Amsden ([Amsden, 1997](#)), Han and Reitz ([Han and Reitz, 1997](#)), Angelberger (Angelberger [et al.](#), 1997), GruMo-UniMORE ([Berni et al., 1993](#)), and Jayatilleke ([Jayatilleke, 1969](#)).

The Han and Reitz model accounts for compressible effects.

The Angelberger model accounts for quasi-isothermal flow (*e.g.*, in intake pipes or compression during engine simulations) and for non-isothermal wall flow (*e.g.*, in the combustion chamber during combustion or in exhaust pipes). The Angelberger model consistently predicts lower wall heat fluxes than the Han and Reitz model.

The GruMo-UniMORE model is recommended for simulating highly-charged/highly-downsized spark-ignited engines, and this model may improve over-estimation of the wall heat transfer as predicted by the Angelberger and Han and Reitz models.

Chapter 9: Boundaries and Boundary Conditions

WALL Temperature Boundary Conditions

The Jayatilleke model is applicable to general flows and accounts for the effects of wall roughness. For fluids with low Prandtl numbers (*e.g.*, air), the Jayatilleke model predicts lower heat transfer for rough walls than for smooth walls.

For the O'Rourke and Amsden model, the wall heat transfer is given by

$$k \frac{\partial T}{\partial x_i} = \frac{\mu_m c_p F (T_f - T_w)}{\text{Pr}_m y} n_i, \quad (9.60)$$

where

$$F = \begin{cases} 1.0 & y^+ < 11.05 \\ \frac{\left(\frac{y^+ \text{Pr}_m}{\text{Pr}_t} \right)}{\frac{1}{\kappa} \ln(y^+) + B + 11.05 \left(\frac{\text{Pr}_m}{\text{Pr}_t} - 1 \right)} & y^+ > 11.05, \end{cases} \quad (9.61)$$

$$y^+ = \frac{\rho u_\tau y}{\mu_m}, \quad (9.62)$$

k is the molecular conductivity, κ is the von Karman constant (typically 0.42 but can be changed via [*turbulence.in*](#) > *Wall_modeling* > *law_kappa*), B is given by [*turbulence.in*](#) > *Wall_modeling* > *law_c*, Pr_m is the molecular Prandtl number, Pr_t is the turbulent Prandtl number, T_f is the fluid temperature, T_w is the wall temperature, and u_τ is the shear speed (taken from the momentum law-of-the-wall).

For the Han and Reitz model, wall heat transfer is given by

Chapter 9: Boundaries and Boundary Conditions

WALL Temperature Boundary Conditions

$$k \frac{dT}{dx_i} = \begin{cases} \frac{\mu c_p (T_f - T_w) n_i}{y^+ \text{Pr}_m} & y^+ < 11.05 \\ \frac{\rho c_p u_\tau T_f \ln\left(\frac{T_f}{T_w}\right) n_i}{2.1 \ln(y^+) + 2.513} & y^+ > 11.05. \end{cases} \quad (9.63)$$

For the Angelberger model, wall heat transfer is given by

$$k \frac{dT}{dx_i} = \frac{\rho_w c_p u_\tau T_w \ln\left(\frac{T_f}{T_w}\right) n_i}{\theta^+}, \quad (9.64)$$

where

$$\theta^+ = \begin{cases} \text{Pr} y^+ & y^+ \leq 13.2 \\ 2.075 \ln(y^+) + 3.9 & y^+ > 13.2. \end{cases} \quad (9.65)$$

For the GruMo-UniMORE model, wall heat transfer is given by

$$k \frac{dT}{dx_i} = \frac{\rho c_p u_\tau (T_f - T_w)}{\theta^+}, \quad (9.66)$$

where

$$\theta^+ = \begin{cases} \text{Pr} y^+, & y^+ \leq 13.2 \\ 2.075 \ln(y^+) + 13.2 \text{ Pr} - 5.34 & y^+ > 13.2. \end{cases} \quad (9.67)$$

For the Jayatilleke model, the relationship between the wall heat transfer (denoted here by \dot{q}), the wall temperature T_w , the near-wall fluid temperature T_f and y^+ is given by

Chapter 9: Boundaries and Boundary Conditions

WALL Temperature Boundary Conditions

$$\frac{(T_w - T_f) \rho c_p C_\mu^{1/4} k_p^{1/2}}{\dot{q}} = \begin{cases} \Pr_m y^+ + \frac{\rho \Pr_m C_\mu^{1/4} k_p^{1/2} u_p^2}{2\dot{q}}, & y^+ < y_T^+ \\ \Pr_t \left[\frac{1}{\kappa} \ln(Ey^+) + P \right] + \\ \frac{\rho C_\mu^{1/4} k_p^{1/2}}{2\dot{q}} \left[\Pr_t u_p^2 + (\Pr_m - \Pr_t) u_c^2 \right], & y^+ > y_T^+ \end{cases} \quad (9.68)$$

where C_μ is the model constant from the turbulence model, k_p is the near-wall turbulent kinetic energy, u_p is the near-wall velocity, E is 9.8, u_c is the mean velocity at $y^+ = y_T^+$, and y_T^+ is the value of y^+ at which the linear and logarithmic portions of the thermal law-of-the-wall intersect.

The factor P takes a different form depending on whether the wall is smooth or rough. For a smooth wall ([boundary.in](#) > `boundary_conditions` > `boundary` > `roughness` > `height` = 0), P is given by

$$P = 9.24 \left[\left(\frac{\Pr_m}{\Pr_t} \right)^{3/4} - 1 \right] \left[1 + 0.28 e^{-0.007 \Pr_m / \Pr_t} \right]. \quad (9.69)$$

For a rough wall ([boundary.in](#) > `boundary_conditions` > `boundary` > `roughness` > `height` > 0), CONVERGE replaces P in Equation 9.68 with

$$P_{rough} = 3.15 \Pr_m^{0.695} \left(\frac{1}{E'} - \frac{1}{E} \right)^{0.359} + \left(\frac{E'}{E} \right)^{0.6} P, \quad (9.70)$$

where $E' = E/f_r$ depends on the [roughness function](#) f_r .

Wall Distance Calculation Schemes

Different wall distance (y) calculation schemes are available. For the first scheme, CONVERGE calculates the distance to a moving boundary as

$$y = 0.5 \left(V_{fullcellsize} \right)^{1/3}, \quad (9.71)$$

where $V_{fullcellsize}$ refers to the volume of a full size cell at the same embedding scale. For a stationary boundary, y is equal to the distance from the effective boundary cell center to the boundary itself. If the boundary cell is not part of a cell pair, the effective boundary cell

Chapter 9: Boundaries and Boundary Conditions

WALL Temperature Boundary Conditions

center is the center of the cell adjacent to the boundary. If the boundary cell is part of a cell pair, the effective boundary cell center is the center of the cell pair.

To set up a heat model with this wall distance calculation scheme, set [*turbulence.in*](#) > *Wall_modeling* > *wall_dist_flag* = 0.

For the second wall distance calculation scheme, CONVERGE calculates the distance to moving and stationary boundaries as

$$y = 0.3 \left(V_{fullcellsize} \right)^{1/3}, \quad (9.72)$$

where $V_{fullcellsize}$ refers to the volume of a full size cell at the same embedding scale and 0.3 is an empirical value.

To set up a heat model with this wall distance calculation scheme, set [*turbulence.in*](#) > *Wall_modeling* > *wall_dist_flag* = 1.

Convection

For the convection boundary condition, heat transfer at the wall occurs by convection and is given by

$$\dot{q}_{convection} = k \frac{\partial T}{\partial x} \Big|_w, \quad (9.73)$$

where k is the thermal conductivity (in $W/m \cdot K$). Note that the sign convention (*i.e.*, $\dot{q} < 0$ is heat transfer into the domain) is addressed by the normal vector orientation at the wall face. CONVERGE evaluates this expression in one of two ways.

For solid walls and laminar fluids, CONVERGE discretizes the temperature gradient to obtain a relation between the wall temperature T_w (in K) and the first wall-normal cell temperature T_i (in K). The wall heat transfer is then expressed as

$$\dot{q}_{convection} = k \frac{T_i - T_w}{\partial x} = h(T_\infty - T_w), \quad (9.74)$$

where T_∞ is the ambient temperature (in K), h is heat transfer coefficient (in $W/m^2 \cdot K$), and T_{solid} is the temperature at the boundary of the solid (in K). The wall temperature T_w is then solved as

Chapter 9: Boundaries and Boundary Conditions

WALL Temperature Boundary Conditions

$$T_w = f(T_i, T_\infty). \quad (9.75)$$

For turbulent fluids, CONVERGE uses a wall function to calculate the heat flux at the wall. A wall temperature is specified, and the wall function is used to calculate the heat flux at the wall as

$$\dot{q}_w = f_w(T_i, T_w). \quad (9.76)$$

From this calculated heat flux, CONVERGE re-calculates the wall temperature. CONVERGE iterates this solution procedure to calculate the temperature at the wall. For more details about wall functions, refer to [Chapter 13 - Turbulence Models](#).

For a convection boundary condition, you must specify the ambient temperature (in K) and the heat transfer coefficient (in $W/m^2\text{-}K$).

Radiation Convection

For the radiation convection boundary condition, heat transfer at the wall occurs by radiation and convection and is given by

$$q_{rad_conv} = h(T_\infty - T) + \varepsilon\sigma(T_{rad}^4 - T^4), \quad (9.77)$$

where T_∞ is the ambient temperature for convection, ε is the emissivity of the surface, σ is the Stefan-Boltzmann constant, and T_{rad} is the ambient temperature for radiation.

For a convection boundary condition, you must specify the ambient temperature (in K) and the heat transfer coefficient (in $W/m^2\text{-}K$), the ambient temperature for radiation (in K), and the emissivity of the surface. To model a pure radiation case, set the ambient temperature for convection and the heat transfer coefficient to 0.0.

Flux

For the flux boundary condition, heat transfer occurs at a user-specified rate across the entire surface area of the boundary.

For a flux temperature boundary condition, you must specify a heat flux value (in W/m^2). A positive heat flux means that heat is flowing out of the domain, while a negative heat flux means that heat is flowing into the domain. For example, if a WALL boundary is adjacent to a fluid, a positive heat flux would indicate that heat is flowing from the fluid to the WALL boundary, and a negative heat flux would indicate that heat is flowing from the WALL boundary into the fluid.

1D Conjugate Heat Transfer

CONVERGE includes a 1D conjugate heat transfer (CHT) model for heat transfer in thin solid layers between a fluid and a solid. This model is available for Dirichlet and law-of-the-wall boundary conditions. Refer to the 1D CHT Model section for information about configuring this model.

Nucleate Boiling Model

Nucleate boiling is a boiling regime that occurs when a single-phase liquid contacts a solid under specific conditions. If the temperature of the solid exceeds the saturation temperature of the liquid by a certain liquid-specific amount and the heat flux from the wall is below the critical heat flux, nucleate boiling will occur. Formation of vapor bubbles at the solid surface and subsequent separation of these vapor bubbles characterizes this boiling regime. Nucleate boiling is a particularly efficient mode of heat transfer because the separation of vapor bubbles causes rapid mixing and enhances convective heat transfer.

CONVERGE accounts for the additional heat transfer from nucleate boiling by computing the heat flux through a wall as the sum of single-phase heat flux and heat flux due to nucleate boiling:

$$q = q_{\text{single-phase}} + q_{\text{bw}}, \quad (9.78)$$

where q_{bw} is the additional heat flux due to boiling. To represent the additional heat flux, CONVERGE uses the Rohsenow correlation ([Rohsenow, 1952](#)):

$$\frac{q_{\text{bw}}}{A} = \mu_l h_{\text{lat}} \sqrt{\frac{g(\rho_l - \rho_v)}{\sigma}} \left(\frac{c_{pl}(T_w - T_{\text{sat}})}{C_{sf} h_{\text{lat}} \text{Pr}_l^{1+m}} \right)^3, \quad (9.79)$$

where μ_l is the liquid viscosity, h_{lat} is the latent heat of vaporization, g is gravitational acceleration, ρ_l is the density of the liquid, ρ_v is the density of the vapor, σ is the surface tension, c_{pl} is the specific heat capacity of the liquid, C_{sf} is a constant that varies with the surface-liquid combination, and Pr_l is the Prandtl number. The quantity $T_w - T_{\text{sat}}$ is the difference between the wall temperature and the saturation temperature. The exponent m is 0.0 when the liquid is water and 0.7 in all other cases. Specify the exponent m via [nucleate_boiling.in > nb_exponent](#). Table 9.2 lists C_{sf} and m values for some commonly simulated surface-fluid combinations ([Incropera, 1990](#)).

The critical heat flux, q_{max} , marks the upper limit of the nucleate boiling regime, and is based on the following equation ([Lienhard](#)):

$$q_{\text{max}} = 0.149 \rho_v^{1/2} h_{\text{lat}} \sqrt[4]{g(\rho_l - \rho_v)\sigma}. \quad (9.80)$$

Chapter 9: Boundaries and Boundary Conditions

WALL Temperature Boundary Conditions

To activate the nucleate boiling model in CONVERGE, set [inputs.in](#) > *feature_control* > *nucleate_boiling_flag* = 1 and include [nucleate_boiling.in](#) in your case setup.

Table 9.2: Rohsenow correlation parameters for some fluid-surface combinations.

Fluid-surface combination		C_{sf}	m
Water-copper	Scored	0.0068	0.0
	Polished	0.0130	0.0
Water-stainless steel	Chemically etched	0.0130	0.0
	Mechanically polished	0.0130	0.0
Ground and polished		0.0060	0.0
Water-brass		0.0060	0.0
Water-nickel		0.006	0.0
Water-platinum		0.0130	0.0
n-Pentane-copper	Polished	0.0154	0.7
	Lapped	0.0049	0.7
Benzene-chromium		0.101	0.7
Ethyl alcohol-chromium		0.0027	0.7

Species Boundary Conditions

You can specify a zero-gradient Neumann species boundary condition for a WALL boundary.

Set up a species boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *species*.

Passive Boundary Conditions

You can specify a zero-gradient Neumann passive boundary condition for a WALL boundary.

Set up a passive boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *passive*.

Turbulent Kinetic Energy Boundary Conditions

You must specify a boundary condition for the turbulent kinetic energy (tke) equation when your simulation includes certain [turbulence models](#).

Dirichlet and Neumann boundary conditions are available to solve the tke equation at WALL boundaries.

Chapter 9: Boundaries and Boundary Conditions

WALL Turbulent Kinetic Energy Boundary Conditions

Set up a tke boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *turbulence* > *tke*.

Dirichlet

You can specify a Dirichlet tke boundary condition for a WALL boundary.

Neumann

You can specify a zero-gradient Neumann tke boundary condition for a WALL boundary.

Law-of-the-Wall

You can specify a law-of-the-wall tke boundary condition for a WALL boundary.

Turbulent Dissipation Boundary Conditions

You must specify a boundary condition for the turbulent dissipation rate (eps) equation when your simulation includes certain [turbulence models](#).

Wall model (Dirichlet) and Neumann boundary conditions are available to solve the eps equation at WALL boundaries.

Set up an eps boundary condition via [boundary.in](#) > *boundary_conditions* > *boundary* > *turbulence* > *eps*.

Wall Model (Dirichlet)

CONVERGE calculates the Dirichlet turbulent dissipation boundary condition at the center of the cell adjacent to the wall via the wall model as

$$\varepsilon = \frac{c_{\mu}^{0.75} k^{1.5}}{\kappa y}, \quad (9.81)$$

in which ε is the turbulent dissipation, k is the turbulent kinetic energy, y is the distance from the wall to the middle of the cell, c_{μ} is a turbulence model constant, and κ is von Karman's constant (typically 0.42 but can be changed via [turbulence.in](#) > *Wall_modeling* > *law_kappa*).

Neumann

A zero-gradient Neumann boundary condition is available to solve the eps equation at WALL boundaries.

Specific Dissipation Rate Boundary Conditions

You must specify a boundary condition for the specific dissipation rate (omega) equation when your simulation includes certain [turbulence models](#).

Dirichlet, wall model, and Neumann boundary conditions are available to solve the omega equation at WALL boundaries.

Chapter 9: Boundaries and Boundary Conditions

WALL Specific Dissipation Rate Boundary Conditions

Set up an omega boundary condition via [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [omega](#) > [type](#) and [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [omega](#) > [value](#). For a [temporally](#) and/or [spatially](#) varying omega boundary condition (not available for all conditions), specify a file name for [value](#).

Dirichlet

You can specify a Dirichlet omega boundary condition for a WALL boundary.

Wall Model (Dirichlet)

CONVERGE calculates the Dirichlet turbulent dissipation boundary condition at the center of the cell adjacent to the wall via the wall model as

$$\varepsilon = \frac{c_{\mu}^{0.75} k^{1.5}}{\kappa y}, \quad (9.82)$$

in which ε is the turbulent dissipation, k is the turbulent kinetic energy, y is the distance from the wall to the middle of the cell, c_{μ} is a turbulence model constant, and κ is von Karman's constant (typically 0.42 but can be changed via [turbulence.in](#) > [Wall_modeling](#) > [law_kappa](#)).

Neumann

You can specify a zero-gradient Neumann omega boundary condition for a WALL boundary.

Electric Potential Boundary Conditions

Boundary conditions for the [electric potential solver](#) are available for solid boundaries when [inputs.in](#) > [feature_control](#) > [electric_potential_flag](#) = 1. You can specify Dirichlet, flux (current per area), or Neumann boundary conditions.

Neumann boundary conditions are appropriate for insulating materials. If you do not set an electric potential boundary condition for a solid boundary in a simulation with [inputs.in](#) > [feature_control](#) > [electric_potential_flag](#) = 1, CONVERGE assumes a Neumann boundary condition for that boundary.

Configure electric potential boundary conditions in [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [electric](#).

9.3 PERIODIC

PERIODIC boundaries allow you to simulate only a portion of a geometry (e.g., an engine sector) to save computational resources. Sector (pie-shaped) and planar (box-shaped) PERIODIC boundaries are available in CONVERGE. You must set up PERIODIC boundaries in pairs.

Chapter 9: Boundaries and Boundary Conditions

PERIODIC

During a simulation, CONVERGE copies the values of each boundary condition on the first PERIODIC boundary to its matching boundary. To ensure physical results, verify that the perimeter geometry of boundary faces on the two PERIODIC boundaries match exactly.

A PERIODIC boundary can be stationary or [moving](#). The only type of motion allowed is translating. If a PERIODIC boundary is translating, vertices within that boundary will move according to the user-specified velocity. Periodic translating boundaries move differently than wall translating boundaries. Only the interior vertices (*i.e.*, vertices not shared by triangles on other boundaries) will translate. The vertices shared by triangles on neighboring boundaries will move (or remain stationary) according to the velocity boundary condition of these neighboring boundaries.

Consider the engine sector example shown below in Figure 9.6. The blue boundary is the moving (translating) piston face. The red boundary is the cylinder wall and the green boundary is the front periodic boundary. Notice that many of the triangles that constitute the periodic boundary are defined by vertices that will not move as part of the piston face.

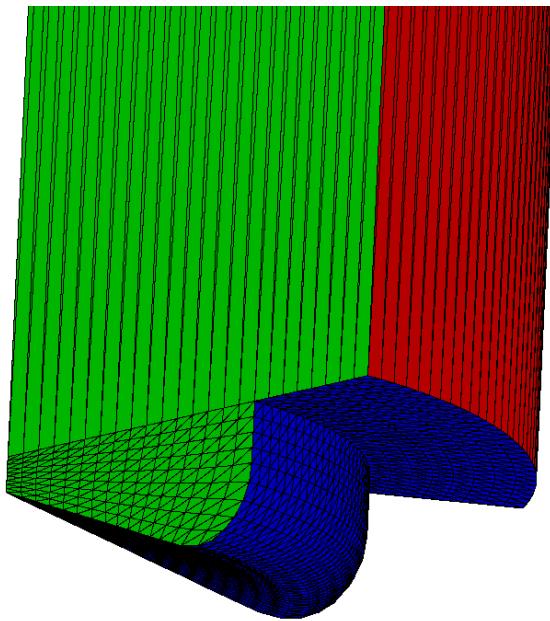


Figure 9.6: A sample sector engine case with a periodic face (green). Define the periodic boundary velocity boundary condition as translating in order to move the interior vertices on the periodic face in conjunction with the piston.

For the simulation to run without error, you must specify that these interior triangle vertices translate in conjunction with the piston. By defining this periodic boundary as translating, the interior triangle vertices will translate. The triangle vertices on the periphery of the

Chapter 9: Boundaries and Boundary Conditions

PERIODIC

periodic face will move according to the boundary type with which they are shared (*e.g.*, vertices shared with the piston will move according to the piston motion, vertices shared with the head will be stationary).

The direction of translation of the vertices on the periodic face must be tangential to the triangle face, so that each triangle will maintain the same normal vector. If there are no triangle vertices wholly contained within the periodic face (interior vertices), it makes no difference if the periodic boundary is defined as a translating or stationary type.

If a periodic boundary is adjacent to a moving boundary (such as the green periodic face adjacent to moving blue piston boundary shown above in Figure 9.6) and if this periodic boundary has interior triangles (*i.e.*, triangles that do not share a vertex with another boundary), then you must specify a translating velocity boundary condition for the periodic boundary.

To set up an PERIODIC boundary pair, set `boundary.in > boundary_conditions > boundary > type = PERIODIC` and configure the subsequent parameters in the `boundary_conditions > boundary` settings block. Typically each boundary has its own `boundary.in > boundary_conditions > boundary` settings block. For a pair of PERIODIC boundaries, however, information for both boundaries is included in a single `boundary_conditions > boundary` settings block. You must include the `boundary_conditions > boundary > match` sub-block for a pair of PERIODIC boundaries.

9.4 SYMMETRY

At SYMMETRY boundaries, CONVERGE sets the velocity boundary condition to slip and all other boundary conditions to Neumann. If a spray parcel interacts with a SYMMETRY boundary, the spray parcel is perfectly reflected (*i.e.*, not affected by friction at the wall) as if it had entered the domain from the opposite side of the boundary.

To set up a SYMMETRY boundary, set `boundary.in > boundary_conditions > boundary > type = SYMMETRY` and configure the subsequent parameters in the `boundary_conditions > boundary` settings block.

In some circumstances, you may want your simulation to include cells which have multiple non-coplanar faces that are assigned to SYMMETRY boundaries. In this case, you must ensure that each non-planar SYMMETRY boundary has a unique boundary ID.

9.5 TWO_D

To run a two-dimensional simulation in CONVERGE, use the TWO_D boundary type. The TWO_D boundary is not available for simulations that include [discrete phase modeling](#).

To set up a two-dimensional case in CONVERGE, create a three-dimensional surface including two identical, parallel boundaries and set their boundary type to TWO_D. Note that the normal vectors of the corresponding TWO_D boundaries must be in the z direction (positive or negative). Thus, the two-dimensional flow occurs in the xy plane. Ensure that in

Chapter 9: Boundaries and Boundary Conditions

TWO_D

the surface geometry, no vertices exist in the thickness between the parallel TWO_D boundaries.

At runtime, CONVERGE compresses the surface definitions in the z direction.

TWO_D boundaries cannot move independently, but they can be connected to moving boundaries. Any deformation of the TWO_D boundary resulting from connected motion must occur in the xy plane. CONVERGE can automatically [reloft](#) the TWO_D boundary if this motion causes the triangles to become degenerate.

To set up a TWO_D boundary, set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *type* = TWO_D and configure the subsequent parameters in the *boundary_conditions* > *boundary* settings block.

9.6 INTERFACE

An INTERFACE boundary separates two regions. You can use an INTERFACE boundary to separate regions with different phases (e.g., solid and fluid phases), such as in a [conjugate heat transfer](#) application. You can also use an INTERFACE boundary as a baffle to separate two fluid regions. Regardless of the application, CONVERGE generates independent grids for the regions on either side of the INTERFACE boundary.

To set up an INTERFACE boundary, set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *type* = INTERFACE and configure the subsequent parameters in the *boundary_conditions* > *boundary* settings block.

For an INTERFACE boundary, you must specify boundary conditions for both the forward and reverse sides of the interface. The forward and reverse sides are solely for boundary condition definition: these sides have no triangles flagged to them in the surface geometry file, nor do their boundary IDs appear in the exported [surface geometry file](#). By convention, the boundary facing the normal vectors of the triangles assigned to the INTERFACE is the forward boundary. The boundary facing the opposite direction of these normal vectors is the reverse boundary.

If you specify existing boundaries as the forward and reverse boundaries, CONVERGE will use the boundary conditions already assigned for these existing boundaries as the boundary conditions for the two sides of the INTERFACE boundary. Alternatively, the forward and reverse boundaries can be so-called virtual boundaries. Virtual boundaries are not composed of any triangles in the surface geometry file, and CONVERGE considers their boundary conditions to be a part of the INTERFACE boundary. In [*boundary.in*](#), however, virtual boundaries are set up as additional boundaries, with their temperature boundary condition set to COUPLED. This boundary condition imposes a thermal continuity across the INTERFACE, so that the temperatures and heat fluxes on both sides are the same.

To model thermal contact resistance between coupled forward and reverse boundaries (two solids or a solid and a fluid), set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *temperature* >

Chapter 9: Boundaries and Boundary Conditions

INTERFACE

value = COUPLED and then specify the contact resistance (in $K \cdot m^2/W$) via [*boundary.in*](#) > *boundary_conditions* > *boundary* > *temperature* > *contact_resistance*.

An INTERFACE boundary represents disconnect triangles similar to those automatically created to [*disconnect fluid regions*](#). If you set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *disconnect* = 0 for an INTERFACE boundary, the INTERFACE triangles cannot be disabled and no fluid flow is allowed between the two regions. Heat transfer across the INTERFACE boundary is allowed. This setup is typical for a conjugate heat transfer simulation of heat transfer between fluid and a solid piston. When *disconnect* = 0, both the forward and reverse virtual boundaries must be [*WALL*](#) boundaries.

If you set *disconnect* = 1, CONVERGE will treat the INTERFACE boundary as a set of [*disconnect triangles*](#) and thus will allow fluid flow across the INTERFACE boundary as specified by events in [*events.in*](#). This configuration works only for an INTERFACE between two fluid regions. You can configure OPEN and CLOSE events in [*events.in*](#) to control the flow across the interface. OPEN events will allow both mass and heat to move across the boundary. CLOSE events will prohibit mass and heat flow across the boundary. When *disconnect* = 1, both the forward and reverse virtual boundaries must be [*SYMMETRY*](#) boundaries.

Flow Through Boundaries

CONVERGE offers a flow-through INTERFACE boundary option. This option does not affect the flow in any way, and flow always passes through this boundary without a change in behavior. For example, if you were to perform a channel flow simulation and then place a flow-through INTERFACE boundary in the channel cross section normal to the axial direction, the results would not change. Unlike disconnect triangles at interfaces, unless otherwise specified, this boundary is always OPEN. You can configure OPEN/CLOSE events for a flow through boundary, but it is not required.

INTERFACE Boundaries for the MRF Approach

For the [*multiple reference frame \(MRF\) approach*](#), INTERFACE boundaries separate the local (moving) reference frame from the inertial reference frame. Each INTERFACE boundary is associated with two virtual boundaries (as described above). For MRF simulations, use the flow through option for the virtual boundaries associated with each INTERFACE that delineates the MRF region. Figure 9.7 below gives an excerpt of [*boundary.in*](#) with the virtual flow through option. After each virtual boundary, specify the region ID associated with the given side of the INTERFACE.

Chapter 9: Boundaries and Boundary Conditions

INTERFACE

```
version: 3.1
---
.
.
- boundary:
    id:          12
    type:        INTERFACE
    name:        interface_bdry_example
    disconnect: 0
    forward:
        id:      36
        type:    FLOW_THROUGH
        name:   region_forward_example
        region: 1
    reverse:
        id:      37
        type:    FLOW_THROUGH
        name:   region_reverse_example
        region: 0
.
```

Figure 9.7: Configuration of a pair of flow-through INTERFACE boundaries.

To separate the local reference frame(s) from the inertial frame, you must create additional boundaries in the geometry. These boundaries must be situated in such a way that the interface between the local and inertial reference frames is smooth. This way, numerical discontinuities will not appear at the reference frame interfaces.

Typically, you create two additional boundaries as interfaces between the inertial and local reference frames. These reference frame boundaries must be a shape such that the cross product of the rotation vector and the position vector from the rotation origin to a point on the interface surface is tangential to the interface boundary at that point. For WALL boundaries within this local reference frame, specify the boundary conditions in the inertial frame (see [Multiple Reference Frame Approach](#)). This way, you can easily convert a moving geometry case to a case that employs the MRF approach.

You can use CONVERGE Studio to create the additional boundaries that delineate the local reference frame. In the example shown in Figure 9.8, the first INTERFACE boundary is situated between the inflow and pump impeller (in solid gray). The second INTERFACE is a band around the circumference of the impeller (in solid red) that separates the impeller blades from the volute chamber. In this example, the local reference frame is the region between the two INTERFACE boundaries.

Chapter 9: Boundaries and Boundary Conditions

INTERFACE

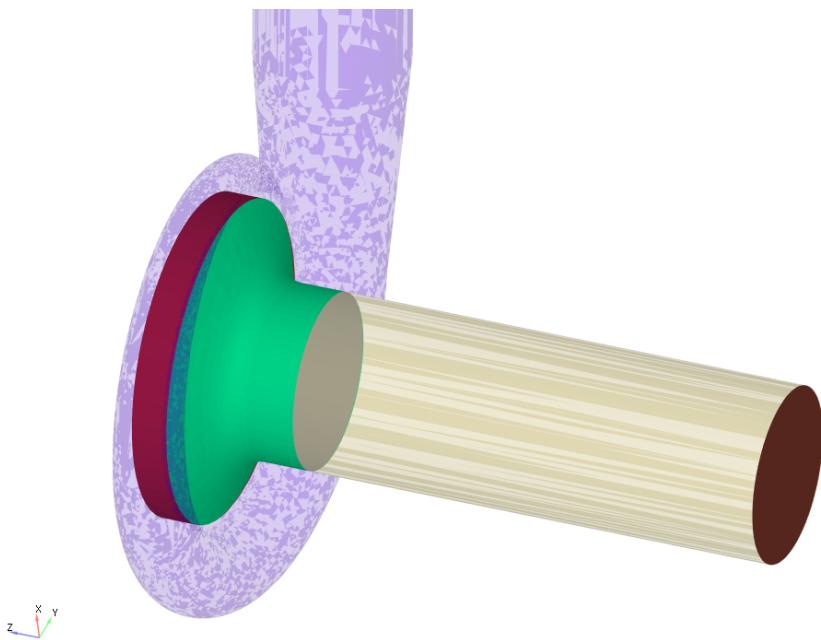


Figure 9.8: The additional INTERFACE boundaries that separate a local reference frame.

INTERFACE Boundaries for the Electric Potential Solver

When [*inputs.in*](#) > *feature_control* > *electric_potential_flag* = 1, CONVERGE solves the electric potential only for solid streams. You can use an INTERFACE boundary to separate a fluid stream from a solid stream or to separate two solids with different material properties. In [*boundary.in*](#), set *boundary_conditions* > *boundary* > *type* = INTERFACE and *boundary_conditions* > *boundary* > forward/reverse > *type* = WALL.

At a fluid-solid interface, set an [*electric potential boundary condition*](#) only on the solid side of the interface.

At a solid-solid interface, set [*electric potential boundary conditions*](#) on both sides. To solve the electric potential at the interface in a coupled manner, you can set [*boundary.in*](#) > *boundary_conditions* > *boundary* > forward/reverse > *electric* > *type* = DIRICHLET and *value* = COUPLED. When the two sides of the interface are coupled, CONVERGE imposes continuity via

$$[\Phi_b]_{\text{forward}} = [\Phi_b]_{\text{reverse}} \quad (9.83)$$

and

$$[\text{Current}]_{\text{forward}} = [\text{Current}]_{\text{reverse}}, \quad (9.84)$$

Chapter 9: Boundaries and Boundary Conditions

INTERFACE

where Φ_b is the electric potential at the boundary.

You can optionally specify an electrical contact resistance to account for irregularities and defects that prevent perfect contact between the two solid surfaces. When the contact resistance is nonzero, the current remains continuous across the interface, but Equation 9.83 is replaced by a voltage drop given by

$$\begin{aligned} [\Phi_b]_{forward} - [\Phi_b]_{reverse} &= [Current\ density]_{forward} R'' \\ &= [Current\ density]_{reverse} R'', \end{aligned} \quad (9.85)$$

where R'' is the electrical contact resistance in $\Omega\text{-m}^2$.

We recommend setting [*solver.in*](#) > *transport* > *electric* > *solver_type* = *NONLINEAR_KRYLOV* when the electric potential is coupled at a solid-solid interface.

9.7 GT-SUITE

In a [CONVERGE + GT-SUITE co-simulation](#), the GT-SUITE boundary type identifies boundaries at which flow coupling or [FSI coupling](#) occurs. For these boundaries, set [boundary.in](#) > *boundary_conditions* > *boundary* > *type* = *GT-SUITE* and configure the subsequent parameters in the *boundary_conditions* > *boundary* settings block.

Do not use this boundary type for [GT-SUITE CHT coupling](#).

9.8 MIXING_PLANE

You can use mixing planes to strategically remove parts of the domain in which the spatial variation of flow variables has a negligible effect on the results of interest. When you replace these parts of the domain with a mixing plane, CONVERGE averages flow variables on each side of the mixing plane and passes them to the other side, removing all spatial variation. For some geometries, the use of mixing planes can greatly shorten runtimes by reducing the total number of cells in the simulation.

Mixing planes are especially useful for turbomachinery applications such as the turbocharger compressor shown below in Figure 9.9. In this geometry, the compressor wheel (red) exhibits six-fold symmetry about the z axis, while the surrounding volute (purple) is not symmetric about the z axis.

Chapter 9: Boundaries and Boundary Conditions

MIXING_PLANE

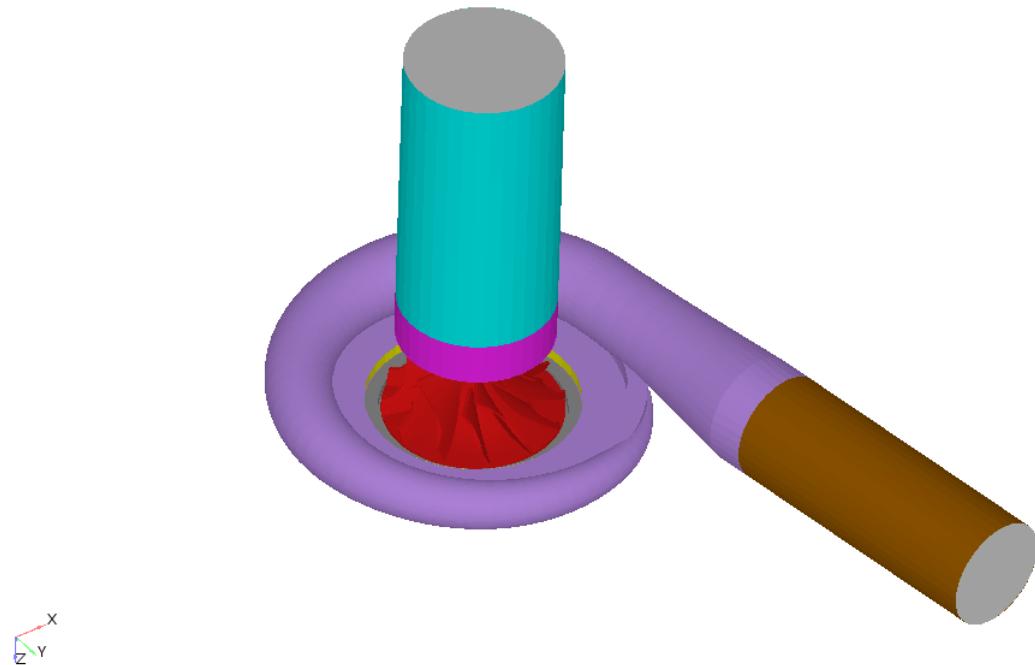


Figure 9.9: Example turbocharger compressor geometry.

With a mixing plane, you can combine a sector of the axisymmetric region with the full non-axisymmetric region, as shown below in Figure 9.10, to simulate the flow through the entire geometry with a reduced cell count.

The axisymmetric and non-axisymmetric regions are coupled through the mixing plane, which is associated with one or more boundaries. At each time-step, CONVERGE averages flow variables in the cells adjacent to side A of the mixing plane and passes them to side B, and vice versa. In the example shown below in Figure 9.11, side A is in the volute region and side B is in the compressor's multiple reference frame (MRF) region.

Chapter 9: Boundaries and Boundary Conditions

MIXING_PLANE

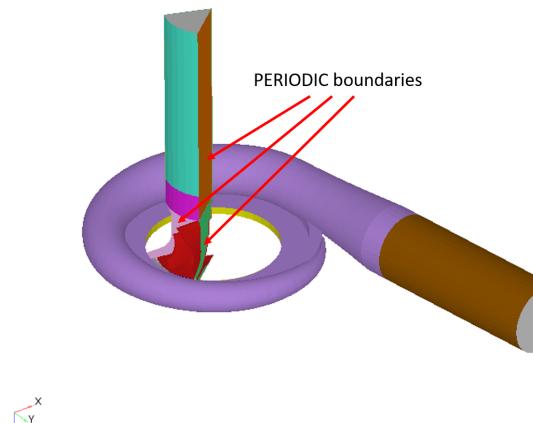


Figure 9.10: Modified turbocharger compressor geometry with a 60 degree sector of the compressor wheel, shaft, and related

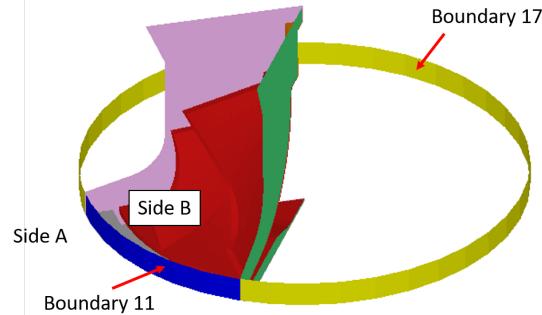


Figure 9.11: Boundaries associated with the mixing plane (boundaries 11 and 17).

You can optionally specify a target mass flow rate for a mixing plane. In this scenario, CONVERGE imposes a velocity boundary condition at the mixing plane instead of averaging the velocities. The face velocity in a cell adjacent to the mixing plane is given by

$$u = \frac{\dot{m}_{target}}{\rho A}, \quad (9.86)$$

where \dot{m}_{target} is the target mass flow rate in kg/s , ρ is the averaged value of the density on the current side of the mixing plane, and A is the area of the cell face. The direction of the velocity is user-specified.

The boundaries associated with a mixing plane must have a type of MIXING_PLANE (either in [boundary.in > boundary > type](#) or, for INTERFACE boundaries, in [boundary.in > boundary > forward/reverse > type](#)). In the example shown above in Figure 9.11, boundary 17 is a MIXING_PLANE boundary and boundary 11 is an INTERFACE boundary with a type of MIXING_PLANE for the forward and reverse sides. (Note that in the original geometry, these two boundaries were combined to form a single flow-through INTERFACE boundary. Now the MRF region has been reduced to a sector, so only the boundary along the length of the sector is marked as an INTERFACE.)

Figure 9.12 below shows results of simulations with and without a mixing plane. In the image on the left, the averaging at the mixing plane generates smoother velocity contours in the volute region.

Chapter 9: Boundaries and Boundary Conditions

MIXING_PLANE

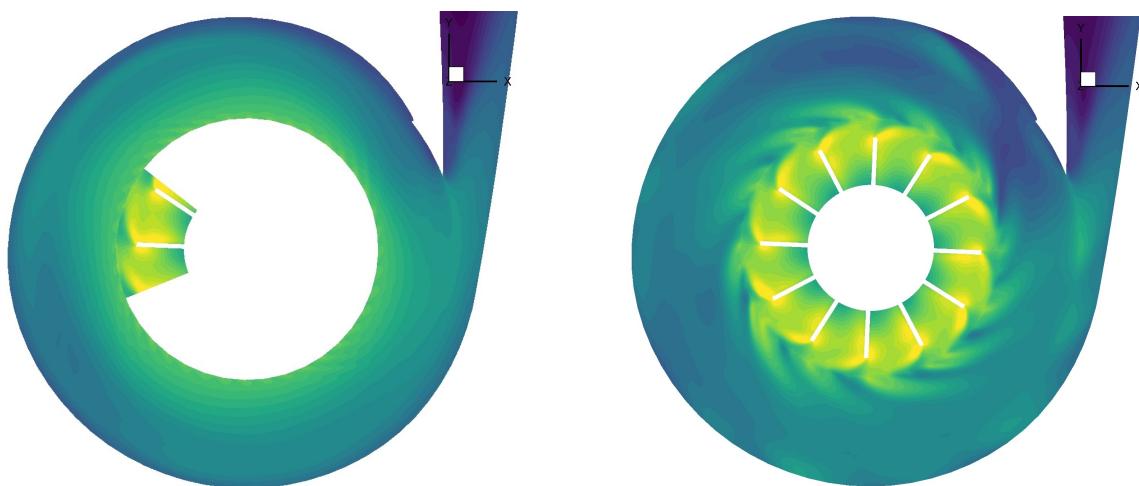


Figure 9.12: Turbocharger compressor results with mixing plane (left) and without mixing plane (right). Contours are colored by velocity magnitude.

To use this feature, you must manipulate your geometry to remove the part of the domain that will not be solved (in turbomachinery applications, this means creating a sector in the axisymmetric region). You might need to redefine some boundaries as in the example above. Then, set [*inputs.in*](#) > *feature_control* > *mixing_plane_flag* = 1 and include a [*mixing_plane.in*](#) file in your case setup. In this file, define an ID for each mixing plane in the geometry. In [*boundary.in*](#), assign a mixing plane ID (as defined in [*mixing_plane.in*](#)) and a mixing plane side to each MIXING_PLANE boundary. Cells on the same side of the mixing plane are averaged together.

In our example above, side A of the mixing plane is assigned to boundary 17 and to the forward (volute) side of boundary 11. Side B of the mixing plane is assigned to the reverse (MRF) side of boundary 11. Figure 9.13 below shows the relevant excerpt from [*boundary.in*](#).

Chapter 9: Boundaries and Boundary Conditions

MIXING_PLANE

```
- boundary:
  id:          11
  type:        INTERFACE
  name:        Interface2
  disconnect: 0
  forward:
    id:          29
    type:        MIXING_PLANE
    name:        Forward side of INTERFACE Interface2 (ID = 11)
    region:     3
    mixing_plane:
      id:          0
      side:        Side_A
  reverse:
    id:          30
    type:        MIXING_PLANE
    name:        Reverse side of INTERFACE Interface2 (ID = 11)
    region:     0
    mixing_plane:
      id:          0
      side:        Side_B
- boundary:
  id:          17
  type:        MIXING_PLANE
  name:        Interface3
  region:     3
  mixing_plane:
    id:          0
    side:        Side_A
```

Figure 9.13: Excerpt of *boundary.in* for the turbocharger example. Region 3 corresponds to the volute and region 0 corresponds to the MRF region.

Chapter



10

Regions and Streams

10 Regions and Streams

In CONVERGE, a [boundary](#) is a collection of surface triangles. A region is a collection of one or more boundaries. A stream is a collection of one or more regions.

This chapter describes [regions](#) and [streams](#). Both regions and streams give you flexibility in customizing your simulation.

10.1 Regions

A region is a collection of one or more boundaries. We recommend using CONVERGE Studio to assign surface triangles to boundaries and boundaries to regions.

CONVERGE uses regions to [initialize variables](#) (temperature, pressure, turbulent kinetic energy, turbulent dissipation, species, and [passives](#)), [control the flow between portions of the geometry](#), and [report simulation results](#).

Note that the concept of regions in CONVERGE is slightly different than in other CFD solvers. In other CFD codes, you must manually create a grid and assign each cell in the grid to a region before running a simulation. When you create a grid in other CFD codes, you must shape the cells so that the cell faces align perfectly with the region boundaries. The region assignment process in CONVERGE is much simpler because CONVERGE automatically creates the grid (and controls the location and orientation of each cell in the grid) at runtime.

It is important to remember that CONVERGE requires each boundary to be assigned to a single region. If you have a portion of the geometry with one set of boundary conditions but you wish to assign half of that geometry to one region and half to another region, you must create two boundaries (with identical boundary conditions). Then you can assign one boundary to each region. Based on the region assignment of each boundary, CONVERGE determines where in the interior of the domain to draw dividing lines for each region. You can assign a boundary to a particular region via [boundary.in > boundary_conditions > boundary > region](#).

Dependent Regions

Typically each boundary is assigned to a single region. In some cases, however, the region assignment of a boundary needs to vary throughout the simulation. CONVERGE's dependent option can dynamically change the region with which a boundary is associated. This option is most commonly used for simulations that involve [sealing](#) (such as two-stroke or Wankel engines), but it may be useful for other cases as well (such as various compressor or pump simulations or modeling blowby within an opposed-piston engine).

To direct CONVERGE to dynamically vary the region associated with a boundary, set [boundary.in > boundary_conditions > boundary > region = DEPENDENT](#).

Solid Regions

Solid regions must be grouped into one or more solid streams in [*stream_settings.in*](#). All solid streams must have [*stream_settings.in*](#) > *stream_list* > *stream* > *solid_flag* = 1.

If your simulation has one or more solid streams, you must define one or more solid species in [*species.in*](#).

CONVERGE looks in the [*solid.dat*](#) file for properties of any species specified as a solid in [*species.in*](#). The [*solid.dat*](#) file includes data for melting point, density, specific heat capacity, and conductivity, all in SI units.

Also note that only the Dirichlet (*di*) boundary condition is allowed for a WALL boundary that is associated with a solid stream ([*stream_settings.in*](#) > *stream_list* > *stream* > *solid_flag* = 1).

Region Connection and Disconnection

At no time during a simulation are the surface triangles allowed to intersect each other. For example, in an engine, there must always be a minimum lift between a valve and the corresponding valve seat. Because of this restriction, you cannot cut off the flow between regions by having one surface come in contact with another. Instead CONVERGE creates disconnect triangles to close the gaps (and thus cut off the flow) between regions. This section describes the different types of disconnect triangles that CONVERGE creates. The [following section](#) describes how to activate or deactivate disconnect triangles to control the flow between regions.

Consider a shock tube as an example. The following two figures show the geometry, which is separated into two regions: the red region has a low initial pressure and the blue region has a high initial pressure. (The boundaries for this geometry are not shown, but in this case the red region consists of one or more boundaries and the blue region consists of one or more [different] boundaries.)

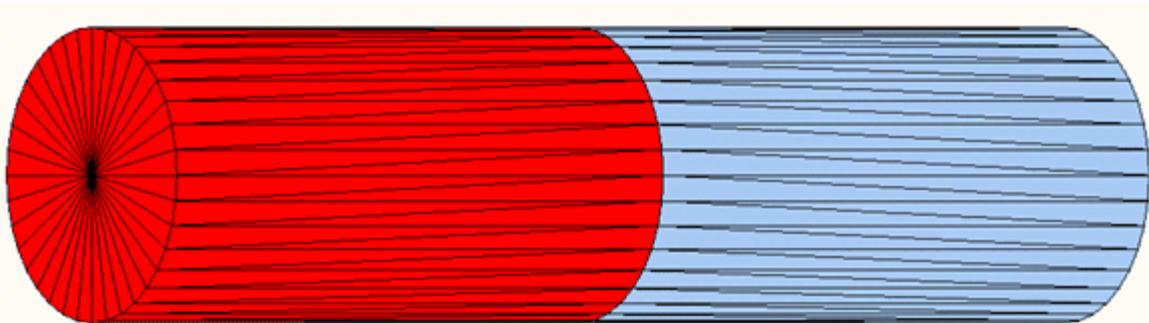


Figure 10.1: Shock tube geometry. The red region has a low initial pressure. The blue region has a high initial pressure.

Chapter 10: Regions and Streams

Regions Region Connection and Disconnection

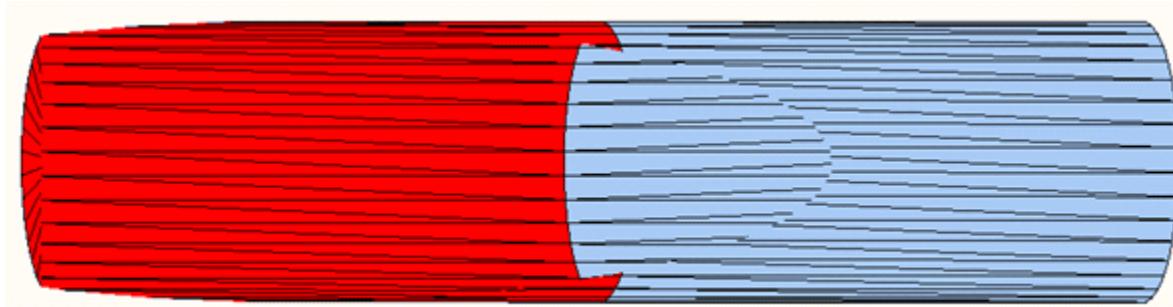


Figure 10.2: Clip plane of the shock tube. From this view you can see that there are no triangles between the two regions in the triangulated surface.

At the start of a simulation, CONVERGE identifies continuous loops of edges that form boundaries between regions. For example, to delineate the red and blue regions CONVERGE identifies the orange loop shown below in Figure 10.3.

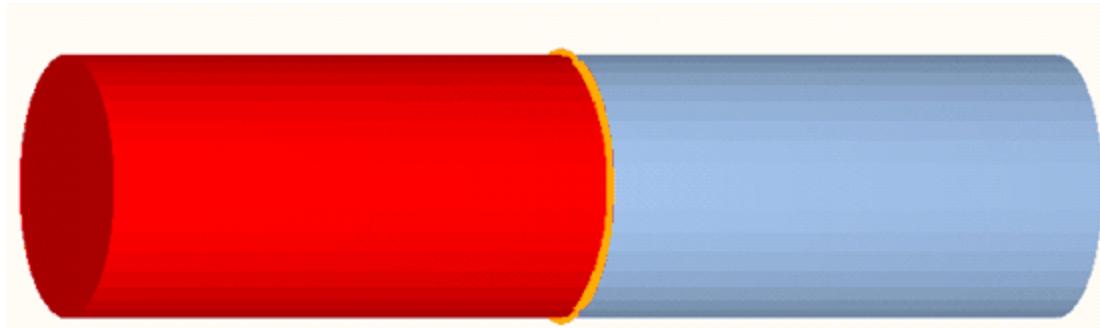


Figure 10.3: The edges highlighted in orange mark the border between the red and blue regions.

At the beginning of the simulation, CONVERGE will activate disconnect triangles to patch the orange loop, as shown below in Figure 10.4. When active, these disconnect triangles prohibit flow between the two regions. Throughout the simulation CONVERGE will activate or deactivate the disconnect triangles to control the flow between these two regions. Note that the disconnect triangles are strictly for numerical purposes and they are not actually a part of the surface geometry.

Chapter 10: Regions and Streams

Regions Region Connection and Disconnection

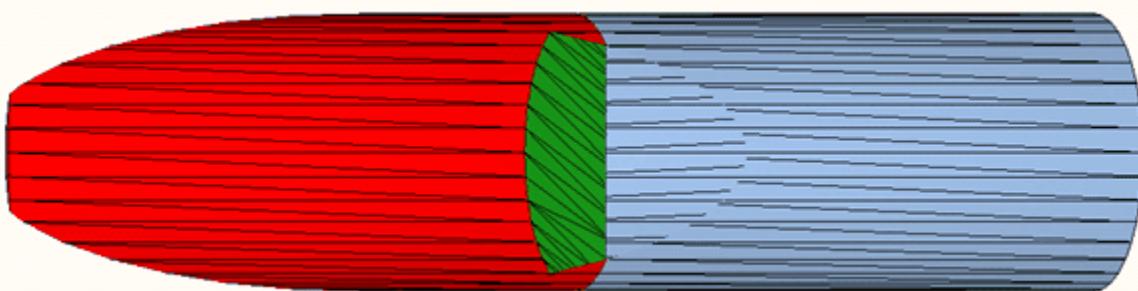


Figure 10.4: Clip plane of the shock tube and green disconnect triangles. Note that these triangles are not part of the surface geometry and will not appear in CONVERGE Studio.

In this shock tube example, CONVERGE creates disconnect triangles within the circular loop that delineates the two regions. CONVERGE can properly triangulate any planar loop, even non-convex or ring-shaped loops. CONVERGE can triangulate most non-planar loops, but success is not guaranteed.

This example illustrates an important point: the surface must be triangulated such that each boundary (collection of triangles) can be assigned to a single region. In the case of the shock tube, the same geometry could have been represented as shown below in Figure 10.5. However, it would be impossible to flag separate boundaries, and thus separate regions, for the left and right sides of the cylinder.

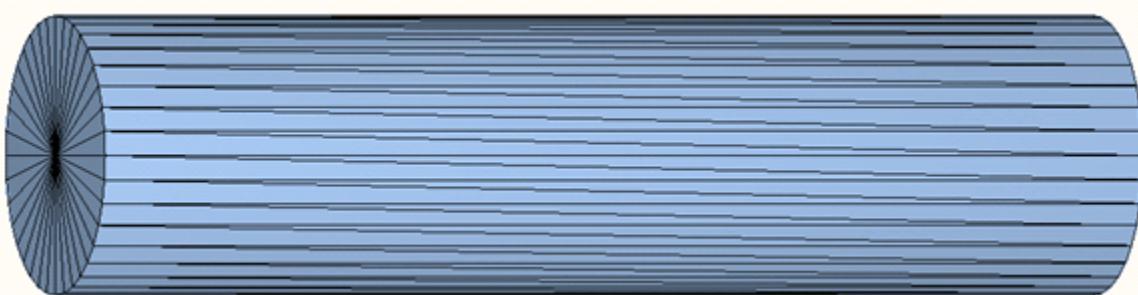


Figure 10.5: The same shock tube geometry but with a surface triangulation that does not allow for separation into two boundaries (and thus two regions).

The type of disconnection illustrated with the shock tube is called a single loop disconnection. A second common type of disconnection is a concentric circle disconnection. For example, in the poppet valves in an engine, there is a concentric circle disconnection between the valve (a circular loop) and the valve seat (another circular loop). Figure 10.6 below shows a cross section of a valve and a valve seat in an engine. In this example, the blue lines represent boundaries assigned to region 1, while the red lines represent boundaries assigned to region 2.

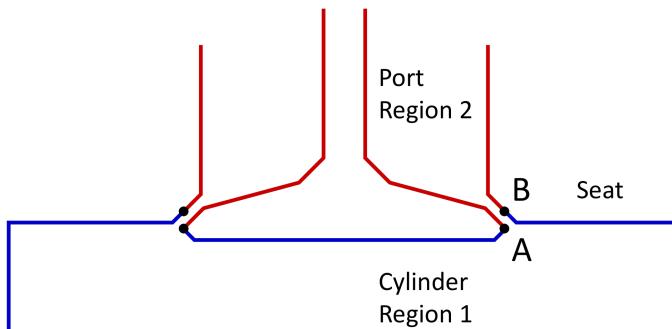


Figure 10.6: Two-dimensional cross section of a valve and a valve seat.

At the start of the simulation, CONVERGE will detect two circular loops where regions 1 and 2 meet: one loop around the base of the valve (point A in Figure 10.6) and one loop around the base of the valve seat (point B). Because there are two loops, CONVERGE will create a ring of disconnect triangles between the two loops (*i.e.*, between the valve and the valve seat). You can activate or deactivate these disconnect triangles to control the flow between regions. Note that in this valve example, the top and bottom of the valve must be assigned to different boundaries (even if identical boundary conditions will be assigned to the top and bottom of the valve) so that they can be assigned to different regions.

CONVERGE determines whether an edge loop is single or paired, and to which other edge loop it could be paired, based on its proximity to any other disconnect edge loop (*i.e.*, [events.in > disconnect_concentric_dist](#) and [events.in > disconnect_concentric_tol](#)). CONVERGE calculates the location of the centroid of each disconnect edge loop, as well as its mean radius. If two edge loops have their centers closer than *disconnect_concentric_dist*, and radii closer than *disconnect_concentric_tol*, CONVERGE will treat the pair of edge loops as a concentric disconnection.

You must specify appropriate values of *disconnect_concentric_dist* and *disconnect_concentric_tol* in order for CONVERGE to pair disconnect loops correctly. Excessively small values will cause CONVERGE to not detect the pairing, and excessively large values can cause incorrect pairing. Consider the three-region annular duct shown in Figure 10.7.

Chapter 10: Regions and Streams

Regions Region Connection and Disconnection

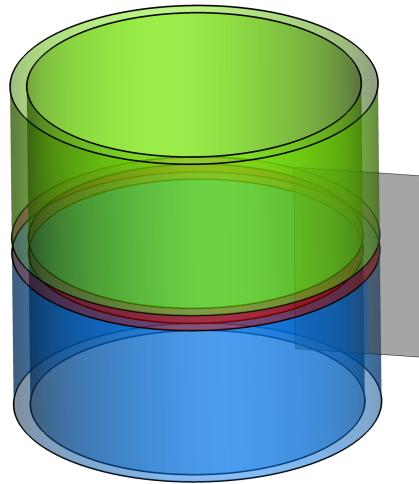


Figure 10.7: Three-region annular duct with cut-plane. Green, red, and blue volumes are each separate boundaries defining separate regions.

This duct has three regions separated by four disconnect edge loops. The gray cut-plane is shown below in 10.8.

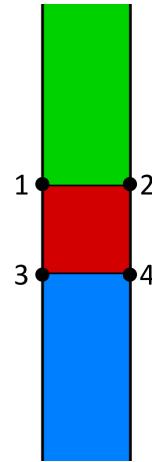


Figure 10.8: Edge loops on the cut-plane.

In this example, edge loops 1 and 2 should form one concentric disconnection, and edge loops 3 and 4 should form a second concentric disconnection. If you specify excessively large *disconnect_concentric_dist* and *disconnect_concentric_tol*, CONVERGE will pair the first two pairs of edge loops found in the surface file, then the next two pairs. This could result in edge loops 1 and 4 being paired, and 2 and 3 being paired. This will cause CONVERGE to crash.

Events

CONVERGE uses events for a variety of purposes.

Close and open events activate or deactivate, respectively, [disconnect triangles](#), which control when and where flow between regions is allowed. An open event deactivates the disconnect triangles and thus allow flow between the specified regions. A close event activates the disconnect triangles and thus prohibits flow between the specified regions. When disconnect triangles are activated, they act as a [SYMMETRY boundary condition](#).

If a case includes at least two adjacent fluid regions, you must set [*inputs.in* > grid_control > events_flag = 1](#) and include an [*events.in*](#) file in your case setup. The [*events.in*](#) file must contain at least one event for each pair of adjacent regions. If there is a time period for which [*events.in*](#) does not contain at least one event for a pair of adjacent fluid regions, CONVERGE prohibits flow between those regions.

Initialization of Events

If [*events.in* > event > temporal_type = SEQUENTIAL](#), you must specify an event at the simulation start time ([*inputs.in* > simulation_control > start_time](#)). If [*events.in* > event > temporal_type = CYCLIC](#), at the beginning of the simulation CONVERGE disconnects all adjacent regions and then processes the events for one full cycle and then up until the simulation start time. This action is necessary because, in a cyclic case, the initial region connection status depends on the region connection status at the end of the previous cycle.

CONVERGE does not alter the time-step to make an event coincide with the start of a time-step. Instead, at the end of each time-step, CONVERGE executes any events that occurred during that time-step (*i.e.*, the actual event will occur at the end of the time-step in which that event is specified to occur). Following an event, the time-step will be reduced by a factor of ten to more accurately simulate the shock introduced in the domain by closing off or connecting regions.

Automatic Valve Events

For engine cases involving valves, CONVERGE can automatically create open and close events based on the valve lift profile specified in [*boundary.in*](#). To implement this option, set [*events.in* > events > event > type = VALVE](#) and set up a valve lift profile to specify the motion of the valve.

Contact Resistance Events

For contact resistance modeling, set [*events.in* > events > event > contact_resistance_flag = 1](#). For contact resistance open and close events, CONVERGE allows or prohibits flow through the contact resistance region.

Moving Triangles

It is important to note that disconnect triangles cannot be activated when the triangles to which the disconnect triangles are attached are moving. For example, in the case of a valve and valve seat, the valve must not be moving when the regions above and below the valve

are disconnected. CONVERGE checks all events at the start of the simulation to ensure that no disconnect triangles are activated (*i.e.*, no CLOSE events are scheduled) while any portions of the relevant regions are moving.

Boundary Events: Flow-through INTERFACE

For cases with an [inlaid mesh](#), you can use events to control the flow-through INTERFACE boundaries that delineate the inlaid grid. In this way, you can begin the simulation with a Cartesian mesh to rapidly obtain an initial solution. At the time(s) specified in [events.in](#), CONVERGE activates the flow-through INTERFACE boundaries delineating the inlaid grid, interpolates the solution onto the inlaid grid, and proceeds with the simulation. In the *event* settings block (after the *boundary_id* parameter), supply the boundary ID of the flow-through INTERFACE boundary to be activated or deactivated. To activate or deactivate the flow-through INTERFACE boundary, set [events.in > events > event > type = CLOSE](#) or [OPEN](#), respectively.

Synchronizing Valve Motion with Open and Close Events

If desired, you can manually specify open and close events for valves instead of using [automatic valve events](#). If you specify the open and close events manually, be sure to prescribe events that correspond with the valve motion, as described below. Automatic valve events use the same logic as described below.

The valve must not be moving when regions are disconnected, so specify an open event before the minimum lift (L_{min}) has been reached when the valve is opening. Similarly, specify a close event after the valve reaches a lift distance lower than the minimum lift (L_{min}), when the valve is closing. The timing of the open and close events should be such that the real flow area is reasonably approximated, as shown below in Figure 10.9. This approach assumes the lift profile to be nearly linear at the opening and closing ends of the valve lift profile.

Chapter 10: Regions and Streams

Regions Events

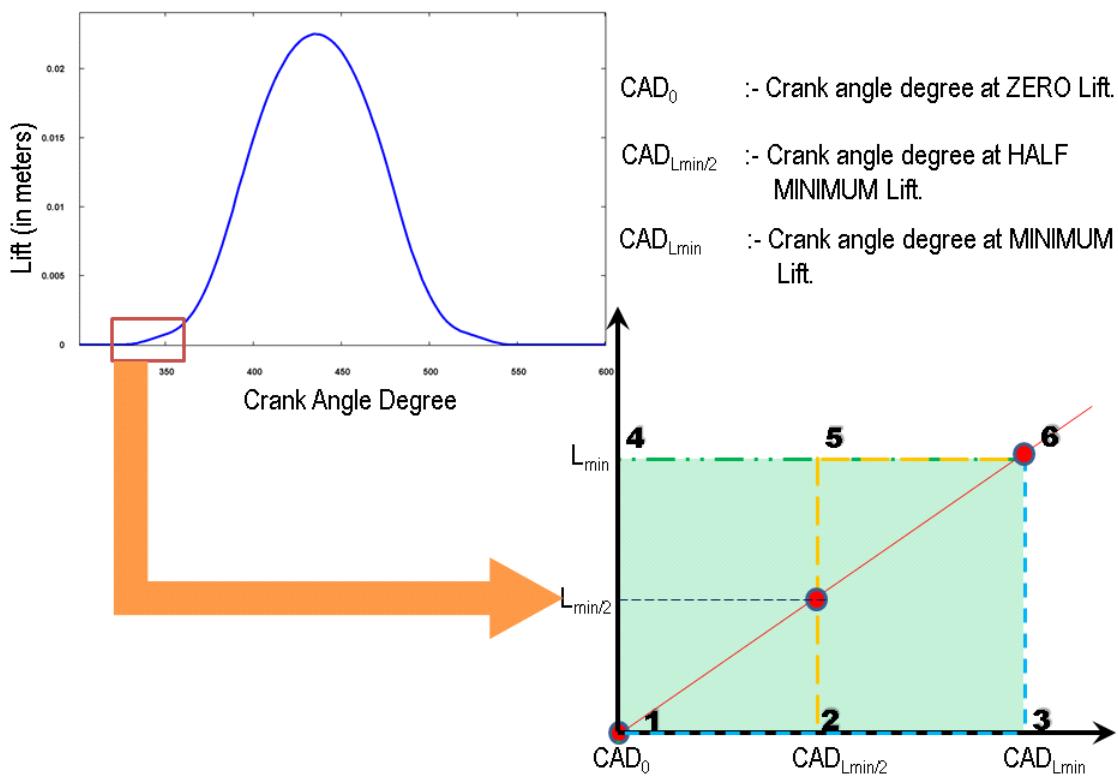


Figure 10.9: Different flow areas will be evaluated for different choice of crank angle degree for open event in events.in.

The crank angle when the valve lift is half the minimum lift is recommended for the open event timing (point 2 in Figure 10.9). This process will capture the flow area between the points 2-3-6-5-2 which approximates the real mass flow in magnitude, (*i.e.*, the area enclosed by the points 1-6-3-2-1).

If the crank angle where the valve actually starts moving (point 1 in Figure 10.9) is chosen as the open event time, the flow area will be too large (area enclosed by the points 1-2-3-6-5-4-1), since the valve is at a non-zero minimum lift. This area is larger than the real flow area, which is the area enclosed by the points 1-2-3-6-1 and for this reason this approach is not recommended.

If the crank angle where the actual valve lift reaches the minimum lift (point 3 in Figure 10.9) is chosen as the event time and all the flow area between the seated position (1) and minimum lift (3) is neglected (area enclosed by the points 1-2-3-6-1) therefore this approach is also not recommended.

Similarly, choose the close event timing to correspond with the time at which the valve lift is approximately half of the minimum lift distance.

10.2 Streams

In CONVERGE, each region is assigned to a stream. A stream is a set of regions that share the same computational grid and the same material properties (*i.e.*, fluid or solid). Connected streams are coupled to each other through INTERFACE boundaries, as shown below in Figure 10.10. The type of INTERFACE boundary varies depending on the application.

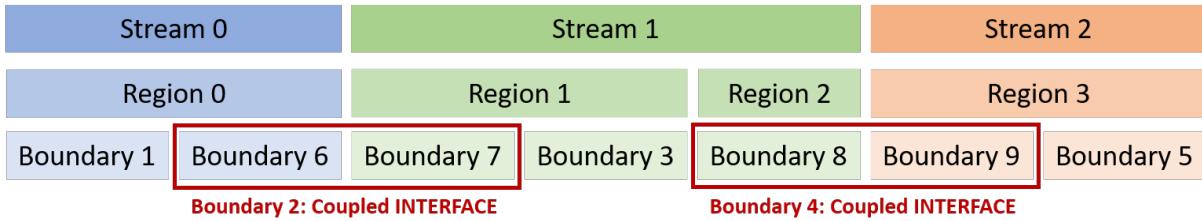


Figure 10.10: Hierarchy of streams, regions, and boundaries in CONVERGE.

Many simulations have only one stream. Examples of simulations that require multiple streams include [fixed flow](#) simulations and [conjugate heat transfer \(CHT\)](#) simulations. You might also define multiple streams if you want to use different settings, such as different solver types or reaction mechanisms, in different streams.

You must specify the number of streams and their characteristics in the [stream_settings.in](#) file. In this file, you assign a stream ID to each stream, specify the regions that belong to the stream, and configure other stream-related settings. This file is required in the Case Directory for all simulations, even if there is only one stream.

Additional setup options are available for multi-stream simulations. This chapter includes setup instructions for [stream-based overrides](#) and [connected fluid streams](#). The [fixed flow method](#) and [conjugate heat transfer](#) are described elsewhere.

Stream-Based Overrides for Multi-Stream Simulations

Some applications require different settings for different streams. For example, in a multi-cylinder engine simulation, you might want to use a detailed reaction mechanism for the primary cylinder (stream 0) and use a less accurate mechanism for the other cylinders (stream 1). This would require different [reaction mechanism files](#) for the two streams. Another example is a combined engine and aftertreatment system simulation that uses the transient solver for the engine (stream 0) and the steady-state solver for the aftertreatment system (stream 1). This would require different settings for the two streams in several input files, including [inputs.in](#) and [solver.in](#).

You can use stream-based overrides to set up cases like these, as shown below in Figure 10.11. Put the input and data files containing settings that apply to all streams in the Case Directory. We refer to these settings as the shared settings. To override the shared settings for a given stream, create a subdirectory for that stream in the Case Directory and enter the name of the subdirectory (*e.g.*, stream0) in [stream_settings.in](#) > [stream_list](#) > [stream](#) > [stream_overwrite](#). Add any input files containing stream-specific settings to the specified subdirectory.

Chapter 10: Regions and Streams

Streams Stream-Based Overrides for Multi-Stream Simulations

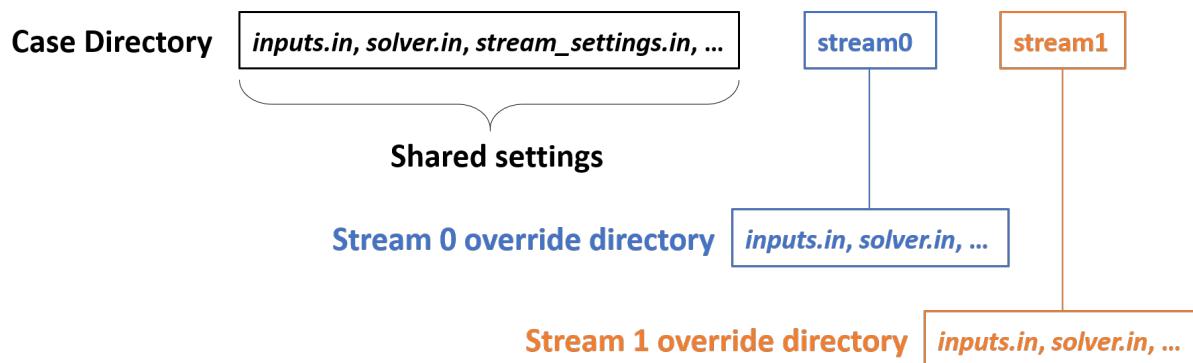


Figure 10.11: Example input directory structure for a multi-stream simulation with stream-based overrides.

When the `stream_overwrite` parameter is populated, CONVERGE reads the input files in the specified subdirectory when solving that stream. If a given file has the same name as a file in the Case Directory, CONVERGE looks for parameter values or lists that are different between the two files and overwrites the shared settings with the stream-specific parameter values and lists. Examples of lists that can be overwritten on a stream-by-stream basis include the lists of species in [`species.in`](#), the embeddings in [`embedded.in`](#), and the sources in [`source.in`](#).

The following figure shows an example of stream-based overrides in [`inputs.in`](#) and [`species.in`](#), and the subsequent table shows the parameter values and lists that CONVERGE uses for each stream.

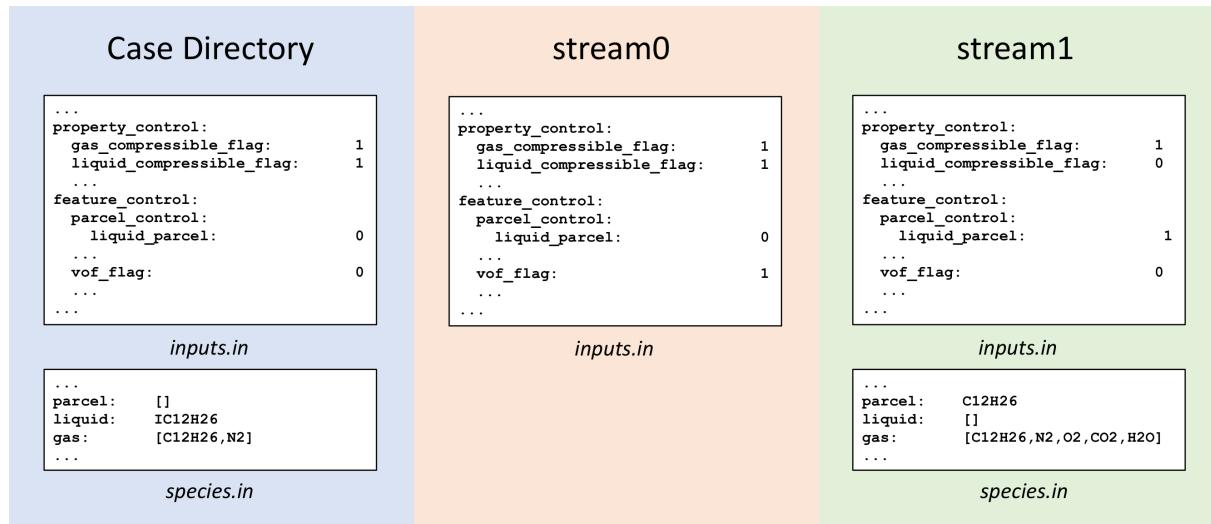


Figure 10.12: Example of input files containing stream-based overrides.

Table 10.1: Parameter values and lists used in each stream for the example in Figure 10.11.

Parameter	Value/list in stream 0	Value/list in stream 1
<i>inputs.in > property_control > gas_compressible_flag</i>	1	1
<i>inputs.in > property_control > liquid_compressible_flag</i>	1	0
<i>inputs.in > feature_control > parcel_mode > liquid_parcel</i>	0	1
<i>inputs.in > feature_control > vof_flag</i>	1	0
<i>species.in > parcel</i>	[]	C12H26
<i>species.in > liquid</i>	IC12H26	[]
<i>species.in > gas</i>	[C12H26, N2]	[C12H26, N2, O2, CO2, H2O]

The override directories need not contain a full set of input files. As a general rule, they should contain only the files that are unnecessary in the Case Directory (because they apply only to one stream) or the files in which some parameter values or lists are different from the shared settings.

Note that the [surface geometry file](#) (e.g., *surface.dat*), [boundary.in](#), and [initialize.in](#) cannot be overridden. CONVERGE ignores these files if they are present in an override directory.

Setup for Connected Fluid Streams

The boundary between two connected fluid streams must be marked as a flow-through INTERFACE boundary, as shown below in Figure 10.13. The flow-through INterface is required for CONVERGE to pass flow variables between the streams and does not obstruct the flow in any way. If you do not configure a flow-through INterface for two neighboring fluid streams, CONVERGE will solve the two streams in a completely uncoupled manner.

Chapter 10: Regions and Streams

Streams Setup for Connected Fluid Streams

```
- boundary:  
  id:          3  
  type:        INTERFACE  
  name:        stream_bound  
  motion:      STATIONARY  
  geometry_motion: FIXED  
  disconnect:  0  
  forward:  
    id:          15  
    type:        FLOW_THROUGH  
    name:        Forward side of stream_bound  
    region:     0  
  reverse:  
    id:          16  
    type:        FLOW_THROUGH  
    name:        Reverse side of stream_bound  
    region:     1
```

Figure 10.13: An excerpt from `boundary.in` showing the setup for a flow-through INterface between two streams. In this example, region 0 and region 1 belong to different streams.

CONVERGE automatically assigns streams to groups to make internal calculations more efficient. A connectivity group defines a group of connected streams (e.g., streams 0-2 in Figure 10.10). Each connectivity group contains one or more transport groups. Streams in the same transport group are solved together in the [PISO](#) or [SIMPLE](#) loop.

Because steady-state (`inputs.in > solver_control > steady_solver = 1`) fluid streams typically have a unique time-step and solver setup, CONVERGE assigns each steady-state stream to a separate transport group (also called a stream group). By contrast, CONVERGE assigns connected transient (`steady_solver = 0`) streams to the same transport group. Thus, connected transient streams must have the same time-step and the same solver setup (i.e., the same values of `ns_solver_scheme` and `ns_solver_type` in [solver.in](#)).

If a transient fluid stream is connected to a steady-state fluid stream, CONVERGE stores time-averaged values of flow variables in the cells on the transient side of the interface at each time-step. These time-averaged values are passed to neighboring cells on the steady-state side, which helps to smooth out any transient fluctuations that would otherwise interfere with steady-state convergence. The time-averaging interval typically starts at the beginning of the simulation and ends at the current time-step in the transient stream. However, if `inputs.in > feature_control > fixed_flow_flag = 1` for the transient stream, the time-averaging is paused during the time intervals in which fixed flow is active ([fixed_flow.in > fixed_time_control > fixed_intervals](#)).

Matching Species and Passives Across Streams

It is possible to define different sets of species or passives for two connected fluid streams. For example, if you use different chemical mechanisms in two different streams, it is possible that there will be different gas species present in each stream. Figure 10.14 below shows a simplified example in which stream 0 has four gas species (N₂, CO₂, H₂O, and CH₄) and stream 1 has three gas species (N₂, CO₂, and H₂O).

Chapter 10: Regions and Streams

Streams Setup for Connected Fluid Streams

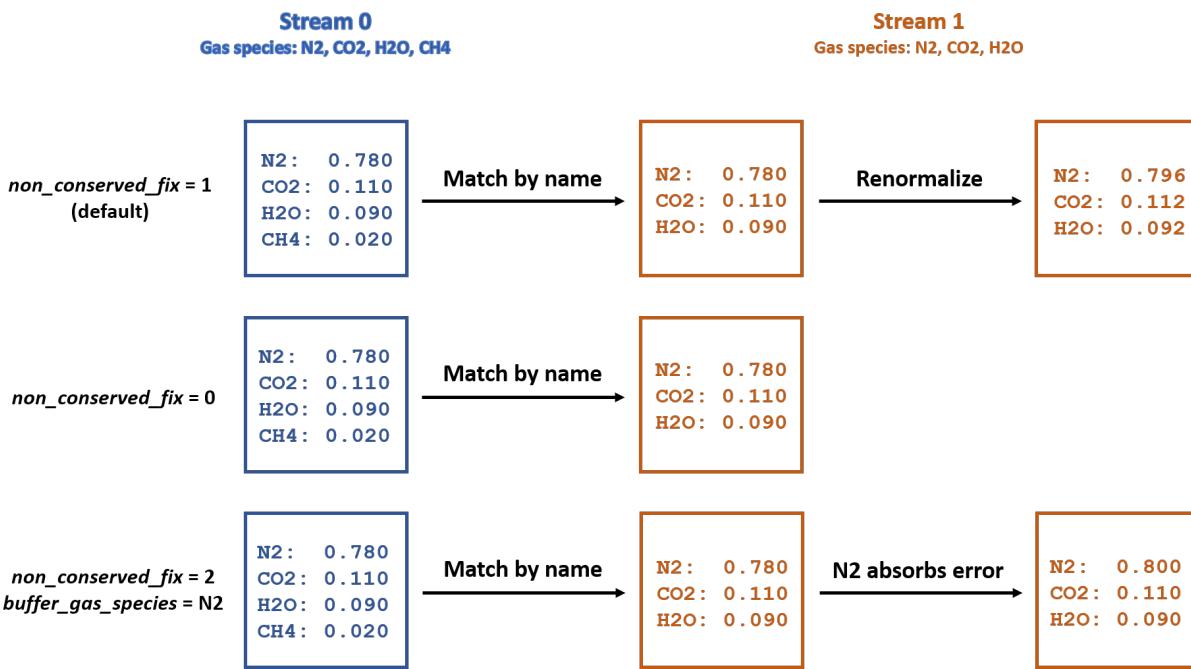


Figure 10.14: Options for matching species across streams.

The first line of Figure 10.14 illustrates the default method for transferring species information across streams. When copying species mass fractions to the new stream, CONVERGE attempts to match species by name. If no match is found in the new stream, the value for that species is not copied. Then, if the mass fractions in a given cell in the new stream do not add up to 1, CONVERGE renormalizes the mass fractions in that cell to avoid mass conservation errors.

The default behavior for [passives](#) is also to attempt to match by name. If no match is found in the new stream, CONVERGE applies a Neumann boundary condition for the passive at the stream-stream interface.

To modify the default behavior, set `stream_match_species_settings = 1` in [`stream_settings.in`](#) and include a [`stream_match_species.in`](#) file in the Case Directory. The latter file provides several options for customizing the species and passive matching behavior between pairs of connected streams. In the `gas_species_match_list`, `liquid_species_match_list`, and `passive_match_list` parameters, you can provide lists of species (or passives) to match between streams, which allows the same species (or passive) to have different names in the two streams.

In the `non_conserved_fix` parameter, you can choose an alternative method for addressing mass conservation errors, as shown in the second and third lines of Figure 10.14. With `non_conserved_fix = 0`, CONVERGE does not attempt to resolve these errors and allows the sum of species mass fractions to be non-unity. With `non_conserved_fix = 2`, you can select a buffer species to absorb the error in each stream.

Chapter 10: Regions and Streams

Streams Setup for Connected Fluid Streams

In [*stream_match_species.in*](#) > *passive_value*, you can specify passive boundary conditions to be applied at the stream-stream interface. This option is available for passives that exist in both streams as well as for passives that exist in only one stream. Only Dirichlet or Neumann boundary conditions are allowed.

Chapter



11

Grid Control

11 Grid Control

This chapter describes CONVERGE's [grid generation](#) and [cell pairing](#) procedures, which are fully automated.

This chapter describes features for controlling the grid size before and during a simulation. These features, all of which are optional, may be useful for refining and/or coarsening the grid when and where it makes sense for your simulation.

- [Grid scaling](#) coarsens or refines the base grid size.
- [Fixed embedding](#) refines the grid at specified locations and times.
- [Adaptive Mesh Refinement](#) (AMR) automatically changes the grid based on fluctuating and moving conditions.

This chapter describes CONVERGE's [inlaid mesh](#) feature, which allows you replace or supplement the Cartesian cut-cell grid generated at runtime with a prescribed grid of arbitrary shape in CONVERGE Studio (or to load a grid generated by another program). This feature is entirely optional.

11.1 Grid Generation

CONVERGE has an innovative approach to grid generation in which the grid is automatically generated at runtime. To make this automatic grid generation possible, CONVERGE uses a modified cut-cell Cartesian grid generation method. The geometry surface is immersed within a Cartesian block. CONVERGE trims the cells at the intersecting surface, after which the intersection information (surface areas, normal vectors, etc.) is reduced before being stored for each cell. This process allows for complex surface intersections to be represented more easily. The figures below shows a post-processing visualization of the cutting process and what a cut-cell looks like in CONVERGE. For this cutting process to be done properly, the [surface geometry must be properly prepared](#).

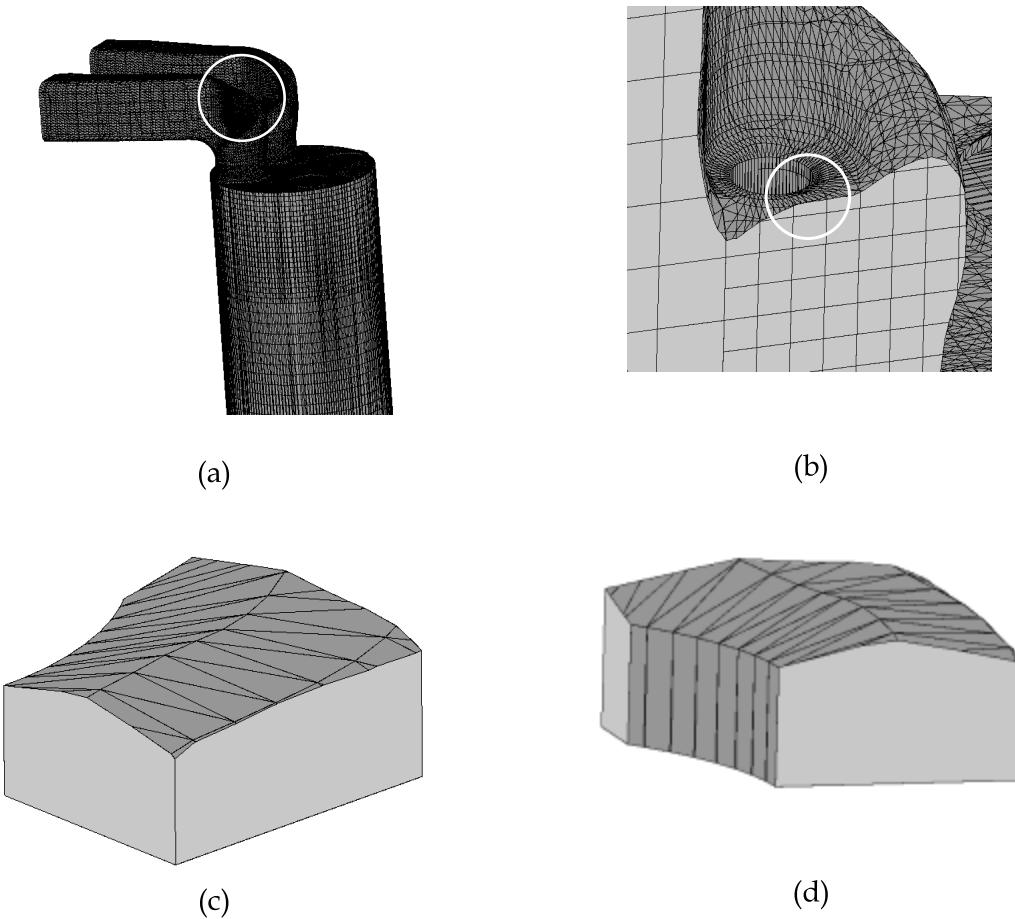


Figure 11.1: A post-processing visualization of the cut-cell method in CONVERGE using a sample cut-cell. (a) The geometry used for modeling a flowbench case. The circle indicates the general location of the sample cut-cell. (b) A cutaway view that shows the specific location of the sample cut-cell. (c) One view of the sample cut-cell. (d) Another view of the same sample cut-cell.

CONVERGE generates the Cartesian grid internally at runtime. This process involves moving the surface to the proper location (if the geometry includes moving components), trimming the boundary cells, refining any embedding areas, and then removing the refinement from the embedding. For stationary geometries, CONVERGE performs this process once at the start of the simulation and again whenever the geometry is refined or coarsened. For moving geometries, CONVERGE performs the grid generation process at each time-step. However, if the moving geometry has a solid stream with only a single rigid motion associated with all its boundaries (*i.e.*, the solid does not deform), CONVERGE will not perform grid generation at each time-step. Instead, the generated grid will move with the solid so that the solid's energy transport equation is solved on a fixed stationary grid. This reduces the numerical diffusion associated with solving the solid stream's energy transport equations on a moving mesh. The grid generation process is computationally efficient so as to minimize CPU usage for this portion of the simulation.

11.2 Cell Pairing

If CONVERGE finds a cut-cell whose volume is less than 30% of the adjacent regular cell, then the cut-cell and the regular cell are paired together to form a single node. This process is known as cell pairing. The center of the paired cell is at the volumetric center of the combined cell. The values of the transport entities (velocity, temperature, pressure) are shared by the regular cell and the cut-cell. Figure 11.2 is a schematic of cell pairing.

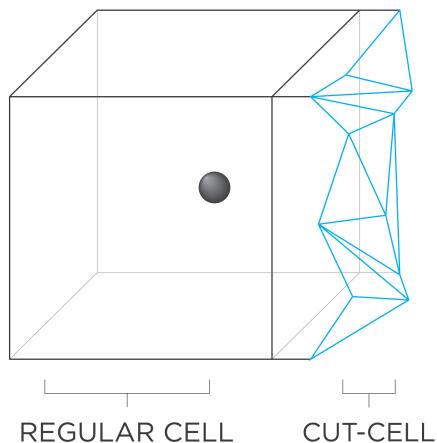


Figure 11.2: Cell pairing.

You will see the combined volume of the cell-pair, as shown below in Figure 11.3, if you include *volume* in the *cells > general settings* block of *post.in*.

Chapter 11: Grid Control

Cell Pairing

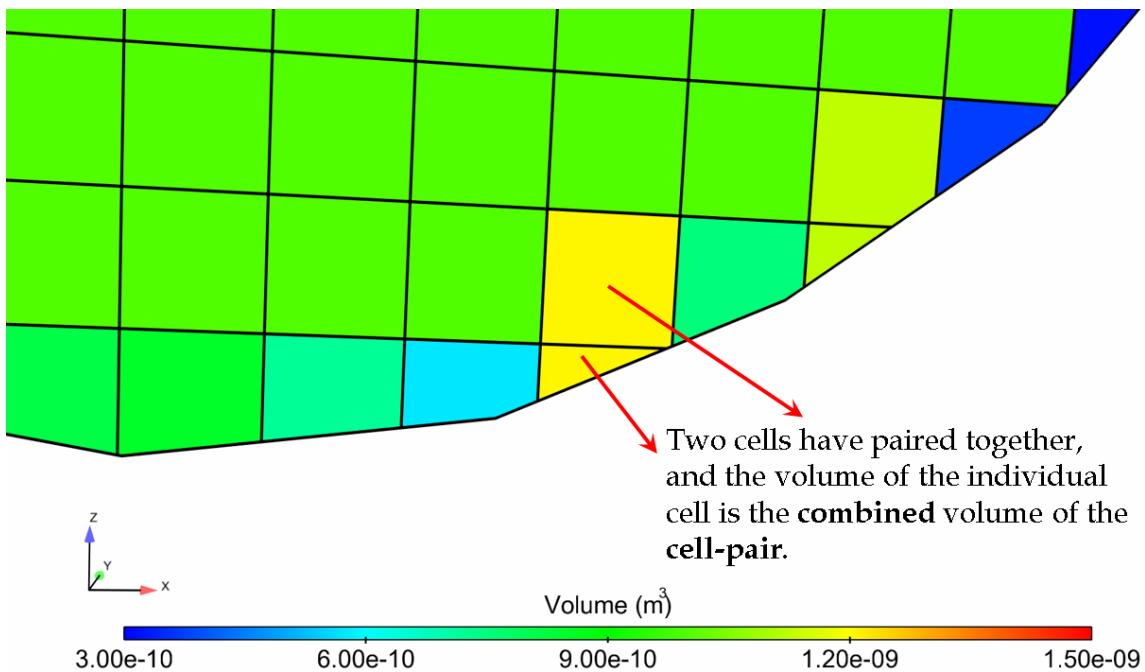


Figure 11.3: Cell pairing.

Cell pairing is an iterative process, and a set of paired cells need not be a set of two. Rather, a set might include five or ten cells all paired together. Too much cell pairing along one direction can lead to the formation of slender cells with a high aspect ratio, which occurs most frequently along the boundaries. Too many cells with a high aspect ratio can lead to numerical dispersion errors and a less robust simulation. In order to avoid this problem, you can adopt one of the following strategies:

- Choose a sufficiently small base grid size.
- Use fixed embedding in the tight areas of the geometry so that the regular cell size is comparable with the cut-cell size.

To monitor cell pairing information, add `cell_pairs` to [`post.in`](#) > `cells` > `geometry`.

11.3 Grid Scaling

Grid scaling changes the base grid size, either for the entire simulation or at specified times during a simulation. Grid scaling can reduce runtimes by coarsening the grid during non-critical simulation times and can help capture critical flow phenomena by refining the grid at other times. For example, for an in-cylinder diesel engine simulation that includes spray and combustion modeling, the grid needs a higher resolution to ensure accurate results during spray and combustion but lower grid resolution may be sufficient during compression. Grid scaling is also useful to determine the grid sensitivity of your case: Run multiple simulations with different grid scaling values and see if your results are affected by the size of the grid.

CONVERGE scales the base grid size according to

$$scaled_grid = base_grid / 2^{grid_scale}, \quad (11.1)$$

where *base_grid* is the size of the base grid in the x, y, and z directions; *grid_scale* is the scaling factor (must be an integer); and *scaled_grid* is the new grid size in all three directions. A *grid_scale* value of 0 will leave the base cells unchanged, a negative value will coarsen the base grid, and a positive value will refine the base grid.

Specify the base grid size via [*inputs.in* > grid_control > base_grid](#). Specify grid scaling via [*inputs.in* > grid_control > grid_scale](#). If you want to scale the grid multiple times throughout your simulation, you must include a file (e.g., [*gridscale.in*](#)) in your case setup.

11.4 Fixed Embedding

Use fixed embedding to refine the grid at specific locations in the domain where a finer resolution is critical to the accuracy of the solution. For example, when simulating sprays, you can add an area of fixed embedding by the nozzle to resolve the complex flow behavior. Fixed embedding allows the rest of the grid to remain coarse to minimize simulation time.

For each fixed embedding, you must specify an embedding scale that indicates how CONVERGE will refine the grid in that location. CONVERGE scales the embedded grid according to

$$embedded_grid = base_grid / 2^{embed_scale}, \quad (11.2)$$

where *base_grid* is the size of the base grid in the x, y, and z directions; *embed_scale* is the scaling factor (must be a positive integer); and *embedded_grid* is the new grid size in all three directions.

Note that CONVERGE requires two-to-one connectivity between cells, *i.e.*, a cell with an *embed_scale* of 2 can be adjacent only to a cell with an *embed_scale* of 1, 2, or 3. To maintain the required connectivity, CONVERGE provides cells with intermediate *embed_scale* values as necessary.

You can specify a specific time period for each fixed embedding, which can further reduce your computational time by refining the grid for only a portion of the simulation.

To include fixed embedding in a simulation, set [*inputs.in* > grid_control > embedded_flag = 1](#) and include an [*embedded.in*](#) file in your case setup.

The specific types of fixed embedding are described below. It is important to note that the shape embedding options (*e.g.*, sphere) may slightly affect the cell count.

Boundary Embedding

Set [*embedded.in* > embedding > embed_type = BOUND](#) to specify fixed embedding near a boundary. For example, when simulating flow around a valve, you may want extra

resolution near the valve surface to more accurately model the flow in this section of the domain (see Figure 11.4 below). For a moving surface (e.g., a valve), the embedding will move with the surface automatically. Identify the boundary by its ID, which is specified in [*boundary.in*](#) > *boundary_conditions* > *boundary* > *id*. The [*embedded.in*](#) > *embedding* > *num_embed* parameter, which must be a positive integer, specifies the number of layers of embedded cells.

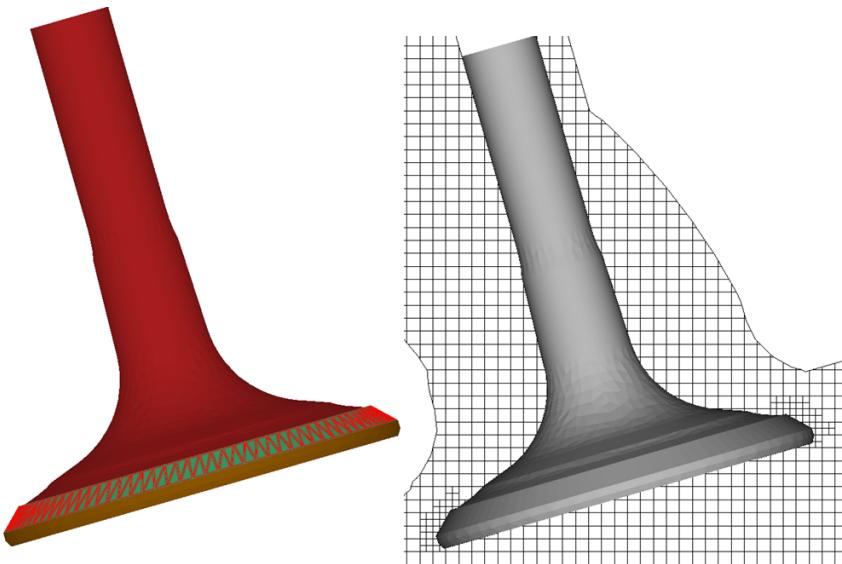


Figure 11.4: A visualization of **BOUND** embedding around a valve angle.

Sphere Embedding

Set [*embedded.in*](#) > *embedding* > *embed_type* = *SPHERE* to specify a spherical area of fixed embedding. The sphere is defined by its center and radius.

Cylinder Embedding

Set [*embedded.in*](#) > *embedding* > *embed_type* = *CYLINDER* to specify a cylindrical or truncated conical area of fixed embedding. The cylinder is defined by the center and radius of one base of the cylinder, followed by the center and radius of the other base.

Nozzle and Injector Embedding

Set [*embedded.in*](#) > *embedding* > *embed_type* = *NOZZLE* to specify a conical area of fixed embedding around a nozzle. Set [*embedded.in*](#) > *embedding* > *embed_type* = *INJECTOR* to specify a conical area of embedding around all nozzles in an injector. (If you specify a nozzle or injector embedding in [*embedded.in*](#) and if [*inputs.in*](#) > *feature_control* > *parcel_mode* > *liquid_parcel*, *solid_parcel*, and *gas_parcel* = 0, CONVERGE will give a warning message and ignore the nozzle or injector embedding.) For either nozzle or injector embedding, you must specify two radii. The first value is the radius of the circular nozzle opening, which is centered on the nozzle exit and has a normal vector pointing in the direction of the spray injection. The second value is the radius of the other circular area of the cone (*i.e.*, the wider base of the cone). You must also specify the length of the cone.

Figure 11.5 shows a post-processing visualization of the grid generated from a nozzle embedding.

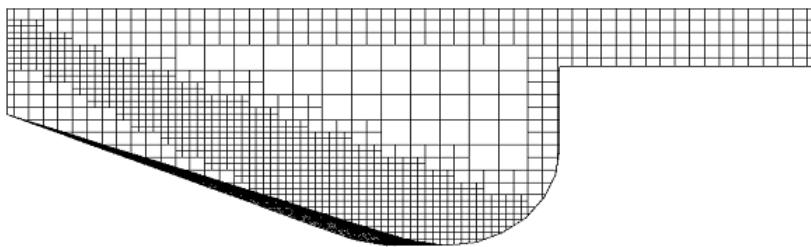


Figure 11.5: A grid generated using a nozzle embedding ([embedded.in > embedding > embed_scale = 2](#)).

Box Embedding

Set [embedded.in > embedding > embed_type = BOX](#) to specify a box of fixed embedding. The box is defined with its center and the half-length of each dimension of the box.

Region Embedding

Set [embedded.in > embedding > embed_type = REGION](#) to specify fixed embedding for an entire region. For example, you could use a region embedding to refine the grid in the cylinder of an engine. Identify the region by [embedding > region_id](#), which must match the value given in [initialize.in](#).

11.5 Adaptive Mesh Refinement

Use Adaptive Mesh Refinement (AMR) to automatically refine the grid based on fluctuating and moving conditions such as temperature or velocity. This feature may be used to obtain a refined grid in fluid or solid regions of interest allowing complex phenomena, such as flame propagation or high-velocity flow, without unnecessarily slowing the simulation with a globally refined grid.

There are two types of AMR: sub-grid scale (SGS)-based and value-based. The AMR type and criteria should be chosen such that a higher grid resolution (embedding) is added where the flow field is most under-resolved, where the sub-grid field is the largest (*i.e.*, where the curvature [gradient] of a specified field variable is the highest), or where a variable satisfies the embed criteria. The default AMR method in CONVERGE estimates the magnitude of the sub-grid field to determine where CONVERGE will add embedding. CONVERGE can also apply AMR based on specified values (*i.e.*, value-based AMR) and/or based on the number of neighboring [inlaid mesh](#) cells.

To include AMR in a simulation, set [*inputs.in*](#) > *grid_control* > *amr_flag* = 1 and include an [*amr.in*](#) file in your case setup. To specify the type of AMR, set [*amr.in*](#) > *amr_** > *amr_type* (where * is *velocity*, *temperature*, etc.).

You can perform AMR on any variable marked for AMR in the [*active variables log*](#) by specifying it as a passive in [*amr.in*](#) > *passive_list* > *passive* > *name*.

To limit the number of embedded cells, you can specify a maximum overall number of cells with [*amr.in*](#) > *amr_settings* > *max_cells*. If the number of cells in the grid reaches the maximum number (*i.e.*, too many cells satisfy the embed criterion), CONVERGE will adjust the user-specified sub-grid value criteria (SGS) or the embed criterion (value-based) as needed to make the criterion less sensitive. That is, if there are too many cells, CONVERGE will increase the value of the user-specified criteria and only the cells with the largest gradients (SGS) or the most extreme values (value-based) will be tagged for embedding.

When [*amr.in*](#) > *amr_** > *amr_type* = *EMBED_BETWEEN*, the value at the center of the range is prioritized. For this type of value-based AMR, if many values are at the center of the specified range, it may not be possible to satisfy the maximum cell count. All other cell types of AMR will therefore be rendered so insensitive that they are useless as the *EMBED_BETWEEN* definition cuts its range in half with every time-step but never quite manages to exclude the center.

You can specify AMR with a different embedding scale and different sub-grid criterion for each condition. You can even specify multiple AMR definitions for each variable simultaneously and in the same region. In addition to the field control, you can specify the time when the AMR will start and when it will end for each field, similar to fixed and boundary embedding timing control.

You can also specify a minimum number of cells with [*amr.in*](#) > *amr_settings* > *min_cells*. If the pre-AMR cell count is less than the value you specify for *min_cells*, CONVERGE will increase the sensitivity of the embed criteria in order to embed additional cells. For example, in the case of sub-grid scale, CONVERGE will lower the user-specified value of *sgs_embed* in an attempt to increase the cell count to the extent allowed by the embed scale. You can maintain a target range of cells using *max_cells* and *min_cells*.

To control the cell size, specify a maximum level of embedding for each condition to which AMR is applied via [*amr.in*](#) > *amr_group* > *amr_** > *embed_scale* (*where ** is *velocity*, *temperature*, etc.). CONVERGE will apply the maximum embed scale to the base grid *after* applying [*grid scaling*](#).

Sub-Grid Field AMR Theory

For a scalar, the sub-grid field is defined as the difference between the actual field and the resolved field or

$$\phi' = \phi - \bar{\phi}, \quad (11.3)$$

where ϕ is the actual scalar field, $\bar{\phi}$ is the resolved scalar field, and ϕ' is the sub-grid scalar field. The subgrid for any scalar can be expressed as an infinite series [[Bedford and Yeo \(1993\)](#) and [Pomraning \(2000\)](#)], as is given by

$$\begin{aligned}\phi' = & -\alpha_{[k]} \frac{\partial^2 \bar{\phi}}{\partial x_k \partial x_k} + \frac{1}{2!} \alpha_{[k]} \alpha_{[l]} \frac{\partial^4 \phi}{\partial x_k \partial x_k \partial x_l \partial x_l} \\ & - \frac{1}{3!} \alpha_{[k]} \alpha_{[l]} \alpha_{[m]} \frac{\partial^6 \bar{\phi}}{\partial x_k \partial x_l \partial x_m \partial x_l \partial x_m} + \dots\end{aligned}\tag{11.4}$$

where $\alpha_{[k]}$ is $dx_k^2/24$ for a rectangular cell and the brackets, [], indicate no summation. Since it is not possible to evaluate the entire series, only the first term (the second-order term) in the series is used to approximate the scale of the subgrid,

$$\phi' \approx -\alpha_{[k]} \frac{\partial^2 \bar{\phi}}{\partial x_k \partial x_k}. \tag{11.5}$$

Note that the above equations can be easily generalized for a vector field, such as velocity. A cell is embedded if the absolute value of the sub-grid field is above a user-specified value. Conversely, a cell is released (*i.e.*, the embedding is removed) if the absolute value of the subgrid is below 1/5th of the user-specified value.

Value-Based AMR

To define embed criteria based on values rather than gradients, activate value-based AMR by setting *amr_type* to *EMBED ABOVE*, *EMBED BELOW*, or *EMBED BETWEEN*. The theory for value-based AMR is straightforward. Any cell that satisfies the embed criterion for the selected AMR type will be embedded, while cells that satisfy the holding criterion will be held. All other cells will be released.

Value-based AMR allows you to specify a value (positive or negative) for the embed criterion rather than using sub-grid scale. For example, you can use value-based AMR to define criteria for void fraction AMR to track the interface between a gas and a liquid and resolve a VOF simulation in more detail. You can set *amr_type* = *EMBED_BETWEEN* and *amr_criterion* = [0.4, 0.6] to embed cells that occur near the center of the transition from one phase to another.

Value-based AMR provides an interface to define arbitrary AMR definitions. You can define a new variable using a UDF (consult the CONVERGE 3.1 UDF Manual for more information), then define custom embedding. The *hold_criterion* is an optional parameter that can be used with the value-based AMR and should be used for AMR variables with values that are changed by embedding, such as parcel count. Cells that have values that fall between the *hold_criterion* and *embed_criterion* will maintain their current level of embedding.

Caution: Unlike SGS AMR, value-based AMR is typically not self-limiting. When SGS AMR adds more cells, the local gradient is lowered and a small portion of the domain will meet the embed criterion. Value-based AMR does not have a similar mechanism for most variables since embedding a cell will not change the value of the variable. For example, value-based AMR will embed every cell that meets the criterion for as many levels as you specify until the case reaches the maximum cell count or the simulation exhausts the system's available memory. Value-based AMR therefore should be applied sparingly and thoughtfully.

y+ AMR Restriction

In some simulations, the flow conditions may be such that sub-grid scale quantities near solid walls trigger AMR. The flow features near the wall, however, may not be of interest. When CONVERGE refines the grid near a wall, the total cell count will increase, which may prevent additional refinement near relevant flow features. Also, excessive refinement near the wall may cause the cell count to exceed [amr.in > amr_settings > max_cells](#), preventing further refinement. Another problem with AMR refinement near walls is that, if the y^+ value of a cell adjacent to a wall falls outside of the range for the chosen model, law-of-the-wall models will no longer produce physically realistic results.

To avoid these problems, CONVERGE includes an option for y^+ AMR restriction. You can specify a target y^+ value $+$ (on a boundary-by-boundary basis), and CONVERGE will remove AMR refinement in an effort to maintain the desired target value. Even if the new y^+ value does not meet the target, CONVERGE will not remove more than one level of AMR refinement at each time-step. If the y^+ value is still below the target value the next time CONVERGE evaluates AMR, it will again try to release one level of AMR refinement to achieve the target value. The parameters used to set up y^+ AMR restriction are in [amr.in > amr_yplus_restrict](#).

CONVERGE performs y^+ AMR restriction after all other AMR-related calculations. If, for example, in one time-step CONVERGE adds cells because of boundary AMR, then it may immediately remove those cells due to y^+ AMR restriction.

Inlaid Neighbor AMR

If a Cartesian cell along a flow-through INTERFACE boundary that demarcates an [inlaid mesh](#) is large relative to the adjacent cells within the inlaid grid, the cell size discrepancy may result in numerical instabilities. To mitigate this problem, inlaid neighbor AMR can refine or coarsen a Cartesian cell along an INTERFACE boundary so that the cell size transition is more gradual.

If the Cartesian cell's number of cell neighbors within the inlaid grid exceeds the value specified for [amr.in > amr_inlaid_neighbors > embed_criterion](#), CONVERGE embeds the Cartesian cell up to the value specified by [amr.in > amr_inlaid_neighbors > embed_scale](#). This way, the Cartesian cell is split into many cells, thus reducing the size discrepancy across the boundary.

Conversely, if the Cartesian cells outside of the inlaid grid are too small (*i.e.*, the inlaid grid cells along the flow-through INTERFACE boundary have more than [amr.in](#) > *amr_inlaid_neighbors* > *embed_criterion* Cartesian neighbors), CONVERGE releases the embedding on the Cartesian cells.

Proximity-Based AMR

To resolve length scales in tight gaps (such as those that occur in rotating machinery), activate proximity-based AMR to refine the mesh based on the small distance between boundaries that make up the geometry. If you activate proximity-based AMR, you must also activate the [proximity boundaries feature](#) (set [inputs.in](#) > *feature_control* > *prox_check_flag* = 1 or 2 and include a [proximity_boundaries.in](#) file in your case setup).

A proximity group contains boundaries that may form small gaps with the boundaries in other proximity groups. When you activate the proximity boundaries feature, supply proximity groups in [proximity_boundaries.in](#) > *proximity_groups*. During the simulation, CONVERGE checks if the proximity groups are within a specified distance.

If the gap between proximity groups is smaller than [proximity_boundaries.in](#) > *seal_tolerance*, CONVERGE flags cells in this gap as possible candidates for proximity AMR. Then, within the flagged cells, CONVERGE checks for cells in regions of the gap that are smaller than the value supplied for [amr.in](#) > *amr_proximity* > *boundary* > *prox_dist* and applies AMR to these cells. Hence, *prox_dist* must be less than or equal to *seal_tolerance*. In *amr.in*, the *embed_scale* and timing configuration control the proximity AMR.

For output purposes, when [inputs.in](#) > *feature_control* > *prox_check_flag* = 1, CONVERGE writes the proximity distance for the cells in regions of the gap that are closer than both *seal_tolerance* and *prox_dist* (*i.e.*, cells to which AMR was applied) to the [post output files](#). When *prox_check_flag* = 2, CONVERGE writes the proximity distance for cells in regions of the gap that are closer than *seal_tolerance*.

AMR Example

Figure 11.6 shows post-processing visualizations of a grid generated by AMR for a time-evolving spray bomb case. In this case, the velocity is allowed to embed three levels while the temperature is allowed to embed only two levels. The figure shows that cells are added only when and where they are needed, which significantly reduces the computational time as compared to a simulation with fixed embedding or with a finer base mesh size.

Chapter 11: Grid Control

Adaptive Mesh Refinement

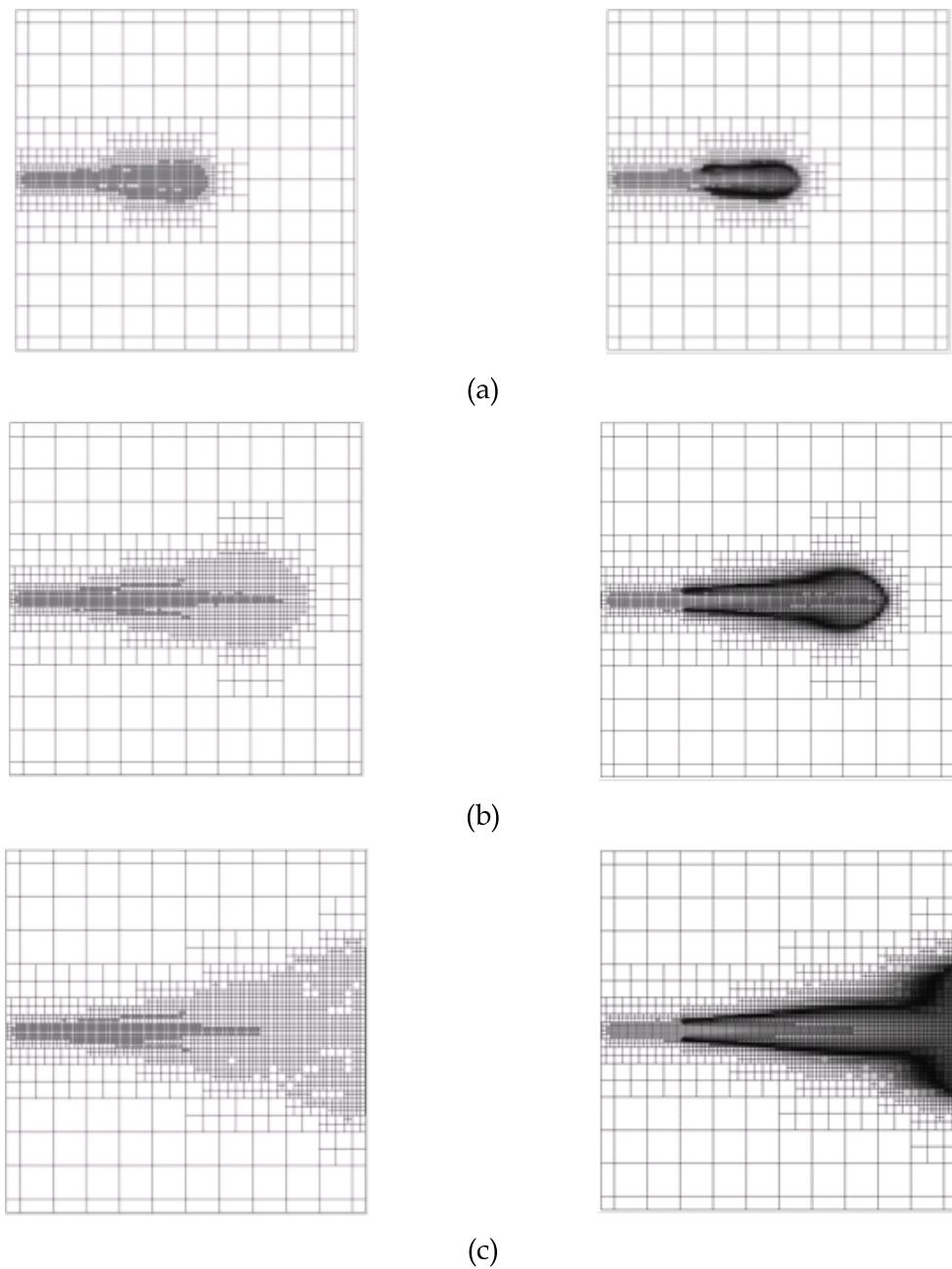


Figure 11.6: Evolution of a combusting spray bomb. This grid was generated by AMR (a) 0.5, (b) 1.0, and (c) 2.0 milliseconds after the start of the simulation. The left column shows only the grid, while the right column shows the grid and temperature values (black represents a temperature of approximately 2800 K) for corresponding times. This case has an ambient temperature of 1000 K, an ambient density of 14.8 kg/m^3 , an orifice diameter of 0.180 mm, and an injection pressure of 136 MPa.

11.6 Inlaid Meshing

CONVERGE offers an inlaid and boundary layer meshing feature, which constructs a non-Cartesian local mesh with prescribed surface generation and the [flow-through INTERFACE](#) boundary condition.

Inlaid meshes and boundary layer meshes are synonymous in CONVERGE, and they are identical from a computational standpoint. They differ primarily in where and how they are constructed. Inlaid meshes are created within the volume of the simulation in order to align grid faces with a flow in a known direction, and they are typically bounded on all sides by flow-through INTERFACE boundaries. Boundary layer meshes are generally formed by extruding surfaces from a solid WALL boundary, and they are bounded on one side by a no-slip wall. Figure 11.7 shows simple examples of (a) an inlaid mesh and (b) a boundary layer mesh. In both cases, the motivation is to align strong gradients with the mesh faces when you know the direction of those gradients before performing the simulation.

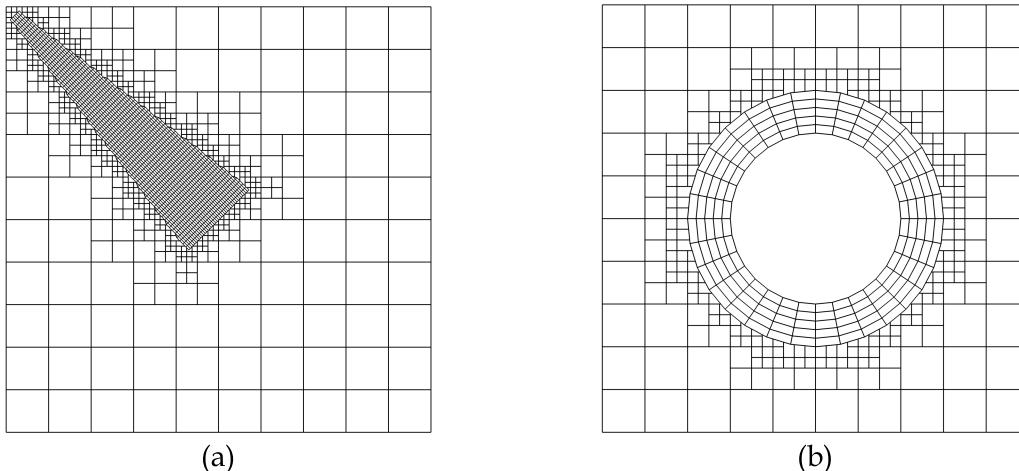


Figure 11.7: Inlaid and boundary layer meshes. (a) A simple inlaid mesh around an injector nozzle spray. (b) Boundary layer mesh of a cylinder. Note that in both cases, fixed embedding or inlaid neighbor AMR is used to approximately match cell sizes at the interface.

Although the cells within the inlaid mesh or boundary layer mesh are strictly controlled by the flow-through INTERFACE boundaries, connectivity must remain physically consistent at the interface between the inlaid mesh and the background Cartesian grid. Consider the example of the inlaid spray mesh from Figure 11.6, but without embedding. In Figure 11.8 (b), there is a face which connects the inlaid cell (dark red) with the background Cartesian cell (light blue). Imagine that the flow is from upper left to lower right, that is, from the blue cell to the red cell at the connecting face.

Chapter 11: Grid Control

Inlaid Meshing

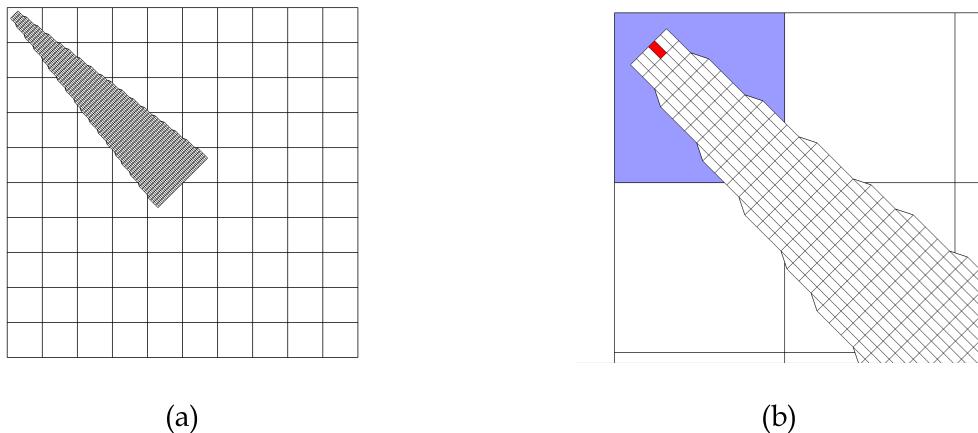


Figure 11.8: A poor interface between an inlaid mesh and a Cartesian mesh. (a) The inlaid spray mesh without embedding. (b) The tip of the inlaid spray mesh, highlighting an inlaid cell in dark red and the interfacing Cartesian cell in light blue.

In computational space, shown below in Figure 11.9 (a), the flow is in the direction of the black arrow, from the blue cell center to the red cell center. In physical space, shown in Figure 11.9 (b), the same connectivity for this face (from blue to red) is backwards. A flux that transports mass out of the blue cell moves it in the direction of the blue cell center. This is non-physical behavior, and it is numerically destabilizing.

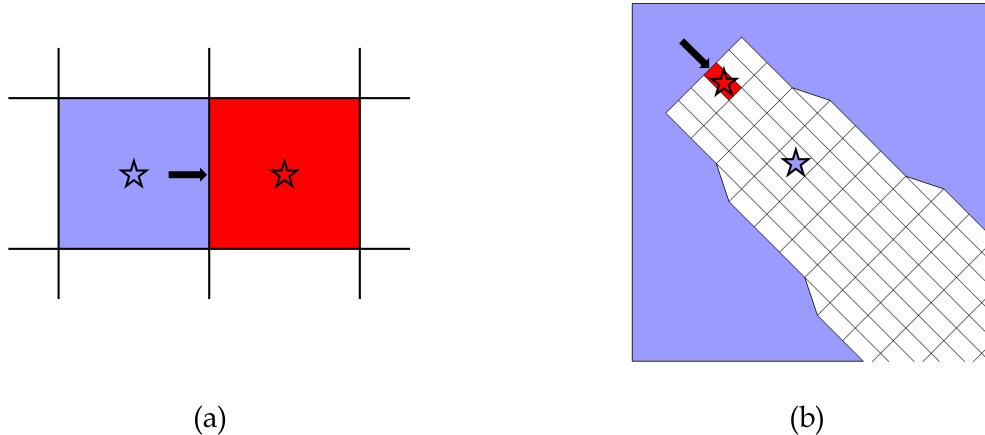


Figure 11.9: A poor interface between an inlaid mesh and a Cartesian mesh. (a) The interface between the two cells in computational space. (b) The interface between the two cells in physical space.

There is no completely robust prescription for avoiding these flipped interfaces. However, they are less likely to occur when the interfacing Cartesian grid is of similar cell spacing to the inlaid/boundary layer mesh. Flipped interfaces will also tend to be less severe if the mismatch is between cells of relatively similar size. A simulation may run acceptably and accurately with some small mismatches. You should use fixed embedding or AMR near to the inlaid/boundary layer mesh to approximately match cell spacings. You can activate CONVERGE's [mesh quality output](#) to monitor the size and location of low-quality interfaces.

Inlaid meshes are controlled via [*inlaid mesh.in*](#), which tells CONVERGE where to switch from the Cartesian grid to the inlaid mesh. In [*inlaid mesh.in*](#) > *boundary_ids*, you specify the flow-through INTERFACE boundaries from which the inlaid mesh is constructed (*i.e.*, the "interior" inlaid INTERFACE boundaries). The geometry must also include flow-through INTERFACE boundaries to mark the interface between the Cartesian grid and the inlaid mesh. These "exterior" inlaid INTERFACE boundaries are not specified in [*inlaid mesh.in*](#).

Because they are not a part of the Cartesian base grid, inlaid meshes cannot be combined with grid scaling, fixed embedding, or AMR. You can run a simulation with inlaid/boundary layer meshes and grid scaling, fixed embedding, and AMR, but these features are automatically disabled within the inlaid/boundary layer mesh. Inlaid cells cannot be paired with Cartesian cells. However, you can specify cell pairing for cells within an inlaid mesh. This feature is controlled by [*inlaid mesh.in*](#).

In principle, you can set up the boundary layer surfaces and flow-through INTERFACE boundaries in the [*surface geometry file*](#), [*boundary.in*](#), and [*inlaid mesh.in*](#), but this is not practical outside of the simplest test cases. CONVERGE Studio features parametrized inlaid mesh surface generation and boundary layer surface extrusion features with semi-automated flow-through INTERFACE setup (region assignment is not automatic), and can also load [PLOT3D](#)-formatted structured grids from a file. We strongly recommend using CONVERGE Studio to set up inlaid and boundary layer meshes. Refer to the CONVERGE Studio manual for more information on this feature.

Inlaid Mesh Motion

You can specify deforming or non-deforming motion for an inlaid mesh.

Non-deforming motion does not cause any change in shape for the inlaid cells and corresponds to standard motion types, such as *TRANSLATING* and *ROTATING*. For an inlaid mesh in free space, non-deforming motion requires that all inlaid INTERFACE boundaries share the same motion. For an extruded boundary layer mesh, the inlaid INTERFACE boundaries must share the same motion as the surface from which they were extruded.

Deforming motion allows the shapes of the inlaid cells to change in response to the motion of connected boundaries. Consider the example in Figure 11.10, in which the left side of the box has a Cartesian grid and the right side has an inlaid mesh. The inlaid mesh, which is bounded by the exterior inlaid INTERFACE, the liner, and the piston, deforms as the piston moves in the -x direction. In this example, the motion of the piston is prescribed, while the deforming motion of the inlaid mesh is calculated by CONVERGE. Figure 11.11 shows the deformation of the inlaid cells in response to the piston motion.

Chapter 11: Grid Control

Inlaid Meshing

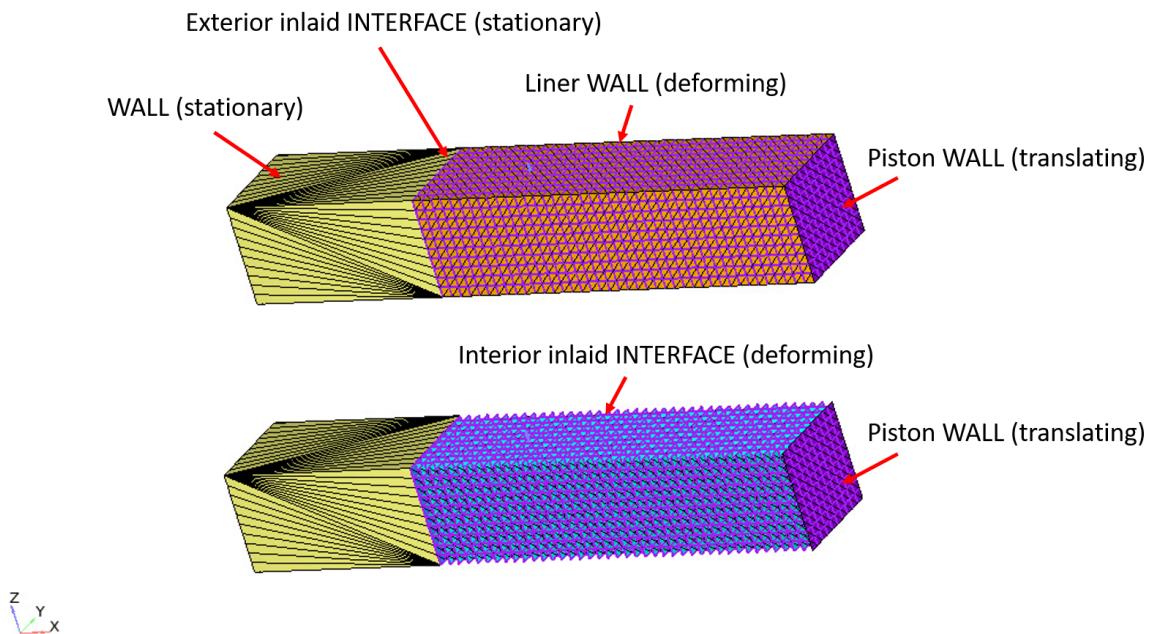


Figure 11.10: Two views of a geometry with a deforming inlaid mesh. The top view shows the WALL boundaries and the exterior inlaid INTERFACE that separates the left and right sides of the box. The bottom view shows the interior inlaid INTERFACE on the right side of the box.

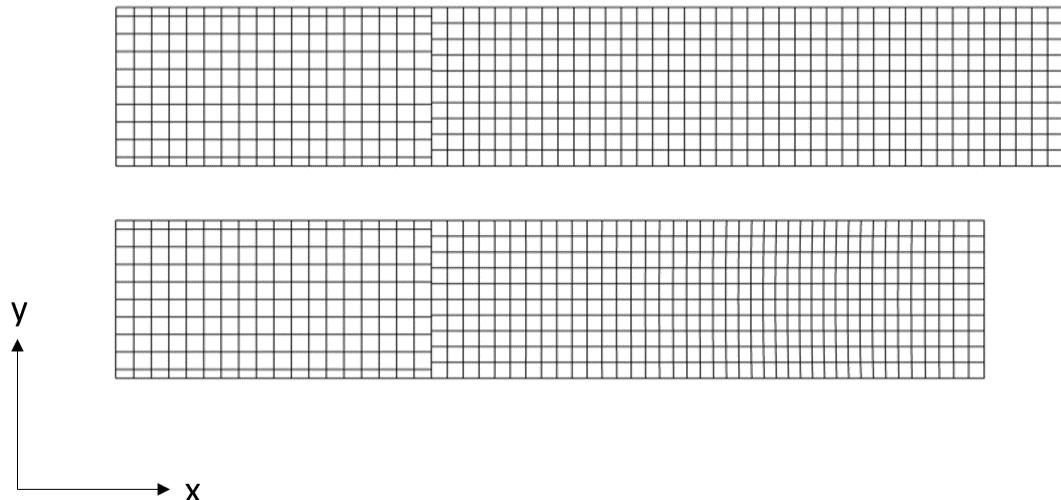


Figure 11.11: The cells of the original inlaid mesh (top) become compressed as the piston moves in the -x direction (bottom).

Chapter 11: Grid Control

Inlaid Meshing

CONVERGE uses the radial basis function approach ([Sheng and Allen, 2013](#)) to calculate vertex displacements for the deforming boundaries. The radial basis function implemented in CONVERGE is given by

$$f(x) = \sum_{p=0}^N \lambda_p \left[\varphi(\|x - x_p\|) \right], \quad (11.6)$$

where x is a point in space and λ_p are the weights assigned to a set of basis points x_p . The function φ , which depends only on the distance (Euclidean norm) between x and x_p , is given by

$$\varphi(\varepsilon) = (1 - \varepsilon^4)(4\varepsilon + 1). \quad (11.7)$$

At each time-step, CONVERGE chooses the weights and basis points such that Equation 11.6 predicts the displacement d of vertex x_n , and thus can be written in matrix form as

$$f(x_n) = d(x_n). \quad (11.8)$$

The basis points are a subset of vertices (s_1, s_2, \dots, s_N) on the moving boundaries connected to the inlaid mesh (e.g., the piston in Figure 11.10). To determine the weights, CONVERGE first solves Equation 11.8 for the basis points. The system of equations can be written as

$$A\lambda_i = d_i, \quad (11.9)$$

where the displacement vectors

$$\begin{aligned} d_x &= [\Delta x_{s_1}, \Delta x_{s_2}, \dots, \Delta x_{s_N}]^T \\ d_y &= [\Delta y_{s_1}, \Delta y_{s_2}, \dots, \Delta y_{s_N}]^T \\ d_z &= [\Delta z_{s_1}, \Delta z_{s_2}, \dots, \Delta z_{s_N}]^T \end{aligned} \quad (11.10)$$

are known from the boundary motion (prescribed or calculated by another model, such as a [fluid-structure interaction](#) model), and the components of the matrix A are given by

$$A_{s_n s_m} = \varphi(\|s_n - s_m\|). \quad (11.11)$$

After obtaining the weights from Equation 11.9, CONVERGE uses Equation 11.8 to calculate vertex displacements for the deforming boundaries.

To allow deforming motion, set [boundary.in](#) > *boundary_conditions* > *boundary* > *motion* = *DEFORM* for the interior inlaid INTERFACE boundaries, as well as any other boundaries that will have their vertex displacements calculated from the radial basis function. In the example above, the liner WALL boundary has *motion* = *DEFORM* so that the liner vertices can move in tandem with the inlaid mesh. If the liner WALL boundary had *motion* = *STATIONARY*, the cells attached to the liner would become abnormally skewed as the inlaid mesh deformed.

Inlaid Mesh Activation and Deactivation

Inlaid meshes can be active throughout the simulation, or you can activate and deactivate them during the simulation via boundary events. This approach activates the inlaid mesh by associating the interior and exterior boundaries with a CLOSE boundary event, and deactivates the inlaid mesh by associating these boundaries with an OPEN boundary event.

To set up inlaid mesh activation and deactivation, set the inner and outer inlaid mesh boundaries as disconnected ([boundary.in](#) > *boundary* > *disconnect* = 1). Then set up boundary-type close or open events in [events.in](#) to activate or deactivate the inlaid mesh.

For cases that use [grid scaling](#), you can also activate or deactivate the inlaid mesh when the simulation reaches a specific grid scaling value. This option is available only when [inputs.in](#) > *grid_control* > *grid_scale* is set to a file name (e.g., [gridscale.in](#)). This option is not available when *grid_scale* is set to an integer. Set [events.in](#) > *event* > *temporal_type* = *GRIDSCALE* and specify the target grid scaling value in [events.in](#) > *event* > *starting_gridscale*. Set [events.in](#) > *event* > *type* to the initial state of the inlaid mesh (OPEN or CLOSE). When the target grid scaling value is applied, CONVERGE changes the state of the inlaid mesh to the opposite value (CLOSE or OPEN). You can also set [events.in](#) > *event* > *delay_time* if you want to delay the activation or deactivation of the inlaid mesh by a certain amount of time. You can define only one grid scaling event per boundary, and you cannot use events with other temporal types for the same boundary. Figure 11.12 below shows an example grid scaling event setup.

```
- event:
  type:          OPEN
  contact_resistance_flag: 0
  temporal_type: GRIDSCALE
  starting_gridscale: 0
  delay_time:    600
  boundary_id:   [11, 12]
```

Figure 11.12: Excerpt from [events.in](#) showing an example grid scaling event for a steady-state case. Boundaries 11 and 12 are associated with the inlaid mesh. CONVERGE activates the inlaid mesh 600 cycles after reaching the target grid scaling value of 0.

When an inlaid-Cartesian event occurs, CONVERGE first generates the new grid (either the inlaid grid or a Cartesian grid propagated through the inlaid volume at the embed level of the interface). CONVERGE then calculates the intersections between the cells in the existing grid and cells in the new grid and calculates the appropriate volume-weighted averages of the conserved quantities for each new cell. Because no interpolation is performed, all such

Chapter 11: Grid Control

Inlaid Meshing

quantities (mass, momentum, energy, species, etc.) are conserved to within machine precision.

Mesh Quality

Simulations with an inlaid grid can require more attention to mesh quality. CONVERGE will calculate and output a set of mesh quality metrics if you specify *mesh_quality* in [*post.in*](#). The following table lists these mesh quality metrics, as well as ideal and threshold values. The ideal values are theoretical, and it is impossible to generate a non-trivial grid that achieves them in every cell.

The threshold values are only general guidelines. CONVERGE will frequently run stably and generate accurate results even if the mesh quality metrics are worse than the threshold values, especially if the lower-quality regions are small and rare, and if they are not near strong flow gradients. If CONVERGE is not running stably, or is generating inaccurate results, you should consider altering the mesh to improve these metrics, especially if the lower-quality regions are not at the Cartesian-inlaid interface.

Chapter 11: Grid Control

Inlaid Meshing

Table 11.1: Mesh quality metrics.

Metric	Description	Ideal value	Threshold value
ASPECT_RATIO	Aspect ratio of this cell.	1.0	No general recommendation.
NON-ORTHOGONALITY	For each neighbor, CONVERGE calculates the angle between the vector connecting two cell centers and the normal of their common face, then prints the maximum value.	0 deg	Less than 70 deg. Near or above 90 deg is very likely to cause a crash.
SKEWNESS	Angle between the vector connecting the cell center to the face center and the normal of the face.	0 deg	Less than 70 deg. Near or above 90 deg is very likely to cause a crash.
NUM_INLAID_NBRS	Number of inlaid neighbors for a Cartesian cell. Returns 0 for inlaid cells.	1 per Cartesian face	Fewer than 15-20 for a 3D simulation.
NUM_CARTESIAN_NBRS	Number of Cartesian neighbors for an inlaid cell. Returns 0 for Cartesian cells.	1 per inlaid cell face	Fewer than 15-20 for a 3D simulation.
FACE_WARPAGE	Maximum angle between two faces of a cell that are shared with the same neighbor.	0 deg	Away from a Cartesian-inlaid interface, less than 10 deg. Cartesian cells that contain a corner of an inlaid cell will have a value close to 90 deg.
STRETCH_RATIO	For each face, CONVERGE calculates the ratio of the distance from the face to either cell center, then prints the maximum value for each cell.	1.0	Within and adjacent to inlaid meshes, less than 1.15. Areas of embedding or AMR will have stretch ratios of 2.0, which is acceptable.

Chapter



12

Initialization

12 Initialization

This chapter describes the different methods by which CONVERGE can initialize physical variables (*e.g.*, velocity, pressure, temperature, and species).

There are three ways to initialize values in CONVERGE:

1. [Specify uniform values](#) for the entire geometry or specify different values for different regions of the domain (assuming you have divided the domain into regions).
2. [Map field values](#) from a file, which allows each cell to be initialized individually.
3. Initialize the domain from the values of the field variables in a [restart file](#).

The following subsections discuss each option in detail.

12.1 Restarting

If you start a simulation from a restart file, CONVERGE will initialize the domain with information from that restart file.

[Chapter 24 - File Overview](#) contains information about [restart files](#), and [Chapter 25 - Input and Data Files](#) describes the parameters in [inputs.in](#) that you can use to set up a restart.

The Getting Started Guide contains additional information about restarting a simulation.

12.2 Mapping

Another way to initialize the domain is via mapping: you specify a map file (*i.e.*, a file that contains three-dimensional location-specific values), and CONVERGE will initialize the domain with the values from that file. CONVERGE allows you to initialize any number of field variables as well as parcel data and fluid-structure interaction (FSI) data via mapping. Mapping is useful if you wish to begin a simulation with output from a different CFD solver, with experimental data (*e.g.*, LDV or PIV data), or with spatially varying initial values that cannot be initialized via [initialize.in](#).

To initialize the domain via mapping, set `inputs.in > simulation_control > map_flag = MAP` and supply a [map.in](#) file. The [map.in](#) file must include the file name of at least one HDF5-formatted data file (*e.g.*, `map.h5`) or restart file that contains spatially varying data. To view or analyze HDF5-formatted files, use the utilities provided by the HDF group, such as HDFView. Additionally, you must list the mapping configuration in [map.in](#).

Map data files from CONVERGE 2.4 and earlier versions are not HDF5-formatted. CONVERGE Studio can convert old map files to HDF5 format. Please refer to the CONVERGE Studio 3.1 Manual for more information.

CONVERGE maps on a region-by-region basis. In [map.in](#), you can list one or multiple `map_region` settings block(s) in which you specify the map file name, the regions and data sets to map, and any manipulations (such as translation or scaling) to apply to the mapped

data sets. If you do not supply manipulations for a data set, CONVERGE implicitly assumes that the manipulation parameters are 1.0 and maps the quantities to the domain directly from the map file.

The variables that are not mapped will be initialized by [initialize.in](#). The regions that are not listed in the map file will be initialized via [initialize.in](#). Note that mapping will take precedence over these [initialize.in](#) values (when [inputs.in](#) > [simulation_control](#) > [map_flag](#) = MAP) for the specified variables in the specified regions.

To restart a simulation by mapping output from a previous CONVERGE simulation, specify the restart file name (*e.g.*, `restart.rst`) as the map file name in [map.in](#). This approach may be useful if, for example, the base grid size for a simulation was inappropriate (either too coarse or too fine). You can initialize a new simulation with the data from the restart file but select a more appropriate grid size.

All cells in the new grid are initialized with the value of the nearest point of the original grid. When creating your own region-by-region map data files from other codes or experimental data, you need to be aware of the locations of the points in the map data file. The mapping algorithm interpolates from the nearest neighboring point, regardless of whether that point lies within the same region that is mapped.

You cannot map density values from one grid to the other. Density is determined by the temperature, pressure, and species mass fractions for compressible flow (*i.e.*, from the equation of state). For incompressible liquids, the density is determined by the species mass fractions, *i.e.*, density for incompressible liquids is a function only of the species that is being transported.

Considerations for Multi-Stream Simulations

CONVERGE writes separate [map files](#) for each stream in a simulation. Each file contains mapping data for only one stream. To initialize a new multi-stream simulation from these data, you can either keep the data in separate files or merge multiple stream-specific map files into a single map file.

If you keep the data in separate files, the new simulation must have the same streams as the simulation from which the files were generated. For the new simulation, specify a subdirectory for each stream in [stream_settings.in](#) > [stream_list](#) > [stream](#) > [stream_overwrite](#). Put the stream-specific `map_*.h5` files in the appropriate subdirectories. If the [map.in](#) settings (including the names of the `map_*.h5` files) are identical for all streams, you can use a single [map.in](#) file in the top-level Case Directory. Otherwise, override the [map.in](#) settings on a stream-by-stream basis as needed.

To merge the stream-specific map files into a single file, run the `merge_mapfiles.py` utility. You can choose which files to merge with this utility, so the new simulation need not have the same streams as the original simulation. Put the merged map file created by the utility in the top-level Case Directory for the new simulation, along with a [map.in](#) file.

Data sets in the merged map file are organized by stream ID. By default, CONVERGE does not map across streams when locating the nearest point in the map file from which to map data (e.g., only data from stream 0 can be mapped to cells that belong to stream 0 in the new simulation). You can set [*map.in* > *map_phase_flag* = 1](#) to allow mapping across streams. When this flag is activated, CONVERGE considers all streams of the same phase (fluid or solid) when locating the nearest point in the map file (e.g., data from any fluid stream in the map file can be mapped to a fluid stream in the new simulation).

The [*merge_mapfiles.py*](#) utility is included in the CONVERGE installation package. To run this utility, you must have Python 3 and h5py installed on your system.

12.3 Specified Initial Values

The most common initialization method is simply to specify initial values for thermodynamic quantities (e.g., pressure, temperature, etc.) for the entire domain or for individual regions (if the domain has been divided into regions) via [*initialize.in*](#).

For engine cases, CONVERGE uses the engine parameters specified in [*engine.in*](#) to initialize the velocity in the [*piston*](#) and in the [*cylinder*](#), as described below. For a multi-cylinder case, only one cylinder will be initialized using this method.

Velocity Initialization in the Piston

The boundaries defined by *piston_id*, *head_id*, and *liner_id* in [*engine.in*](#) must be associated with the same region. CONVERGE sets the *w* component (z direction) of velocity to the piston velocity for all points in the piston face. You can specify piston velocity in one of three ways:

- *u, v, w* velocities: Specify the values of the *u, v, w* velocities (in m/s) of the piston in [*boundary.in*](#).
- [*Piston motion*](#): Specify engine parameters (including rpm and stroke) in [*engine.in*](#). CONVERGE will use the [*crank-slider mechanism*](#) to determine the piston velocity throughout the simulation. We recommend this option for engine applications.
- [*Piston position*](#): Specify temporally varying position data from which CONVERGE will calculate piston velocity. Note that this file specifies only position (not velocity) data.

Piston Motion

If you specify piston motion as the velocity boundary condition for the piston boundary, CONVERGE will internally generate the position tables for the piston boundary using the [*crank-slider equation*](#). If you use this option, you must verify that the [*surface geometry file*](#) is prepared with the piston at bottom dead center (BDC) position. (Even if you do not use this option, you must verify that the piston is at BDC prior to running a simulation.)

To use piston motion as the piston's velocity boundary condition, set [*boundary.in* > *boundary_conditions* > *boundary* > *velocity* > *value* = *AUTO_GENERATE*](#). CONVERGE will use the engine geometry parameters in [*engine.in*](#) to determine the piston position table (using a crank-slider motion with options for the crank and wrist pin offset). We recommend this

option for most typical engine cases. The piston position is calculated at each time-step, based on a piston profile calculated with a resolution of 0.1 *crank angle degrees*.

To direct CONVERGE to write a piston position output file, set `boundary.in > boundary_conditions > boundary > velocity > echo_profile = 1`. The output file, named `piston_profile<boundary ID>.out`, contains piston position information calculated every 0.1 *crank angle degrees*.

Piston Position

To enter your own piston position table for the piston's velocity boundary condition, specify a file name (e.g., `piston_position.in`) for `boundary.in > boundary_conditions > boundary > velocity > value`. Then save a file with that name in your case setup. The figure below shows a properly formatted piston position file that contains temporally varying position data. CONVERGE will interpolate position values for *crank angle degrees* that are not specified in this file.

TEMPORAL CYCLIC			
crank	x	y	z
0	0	0	0
180	0	0	.085
360	0	0	0
540	0	0	.085
720	0	0	0

Figure 12.1: An example of a piston position profile (e.g., `piston_position.in`).

Velocity Initialization in the Cylinder

CONVERGE gives the remaining grid cells in the cylinder region an initial w velocity (z direction) consistent with a field that linearly interpolates from the piston speed at the piston face to zero at the head, as

$$w(z) = w_{piston} \left(\frac{z_{head} - z}{z_{head} - z_{piston}} \right), \quad (12.1)$$

where w_{piston} is the speed of the piston and z_{head} and z_{piston} are the z coordinates of the head and piston, respectively. CONVERGE calculates z_{piston} as the difference between the *zhead* and *stroke* you specify in `engine.in`, as shown in Figure 12.2. Note that CONVERGE performs the linear interpolation only between z_{head} and z_{piston} . Cells in the piston bowl have the same velocity as the piston boundary, and any cells above z_{head} (e.g., a non-flat cylinder head) are initialized to zero velocity. Your choice of *zhead* affects only the initialization of the cylinder region, and it is not used elsewhere in the simulation.

Chapter 12: Initialization

Specified Initial Values Velocity Initialization in the Cylinder

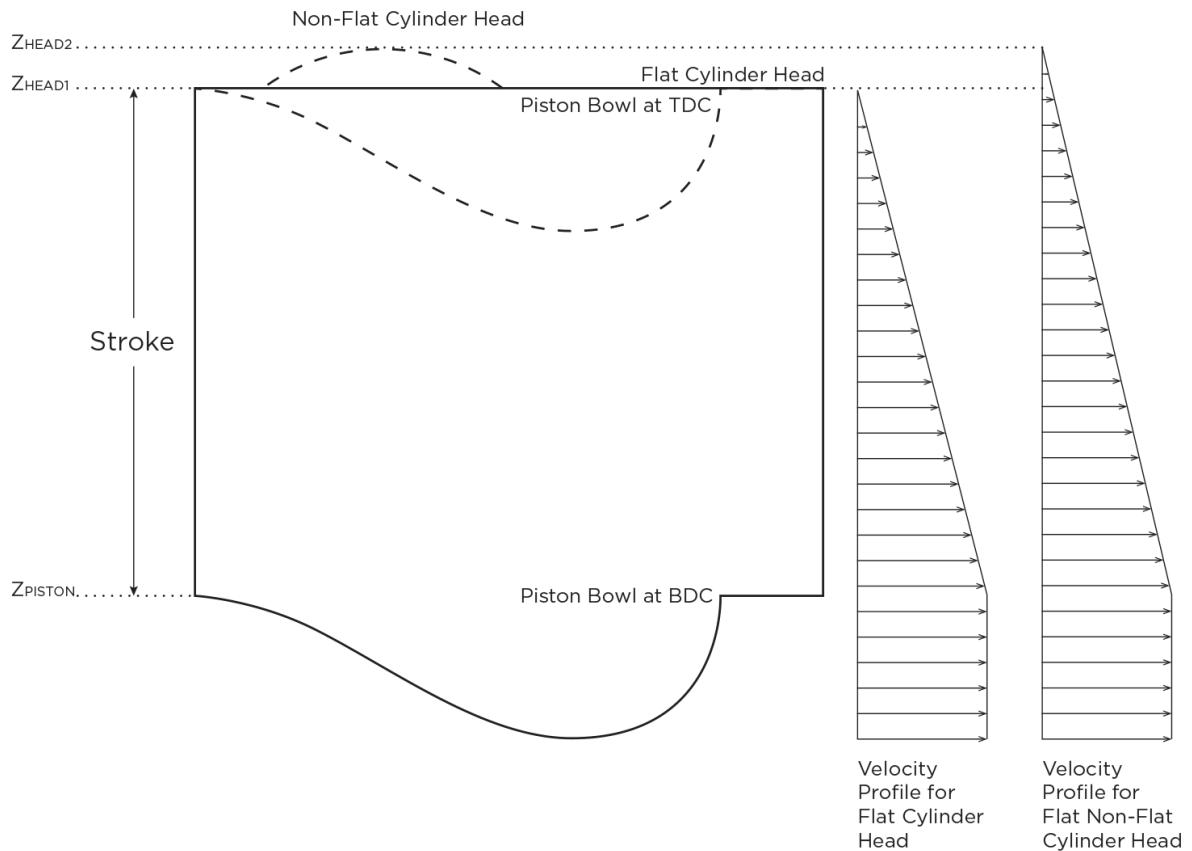


Figure 12.2: Initialization of w velocity for two different choices of z_{head} . Note that squish height is assumed to be zero for w velocity initialization.

CONVERGE uses *swirl* and *swirl_profile* in [engine.in](#) to set the u (x direction) and v (y direction) components of the velocity in the cylinder region. The *swirl* is the ratio of the angular velocity of the flow, Ω_{flow} (radians/second), to the angular velocity of the crankshaft, $\Omega_{crankshaft}$ (radians/second). The direction of swirl consistent with the right-hand rule. In other words,

$$swirl = \frac{\Omega_{flow}}{\Omega_{crankshaft}}. \quad (12.2)$$

The simplest approach to initializing swirl is to assume wheel flow. Assuming the cylinder axis is aligned with the z axis, the u (x direction) and v (y direction) components of velocity are initialized as

$$u = \Omega_{flow}y \text{ and } v = \Omega_{flow}x. \quad (12.3)$$

The wheel-flow assumption is not usually realistic since the velocity should diminish significantly near the cylinder wall. It has been observed that a Bessel function more accurately represents the velocity profile in an engine (Amsden et al., 1989). The *swirl_profile* is a dimensionless constant used in the Bessel function calculation with a minimum value of 0.0 for wheel flow and a maximum value of 3.83 for zero velocity at the wall. For IC engines, typically *swirl_profile* = 3.11. To specify swirl for a V engine, use [*dynamic.in*](#).

Hydrostatic Pressure Initialization

You can initialize a pressure field at time = 0 with a hydrostatic gradient. This feature is especially important in liquid simulation cases with gravity turned ON ([*inputs.in*](#) > *property_control* > *gravity*).

CONVERGE solves the following equation with a given reference point,

$$\frac{\partial p}{\partial x_i} = \rho g_i. \quad (12.4)$$

The reference point can be selected with either the *COPY_FROM_BOUNDARY* method or the *MANUAL_INPUT* method in [*initialize.in*](#). Only one reference point should be provided for a group of connected regions or streams. That is, if *region_id* 1 and *region_id* 2 are connected, the *hydrostatic_pres_init* option should only be used in one of the regions in [*initialize.in*](#). When the *COPY_FROM_BOUNDARY* method is used, the *reference_center* provided on the pressure boundary must lie on the boundary.

Chapter



13

Turbulence Modeling

13 Turbulence Modeling

Turbulence significantly increases the rate of mixing of momentum, energy, and species. For a wide variety of applications, it is difficult to attain accurate CFD simulation results without modeling turbulence.

Turbulence-enhanced mixing is a convective process that results from the presence of turbulent eddies in the flow. These turbulent eddies occur at many length scales. If a CFD solver does not contain a discretized domain (grid) that can resolve the smallest eddy length scales, then the solver cannot entirely account for the enhanced mixing effects of turbulence in the simulation. Often it is not practical to resolve all of the length scales in a typical CFD simulation, and thus turbulence models are used to account for the additional mixing.

To activate turbulence modeling, set `inputs.in > solver_control > turbulence_solver = 1` and include a `turbulence.in` file in your case setup. The following table summarizes the turbulence models available in CONVERGE and the corresponding values of the `turbulence_model` parameter in `turbulence.in`.

Table 13.1: Turbulence models in CONVERGE.

Model type	Model name	<i>turbulence_model</i> value
RANS	Standard $k-\varepsilon$	<code>RANS_K_EPS_STD</code>
	RNG $k-\varepsilon$	<code>RANS_K_EPS RNG</code>
	Rapid Distortion RNG $k-\varepsilon$	<code>RANS_K_EPS RNG_RD</code>
	Generalized RNG $k-\varepsilon$	<code>RANS_K_EPS RNG_GEN</code>
	Realizable $k-\varepsilon$	<code>RANS_K_EPS REAL</code>
	$\overline{v^2} - f$	<code>RANS_K_EPS_V2F</code>
	$\zeta - f$	<code>RANS_K_EPS_ZETAF</code>
	Standard $k-\omega$ (1998)	<code>RANS_K_OMEGA_STD_98</code>
	Standard $k-\omega$ (2006)	<code>RANS_K_OMEGA_STD</code>
	$k-\omega$ SST	<code>RANS_K_OMEGA_SST</code>
LES	RSM SSG	<code>RANS_RSM_SSG_EPS</code>
	RSM LRR	<code>RANS_RSM_LRR_EPS</code>
	Spalart-Allmaras	<code>RANS_SPALART_ALLMARAS</code>
	Upwind LES	<code>LES_UPWIND</code>
	One-equation viscosity	<code>LES_ONE_EQN_VISC</code>

	Smagorinsky	<i>LES_SMAG</i>
	Dynamic Smagorinsky	<i>LES_DYN_SMAG</i>
	Dynamic structure	<i>LES_DYN_STRUCT</i>
	Consistent dynamic structure	<i>LES_CON_DYN_STRUCT</i>
	Sigma LES model	<i>LES_SIGMA</i>
DES	Delayed DES	<i>DDES_K_OMEGA_SST</i>
	Improved delayed DES	<i>IDDES_K_OMEGA_SST</i>

CONVERGE requires turbulence modeling (*i.e.*, you must set [*inputs.in* > *solver_control* > *turbulence_solver* = 1](#) and include a [*turbulence.in*](#) file) when any one or more than one of the following conditions apply.

- The law-of-the-wall (*la*) boundary condition for velocity is specified for any boundary in the surface (refer to [Chapter 9 - Boundaries and Boundary Conditions](#)).
- The law-of-the-wall (*la*) boundary condition for temperature is specified for any boundary.
- The O'Rourke or tke-preserving model is specified for turbulent dispersion (*i.e.*, [*parcels.in* > \[type\] > *parcel_list* > *parcel* > *turb_dispersion_control* > *dispersion_model* = *OROURKE* or *TKE_PRESERVING*](#)).
- The CTC (Characteristic Time Combustion) model has been enabled ([*combust.in* > *ctc_flag* = 1](#)).

If you are using CONVERGE Studio to configure the input files, CONVERGE Studio will automatically activate turbulence modeling when any of the above conditions are true.

13.1 RANS Models

In Reynolds-Averaged Navier-Stokes (RANS) turbulence models, the flow variables (*e.g.*, velocity) are decomposed into an ensemble mean and a fluctuating term as

$$\underbrace{u_i}_{\text{instantaneous velocity}} = \underbrace{\bar{u}_i}_{\text{ensemble mean}} + \underbrace{u'_i}_{\text{fluctuating}}. \quad (13.1)$$

To derive the RANS transport equations and averaging, substitute the RANS decomposition (Equation 13.1) into the Navier-Stokes equations. The compressible RANS equations for mass and momentum transport are

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial \bar{\rho} \bar{u}_j}{\partial x_j} = 0 \quad (13.2)$$

and

Chapter 13: Turbulence Modeling

RANS Models

$$\frac{\partial \bar{\rho} \tilde{u}_i}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_i \tilde{u}_j}{\partial x_j} = -\frac{\partial \bar{P}}{\partial x_i} + \frac{\partial}{\partial x_j} \left[\mu \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) - \frac{2}{3} \mu \frac{\partial \tilde{u}_k}{\partial x_k} \delta_{ij} \right] + \frac{\partial}{\partial x_j} \left(-\bar{\rho} \tilde{u}'_i \tilde{u}'_j \right). \quad (13.3)$$

where the Favre average, $\tilde{\cdot}$, is defined for velocity as

$$\tilde{u}_i \equiv \frac{\overline{\rho u'_i}}{\rho}. \quad (13.4)$$

The ensemble averaging of the equations introduces additional terms called the Reynolds stresses that represent the effects of turbulence. The Reynolds stress τ_{ij} is given by

$$\tau_{ij} = -\widetilde{\rho u'_i u'_j}, \quad (13.5)$$

which is included in the last term on the right side of Equation 13.3. The turbulence model must model the Reynolds stress to obtain closure for Equation 13.3. RANS models solve additional transport equations for turbulence modeling variables (*e.g.*, the turbulent kinetic energy) to close this term.

The following RANS turbulence models are available in CONVERGE: Standard $k-\varepsilon$, RNG (Renormalization Group) $k-\varepsilon$, Rapid Distortion RNG $k-\varepsilon$ ([Han and Reitz, 1995](#)), Realizable $k-\varepsilon$, Standard $k-\omega$ 1998 ([Wilcox, 1998](#)), Standard $k-\omega$ 2006 ([Wilcox, 2006](#)), $k-\omega$ SST, and Spalart-Allmaras ([Spalart and Allmaras, 1994](#)).

k- ε Models

Traditionally, RANS models use an effective turbulent viscosity to model the Reynolds stress term. Thus, additional turbulent diffusion (*i.e.*, diffusive mixing) models the turbulent convective mixing.

For all two-equation $k-\varepsilon$ models, the turbulence length scale le is given by

$$le = C_\mu^{3/4} \frac{k^{3/2}}{\varepsilon}. \quad (13.6)$$

Standard and RNG k- ε

The modeled Reynolds stress for the Standard $k-\varepsilon$ and RNG models is given by

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

$$\tau_{ij} = -\overline{\rho \widetilde{u'_i u'_j}} = 2\mu_t S_{ij} - \frac{2}{3}\delta_{ij} \left(\rho k + \mu_t \frac{\partial \widetilde{u_i}}{\partial x_i} \right). \quad (13.7)$$

The turbulent kinetic energy, k , is defined as half of the trace of the stress tensor:

$$k = \frac{1}{2} \widetilde{u'_i u'_i}, \quad (13.8)$$

where the turbulent viscosity, μ_t , is given by

$$\mu_t = C_\mu \rho \frac{k^2}{\varepsilon}. \quad (13.9)$$

In the previous equation, C_μ is a model constant that you can tune for a particular flow and ε is the dissipation of turbulent kinetic energy. The mean strain rate tensor S_{ij} is given by

$$S_{ij} = \frac{1}{2} \left(\frac{\partial \widetilde{u_i}}{\partial x_j} + \frac{\partial \widetilde{u_j}}{\partial x_i} \right). \quad (13.10)$$

The models use turbulent diffusion and turbulent conductivity terms to account for the presence of turbulence in mass transport and energy transport. The turbulent diffusion and conductivity terms are

$$D_t = \frac{\mu_t}{\rho Sc_t} \quad (13.11)$$

and

$$K_t = \frac{\mu_t}{\rho Pr_t} C_p, \quad (13.12)$$

where Sc_t is the turbulent Schmidt number, Pr_t is the turbulent Prandtl number, D_t is the turbulent diffusion, and K_t is the turbulent conductivity.

The standard k - ε and RNG k - ε models require additional transport equations to obtain the turbulent viscosity given by Equation 13.9. One equation is needed for the turbulent kinetic energy, k , and one for the dissipation of turbulent kinetic energy, ε . The turbulent kinetic energy transport equation is given by

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

$$\frac{\partial \rho k}{\partial t} + \frac{\partial (\rho u_i k)}{\partial x_i} = \tau_{ij} \frac{\partial u_i}{\partial x_j} + \frac{\partial}{\partial x_j} \frac{\mu + \mu_t}{Pr_k} \frac{\partial k}{\partial x_j} - \rho \varepsilon + \frac{C_s}{1.5} S_s, \quad (13.13)$$

where the factor of 1.5 is an empirical constant. The transport equation for the dissipation of turbulent kinetic energy is given by

$$\begin{aligned} \frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial (\rho u_i \varepsilon)}{\partial x_i} &= \frac{\partial}{\partial x_j} \left(\frac{\mu + \mu_t}{Pr_\varepsilon} \frac{\partial \varepsilon}{\partial x_j} \right) + C_{\varepsilon 3} \rho \varepsilon \frac{\partial u_i}{\partial x_i} \\ &\quad + \left(C_{\varepsilon 1} \frac{\partial u_i}{\partial x_j} \tau_{ij} - C_{\varepsilon 2} \rho \varepsilon + C_s S_s \right) \frac{\varepsilon}{k} + S - \rho R_\varepsilon, \end{aligned} \quad (13.14)$$

where S is the user-supplied source term and S_s is the source term that represents interactions with discrete phase (spray). Note that these two terms are distinct. The $C_{\varepsilon i}$ terms are model constants that account for compression and expansion. In the previous equation, $R_\varepsilon = 0$ for the standard $k-\varepsilon$ model and

$$R_\varepsilon = \frac{C_\mu \eta^3 (1 - \eta/\eta_0) \varepsilon^2}{(1 + \beta \eta^3)} \frac{k}{k} \quad (13.15)$$

for the RNG $k-\varepsilon$ model. In Equation 13.15, the expression for η is

$$\eta = \frac{k}{\varepsilon} |S_{ij}| = \frac{k}{\varepsilon} \sqrt{2 S_{ij} S_{ij}}. \quad (13.16)$$

To activate the standard $k-\varepsilon$, set [*turbulence.in*](#) > *turbulence_model* = *RANS_K_EPS_STD*. To activate the RNG $k-\varepsilon$ model, set *turbulence_model* = *RANS_K_EPS_RNG*.

Rapid Distortion RNG $k-\varepsilon$

The Rapid Distortion RNG $k-\varepsilon$ model uses the transport equation for ε given by

$$\begin{aligned} \frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial (\rho u_i \varepsilon)}{\partial x_i} &= \frac{\partial}{\partial x_j} \left(\frac{\mu + \mu_t}{Pr_\varepsilon} \frac{\partial \varepsilon}{\partial x_j} \right) - \left[\frac{2}{3} C_{\varepsilon 1} - C_{\varepsilon 3} + \frac{2}{3} C_\mu C_\eta \frac{k}{\varepsilon} \frac{\partial u_k}{\partial x_k} \right] \rho \varepsilon \frac{\partial u_i}{\partial x_i} \\ &\quad + \left((C_{\varepsilon 1} - C_\eta) \frac{\partial u_i}{\partial x_j} \tau_{ij}^* - C_{\varepsilon 2} \rho \varepsilon + C_s S_s \right) \frac{\varepsilon}{k}, \end{aligned} \quad (13.17)$$

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

where

$$C_\eta = \frac{\eta(1 - \eta/\eta_0)}{1 + \beta\eta^3} \quad (13.18)$$

and

$$\tau_{ij}^* = \tau_{ij} - \frac{1}{3}\delta_{ij}\tau_{kk} = 2\mu_t \left(S_{ij} - \frac{1}{3}\delta_{ij}\frac{\partial u_k}{\partial x_k} \right). \quad (13.19)$$

The source term, S_s , in Equations 13.13, 13.14, and 13.17 is included to account for the interactions of turbulence with the discrete phase. This term is modeled as

$$S_s = -\frac{\sum_p N_p (F'_{drag,i} u'_i)_p}{V}, \quad (13.20)$$

where the summation is over all parcels in the cell, N_p is the number of drops in a parcel, V is the cell volume, u'_i is the fluctuating component of the gas-phase velocity, and

$$F'_{drag,i} = \frac{F_{drag,i}}{(u_i + u'_i - v_i)} u'_i, \quad (13.21)$$

where $F_{drag,i}$ is the drag force on a drop. It is important to note that although the c_s model constant appears in front of S_s in only the ε transport equation, setting c_s to zero in the input files actually deactivates the source term in both the k and ε equations.

To activate the Rapid Distortion RNG k - ε model, set [turbulence.in > turbulence_model = RANS_K_EPS_RNG_RD](#).

Generalized RNG k- ε

The generalized RNG k - ε model ([Wang et al., 2011](#)) reformulates the closure parameters to take into account to the dimensionality of the flow strain rate (*i.e.*, unidirectional and omnidirectional compression have closure terms with different values). These reformulated closure parameters model the impact of bulk flow compression and expansion. The transport equation for turbulent dissipation is written as

$$\rho \frac{D\varepsilon}{Dt} = \frac{\varepsilon}{k} C_1 P + C'_1 \rho \nu_t \frac{\varepsilon}{k} (\nabla \cdot u)^2 - C_{2n} \rho \frac{\varepsilon^2}{k} - \rho R_\varepsilon + C_3 \rho \varepsilon (\nabla \cdot u) + diffusion, \quad (13.22)$$

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

where the model parameters are calculated according to

$$\begin{aligned} C'_1 &= a \left(1 - \frac{2}{3} C_1 \right) \\ C_{2n} &= b_0 + b_1 n + b_2 n^2 \\ C_3 &= -\frac{n+1}{n} + \frac{2}{3} C_1 + \sqrt{\frac{2(1+a)}{3}} C_\mu C_\eta \eta (-1)^\delta, \end{aligned} \quad (13.23)$$

where C_η shares the Rapid Distortion RNG definition of Equation 13.18. In Equation 13.23, a is defined as

$$a = \frac{3(S_{11}^2 + S_{22}^2 + S_{33}^2)}{(|S_{11}| + |S_{22}| + |S_{33}|)^2} - 1, \quad (13.24)$$

and n is defined as

$$n = 3 - \sqrt{2a}. \quad (13.25)$$

Coefficients in these calculations are derived from compressible and incompressible jet flows ([Wang et al., 2013](#)) and take the following values.

Table 13.2: Model coefficients for the generalized RNG $k-\varepsilon$ model.

Parameter	Value
C_1	1.42
C_μ	0.0845
η_0	4.38
β	0.012
b_0	2.0725
b_1	-0.3865
b_2	0.083

To activate the generalized RNG $k-\varepsilon$ model, set [*turbulence.in*](#) > *turbulence_model* = *RANS_K_EPS_RNG_GEN*.

Realizable k- ε

In some simulations, such as when the mean strain rate is high, standard k - ε models can produce negative values for turbulent kinetic energy. Such a result is non-realizable (non-physical). The Realizable k - ε model in CONVERGE imposes realizability constraints to ensure the non-negativity of turbulent normal stresses and that the results do not violate Schwarz's inequality ([Shih et al., 1995](#)). As a result, the turbulent kinetic energy calculated by the Realizable k - ε model is always a physically realistic value. The Realizable k - ε model works well for rotational flows.

An important difference between the Realizable k - ε model and the Standard and RNG k - ε models is that while the expressions for turbulent viscosity are the same (see Equation 13.11), the model constant C_μ for the Realizable k - ε model varies as a function of k , ε , and U^* (friction velocity). Equation 13.26 below defines C_μ for the Realizable k - ε model:

$$C_\mu = \frac{1}{A_0 + A_s \frac{k U^*}{\varepsilon}}, \quad (13.26)$$

where A_0 and A_s are model constants and U^* is the friction velocity. The friction velocity is a function of the strain rate S_{ij} and the rotation rate $\bar{\Omega}_{ij}$, which are defined as:

$$U^* = \sqrt{S_{ij} S_{ij} + \bar{\Omega}_{ij} \bar{\Omega}_{ij}} \quad (13.27)$$

The transport equation for turbulent kinetic energy, k , in the Realizable k - ε model is the same as the transport equation for the standard and RNG models (given by Equation 13.15). For turbulent dissipation, however, the transport equation is

$$\begin{aligned} \frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial (\rho u_i \varepsilon)}{\partial x_i} &= \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] + C_1 \rho \varepsilon S \\ &- C_2 \rho \frac{\varepsilon^2}{k + \sqrt{\nu \varepsilon}} + C_{\varepsilon 3} \rho \varepsilon \frac{\partial u_i}{\partial x_i} + S_\varepsilon. \end{aligned} \quad (13.28)$$

Equation 13.28 above includes additional model constants C_2 and $C_{\varepsilon 3}$. The production term, P_b , is the same as that for the standard k - ε model. The model constants and auxiliary functions included in Equations 13.26 and 13.28 are

$$\begin{aligned}
 A_0 &= 4.0 & A_s &= \sqrt{6} \cos \phi & \phi &= \frac{1}{3} \cos^{-1}(\sqrt{6}W) \\
 W &= \frac{S_{ij}S_{jk}S_{ki}}{\tilde{S}^3} & \tilde{S} &= \sqrt{S_{ij}S_{ij}} \\
 C_2 &= 1.9 & C_{1\varepsilon} &= 1.44 & C_{3\varepsilon} &= -1 & \sigma_\varepsilon &= 1.2
 \end{aligned} \tag{13.29}$$

Analysis of experimental results from boundary layer and shear flows reveals that the constant C_1 is a function of the time scale ratio of the turbulence to the mean strain. Equation 13.30 below gives the expression for C_1 :

$$\begin{aligned}
 C_1 &= \max \left[0.43, \frac{\eta}{5+\eta} \right] \\
 \eta &= \frac{Sk}{\varepsilon} & S &= \sqrt{2S_{ij}S_{ij}}
 \end{aligned} \tag{13.30}$$

where η is the mean strain.

To activate the realizable $k-\varepsilon$ model, set [*turbulence.in*](#) > *turbulence_model* = *RANS_K_EPS_REAL*.

v2-f

The $\overline{v^2} - f$ (v2-f) model in CONVERGE is a subset of the previously described $k-\varepsilon$ models. CONVERGE features the variant described by Lien and Kalitzin (2001). As with the $k-\varepsilon$ models, the $\overline{v^2} - f$ model represents the Reynolds stress term τ_{ij} via an effective turbulent viscosity. The formulations for the Reynolds stress term and the production term P in the $\overline{v^2} - f$ model are the same as those for the standard $k-\varepsilon$ models (described in Equation 13.9).

In addition to the transport equations for k and ε , the $\overline{v^2} - f$ model solves transport equations for velocity variance normal to the streamline, $\overline{v^2}$, and elliptic relaxation function, f . By introducing the velocity variance, the $\overline{v^2} - f$ model accounts for the damping of turbulence transport near walls or other impermeable boundaries. The elliptic relaxation function f models the anisotropic wall effects. Additionally, this model represents the near-wall effects of pressure-deformation fluctuations.

To evaluate the eddy viscosity, this model uses the velocity variance. The formulation for turbulent viscosity μ_t is

$$\mu_t = C_\mu \rho \overline{v^2} T. \tag{13.31}$$

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

The transport equation for $\overline{v^2}$ in compressible form is

$$\frac{\partial \rho \overline{v^2}}{\partial t} + U_j \frac{\partial \rho \overline{v^2}}{\partial x_j} = k \rho f - \frac{\overline{v^2} \rho}{k} \varepsilon + \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\nu_t}{\sigma_k} \right) \frac{\partial \overline{v^2}}{\partial x_j} \right]. \quad (13.32)$$

For the elliptic relaxation function, f , the transport equation is

$$L^2 \frac{\partial^2 f}{\partial x_j^2} - f = \frac{1}{T} \left((C_1 - 6) \frac{\overline{v^2}}{k} - \frac{2}{3} (C_1 - 1) \right) - C_2 \frac{P_k}{\varepsilon}. \quad (13.33)$$

The turbulence length scale L (from Equation 13.33) is

$$L = C_L \max \left[\frac{k^{3/2}}{\varepsilon}, C_\eta \left(\frac{\nu^3}{\varepsilon} \right)^{1/4} \right]. \quad (13.34)$$

The formulation for the time scale, T , imposes an upper bound on the traceless portion of the production term:

$$T = \max \left[\frac{k}{\varepsilon}, 6 \left(\frac{\nu}{\varepsilon} \right)^{1/2} \right]. \quad (13.35)$$

Finally, the model coefficients present in Equations 13.32 through 13.34:

$$\begin{aligned} C_\mu &= 0.22 & C_L &= 0.23 \\ \sigma_k &= 1 & C_\eta &= 70 \\ C_1 &= 1.4 & C_2 &= 0.3 \end{aligned} \quad (13.36)$$

and

$$C_{\varepsilon 1} = 1.4 \left(1 + 0.05 \left(k / \overline{v^2} \right)^{1/2} \right). \quad (13.37)$$

This model is subject to the stagnation point anomaly—when flow impinges on a surface (e.g., near the leading edge of an airfoil), over-prediction of turbulent kinetic energy and

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

eddy viscosity may occur. To avoid this phenomenon, the $\overline{v^2} - f$ model in CONVERGE uses a realizability constraint on the production of turbulent kinetic energy and turbulent dissipation. The constraint on production is

$$\mathcal{P} \leq \frac{k|\mathbf{S}|}{\sqrt{6}} \quad (13.38)$$

Note that for this model, you do not have to explicitly supply boundary conditions for $\overline{v^2}$ and f . Instead, supply boundary conditions for k and ε and CONVERGE determines the appropriate boundary conditions from these inputs.

To activate the $\overline{v^2} - f$ model, set [*turbulence.in*](#) > *turbulence_model* = *RANS_K_EPS_V2F*.

ζ -f

As with the $\overline{v^2} - f$ model, the $\zeta - f$ (zeta-f) model in CONVERGE is a subset of the previously described $k-\varepsilon$ models. It was introduced by Hanjalic et al. (2004). The main function of the $\zeta - f$ model is to improve the computational performance of the $\overline{v^2} - f$ model, specifically with regards to sensitivity to the near-wall grid spacing. Instead of solving a transport equation for velocity variance scale ($\overline{v^2}$), this model solves a transport equation for velocity scales ratio, ζ . Additionally, this model applies a quasi-linear pressure-strain model in the f equation.

The definition of the velocity scales ratio, ζ , is

$$\zeta = \frac{\overline{v^2}}{k}. \quad (13.39)$$

The turbulent viscosity, μ_t , is

$$\mu_t = C_\mu \rho \zeta k T. \quad (13.40)$$

The transport equation for ζ is

$$\frac{\partial \rho \zeta}{\partial t} = \rho f - \frac{\zeta \rho}{k} \mathcal{P} + \frac{\partial}{\partial x_k} \left[\left(\mu + \frac{\nu_t}{\sigma_\zeta} \right) \frac{\partial \zeta}{\partial x_k} \right]. \quad (13.41)$$

Note that for the $\zeta - f$ model, CONVERGE still solves the k and ε transport equations. The ε equation, however, is modified to impose the Kolmogorov time and length scales as lower bounds on turbulence dissipation. Equation 13.42 below describes the ε equation and the time- and length scale formulations, respectively.

$$\begin{aligned} \frac{\partial \rho \varepsilon}{\partial t} &= \frac{(C_{\varepsilon 1} \mathcal{P} - C_{\varepsilon 2} \varepsilon)}{\tau} + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon} \right) \frac{\partial \varepsilon}{\partial x_j} \right] \\ T &= \max \left[\frac{k}{\varepsilon}, C_\tau \left(\frac{\nu}{\varepsilon} \right)^{1/2} \right] \\ L &= C_L \max \left[\frac{k^{3/2}}{\varepsilon}, C_\eta \left(\frac{\nu^3}{\varepsilon} \right)^{1/4} \right] \end{aligned} \quad (13.42)$$

For the elliptic relaxation function, f , the transport equation is

$$L^2 \nabla^2 f - f = \frac{1}{T} \left(c_1 + C_2' \frac{\mathcal{P}}{\varepsilon} \right) \left(\zeta - \frac{2}{3} \right). \quad (13.43)$$

Finally, the model coefficients present in Equations 13.40 through 13.42:

$$\begin{aligned} C_1 &= 0.4 & C_2' &= 0.65 \\ C_\mu &= 0.22 & C_\eta &= 85.0 \\ C_L &= 0.36 & \text{Reciprocal Pradtl} &= 0.8333 \end{aligned} \quad (13.44)$$

Like the v2-f model, this model is subject to the stagnation point anomaly—when flow impinges on a surface (e.g., near the leading edge of an airfoil), over-prediction of turbulent kinetic energy and eddy viscosity may occur. To avoid this phenomenon, this model applies a realizability constraint on the production of turbulent kinetic energy and turbulent dissipation.

Note that for this model, you do not have to explicitly supply boundary conditions for ζ and f . Instead, supply boundary conditions for k and ε and CONVERGE determines the appropriate boundary conditions from these inputs.

To activate the $\zeta - f$ model, set [*turbulence.in*](#) > *turbulence_model* = *RANS_K_EPS_ZETAf*.

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

Buoyancy Effects in k- ε Models

Buoyancy effect modeling is available with k - ε turbulence models in CONVERGE. These effects are modeled by an extra term P_b in the turbulence transport equations:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho k u_j}{\partial x_j} = \tau_{ij} \frac{\partial u_i}{\partial x_j} + P_b + \frac{\partial}{\partial x_j} \left(\frac{\mu}{Pr_k} \frac{\partial k}{\partial x_j} \right) - \rho \varepsilon + S_s \quad (13.45)$$

and

$$\begin{aligned} \frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon u_j}{\partial x_j} &= \frac{\partial}{\partial x_j} \left(\frac{\mu}{Pr_\varepsilon} \frac{\partial \varepsilon}{\partial x_j} \right) + C_{\varepsilon 3} \rho \varepsilon \frac{\partial u_k}{\partial x_k} \\ &+ \left(C_{\varepsilon 1} \tau_{ij} \frac{\partial u_i}{\partial x_j} + C_{\varepsilon 1} C_{\varepsilon b} P_b - C_{\varepsilon 2} \rho \varepsilon + C_s S_s \right) \frac{\varepsilon}{k} + \rho R_\varepsilon. \end{aligned} \quad (13.46)$$

The additional term P_b is calculated according to

$$P_b = \frac{\mu_t}{Pr_t} \left(\frac{g_i}{\rho} \frac{\partial \rho}{\partial x_i} \right), \quad (13.47)$$

where μ_t is the turbulent viscosity, Pr_t is the turbulent Prandtl number, and x_i is the gravity vector.

In Equation 13.46, the coefficient $C_{\varepsilon b}$ determines the degree to which ε is affected by buoyancy, and is calculated based on the local flow condition according to

$$C_{\varepsilon b} = \tanh \left| \frac{u_t}{u_n} \right|, \quad (13.48)$$

where u_t is the magnitude of the velocity component that is parallel to the gravity direction and u_n is the magnitude of the velocity component that is perpendicular to the gravity direction.

In most flows, buoyancy effects are negligible, and you do not need to direct CONVERGE to account for buoyancy effects. For cases like natural convection in which buoyancy is important, you can activate the buoyancy model by setting [*turbulence.in* > *Physics_effects* > *buoyancy_flag*](#). Since the buoyancy effects on k are well understood but the effects on ε are less clear, CONVERGE provides two options for *buoyancy_flag*. For *buoyancy_flag* = 1, the P_b

Chapter 13: Turbulence Modeling

RANS Models k- ε Models

term is applied only to Equation 13.45, the k transport equation. For *buoyancy_flag* = 2, the P_b term is applied to both Equations 13.45 and 13.46, the ε transport equation.

CONVERGE accounts for the buoyancy effects based on [inputs.in > property_control > gravity](#). The buoyancy model is available only for k - ε turbulence models.

k- ω Models

As with the RANS k - ε models, each of the k - ω models in CONVERGE uses an effective turbulent viscosity to model the Reynolds stress term τ_{ij} . The expression for the Reynolds stress used in the k - ω models is

$$\tau_{ij} = \mu_t \left(2S_{ij} - \frac{2}{3}\delta_{ij}\frac{\partial \tilde{u}_k}{\partial x_k} \right) - \frac{2}{3}\rho k \delta_{ij}. \quad (13.49)$$

Each k - ω model uses a different formulation for the effective turbulent viscosity (μ_t). To obtain the turbulent viscosity, CONVERGE must solve two additional transport equations: one for the transport of turbulent kinetic energy k and one for the transport of specific dissipation rate ω . All of the k - ω models in CONVERGE use the same definition of turbulent kinetic energy as the RANS k - ε models (see Equation 13.11). The following sections list the formulations of ω , the transport equations for k and ω , and the associated model constants for each k - ω model.

For all two-equation k - ω models, the turbulence length scale le is given by

$$le = \frac{k^{1/2}}{C_\mu^{1/4} \omega}. \quad (13.50)$$

Standard k- ω (1998)

The k - ω model of [Wilcox \(1998\)](#) defines the turbulent viscosity as

$$\mu_t = \frac{\rho k}{\omega}. \quad (13.51)$$

The transport equation for k is

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_j k}{\partial x_j} = P - \beta^* \rho \omega k + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_k \mu_t) \frac{\partial k}{\partial x_j} \right] \quad (13.52)$$

and the transport equation for ω is

Chapter 13: Turbulence Modeling

RANS Models k- ω Models

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} = \frac{\alpha \omega}{k} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right]. \quad (13.53)$$

The equations below define the mean strain rate tensor S_{ij} and the production term P :

$$S_{ij} = \frac{1}{2} \left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} \right) \quad (13.54)$$

$$P = \tau_{ij} \frac{\partial \tilde{u}_i}{\partial x_j}.$$

The model constants included in Equations 13.52 and 13.53 are

$$\sigma_k = 0.5 \quad \sigma_\omega = 0.5 \quad \alpha = \frac{13}{25}$$

$$\beta^* = \beta_0 f_{\beta^*} \quad \beta_0^* = 0.09$$

$$\beta = \beta_0 f_\beta \quad \beta_0 = \frac{9}{125} \quad (13.55)$$

$$f_{\beta^*} = \begin{cases} 1 & \text{if } \chi_k \leq 0 \\ \frac{1+680\chi_k^2}{1+400\chi_k^2} & \text{if } \chi_k > 0 \end{cases} \quad \chi_k = \frac{1}{\omega^3} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}$$

$$f_\beta = \frac{1+70\chi_\omega}{1+80\chi_\omega} \quad \chi_\omega = \left| \frac{\Omega_{ij} \Omega_{jk} S_{ki}}{(\beta_0^* \omega)^3} \right| \quad \Omega_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right).$$

Finally, the equation for the model constant α is

$$\alpha = \frac{\beta_0}{\beta_0^*} - \frac{\sigma_\omega \kappa^2}{\sqrt{\beta_0^*}}. \quad (13.56)$$

Chapter 13: Turbulence Modeling

RANS Models k- ω Models

Note that this model uses a value of α chosen to yield an appropriate value for von Karman's constant ($\kappa \approx 0.41$) via the expression in 13.56.

To activate the standard $k-\omega$ model (1998), set [*turbulence.in*](#) > *turbulence_model* = *RANS_K_OMEGA_STD_98*.

Standard $k-\omega$ (2006)

The $k-\omega$ model of [Wilcox \(2006\)](#) defines turbulent viscosity as

$$\mu_t = \frac{\rho k}{\hat{\omega}}, \quad (13.57)$$

where

$$\hat{\omega} = \max \left[\omega, C_{\text{lim}} \sqrt{\frac{2S_{ij}^* S_{ij}^*}{\beta^*}} \right] \quad (13.58)$$

and

$$S_{ij}^* = S_{ij} - \frac{1}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij}. \quad (13.59)$$

The transport equation for k is the same as that in the 1998 model (given by Equation 13.52). The transport equation for ω , however, includes an additional term:

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} = \frac{\alpha \omega}{k} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[\left(\mu + \sigma_\omega \frac{\rho k}{\omega} \right) \frac{\partial \omega}{\partial x_j} \right] + \frac{\rho \sigma_d}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, \quad (13.60)$$

where Equation 13.55 gives the expressions for the mean strain rate tensor and the production term.

Finally, the model constants and auxiliary functions included in Equations 13.52 and 13.60 are

Chapter 13: Turbulence Modeling

RANS Models k- ω Models

$$\sigma_k = 0.6$$

$$\sigma_\omega = 0.5$$

$$\alpha = \frac{13}{25}$$

$$\beta^* = 0.09$$

$$C_{\text{lim}} = \frac{7}{8}$$

$$\beta = \beta_0 f_\beta$$

$$\beta_0 = 0.0708$$

(13.61)

$$f_\beta = \frac{1+85\chi_\omega}{1+100\chi_\omega}$$

$$\chi_\omega = \left| \frac{\Omega_{ij}\Omega_{jk}\hat{S}_{ki}}{\left(\beta_0^*\omega\right)^3} \right|$$

$$\hat{S}_{ki} = S_{ki} - \frac{1}{2} \frac{\partial u_m}{\partial x_m} \delta_{ki}$$

$$\sigma_a = \begin{cases} 0 & \text{if } \chi_k \leq 0 \\ 0.125 & \text{if } \chi_k > 0 \end{cases} \quad \chi_k = \frac{1}{\omega^3} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}.$$

As with the [k- \$\omega\$ \(1998\)](#) model, the equation for α is

$$\alpha = \frac{\beta_0}{\beta_0^*} - \frac{\sigma_\omega \kappa^2}{\sqrt{\beta_0}} \quad (13.62)$$

and the α coefficient yields an appropriate value for von Karman's constant ($\kappa \approx 0.41$) via the expression in Equation 13.62.

To activate the standard $k-\omega$ model (2006), set [`turbulence.in`](#) > `turbulence_model` = `RANS_K_OMEGA_STD`.

Low-Reynolds Number Correction

The standard $k-\omega$ model (2006) can be combined with a low-Reynolds number correction, also proposed by [Wilcox \(2006\)](#). The low-Reynolds number correction alters the model constants and auxiliary functions of Equation 13.61 as

Chapter 13: Turbulence Modeling

RANS Models k- ω Models

$$\begin{aligned}\beta^* &= 0.09 \left(\frac{100\beta_0 / 27 + (\text{Re}_T / R_\beta)^4}{1 + (\text{Re}_T / R_\beta)^4} \right) & \alpha &= \frac{13}{25} \left(\frac{\alpha_0 + \text{Re}_T / R_\omega}{1 + \text{Re}_T / R_\omega} \right) \\ \mu_t &= \alpha^* \frac{\rho k}{\hat{\omega}} & \hat{\omega} &= \max \left(\omega, C_{lim} \sqrt{\frac{2\bar{S}_{ij}\bar{S}_{ij}}{\beta_0^* / \alpha^*}} \right) \\ \text{Re}_T &= \frac{\rho k}{\mu \omega} & \alpha^* &= \frac{\alpha_0^* + \text{Re}_T / R_k}{1 + \text{Re}_T / R_k}\end{aligned}\quad (13.63)$$

where

$$\begin{aligned}R_\beta &= 8 & R_k &= 6 & R_\omega &= 2.61 \\ \alpha_0 &= \frac{1}{9} & \alpha_0^* &= \frac{\beta_0}{3} & \beta_0^* &= 0.09.\end{aligned}\quad (13.64)$$

The transport equations for k and ω are replaced with

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_j k}{\partial x_j} = P - \beta^* \rho \omega k + \frac{\partial}{\partial x_j} \left[\left(\mu + \sigma_k \alpha^* \mu_t \right) \frac{\partial k}{\partial x_j} \right] \quad (13.65)$$

and

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} = \frac{\alpha \omega}{k} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[\left(\mu + \sigma_\omega \alpha^* \frac{\rho k}{\omega} \right) \frac{\partial \omega}{\partial x_j} \right] + \frac{\rho \sigma_d}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}. \quad (13.66)$$

The low-Reynolds number correction is not recommended for general use with the $k-\omega$ 2006 model.

The correction can be applied to the $k-\omega$ 1998 and $k-\omega$ SST models, but because the correction was designed for the 2006 model, these two combinations are never recommended.

k- ω SST

The RANS $k-\omega$ shear stress transport (SST) model combines the advantages of a standard $k-\omega$ model and a standard $k-\varepsilon$ model ([Menter et al., 2003](#)). In general, the $k-\omega$ SST model performs well when simulating external flows but is not used for simulations involving combustion.

The $k-\omega$ SST model in CONVERGE expresses the turbulent viscosity as

Chapter 13: Turbulence Modeling

RANS Models k- ω Models

$$\mu_t = \frac{\rho a_1 k}{\max[a_1 \omega, SF_2]}' \quad (13.67)$$

where

$$S = \sqrt{2S_{ij}S_{ij}}. \quad (13.68)$$

Like the [k- \$\omega\$ \(2006\)](#) model, the transport equation for k is the same as that in the 1998 model (given by Equation 13.57). The transport equation for ω is

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} = \frac{\alpha}{\nu_t} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + 2(1 - F_1) \frac{\rho \sigma_{\omega 2}}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}. \quad (13.69)$$

The expression for the mean strain rate tensor S_{ij} is the same as that in Equation 13.58. The expression for the production term P , however, differs from that used in the $k-\omega$ (1998) and $k-\omega$ (2006) models. Equation 13.70 below defines the production term for the $k-\omega$ SST model:

$$P = \min \left[\tau_{ij} \frac{\partial u_i}{\partial x_j}, 10\beta^* \rho \omega k \right]. \quad (13.70)$$

Each constant in the transport equations for k and ω (such as σ_k) is a blend of inner (denoted by the subscript 1) and outer (denoted by the subscript 2) constants. For some constant ϕ , Equation 13.71 below defines the blending function:

$$\phi = F_1 \phi_1 + (1 - F_1) \phi_2. \quad (13.71)$$

CONVERGE calculates F_1 and F_2 using the expressions below:

$$F_1 = \tanh \left(\min \left[\max \left[\frac{\sqrt{k}}{\beta^* \omega d}, \frac{500\nu}{d^2 \omega} \right], \frac{4\rho \sigma_{\omega 2} k}{CD_{kw} d^2} \right]^4 \right) \\ F_2 = \tanh \left(\max \left[2 \frac{\sqrt{k}}{\beta^* \omega d}, \frac{500\nu}{d^2 \omega} \right]^2 \right), \quad (13.72)$$

where

Chapter 13: Turbulence Modeling

RANS Models k- ω Models

$$CD_{k\omega} = \max \left[2\rho\sigma_{\omega^2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 10^{-10} \right]. \quad (13.73)$$

Finally, Equation 13.74 below defines the inner and outer model constants and parameters used in the $k-\omega$ SST model:

$$\begin{aligned} \alpha_1 &= \frac{5}{9} & \alpha_2 &= 0.44 \\ \sigma_{k1} &= 0.85 & \sigma_{\omega 1} &= 0.500 & \beta_1 &= 0.075 \\ \sigma_{k2} &= 1.00 & \sigma_{\omega 2} &= 0.856 & \beta_2 &= 0.0828 \\ \beta^* &= 0.09 & \kappa &= 0.41 & a_1 &= 0.31. \end{aligned} \quad (13.74)$$

To activate the standard $k-\omega$ model (1998), set [*turbulence.in*](#) > *turbulence_model* = *RANS_K_OMEGA_SST*.

Reynolds Stress Models

The Reynolds stress models (RSM) in CONVERGE solve the model transport equations for the Reynolds stresses $R_{ij} = u_i u_j$ themselves. In addition, these models solve an equation for a quantity that provides either a length or time scale for the turbulence (usually ε or ω [[Hanjalic and Launder, 2011](#)]). This approach is more direct than that employed by the turbulent viscosity models discussed previously.

Since the RSM approach obviates the use of the isotropic turbulent-viscosity hypothesis, the RSM approach usually is more suitable for simulating complex turbulent flows involving strong anisotropy (e.g., flows with mean streamline curvature, strong swirl, or mean rotation; secondary flows in ducts; and flows with rapid variations in the mean flow [[Pope, 2010](#)]). For a fully three-dimensional flow, CONVERGE solves an independent equation for each of the six Reynolds stresses and for the dissipation rate.

It is important to note that even though the transport equations for the Reynolds stresses can be written in exact form, some of the terms in those equations are unknown and need to be modeled in order to ensure closure of the system. This requirement leads to certain situations in which the RSM approach might not give results that are superior to the simpler (and less computationally expensive) turbulent-viscosity models. It is, however, imperative that flows that involve significant Reynolds-stress anisotropy be solved using the RSM approach.

Chapter 13: Turbulence Modeling

RANS Models Reynolds Stress Models

We obtain the exact transport equations for the Reynolds stresses from the Navier-Stokes equations and write the Reynolds stress transport equations as

$$\frac{\partial \widetilde{\rho u'_i u'_j}}{\partial t} + \frac{\partial u_k \widetilde{\rho u'_i u'_j}}{\partial x_k} = - \underbrace{\frac{\partial}{\partial x_k} \left(\widetilde{\rho u'_i u'_j u'_k} + p \left(\delta_{kj} u'_i + \delta_{ik} u'_j \right) \right)}_{D_{T,ij}} + \underbrace{\frac{\partial}{\partial x_k} \left(\mu \frac{\partial}{\partial x_k} \left(\widetilde{u'_i u'_j} \right) \right)}_{D_{M,ij}} \\ - \underbrace{\overline{\rho} \left(\widetilde{u'_i u'_k} \frac{\partial u_j}{\partial x_k} + \widetilde{u'_j u'_k} \frac{\partial u_i}{\partial x_k} \right)}_{P_{ij}} + \underbrace{p \left(\frac{\partial u'_i}{\partial x_j} + \frac{\partial u'_j}{\partial x_i} \right)}_{\phi_{ij}} - \underbrace{2\mu \frac{\partial u'_i}{\partial x_k} \frac{\partial u'_j}{\partial x_k}}_{\varepsilon_{ij}}, \quad (13.75)$$

where C_{ij} represents the convective term, $D_{T,ij}$ is the turbulent diffusion term, $D_{M,ij}$ is the molecular diffusion term, P_{ij} is the stress production term, ϕ_{ij} is the pressure-strain term, and ε_{ij} is the dissipation term. We must model the turbulent diffusion ($D_{T,ij}$), pressure-rate-of-strain (ϕ_{ij}), and the dissipation (ε_{ij}) terms in order to close the equations, while the rest of the terms do not require modeling. The following sections discuss the modeling assumptions used for evaluating $D_{T,ij}$, ϕ_{ij} , and ε_{ij} .

Modeling the Turbulent Diffusion Term

Various studies indicate that the turbulent diffusion term is in general smaller compared to the other terms. Also, within the turbulent diffusion term, the contribution of the pressure-

velocity correlation, $\widetilde{p \left(\delta_{kj} u'_i + \delta_{ik} u'_j \right)}$, is small enough to safely neglect [2, 3]. We then model the triple velocity correlation, $\widetilde{u'_i u'_j u'_k}$, using the eddy viscosity hypothesis ([Lien and Leschziner, 1996](#)). This approach adopts the same isotropic diffusivity concept as does the standard $k-\varepsilon$ model. Equation 13.76 gives the equation for this model:

$$D_{T,ij} = \frac{\partial}{\partial x_k} \left(\frac{\mu + \mu_t}{\sigma_k} \frac{\partial \left(\widetilde{u'_i u'_j} \right)}{\partial x_k} \right), \quad (13.76)$$

where, in the effective turbulent viscosity $\mu_t = \overline{\rho} C_\mu \left(k^2 / \varepsilon \right)$, k^2 is the turbulent kinetic energy. Note that k can also be expressed in terms of the trace of the Reynolds stress tensor:

$$k = \frac{1}{2} R_{ii}. \quad (13.77)$$

Chapter 13: Turbulence Modeling

RANS Models Reynolds Stress Models

Table 13.3 gives the values of the constants σ_k and C_μ . The EVH model assumes isotropic stress-diffusion and leads to a more numerically stable implementation compared to other models for the turbulent diffusion term ([Siow, 2003](#)).

Modeling the Pressure-Rate-of-Strain Term

The pressure-rate-of-strain term ϕ_{ij} serves to redistribute energy among the Reynolds stresses and is generally considered to be the most important in terms of modeling. CONVERGE uses the SSG model ([Speziale et al., 1991](#)) or the LRR (Launder-Reece-Rodi [formulation of [Gibson and Launder, 1975](#)]) model to evaluate ϕ_{ij} .

The SSG model calculates ϕ_{ij} as

$$\begin{aligned} \phi_{ij} = & -\bar{\rho} \left(C_1 \varepsilon + C_1^* P_k \right) b_{ij} + C_2 \bar{\rho} \varepsilon \left(b_{ik} b_{kj} - \frac{1}{3} \delta_{ij} b_{kl} b_{kl} \right) + \bar{\rho} \left(C_3 - C_3^* \sqrt{b_{kl} b_{kl}} \right) k S_{ij}^* \\ & + C_4 \bar{\rho} k \left(b_{ik} S_{jk} + b_{jk} S_{ik} - \frac{2}{3} \delta_{ij} b_{kl} S_{kl} \right) + C_5 \bar{\rho} k \left(b_{ik} W_{jk} + b_{jk} W_{ik} \right) \end{aligned}, \quad (13.78)$$

where the Reynolds-stress anisotropy tensor, b_{ij} , is

$$b_{ij} = 0.5 \left(\frac{R_{ij}}{k} - \frac{2}{3} \delta_{ij} \right), \quad (13.79)$$

the strain rate tensor, S_{ij}^* , is

$$\begin{aligned} S_{ij}^* &= S_{ij} - \frac{1}{3} S_{kk} \delta_{ij}, \\ S_{ij} &= \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \end{aligned} \quad (13.80)$$

and W_{ij} is the mean rate of rotation term,

$$W_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i} \right). \quad (13.81)$$

The LRR model calculates ϕ_{ij} as the sum of four terms:

Chapter 13: Turbulence Modeling

RANS Models Reynolds Stress Models

$$\phi_{ij} = \phi_{ij1} + \phi_{ij2} + \phi_{ijw1} + \phi_{ijw2}. \quad (13.82)$$

The first term, known as the slow (return to isotropy) term, is calculated according to the proposal of [Rotta \(1951\)](#):

$$\phi_{ij1} = -C_1 \frac{\varepsilon}{k} \left(\overline{u_i' u_j'} - \frac{2}{3} \delta_{ij} k \right). \quad (13.83)$$

The second term, known as the fast term, is the second term of the LRR model:

$$\phi_{ij2} = -C_2 \left(P_{ij} - \frac{2}{3} \delta_{ij} P_{kk} \right). \quad (13.84)$$

The third and fourth terms are wall reflection terms. There is a corresponding wall reflection term for each of the slow term and the fast term. The wall reflection term ϕ_{ijw1} corresponds to the slow term, and is calculated as proposed by [Shir \(1973\)](#):

$$\phi_{ijw1} = C_{1w} \frac{\rho \varepsilon}{k} \left(\overline{u_k' u_m'} n_k n_m \delta_{ij} - \frac{3}{2} \overline{u_i' u_j'} n_i n_k - \frac{3}{2} \overline{u_j' u_k'} n_i n_k \right) \left(\frac{k^{3/2}}{2.5 \varepsilon d_w} \right), \quad (13.85)$$

where n_i is the unit vector normal to the wall and d_w is the distance to the wall.

There are two models available for the second (fast) wall reflection term. The first was proposed by Gibson and Launder and has the form

$$\phi_{ijw2} = C_{2w} \left(\phi_{km2} n_k n_m \delta_{ij} - \frac{3}{2} \phi_{ik2} n_i n_k - \frac{3}{2} \phi_{jk2} n_j n_k \right) \left(\frac{k^{3/2}}{2.5 \varepsilon d_w} \right). \quad (13.86)$$

The second model was proposed by [Craft and Launder \(1992\)](#) and has the form

Chapter 13: Turbulence Modeling

RANS Models Reynolds Stress Models

$$\begin{aligned}\phi_{ijw2} = & -0.08 \frac{\partial \bar{u}_k}{\partial x_m} \bar{u}'_k \bar{u}'_m (\delta_{ij} - 3n_i n_j) \left(\frac{k^{3/2}}{2.5 \varepsilon d_w} \right) \\ & -0.1k a_{km} \left(\frac{\partial \bar{u}_s}{\partial x_m} n_k n_s \delta_{ij} - \frac{3}{2} \frac{\partial \bar{u}_i}{\partial x_m} n_k n_j - \frac{3}{2} \frac{\partial \bar{u}_j}{\partial x_m} n_k n_i \right) \left(\frac{k^{3/2}}{2.5 \varepsilon d_w} \right) \\ & + 0.4k \frac{\partial \bar{u}_k}{\partial x_m} n_k n_m \left(n_i n_j - \frac{1}{3} \delta_{ij} \right) \left(\frac{k^{3/2}}{2.5 \varepsilon d_w} \right),\end{aligned}\quad (13.87)$$

where

$$a_{ij} = \frac{R_{ij}}{k} - \frac{2}{3} \delta_{ij}. \quad (13.88)$$

Table 13.3 lists the values of the various constants for the SSG model.

Table 13.3: SSG model constants.

Model constant	Value
σ_k	1.0
C_μ (SSG)	0.0986
C_1	3.4
C_1^*	1.8
C_2	4.2
C_3	0.80
C_3^*	1.3
C_4	1.25
C_5	0.4

To activate the SSG model, set [turbulence.in](#) > *turbulence_model* = *RANS_RSM_SSG_EPS*.

Table 13.4 lists the values of the various constants for the LRR model.

Table 13.4: LRR model constants.

Model constant	Value
σ_k	1.0
C_μ	0.0655

Chapter 13: Turbulence Modeling

RANS Models Reynolds Stress Models

Model constant	Value
C_1	1.8
C_2	0.6
C_{1w}	0.5
C_{2w}	0.3

To activate the LRR model, set [*turbulence.in*](#) > *turbulence_model* = *RANS_RSM_LRR_EPS*.

Modeling the Dissipation Term

We calculate the dissipation rate ε_{ij} according to

$$\varepsilon_{ij} = \frac{2}{3} \delta_{ij} \varepsilon, \quad (13.89)$$

where we model ε as:

$$\begin{aligned} \frac{\partial(\bar{\rho}\varepsilon)}{\partial t} + \frac{\partial(\bar{\rho}\bar{u}_k\varepsilon)}{\partial x_k} = & -C_{\varepsilon 1} \frac{\varepsilon}{k} \bar{\rho} \bar{u}'_i \bar{u}'_j \frac{\partial \bar{u}_i}{\partial x_j} - C_{\varepsilon 2} \bar{\rho} \frac{\varepsilon^2}{k} + \\ & \frac{\partial}{\partial x_j} \left[\frac{\mu_t}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial x_j} \right] + C_{\varepsilon 3} \bar{\rho} \varepsilon \frac{\partial \bar{u}_i}{\partial x_i}. \end{aligned} \quad (13.90)$$

This form of the dissipation equation accounts for the length scale changes due to velocity dilation ([Amsden et al., 1989](#)). Table 13.5 lists the values of the constants in this dissipation equation.

Table 13.5: Dissipation term model constants.

Model constant	Value
$C_{\varepsilon 1}$	1.44
$C_{\varepsilon 2}$ (SSG)	1.83
$C_{\varepsilon 2}$ (LRR)	1.92
$C_{\varepsilon 3}$	-1.96
σ_ε	1.3

Spalart-Allmaras Model

The Spalart-Allmaras (SA) model ([Spalart and Allmaras, 1994](#)) is a one-equation RANS model developed initially for external aerodynamics. In SA, the Reynolds stress is modeled according to

$$\tau_{ij} = -\overline{\rho \widetilde{u'_i u'_j}} = 2\mu_t S_{ij}, \quad (13.91)$$

where the turbulent viscosity μ_t is a function of a transported working variable $\tilde{\nu}$ and parameters derived from the distance to the nearest wall. The turbulent viscosity is calculated as

$$\mu_t = \rho \tilde{\nu} f_{v1}. \quad (13.92)$$

The working variable $\tilde{\nu}$ is transported according to

$$\frac{\partial \tilde{\nu}}{\partial t} + u_j \frac{\partial \tilde{\nu}}{\partial x_j} = c_{b1} \tilde{S} \tilde{\nu} - c_{w1} f_w \left(\frac{\tilde{\nu}}{d} \right)^2 + \frac{1}{\sigma} \left(\frac{\partial}{\partial x_j} \left[(\nu + \tilde{\nu}) \frac{\partial \tilde{\nu}}{\partial x_j} \right] + c_{b2} \frac{\partial \tilde{\nu}}{\partial x_i} \frac{\partial \tilde{\nu}}{\partial x_i} \right), \quad (13.93)$$

where the c terms are model constants and the f terms are functions of the distance to the wall. The latter terms are calculated according to

$$\begin{aligned} f_{v1} &= \frac{\chi^3}{\chi^3 + c_{v1}^3}, \\ \chi &= \frac{\tilde{\nu}}{\nu}, \\ \tilde{S} &= S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \\ f_{v2} &= 1 - \frac{\chi}{1 + \chi f_{v1}}, \\ f_w &= g \left(\frac{1 + c_{w3}^6}{g^6 + c_{we}^6} \right)^{1/6}, \\ g &= r + c_{w2} (r^6 - r), \\ r &= \frac{\tilde{\nu}}{\tilde{S} \kappa^2 d^2}, \end{aligned} \quad (13.94)$$

where S is the magnitude of the vorticity and d is the distance to the nearest wall. CONVERGE applies a limit to the working variable \tilde{S} to enhance numerical stability,

$$\tilde{S} = \max(\tilde{S}, C_s \Omega)$$

The CONVERGE implementation of the SA model omits trip functions, which act as source terms of \tilde{v} in the original formulation. The default values of the model parameters for this turbulence model are presented in Table 13.6. We do not recommend altering the values of these parameters.

Table 13.6: Model parameter values for the Spalart-Allmaras turbulence model.

Model parameter	Parameter name in turbulence.in	Value
c_{b1}	<code>RANS_models > spalart_allmaras_model > sa_cb1</code>	0.1355
σ	<code>RANS_models > spalart_allmaras_model > sa_rpr_sigma</code>	2/3 (prescribed with a reciprocal value in turbulence.in)
c_{b2}	<code>RANS_models > spalart_allmaras_model > sa_cb2</code>	0.622
κ	<code>Wall_modeling > law_kappa</code>	0.42
c_{w2}	<code>RANS_models > spalart_allmaras_model > sa_cw2</code>	0.3
c_{w3}	<code>RANS_models > spalart_allmaras_model > sa_cw3</code>	2
c_{v1}	<code>RANS_models > spalart_allmaras_model > sa_cv1</code>	7.1
c_s	<code>RANS_models > spalart_allmaras_model > sa_cs</code>	0.3
c_w	(Derived from other parameters)	$\frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}$

RANS Boundary Conditions

When you activate a k - ϵ turbulence model, you must specify boundary conditions for the turbulent kinetic energy and turbulent dissipation equations. When you activate a k - ω turbulence model, you must specify boundary conditions for the turbulent kinetic energy and specific dissipation rate equations. The following sections describe these boundary conditions.

Note that some RANS models employ the [law-of-the-wall model](#) to calculate quantities within the boundary layer.

k- ϵ Boundary Conditions

Turbulent Kinetic Energy Boundary Conditions

There are three types of INFLOW/OUTFLOW boundary conditions available for the turbulent kinetic energy (tke) equation: Dirichlet, turbulence intensity, and Neumann. The Dirichlet and Neumann boundary conditions are not unique to the turbulence models and are explained in detail in [Chapter 9 - Boundaries and Boundary Conditions](#).

To use a turbulence intensity boundary condition, set `boundary.in > boundary_conditions > boundary > turbulence > tke > type = INTENSITY`. The turbulence intensity is a special case of a Dirichlet boundary condition. For the turbulence intensity condition, the boundary tke is given as

$$k = \frac{3}{2} u_i^2 I^2, \quad (13.95)$$

where k is the turbulent kinetic energy and I is the turbulence intensity specified in `boundary.in > boundary_conditions > boundary > turbulence > tke > value`. The turbulence intensity is usually set to a value between 0.01 and 0.10.

For a WALL boundary type there is only one valid boundary condition for the turbulent kinetic energy: Neumann. Thus, the WALL boundary condition for the turbulent kinetic energy equation is given as

$$\frac{\partial k}{\partial n} = 0.0, \quad (13.96)$$

where k is the tke and n is the wall normal vector.

Turbulent Dissipation Boundary Conditions

There are three types of INFLOW/OUTFLOW boundary conditions available for the turbulent dissipation (ϵ) equation: Dirichlet (di), Neumann (ne), and turbulence length scale (le). The Dirichlet and Neumann boundary conditions are not unique to the turbulence models and are explained in detail in [Chapter 9 - Boundaries and Boundary Conditions](#).

The turbulence length scale is a special case of a Dirichlet boundary condition. For this case, the boundary turbulent dissipation is given as

$$\epsilon = \frac{c_\mu^{3/4} k^{3/2}}{le}, \quad (13.97)$$

where c_μ is a model constant (usually 0.09), k is the turbulent kinetic energy, and le is the turbulent length scale. The length scale can sometimes be estimated from a physical

dimension in the domain. For example, while simulating flow in a duct, the length scale could be set to a fraction of an intake duct diameter.

For a WALL boundary type, there are two types of boundary conditions available for turbulent dissipation (Neumann and Dirichlet). The Dirichlet boundary condition is unique to the turbulent dissipation equation. The ε Dirichlet WALL boundary condition is

$$\varepsilon = \frac{c_\mu^{0.75} k^{1.5}}{\kappa y}, \quad (13.98)$$

where ε is the turbulent dissipation in the center of the near wall cell (not at the wall surface), y is the distance from the wall to the middle of the cell, c_μ is a turbulence model constant, and κ is von Karman's constant.

Wall Treatments

In some simulations, the grid density required to resolve the viscous sublayer may be prohibitively expensive. CONVERGE can model the under-resolved viscous sublayer using one of several wall treatments for the turbulent dissipation ε . Specify a wall treatment in [*turbulence.in*](#) > *Wall_modeling* > *near_wall_treatment*. If you want to use a different wall treatment for a specific boundary, specify that value in [*boundary.in*](#) > *boundary_conditions* > *boundary* > *turbulence* > *near_wall_treatment*.

Standard Wall Treatment

The standard wall treatment (*near_wall_treatment* = STANDARD) makes use of the law-of-the-wall assumption for velocity in the log-law region of a turbulent boundary layer. To calculate the value of specific dissipation rate at the cell centroid (ε_p) of a cell adjacent to a solid wall, CONVERGE solves

$$\varepsilon_p = \frac{C_\mu^{3/4} k_p^{3/2}}{\kappa y_p}, \quad (13.99)$$

where y_p is the distance from the wall to the cell centroid. Note that the standard wall function assumes that the cell adjacent to a wall lies in the log-law region of the boundary layer.

Scalable Wall Treatment

The scalable wall treatment (*near_wall_treatment* = SCALABLE) relaxes the log-law assumption of the standard wall function. If the cell adjacent to the wall lies in the buffer region of the boundary layer (below the log layer), the influence of the wall function is scaled down to recover the proper boundary layer behavior. CONVERGE solves the same equation for ε_p , but replaces the cell centroid distance y_p with a scaled distance,

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$\varepsilon_p = \frac{C_\mu^{3/4} k_p^{3/2}}{\kappa y_p^*}, \quad (13.100)$$

$$y_p^* = \text{MAX}(y_p, y_p^{\text{lim}}),$$

where $y_p^{\text{lim}} = 11.05$. With this formulation, if the first cell centroid distance is within the log-law region of the boundary layer, the scalable wall function identically reproduces the behavior of the standard wall function.

Non-Equilibrium Wall Treatment

The non-equilibrium wall treatment (*near_wall_treatment = NON_EQUILIBRIUM*) of [Kim and Choudhury \(1995\)](#) is a more complex model that does not assume an equilibrium boundary layer. This model is suitable for flows that have strong pressure gradients in the neighborhood of the boundary layer. The model assumes that the wall-neighboring cells have a velocity profile that is comprised of a viscous sublayer and a log-law turbulent layer. CONVERGE uses this two-layer approach to compute the turbulent kinetic energy budget in the wall-neighboring cells. Mean temperature and species mass fraction are solved with the standard wall function.

The generalized log-law for mean velocity is written as

$$\frac{\tilde{U} C_\mu^{1/4} k^{1/2}}{\tau_w / \rho} = \frac{1}{\kappa} \ln \left(E \frac{\rho C_\mu^{1/4} k^{1/2} y}{\mu} \right), \quad (13.101)$$

where

$$\tilde{U} = U - \frac{1}{2} \frac{dp}{dx} \left(\frac{y_v}{\rho \kappa \sqrt{k}} \ln \left[\frac{y}{y_v} \right] + \frac{y - y_v}{\rho \kappa \sqrt{k}} + \frac{y_v^2}{\mu} \right) \quad (13.102)$$

and where y_v is the thickness of the viscous sublayer, which is calculated from

$$y_v = \frac{\mu y_v^*}{\rho C_\mu^{1/4} k_p^{1/2}}, \quad (13.103)$$

where y_v^* is equal to 11.05. Model parameters and definitions take the following forms:

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$\begin{aligned}\tau_t &= \tau_w, \\ k &= k_p, \\ \epsilon &= \frac{k^{3/2}}{C_l^* y},\end{aligned}\tag{13.104}$$

where $C_l^* = \kappa C_\mu^{-3/4}$.

Enhanced Wall Treatment

The enhanced wall treatment (*near_wall_treatment* = ENHANCED) is a two-layer approach that blends a modified law-of-the-wall for the log layer with an expression for the viscous sublayer. The resulting wall function is insensitive to near-wall grid spacing, so the first cell from the wall can be in the viscous sublayer, in the log-law region, or in the buffer region between them without introducing excessive errors. The enhanced wall treatment also accounts for pressure gradient, heat transfer, and compressibility effects in the momentum equation.

The compressible log-layer law-of-the-wall with heat transfer and pressure gradient ([White and Christoph, 1972](#)) is written

$$\frac{\partial u_l^+}{\partial y^+} = \frac{1}{\kappa y^+} (1 + \alpha y^+)^{1/2} \left(1 + \beta u_l^+ - \gamma [u_l^+]^2 \right)^{1/2},\tag{13.105}$$

where α (the pressure gradient parameter), β (the heat transfer parameter), and γ (the compressibility parameter) are given as

$$\alpha = \frac{\nu_w}{\tau_w u^*} \frac{dp}{dx},\tag{13.106}$$

$$\beta = \frac{q_w \nu_w}{T_w k_w v^*},\tag{13.107}$$

and

$$\gamma = \frac{\sigma_i (u^*)^2}{2 C_p T_w},\tag{13.108}$$

where k_w is the thermal conductivity and v^* is the shear speed.

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

The analytic solution to Equation 13.105 for initial conditions $(u_{l,0}^+, y_0^+)$ is given by

$$u_l^+ = \frac{1}{2\gamma} \left[\beta + Q \sin \left(\phi + \frac{\gamma^{1/2}}{\kappa} \left[2(S - S_0) + \ln \left(\frac{(S-1)(S_0+1)}{(S+1)(S_0-1)} \right) \right] \right) \right], \quad (13.109)$$

where

$$Q = (\beta^2 + 4\gamma)^{1/2}, \quad (13.110)$$

$$\phi = \sin^{-1} \left(\frac{2\gamma u_{l,0}^+ - \beta}{Q} \right), \quad (13.111)$$

and

$$S = (1 + \alpha y^+)^{1/2}, \quad S_0 = (1 + \alpha y_0^+)^{1/2}. \quad (13.112)$$

Within the viscous sublayer, pressure effects are neglected, and the velocity is given as

$$u_{visc}^+ = y^+. \quad (13.113)$$

The enhanced wall treatment blends Equations 13.109 and 13.113 based on a blending coefficient Γ (Kader, 1981), which is calculated as

$$\Gamma = \frac{0.01(y^+)^4}{1 + 5y^+}. \quad (13.114)$$

The final velocity equation near the wall is expressed as

$$u^+ = \exp(-\Gamma)u_{visc}^+ + \exp(-1/\Gamma)u_l^+. \quad (13.115)$$

The wall shear stress takes the same form as in the standard wall treatment, given by

$$\tau_w = \frac{\rho u^* U_p}{u^+}, \quad (13.116)$$

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

where u^+ is evaluated from Equation 13.115.

Because neither the turbulent stress nor the velocity gradient near the wall can be obtained correctly from the wall function, the kinetic energy production term in the transport equation for turbulent kinetic energy requires special treatment. The method used in CONVERGE is to calculate the velocity gradient near the wall from

$$\left(\frac{\partial U}{\partial y} \right)_w = \frac{\tau_w}{\mu}. \quad (13.117)$$

The turbulent dissipation is blended with a modified blending function:

$$\Gamma_\varepsilon = \frac{0.001(y^+)^4}{1 + y^+}, \quad (13.118)$$

so that the blended turbulent dissipation is calculated as

$$\varepsilon = \exp(-\Gamma_\varepsilon) \varepsilon_{visc} + \exp(-1/\Gamma_\varepsilon) \varepsilon_l, \quad (13.119)$$

where ε_l is calculated as

$$\varepsilon_l = \frac{C_\mu^{3/4} k^{3/2}}{\kappa y} \quad (13.120)$$

and ε_{visc} is calculated according to Equation 13.98 (with $C_\mu = 1$), except for the $\overline{v^2} - f$ and $\zeta - f$ models, where ([Popovac and Hanjalic, 2007](#))

$$\varepsilon_{visc} = \frac{2vk}{y^2}. \quad (13.121)$$

For the $\overline{v^2} - f$ and $\zeta - f$ models, the elliptic relaxation function is modeled with the same two-layer approach,

$$f = \exp(-\Gamma) f_{visc} + \exp(-1/\Gamma) f_l, \quad (13.122)$$

where Γ is given by Equation 13.114 and f_l is given as

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$f_l = 6 \frac{\bar{v}^2}{k} \epsilon. \quad (13.123)$$

The viscous sublayer term f_{visc} is calculated as

$$f_{visc} = -\frac{2v\zeta_p}{y_p^2} \quad (13.124)$$

for the $\zeta - f$ model and as

$$f_{visc} = C_1 (y^+ + C)^{1/2+\sqrt{D}} + C_2 (y^+ + C)^{1/2-\sqrt{D}} - \frac{0.1188}{(y^+ + C)^2}. \quad (13.125)$$

for the $\bar{v}^2 - f$ model ([Kalitzin et al., 2005](#)).

Except for the $\bar{v}^2 - f$ and $\zeta - f$ models, which do not require any specific treatment for turbulent viscosity ([Popovac and Hanjalic, 2007](#)), k - ϵ models calculate the turbulent viscosity with a two-layer approach ([Wolfshtein, 1969](#)),

$$\mu_t = \exp(-\Gamma) \mu_{t,visc} + \exp(-1/\Gamma) \mu_{t,l}, \quad (13.126)$$

where $\mu_{t,l}$ is the far-field expression from the base turbulence model and $\mu_{t,visc}$ is calculated as

$$\mu_{t,visc} = \rho C_\mu l_\mu \sqrt{k}, \quad (13.127)$$

where

$$l_\mu = y C_l^* \left(1 - \exp \left[-Re_y / A_\mu \right] \right), \quad (13.128)$$

$$Re_y = \frac{\rho y \sqrt{k}}{\mu}, \quad (13.129)$$

and A_μ is a model constant equal to 70.

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

The unified thermal wall function uses the method of [Kader \(1981\)](#) to blend the viscous sublayer and log-layer profiles:

$$T^+ = \frac{(T_w - T_p) \rho c_p u^*}{\dot{q}} = \exp(-\Gamma_T) T_{visc}^+ + \exp(-1/\Gamma_T) T_l^+, \quad (13.130)$$

where

$$\Gamma_T = \frac{0.01(Pr y^+)^4}{1 + 5Pr^3 y^+}. \quad (13.131)$$

The viscous sublayer and log-layer temperatures are expressed as

$$T_{visc}^+ = Pr \cdot u_{visc}^+ \quad (13.132)$$

and

$$T_l^+ = Pr_t (u_l^+ + P), \quad (13.133)$$

where P is the sublayer resistance factor ([Jayatilleke, 1969](#)), given as

$$P = 9.24 \left[\left(\frac{Pr}{Pr_t} \right)^{3/4} - 1 \right] \left[1 + 0.28 \exp \left(-0.007 \frac{Pr}{Pr_t} \right) \right]. \quad (13.134)$$

The additional complexity of the enhanced wall treatment results in greater computational expense. However, this treatment is appropriate for any y^+ smaller than several hundred, and performs better than other wall treatments when near-wall pressure gradients and heat transfer are important.

Analytic Wall Treatment

The analytic wall treatment (`near_wall_treatment = ANALYTIC`) uses analytic functions obtained by integrating simplified momentum and energy equations near the wall. This treatment is appropriate for smooth walls (*i.e.*, it does not account for `boundary.in > boundary_conditions > boundary > roughness > height` greater than zero) and $y^+ \geq 30$.

The model includes functions for the near-wall shear stress, heat flux, turbulent viscosity, turbulent kinetic energy production, and turbulent dissipation. In the original implementation, these functions were evaluated using the tangential velocity U_n and wall distance y_n at face n of the cell adjacent to the wall (Craft et al., 2002 and Suga et al., 2006).

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

Because CONVERGE employs a collocated grid arrangement, we replace U_n and y_n with the cell-centered values U_p and y_p .

The transition from the log-law region to the viscous sublayer occurs at $y_p^* = y_v^*$, where

$$y_p^* = y_p \frac{\rho \sqrt{k_p}}{\mu}, \quad (13.135)$$

μ is the molecular viscosity (without the contribution from turbulence) and $y_v^* = 10.7$. The shear stress at the wall is given by

$$\tau_{wall} = -\frac{\rho \sqrt{k_p}}{\mu} A. \quad (13.136)$$

When $y_p^* > y_v^*$, cell p is in the log-law region and the integration constant A is calculated as

$$A = \frac{\mu U_p - N}{(1/\alpha) \ln [1 + \alpha (y_p^* - y_v^*)] + y_v^*}, \quad (13.137)$$

where

$$N = \frac{C}{\alpha} \left[y_p^* - y_v^* - \left(\frac{1}{\alpha} - y_v^* \right) \ln [1 + \alpha (y_p^* - y_v^*)] + \frac{\alpha y_v^{*2}}{2} \right] \quad (13.138)$$

and

$$C = \frac{\mu^2}{\rho^2 k_p} \left[\frac{\partial}{\partial x} (\rho U U) + \frac{\partial P}{\partial x} \right], \quad (13.139)$$

where the gradients are tangential to the wall. The constant α is defined as $\alpha = c_l c_\mu$, where $c_l = 2.55$ and $c_\mu = 0.09$.

When $y_p^* < y_v^*$, cell p is in the viscous sublayer and

$$A = \frac{\mu U_p - N}{y_p^*}, \quad (13.140)$$

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

where

$$N = \frac{C}{2} y_p^{*2} \quad (13.141)$$

and C is given by Equation 13.139.

The heat flux at the wall is given by

$$q_{wall} = -\frac{\rho c_p \sqrt{k_p}}{\mu} A_{th}, \quad (13.142)$$

where c_p is the specific heat at constant pressure. The expressions for the integration constant A_{th} are

$$A_{th} = \frac{1}{(1/\alpha_t) \left[1 + \alpha_t (y_p^* - y_v^*) \right] + y_v^*} \times \begin{bmatrix} (T_p - T_{wall}) (\mu/\text{Pr}) - (1/\alpha_t) C_{th} (y_p^* - y_v^*) \\ -(C_{th}/\alpha_t) (y_v^* - 1/\alpha_t) \ln \left[1 + \alpha_t (y_p^* - y_v^*) \right] \\ -(C_{th}/2) y_v^{*2} \end{bmatrix} \quad (13.143)$$

for $y_p^* > y_v^*$ and

$$A_{th} = \frac{(T_p - T_{wall}) (\mu/\text{Pr}) - (C_{th}/2) y_p^{*2}}{y_p^*} \quad (13.144)$$

for $y_p^* < y_v^*$, where

$$C_{th} = \frac{\mu^2}{\rho^2 k_p} \left(\rho U_p \frac{\partial T}{\partial x} \right) \quad (13.145)$$

and

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$\alpha_t = \frac{\alpha \text{Pr}}{0.9}. \quad (13.146)$$

The turbulent viscosity at the wall is modeled as

$$\mu_t = \max\left[0, \alpha \mu \left(y_p^* - y_v^*\right)\right]. \quad (13.147)$$

The turbulent kinetic energy (tke) at the wall is calculated from the tke transport equation, with the production term set to zero for $y_p^* < y_v^*$ and modeled as

$$\overline{P}_k = \frac{1}{y_p} \frac{\rho \sqrt{k_p} \alpha}{\mu^2} \times \left[\begin{array}{l} \frac{C^2}{2\alpha^2} y^{*2} + \frac{C(2A + Cy_v^* - 2C/\alpha)}{\alpha^2} y^* + \frac{(A + Cy_v^* - C/\alpha)^2}{\alpha^2 [1 + \alpha(y^* - y_v^*)]} \\ + \frac{(A + Cy_v^* - C/\alpha)(A + Cy_v^* - 3C/\alpha)}{\alpha^2} \ln \left[1 + \alpha(y^* - y_v^*) \right] \end{array} \right]_{y^* = y_v^*}^{y^* = y_p^*} \quad (13.148)$$

for $y_p^* > y_v^*$. The dissipation term in the tke equation is modeled as $-\rho \bar{\varepsilon}$, where

$$\bar{\varepsilon} = \begin{cases} \frac{2\rho k_p^2}{\mu y_\varepsilon^{*2}}, & \text{if } y_\varepsilon^* > y_p^* \\ \frac{\rho k_p^2}{\mu y_p} \left[\frac{2}{y_\varepsilon^*} + \frac{1}{c_l} \ln \left(\frac{y_p^*}{y_\varepsilon^*} \right) \right], & \text{if } y_\varepsilon^* \leq y_p^* \end{cases} \quad (13.149)$$

and $y_\varepsilon^* = 5.1$.

The turbulent dissipation at cell p is given by

$$\varepsilon_p = \begin{cases} \frac{2\mu k_p}{\rho y_\varepsilon^2}, & \text{if } y_\varepsilon > y_p \\ \frac{k_p^{3/2}}{c_l y_p}, & \text{if } y_\varepsilon \leq y_p \end{cases} \quad (13.150)$$

where

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$y_\varepsilon = y_\varepsilon^* \frac{\mu}{\rho \sqrt{k_p}}. \quad (13.151)$$

k- ω Boundary Conditions

Turbulent Kinetic Energy Boundary Conditions

There are three types of INFLOW/OUTFLOW boundary conditions available for the turbulent kinetic energy (tke) equation: Dirichlet (*di*), turbulence intensity (*in*), and Neumann (*ne*). The Dirichlet and Neumann boundary conditions are not unique to the turbulence models and are explained in detail in [Chapter 9 - Boundaries and Boundary Conditions](#).

To use a turbulence intensity boundary condition, set *boundary.in* > *boundary_conditions* > *boundary* > *turbulence* > *tke* > *type* = INTENSITY. The turbulence intensity is a special case of a Dirichlet boundary condition. For the turbulence intensity condition, the boundary tke is given as

$$k = \frac{3}{2} u_i^2 I^2, \quad (13.152)$$

where *k* is the turbulent kinetic energy and *I* is the turbulence intensity specified in *boundary.in* > *boundary_conditions* > *boundary* > *turbulence* > *tke* > *value*. The turbulence intensity is usually set to a value between 0.01 and 0.10.

For a WALL boundary type there is only one valid boundary condition for the turbulent kinetic energy: Neumann. Thus, the WALL boundary condition for the turbulent kinetic energy equation is given as

$$\frac{\partial k}{\partial n} = 0.0, \quad (13.153)$$

where *k* is the tke and *n* is the wall normal vector.

Specific Dissipation Rate Boundary Conditions

There are three types of INFLOW/OUTFLOW boundary conditions available for the specific dissipation rate (*omega*) equation: Dirichlet (*di*), Neumann (*ne*), and turbulence length scale (*le*). The Dirichlet and Neumann boundary conditions are not unique to the turbulence models and are explained in detail in [Chapter 9 - Boundaries and Boundary Conditions](#).

The turbulence length scale is a special case of a Dirichlet boundary condition. For this case, the boundary specific dissipation rate is

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$\omega = \frac{k^{1/2}}{C_\mu^{1/4} le}, \quad (13.154)$$

where c_μ is a model constant (usually 0.09), k is the turbulent kinetic energy, and le is the turbulent length scale. The length scale can sometimes be estimated from a physical dimension in the domain. For example, while simulating flow in a duct, the length scale could be set to a fraction of an intake duct diameter.

For a WALL boundary type there is only one valid boundary condition for the specific dissipation rate: *Neumann*. Thus, the WALL boundary condition for the specific dissipation rate equation is given as

$$\frac{\partial \omega}{\partial n} = 0.0, \quad (13.155)$$

where ω is the specific dissipation rate and n is the wall normal vector.

Wall Treatments

In some simulations, the grid spacing required to resolve the viscous sublayer may be prohibitively expensive. In these cases, use one of the wall treatments for the specific dissipation rate (ω) in order to model the under-resolved viscous sublayer.

The standard wall function ([*turbulence.in*](#) > *Wall_modeling* > *near_wall_treatment* = STANDARD) makes use of the law-of-the-wall assumption for velocity in the log-law region of a turbulent boundary layer. Note that this wall function assumes that the cell adjacent to a wall lies in the log-law region. To calculate the value of specific dissipation rate at the cell centroid (ω_p) of a cell adjacent to a solid wall, CONVERGE uses the equation below:

$$\omega_p = \frac{\mu_\tau}{\sqrt{C_\mu \kappa y_p}} = \frac{\sqrt{k_p}}{C_\mu^{1/4} \kappa y_p} \quad (13.156)$$

An automatic wall treatment ([*turbulence.in*](#) > *Wall_modeling* > *near_wall_treatment* = AUTOMATIC) in CONVERGE uses a blend of the known solutions for ω in the viscous sublayer and the log-law region to calculate the specific dissipation rate at the cell centroid (ω_p) in the cell adjacent to a solid wall. Therefore, the cell adjacent to the wall can lie in the viscous sublayer, in the log-law region, or in the buffer region between them. Equation 13.157 below gives the solutions for ω in the viscous sublayer and log-law regions, respectively:

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$\omega_{visc} = \frac{6\nu}{\beta y_p^2} \quad \omega_{log} = \frac{\sqrt{k_p}}{C_\mu^{1/4} \kappa y_p} \quad (13.157)$$

where y_p is the distance from the wall to the cell centroid and ν is the kinematic viscosity. CONVERGE blends the viscous and log-law solutions as

$$\omega_p = \sqrt{\omega_{visc}^2 + \omega_{log}^2}. \quad (13.158)$$

To use the automatic wall treatment (described above) with Wilcox's low Reynolds number corrections, set [*turbulence.in*](#) > *Wall_modeling* > *near_wall_treatment* = ASYMPTOTIC. This wall treatment calculates ω_p via Equations 13.158 and 13.157, but includes low Reynolds number corrections.

If you enable the $k-\omega$ SST model and set [*turbulence.in*](#) > *Wall_modeling* > *near_wall_treatment* = ASYMPTOTIC, CONVERGE instead sets Menter's boundary conditions, using the following expressions for k and ω at a WALL boundary:

$$k_{wall} = 0$$

$$\omega_{wall} = 10 \frac{6\nu}{\beta_1 (\Delta d_1)^2}. \quad (13.159)$$

The enhanced wall treatment [described above](#) in the $k-\varepsilon$ section is also available for $k-\omega$ models. It adopts the hybrid wall momentum function, Equation 13.132, and uses the following hybrid expression for the wall shear stress ([Popovac and Hanjalic, 2007](#)):

$$\tau_w^+ = \left[\left(\tau_{visc}^+ \right)^4 + \left(\tau_l^+ \right)^4 \right]^{1/4}, \quad (13.160)$$

where

$$\tau_{visc}^+ = \frac{\rho u^* U_p}{u_{visc}^+} \quad (13.161)$$

and

$$\tau_l^+ = \frac{\rho u^* U_p}{u_l^+}. \quad (13.162)$$

Reynolds Stress Model Boundary Conditions

Near the wall, the Reynolds shear stress is corrected according to the Boussinesq assumption:

$$-\rho \overline{u'_i u'_j} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \delta_{ij} \frac{\partial u_k}{\partial x_k} \right) - \frac{2}{3} \rho k \delta_{ij}. \quad (13.163)$$

The Reynolds normal stress is set as zero-gradient (*i.e.*, Neumann) without a correction, which is consistent with the tke wall boundary condition.

The Reynolds stress and shear stress require initialization values. If information about these stresses is available (*i.e.*, starting a simulation from a restart file or a map file), the stresses are initialized to those values. If CONVERGE does not have this information available, the Reynolds stress is initialized at 2/3 of the local tke, and the shear stress is initialized at zero.

The enhanced wall treatment [described above](#) in the k - ε section is available for RSM. It adopts the hybrid wall momentum function, Equation 13.132, and the hybrid turbulent dissipation rate, Equation 13.136.

Spalart-Allmaras Boundary Conditions

The Spalart-Allmaras model requires a condition on $\tilde{\nu}$ at the wall. For a highly refined grid (*i.e.*, $y^+ << 1$), this condition reduces to $\tilde{\nu} = 0$. On grids which do not satisfy this condition, the boundary condition is less trivial. CONVERGE enforces a condition on $\tilde{\nu}$ based on the wall units u^+ and y^+ . First, CONVERGE evaluates the Reynolds number at the first cell off of the wall, Re_y , as

$$Re_y = \frac{\rho U y}{\nu}, \quad (13.164)$$

where U is the velocity, ρ is the density, y is the distance to the wall, and ν is the kinematic viscosity in the first cell. The first cell Reynolds number can also be expressed in wall units,

$$Re_y = u^+ y^+. \quad (13.165)$$

Within the viscous sublayer,

$$u^+ = y^+. \quad (13.166)$$

Within the log-law region,

Chapter 13: Turbulence Modeling

RANS Models RANS Boundary Conditions

$$u^+ = \frac{1}{\kappa} \ln y^+ + C^+, \quad (13.167)$$

where C^+ is a constant. CONVERGE takes $Re_y = 169$ (*i.e.*, $y^+ = 13$) as the boundary between these two regions, then solves either Equations 13.165 through 13.166 or Equation 13.167 for the local wall units. Finally, $\tilde{\nu}$ is calculated at the wall according as

$$\tilde{\nu} = \nu \kappa y^+. \quad (13.168)$$

Typical Parameter Values for RANS Models

Typical values for the $k-\varepsilon$ turbulence model constants differ based on which model you choose. Table 13.7 lists typical values for different $k-\varepsilon$ models.

Table 13.7: RANS $k-\varepsilon$ parameters and typical values.

Coefficient	<i>turbulence.in</i> parameter	Standard $k-\varepsilon$	RNG $k-\varepsilon$ / Rapid Distortion RNG $k-\varepsilon$	Realizable $k-\varepsilon$
C_μ	<i>RANS_models > k_eps_models > keps_cmu</i>	0.09	0.0845	0.09
$1/Pr_k$	<i>RANS_models > k_eps_models > keps_rpr_tke</i>	1.0	1.39	1.0
$1/Pr_t$	<i>RANS_models > k_eps_models > keps_rpr_eps</i>	0.77	1.39	0.833
$c_{\varepsilon 1}$	<i>RANS_models > k_eps_models > keps_ceps1</i>	1.44	1.42	1.44
$c_{\varepsilon 2}$	<i>RANS_models > k_eps_models > keps_ceps2</i>	1.92	1.68	1.9
$c_{\varepsilon 3}$	<i>RANS_models > k_eps_models > keps_ceps3</i>	-1.0	-1.0	-1.0
β	<i>RANS_models > k_eps_models > keps_beta</i>	N/A	0.012	0.012
η_0	<i>RANS_models > k_eps_models > keps_eta0</i>	N/A	4.38	4.38

Chapter 13: Turbulence Modeling

RANS Models Typical Parameter Values for RANS Models

Coefficient	<i>turbulence.in</i> parameter	Standard $k-\varepsilon$	RNG $k-\varepsilon$ /Rapid Distortion RNG $k-\varepsilon$	Realizable $k-\varepsilon$
c_s	<i>Physics_effects > discrete_c_s</i>	0.0	0.0	0.0
c_{ps}	<i>Physics_effects > discrete_c_ps</i>	0.03	0.03	0.03

Table 13.8 lists typical values for the constants in the $\overline{v^2} - f$ and $\zeta - f$ model formulations for each of the models.

Table 13.8: RANS v^2 -f and ζ -f parameters and typical values.

Coefficient	<i>turbulence.in</i> parameter	v^2 -f	ζ -f
C_μ	<i>RANS_models > v2f_zetaf_models > keps_v2f_cmu</i>	0.22	0.22
C_1	<i>RANS_models > v2f_zetaf_models > keps_v2f_c1</i>	1.4	0.4
C_2	<i>RANS_models > v2f_zetaf_models > keps_v2f_c2</i>	0.3	N/A
C_L	<i>RANS_models > v2f_zetaf_models > keps_v2f_cl</i>	0.23	0.36
C_η	<i>RANS_models > v2f_zetaf_models > keps_v2f_ceta</i>	70	85
$1/\zeta$	<i>RANS_models > v2f_zetaf_models > keps_zetaf_rpr_zeta</i>	N/A	1.0
C_2'	<i>RANS_models > v2f_zetaf_models > keps_zetaf_c2prime</i>	N/A	0.65

Select appropriate values for the turbulence model constants based on which $k-\omega$ model you choose. Table 13.9 lists typical values for each of the $k-\omega$ models.

Table 13.9: RANS $k-\omega$ parameters and typical values.

Coefficient	<i>turbulence.in</i> parameter	$k-\omega$ (1998)	$k-\omega$ (2006)	$k-\omega$ (SST)
C_μ	<i>RANS_models > k_omega_models > komega_cmu</i>	0.09	0.09	0.09
$1/Pr_k$	<i>RANS_models > k_omega_models > komega_rpr_tke</i>	0.5	0.6	0.85

Chapter 13: Turbulence Modeling

RANS Models Typical Parameter Values for RANS Models

Coefficient	<i>turbulence.in</i> parameter	<i>k-ω</i> (1998)	<i>k-ω</i> (2006)	<i>k-ω</i> (SST)
$1/Pr_{\omega}$	<i>RANS_models > k_omega_models > komega_rpr_omega</i>	0.5	0.5	0.5
α	<i>RANS_models > k_omega_models > komega_alpha</i>	13/25	13/25	5/9
β	<i>RANS_models > k_omega_models > komega_beta</i>	0.0702	0.0708	0.075
C_{lim}	<i>RANS_models > k_omega_models > komega_clim</i>	N/A	0.875	N/A
a_1	<i>RANS_models > k_omega_models > komega_sst_a1</i>	N/A	N/A	0.31
$1/Pr_{k,2}$	<i>RANS_models > k_omega_models > komega_rpr_tke_outer</i>	N/A	N/A	1.0
$1/Pr_{\omega,2}$	<i>RANS_models > k_omega_models > komega_rpr_omega_outer</i>	N/A	N/A	0.856
α_2	<i>RANS_models > k_omega_models > komega_alpha_outer</i>	N/A	N/A	0.44
β_2	<i>RANS_models > k_omega_models > komega_beta_outer</i>	N/A	N/A	0.0828
N/A	<i>RANS_models > k_omega_models > komega_near_wall_treatment</i>	0 = Use standard wall function (log law) for ω , 1 = Use enhanced wall function: low Reynolds number model, 2 = Use enhanced wall function with Wilcox's low Reynolds number corrections, 3 = Menter's wall boundary conditions.		

13.2 LES Models

In the Large Eddy Simulation (LES) approach, the field is decomposed into a resolved field and a sub-grid field as

$$\overbrace{u_i}^{velocity} = \overbrace{\bar{u}_i}^{resolved field} + \overbrace{u'_i}^{sub-grid field}, \quad (13.169)$$

where the over-bar indicates the resolved field and the prime indicates the sub-grid field. The resolved velocity field is defined as a spatial average of the actual velocity field, which differs from the RANS approach, where the mean velocity field is an ensemble average. Because of these decomposition differences, the LES filter has properties, unlike RANS, given by

$$\overline{\overline{u_i}} \neq \bar{u}_i \quad (13.170)$$

Chapter 13: Turbulence Modeling

LES Models

and

$$\overline{\dot{u}_i} \neq 0. \quad (13.171)$$

If the LES decomposition is applied to the conservation of momentum equation, the following LES equation can be derived:

$$\frac{\partial \overline{\rho} \widetilde{u}_i}{\partial t} + \frac{\partial \overline{\rho} \widetilde{u}_i \widetilde{u}_j}{\partial x_j} = - \frac{\partial \overline{P}}{\partial x_i} + \frac{\partial \overline{\sigma}_{ij}}{\partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} \quad (13.172)$$

where

$$\widetilde{u}_i \equiv \frac{\overline{\rho u_i}}{\overline{\rho}}, \quad (13.173)$$

and

$$\tau_{ij} = \overline{\rho} \left(\widetilde{u}_i \widetilde{u}_j - \widetilde{u}_i \widetilde{u}_j \right) \quad (13.174)$$

Most LES models focus on modeling the expression for the sub-grid stress tensor, τ_{ij} , given above. However, it is also possible to allow upwinding (*i.e.*, numerical viscosity) to be used as the LES model. In this case, no additional term is added for the sub-grid stress tensor.

In CONVERGE, the turbulence length scale le is given by

$$le = \sqrt[3]{V}, \quad (13.175)$$

where V is the volume of the cell. This expression applies to all LES models with the exception of the two-equation k - ε model.

In CONVERGE, there are three classes of LES models: zero-equation, one-equation, and two-equation. For zero-equation models, CONVERGE does not solve any additional transport equations. For one-equation models, CONVERGE solves an additional transport equation for sub-grid kinetic energy. For two-equation models, CONVERGE solves transport equations for sub-grid kinetic energy and sub-grid dissipation.

Zero-Equation LES Models

Upwind LES Model

This model employs a dissipative upwind differencing scheme as an alternative to modeling the sub-grid tensor. The upwind numerical scheme serves to keep the simulation stable. Many of the submodels require a sub-grid kinetic energy, k , or a sub-grid dissipation, ε . CONVERGE can compute these values by approximating the unresolved sub-grid velocity. These values can be used in place of the turbulent kinetic energy and the turbulent dissipation in the appropriate submodels.

This model expresses the sub-grid velocity as an infinite Taylor series expansion ([Bedford and Yeo, 1993](#); [Pomraning, 2000](#)). That is,

$$u'_i = -\alpha_{[k]} \frac{\partial^2 \bar{u}_i}{\partial x_k \partial x_k} + \frac{1}{2!} \alpha_{[k]} \alpha_{[l]} \frac{\partial^4 \bar{u}_i}{\partial x_k \partial x_k \partial x_l \partial x_l} \\ - \frac{1}{3!} \alpha_{[k]} \alpha_{[l]} \alpha_{[m]} \frac{\partial^6 \bar{u}_i}{\partial x_k \partial x_k \partial x_l \partial x_l \partial x_m \partial x_m} + \dots, \quad (13.176)$$

where

$$\alpha_{[k]} = \frac{dx_{[k]} dx_{[k]}}{24} \quad (13.177)$$

for a cubic cell and the brackets indicate no summation. Since it is not possible to evaluate all of the terms in the series, the sub-grid velocity is approximated by the first term:

$$u'_i = C_{les} \left[-\alpha_{[k]} \frac{\partial^2 \bar{u}_i}{\partial x_k \partial x_k} \right], \quad (13.178)$$

where C_{les} is a model constant. The sub-grid kinetic energy is then approximated as

$$k_{subgrid} \approx \frac{1}{2} u'_i u'_i, \quad (13.179)$$

and the sub-grid dissipation is approximated as

$$\varepsilon_{subgrid} = \frac{k_{subgrid}^{1.5}}{\Delta}, \quad (13.180)$$

Chapter 13: Turbulence Modeling

LES Models Zero-Equation LES Models

where Δ is the *grid* filter which is the cube root of the cell volume.

To activate this model, set [*turbulence.in*](#) > *turbulence_model* = *LES_UPWIND*.

Smagorinsky Model

The Smagorinsky model is a zero-equation LES model which relates the sub-grid viscosity to the magnitude of the strain rate tensor and cell size ([Deardorff, 1970](#); [Lilly, 1967](#); [Smagorinsky, 1963](#); [Speziale, 1998](#)). The model for the sub-grid stress tensor is

$$\tau_{ij} = -2\nu_t \bar{S}_{ij}, \quad (13.181)$$

where the sub-grid viscosity, ν_t , is

$$\nu_t = C_s^2 \Delta^2 \sqrt{\bar{S}_{ij} \bar{S}_{ij}}. \quad (13.182)$$

Here, Δ is the *grid* filter, which is related to the cell volume by the following expression:

$$\Delta = \sqrt[3]{Vol}. \quad (13.183)$$

To tune the Smagorinsky model, adjust the constant C_s in the expression for sub-grid viscosity.

To activate this model, set [*turbulence.in*](#) > *turbulence_model* = *LES_SMAG*.

Dynamic Smagorinsky Model

One of the problems with the Smagorinsky model is that the appropriate value of the coefficient C_s is different in different flow regimes. The dynamic Smagorinsky model provides a methodology for determining the local value of the Smagorinsky coefficient ([Lilly, 1992](#); [Germano et al., 1991](#); [Meneveau, 1994](#)).

The formulation of a dynamic model requires a second filtering operation designated a *test* level filter $\hat{\Delta}$. This *test* filter is typically twice the value of the *grid* filter, Δ . The residual stresses based on single (*grid*) and double filtering (*test*) operations are:

$$\tau_{ij} = (\bar{u}_i \bar{u}_j - \overline{\bar{u}_i} \overline{\bar{u}_j}) \quad (13.184)$$

and

$$T_{ij} = (\widehat{\bar{u}_i \bar{u}_j} - \widehat{\bar{u}_i} \widehat{\bar{u}_j}) \quad (13.185)$$

Chapter 13: Turbulence Modeling

LES Models Zero-Equation LES Models

The Germano identity ([Germano et al., 1991](#)) relates the *grid* level stress tensor and the *test* level stress tensor. That is,

$$L_{ij} = T_{ij} - \hat{\tau}_{ij} = \left(\widehat{u_i u_j} - \widehat{\bar{u}_i \bar{u}_j} \right) \quad (13.186)$$

where L_{ij} is the Leonard stress term. The Smagorinsky model of the deviatoric part of L_{ij} is given by

$$L_{ij} - \frac{1}{3} L_{kk} \delta_{ij} = C_{s-dynamic} M_{ij}, \quad (13.187)$$

where

$$M_{ij} = 2\bar{\Delta}^2 \left| \widehat{\bar{S}} \right| \widehat{S}_{ij} - 2\bar{\Delta}^2 \left| \widehat{\bar{S}} \right| \widehat{\bar{S}}_{ij} \quad (13.188)$$

and $C_{s-dynamic}$ in Equation 13.187 is the dynamic Smagorinsky coefficient. Recall that S_{ij} in Equation 13.188 is the rate of strain tensor. The sub-grid scale stress tensor is given by

$$\tau_{ij} = -2C_{s-dynamic} \bar{\Delta}^2 \left| \widehat{\bar{S}} \right| \widehat{S}_{ij}. \quad (13.189)$$

By performing the least squares technique to minimize the error ([Lilly, 1992](#)), $C_{s-dynamic}$ is specified by:

$$C_{s-dynamic} = \frac{M_{ij} L_{ij}}{M_{kl} M_{kl}}. \quad (13.190)$$

To activate the dynamic Smagorinsky model, set `turbulence.in > turbulence_model = LES_DYN_SMAG`.

Sigma LES

The sigma LES model ([Nicoud et al., 2013](#)) is a zero-equation model that constructs the sub-grid viscosity from the singular values of the resolved velocity gradient tensor. The construction is designed such that the sub-grid viscosity has a variety of desirable properties. The model produces zero sub-grid scale sub-grid viscosity in two-dimensional or two-velocity-component flowfields, as well as in axisymmetric flows and isotropic contraction or expansion. The sub-grid term also produces appropriate scaling behavior near solid boundaries.

Chapter 13: Turbulence Modeling

LES Models Zero-Equation LES Models

The sigma LES model first builds the matrix G as the product of the resolved velocity gradient tensor with its transpose:

$$G_{ij} = \frac{\partial \bar{u}_k}{\partial x_i} \frac{\partial \bar{u}_k}{\partial x_j}. \quad (13.191)$$

The principle invariants of G are given as

$$\begin{aligned} I_1 &= G_{ii}, \\ I_2 &= \frac{1}{2} \left[(G_{ii})^2 - G_{ii}^2 \right], \\ I_3 &= \det(G), \end{aligned} \quad (13.192)$$

and from these we calculate

$$\begin{aligned} \alpha_1 &= \frac{I_1^2}{9} - \frac{I_2}{3}, \\ \alpha_2 &= \frac{I_1^3}{27} - \frac{I_1 I_2}{6} + \frac{I_3}{2}, \\ \alpha_3 &= \frac{1}{3} \arccos \frac{\alpha_2}{\alpha_1^{3/2}}. \end{aligned} \quad (13.193)$$

We calculate from these the singular values of G :

$$\begin{aligned} \sigma_1 &= \left(\frac{I_1}{3} + 2\sqrt{\alpha_1} \cos \alpha_3 \right)^{1/2}, \\ \sigma_2 &= \left(\frac{I_1}{3} - 2\sqrt{\alpha_1} \cos \left(\frac{\pi}{3} + \alpha_3 \right) \right)^{1/2}, \\ \sigma_3 &= \left(\frac{I_1}{3} - 2\sqrt{\alpha_1} \cos \left(\frac{\pi}{3} - \alpha_3 \right) \right)^{1/2}. \end{aligned} \quad (13.194)$$

Note that the singular values have the property $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$, which is required to construct the sub-grid viscous term. This is a function of the differential operator D_σ :

$$D_\sigma = \frac{\sigma_3 (\sigma_1 - \sigma_2)(\sigma_2 - \sigma_3)}{\sigma_1^2}. \quad (13.195)$$

Chapter 13: Turbulence Modeling

LES Models Zero-Equation LES Models

Finally, we calculate the sub-grid viscosity, ν_t , as

$$\nu_t = (1.5\Delta)^2 D_\sigma, \quad (13.196)$$

where Δ is the grid filter, the cube root of the cell volume.

One-Equation LES Models

One-Equation Viscosity Model

The one-equation viscosity model adds a transport equation for the sub-grid kinetic energy as formulated by [Yoshizawa and Horiuti \(1985\)](#) and [Menon et al. \(1996\)](#). This model uses the sub-grid kinetic energy in modeling the turbulent viscosity. The sub-grid kinetic energy equation is given by

$$\frac{\partial k}{\partial t} + \bar{u}_i \frac{\partial k}{\partial x_i} = -\tau_{ij} \frac{\partial \bar{u}_i}{\partial x_j} - \varepsilon + \frac{\partial}{\partial x_i} \left(\frac{\nu_t}{\sigma_k} \frac{\partial k}{\partial x_i} \right). \quad (13.197)$$

Here the sub-grid kinetic energy is given by

$$k = \frac{1}{2} (\bar{u}_i \bar{u}_i - \bar{u}_i \bar{u}_i). \quad (13.198)$$

The model for the sub-grid stress tensor is

$$\tau_{ij} = -2\nu_t \bar{S}_{ij} + \frac{2}{3} k \delta_{ij}, \quad (13.199)$$

where the turbulent viscosity, ν_t , for the one-equation model is given as

$$\nu_t = C_k k^{1/2} \Delta. \quad (13.200)$$

You can tune the turbulent viscosity by adjusting C_k (see previous equation) via [*turbulence.in*](#) > *LES_models* > *les_c_sgs_ke_visc*. The sub-grid dissipation is

$$\varepsilon = \frac{C_\varepsilon k^{3/2}}{\Delta}. \quad (13.201)$$

You can also tune the sub-grid dissipation by adjusting the constant C_ε in the above expression.

Chapter 13: Turbulence Modeling

LES Models One-Equation LES Models

To activate this one-equation model, set [*turbulence.in*](#) > *turbulence_model* = *LES_ONE_EQN_VISC*.

Dynamic Structure Model

The dynamic structure model does not use turbulent viscosity to model the sub-grid stress tensor in the momentum equation ([Pomraning, 2000](#)). To enforce a budget on the energy flow between the resolved and the sub-grid scales, this model adds a transport equation (given by Equation 13.197) for the sub-grid kinetic energy. To that end, the sub-grid stress tensor models must be a function of the sub-grid turbulent kinetic energy. The modeled stress tensors are given by

$$\begin{aligned}\tau_{ij} &= c_{ij} k \\ T_{ij} &= c_{ij} K,\end{aligned}\tag{13.202}$$

where the *test* level kinetic energy is defined by

$$K = \frac{1}{2} \left(\widehat{\bar{u}_i u_i} - \widehat{\bar{u}_i} \widehat{\bar{u}_i} \right)\tag{13.203}$$

The trace of the Leonard term relates the *test* and *grid* level kinetic energies so that an additional transport equation for *K* is not required. That is,

$$K = \widehat{k} + \frac{1}{2} L_{ii}.\tag{13.204}$$

Substituting these models for the two stress tensors into the Germano identity yields

$$L_{ij} = K c_{ij} - \widehat{k} c_{ij}.\tag{13.205}$$

Note that the coefficient tensor, c_{ij} , is properly left inside the integral as indicated by the curve over the bar. The result is a set of six (the stress tensor is symmetric) Fredholm integral equations of the second kind that can be solved via an iterative method. Alternatively, in an algebraic model, we remove the tensor coefficient from the integral and then solve for τ_{ij} . Thus the model for the sub-grid tensor becomes

$$\tau_{ij} = 2k \left(\frac{L_{ij}}{L_{ii}} \right).\tag{13.206}$$

Chapter 13: Turbulence Modeling

LES Models One-Equation LES Models

The dynamic structure model uses the sub-grid viscosity to close equations other than the momentum equation, and to solve for, *e.g.*, the turbulent Prandtl and turbulent Schmidt numbers. With this model, CONVERGE solves for the sub-grid viscosity according to the grid spacing and the tke, as shown in Equation 13.200.

To activate the dynamic structure model, set `turbulence.in > turbulence_model = LES_DYN_STRUCT`.

Consistent Dynamic Structure Model

An alternate version of the dynamic structure model is the consistent dynamic structure model. This model is more appropriate for a rotating frame of reference ([Lu et al., 2007](#)).

For the consistent dynamic structure model, the sub-grid stress tensor is

$$\tau_{ij} = 2k \left(\frac{G_{ij}}{G_{ii}} \right), \quad (13.207)$$

where

$$G_{ij} = \frac{\partial \bar{u}_i}{\partial x_k} \frac{\partial \bar{u}_j}{\partial x_k}. \quad (13.208)$$

To activate the consistent dynamic structure model, set `turbulence.in > turbulence_model = LES_CON_DYN_STRUCT`.

Two-Equation LES Models

In zero- and one-equation LES models, the turbulent viscosity and the turbulence length scale are proportional to the grid filter Δ , which causes unrealistic behavior in some applications. For example, if you use AMR to resolve the flame front in a combustion simulation, the turbulent viscosity and the turbulence length scale will decrease sharply near the flame front due to local refinement of the grid, which causes the flame to burn more slowly. A two-equation LES model can eliminate such effects by smoothing out the transitions of these quantities.

Two-Equation $k-\varepsilon$ Model

The two-equation $k-\varepsilon$ model uses the same transport equation for sub-grid kinetic energy k as the one-equation [Dynamic Structure](#) model. The transport equation for the sub-grid dissipation ε is given by

$$\frac{\partial \varepsilon}{\partial t} + \bar{u}_i \frac{\partial \varepsilon}{\partial x_i} = \left(\frac{3C_\varepsilon k^{1/2}}{2l_f} \right) P - \frac{3\varepsilon^2}{2k} + \frac{\partial}{\partial x_i} \left(\frac{\nu_t}{\sigma_\varepsilon} \frac{\partial \varepsilon}{\partial x_i} \right) + \frac{\varepsilon}{l_f} \left(\frac{l_f - \Delta}{\tau} \right), \quad (13.209)$$

Chapter 13: Turbulence Modeling

LES Models Two-Equation LES Models

where P is the sub-grid production, l_f is the local filter size, and τ is the time-scale over which l_f relaxes to the local grid size Δ . The dissipation depends on l_f via

$$\varepsilon = \frac{C_\varepsilon k^{3/2}}{l_f} \quad (13.210)$$

and the turbulent viscosity is defined as

$$\nu_t = C_k k^{1/2} l_f = C_k C_\varepsilon \frac{k^2}{\varepsilon}, \quad (13.211)$$

where C_ε and C_k are tuning parameters. Equations 13.210 and 13.211 have the same functional form as Equations 13.200 and 13.201, but they use l_f as the turbulence length scale instead of Δ . The relaxation time-scale is calculated from

$$\tau = \frac{C_\tau k}{\varepsilon}. \quad (13.212)$$

In CONVERGE, the constant C_τ is set to 0.5.

To activate the two-equation k - ε model, set [*turbulence.in*](#) > *turbulence_model* = *LES_TWO_EQN*.

LES Boundary Conditions

Ideally, the grid resolution for an LES simulation is sufficient to resolve the flow into the viscous sublayer. In this case, the walls are modeled directly with no-slip boundary conditions and Dirichlet temperature conditions. In many cases, however, it is not feasible to add enough resolution to resolve the viscous sublayer. Thus, you must employ wall models.

The LES wall models are available for both the [momentum equation](#) and the [energy equation](#). The Werner and Wengle wall model is designed to work with LES models and thus should be activated when running one of the LES models. The model can be activated by setting [*turbulence.in*](#) > *LES_models* > *les_wall_model* = 1.

Many of the spray and combustion models require a turbulent kinetic energy term and a turbulent dissipation term to complete (close) the model. In the one- and two-equation models, the turbulent kinetic energy term and turbulent dissipation term are simply replaced by the sub-grid kinetic energy and the sub-grid dissipation given by the expressions above. For zero-equation models, a sub-grid kinetic energy term is not readily available to close the model. Thus, for zero-equation models the expression below is used to approximate the sub-grid kinetic energy:

Chapter 13: Turbulence Modeling

LES Models LES Boundary Conditions

$$k \cong C_{les} \frac{\Delta^2}{24} \frac{\partial \bar{u}_i}{\partial x_j} \frac{\partial \bar{u}_i}{\partial x_j}. \quad (13.213)$$

Tune the above expression by adjusting the constant C_{les} .

Typical Parameter Values for LES Models

Table 13.9 summarizes the LES model inputs in CONVERGE.

Table 13.10: LES model inputs.

Model constant	<i>turbulence.in</i> parameter	Typical value
C_s	<i>LES_models > les_c_sgs_ke</i> (<i>turbulence_model = LES_ONE_EQN_VISC, LES_SMAG, LES_DYN_STRUCT</i>)	1.0 to 5.0
C_k	<i>LES_models > les_c_sgs_ke_visc</i> (<i>turbulence_model = LES_ONE_EQN_VISC, LES_SMAG, LES_DYN_STRUCT, LES_TWO_EQN</i>)	0.05
C_ε	<i>LES_models > les_c_sgs_eps</i> (<i>turbulence_model = LES_ONE_EQN_VISC, LES_DYN_STRUCT, LES_CON_DYN_STRUCT, LES_TWO_EQN</i>)	1.0
C_{les}	<i>LES_models > les_c_sgs_eps</i> (<i>turbulence_model = LES_UPWIND, LES_SMAG, LES_DYN_SMAG</i>)	2.0
C_{ps}	<i>Physics_effects > discrete_c_ps</i> (<i>turbulence_model = LES_UPWIND, LES_ONE_EQN_VISC, LES_SMAG, LES_DYN_SMAG, LES_DYN_STRUCT, LES_CON_DYN_STRUCT</i>)	0.03
$1/\Pr_k$	<i>LES_models > les_rpr_tke</i> (<i>turbulence_model = LES_ONE_EQN_VISC, LES_DYN_STRUCT, LES_CON_DYN_STRUCT</i>)	1.0

13.3 Hybrid RANS/LES Models

DES Models

Detached Eddy Simulation (DES) is a hybrid approach designed to combine the strongest features of the RANS and LES methodologies. Near walls, DES behaves like a RANS model. Far from a wall, where large-scale unsteady structures are detached from the boundary layer, the DES model behaves like LES. The same sub-grid scale model functions as the RANS turbulence model and as the LES sub-grid filter. DES was originally formulated by [Spalart et al. \(1997\)](#), using the one-equation Spalart-Allmaras RANS model ([Spalart and Allmaras, 1992](#)) as the sub-grid closure. This original formulation uses grid cell metrics to switch between RANS and LES behavior, limiting its applicability.

The Delayed DES ([Spalart et al., 2006](#)) and Improved Delayed DES ([Shur et al., 2008](#)) approaches, denoted DDES and IDDES, respectively, are improvements on the original DES

Chapter 13: Turbulence Modeling

Hybrid RANS/LES Models

formulation. Rather than using only grid metrics, DDES and IDDES switch modes based on several flow-based and geometry-based blending and shielding functions. These improvements were originally designed to prevent ("delay") the LES behavior from appearing in near-wall regions with excess grid density. These formulations also use Spalart-Allmaras as the sub-grid closure.

CONVERGE incorporates variants of the DDES and IDDES models, using the $k-\omega$ SST model [described above](#). [Strelets \(2001\)](#) adapted the DES methodology to $k-\omega$ SST, and [Gritskevich et al. \(2012\)](#) formulated the DDES and IDDES improvements. For exact model formulation details, refer to this latter source.

To run CONVERGE with the DDES-SST turbulence model, set [*turbulence.in* > turbulence_model = DDES_K_OMEGA_SST](#). To run with the IDDES-SST model, set [*turbulence.in* > turbulence_model = IDDES_K_OMEGA_SST](#). When running DDES-SST or IDDES-SST, CONVERGE refers to the same $k-\omega$ SST parameters as when running with this model in RANS mode. The DDES and IDDES formulations also refer to a set of DDES parameters. Refer to [Chapter 25 - Input and Data Files](#) for information on [these parameters, as well as recommended values](#).

13.4 Turbulence Statistics Output

To generate output of turbulence statistics (mean and RMS quantities for density, pressure, temperature, etc.) for transient simulations, define these quantities as non-transport [passives](#). Refer to [Chapter 25 - Input and Data Files](#) for a list of [predefined turbulence statistics](#) that you can specify as non-transport passives in [*species.in*](#).

You must also set several parameters in [*turbulence.in*](#) to use the turbulence statistics option. Table 13.11 below describes these parameters.

Finally, to include the turbulence statistics in the *post*.h5* files, you must list the turbulence statistics non-transport passives [in the form *passive(<turbulence statistic name>)*] in the *<cells>* section of the [*post.in*](#) file.

Table 13.11: Turbulence statistics parameters.

Parameter	Description
<i>turb_stat_flag</i>	0 = Do not calculate turbulence statistics, 1 = Calculate turbulence statistics.
<i>turb_stat_start_time</i>	The start time in seconds (if <i>inputs.in</i> > simulation_control > crank_flag = 0) or crank angle degrees (if <i>crank_flag</i> is non-zero) for the turbulence statistics calculation.
<i>turb_stat_end_time</i>	The end time in seconds (if <i>inputs.in</i> > simulation_control > crank_flag = 0) or crank angle degrees (if <i>crank_flag</i> is non-zero) for the turbulence statistics calculation.
<i>turb_stat_tol</i>	Relative tolerance for turbulence statistics convergence = $(abs(delta(mean)) / mean))$. For monitoring purposes only.

Chapter 13: Turbulence Modeling

Turbulence Statistics Output

If [*inputs.in*](#) > *simulation_control* > *reread_input* = 1, CONVERGE will reread the turbulence statistics parameters listed above before each time-step.

- If you change the value of either [*turbulence.in*](#) > *Turbulence_statistics* > *turb_stat_start_time* or *Turbulence_statistics* > *turb_stat_end_time*, CONVERGE uses only the revised time range to calculate the updated turbulence statistics.
- Set *Turbulence_statistics* > *turb_stat_start_time* to a time prior to the current simulation time to have CONVERGE begin the turbulence statistics calculations immediately.
- If CONVERGE has already calculated turbulence statistics and if you want CONVERGE to resume turbulence statistics calculations and incorporate both the new and the old data into the calculations, set *Turbulence_statistics* > *turb_stat_end_time* to a time beyond the current simulation time.
- If CONVERGE has already calculated turbulence statistics and if you wish to overwrite these calculations, set *Turbulence_statistics* > *turb_stat_end_time* and *Turbulence_statistics* > *turb_stat_end_time* to times beyond the current simulation time.

It is important to verify that all of the initial transients (non-physical values of conditions in the domain) have been eliminated prior to the *Turbulence_statistics* > *turb_stat_start_time*.

To monitor turbulence statistics, look at the *post*.h5* files after converting them with the [*post_convert*](#) utility. You can find additional information regarding the convergence of these statistics in the log file when you set [*inputs.in*](#) > *output_control* > *log_level* = 2 or higher.

The turbulence statistics calculated near moving boundaries (*e.g.*, volume swept by moving valves) may not be accurate. Carefully evaluate these near-moving-object statistics to gauge their suitability.

You can calculate user-defined turbulence statistics with the *user_turbulent_statistics.c* [user-defined function](#).

Turbulence Statistics Theory

For the turbulence statistics, CONVERGE calculates the time average by

$$\overline{f(x)} = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} f(x, t) dt \quad (13.214)$$

and the fluctuations by

Chapter 13: Turbulence Modeling

Turbulence Statistics Output

$$\begin{aligned}\overline{f'g'} &= \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} [f(x, t) - \overline{f(x)}] [g(x, t) - \overline{g(x)}] dt \\ &= \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} [f(x, t)g(x, t) - \overline{f(x)}g(x, t) - f(x, t)\overline{g(x)} + \overline{f(x)g(x)}] dt \\ &= \overline{f(x, t)g(x, t)} - \overline{f(x)g(x)} - \overline{g(x)f(x)} + \overline{f(x)g(x)} \\ &= \overline{fg} - \overline{fg}\end{aligned}\tag{13.215}$$

By calculating \overline{fg} together with \overline{f} and \overline{g} , $\overline{f'g'}$ can be computed simultaneously.

Chapter



14

Discrete Phase Modeling

14 Discrete Phase Modeling

This chapter describes the discrete phase models included in CONVERGE. This capability is designed to simulate the motion of compact packets of mass that are small relative to the local cell spacing, and that move through the domain while generally maintaining their coherence. Packet motion is represented in a Lagrangian frame of reference, within the finite-volume cells whose equations of motion are represented in the Eulerian frame. Physical examples of compact packets of mass include solid particles, liquid spray drops, or components of a liquid film on a boundary.

Discrete phase modeling in CONVERGE is based on the concept of a parcel. Each parcel represents a group of identical mass packets (*i.e.*, same radius, velocity, temperature, etc.). A field of such parcels are used to statistically represent the entire discrete phase field. By using the concept of parcels, CONVERGE makes the computational time only weakly dependent on the number of physical mass packets in a simulation.

In CONVERGE, descriptions of parcel physics and models are grouped into four categories, each with its own input file. These are intrinsic parcel properties and settings ([parcels.in](#)), introduction of parcels to the domain ([parcel_introduction.in](#)), how parcels interact with each other ([parcel.ParcelInteraction.in](#)), and how parcels interact with boundaries ([parcel.WallInteraction.in](#)). The following sections provide detailed descriptions of these physics and models.

Note that some models, such as injection distribution models, collision models, and turbulent dispersion models, require the use of a random number generator. CONVERGE uses a pseudorandom number generator that is initialized from a user-specified seed value ([inputs.in > simulation_control > random_seed](#)). If you run two simulations on the same computer hardware with the same case setup and the same value of *random_seed*, CONVERGE will generate identical sequences of random numbers for those simulations. In some situations, you might want to change the value of *random_seed* to obtain different realizations for different simulations.

14.1 Parcel Equations of Motion

In CONVERGE, the parcel velocity, v_i , is obtained from the equation of motion

$$\rho_p V_p \frac{dv_i}{dt} = F_{p,i}, \quad (14.1)$$

where ρ_p is the parcel density, V_p is the parcel volume, and $F_{p,i}$ is the total force on the parcel.

For liquid parcels, $F_{p,i}$ is given by the sum of the drag force and the gravitational body force as

Chapter 14: Discrete Phase Modeling

Parcel Equations of Motion

$$F_{p,i} = F_{drag,i} + F_{g,i} = C_D \pi r_p^2 \frac{\rho |U_i|}{2} U_i + \rho_p V_p g_i. \quad (14.2)$$

In Equation 14.2, C_D is the [drag coefficient](#), r_p is the parcel radius, ρ is the fluid density, U_i is the parcel-fluid relative velocity given by

$$U_i = u_i + u'_i - v_i, \quad (14.3)$$

where u_i and u'_i are the local mean and turbulent fluctuating fluid velocities, respectively, and g_i is the gravitational acceleration specified in [inputs.in > property_control > gravity](#).

For solid parcels, the total force is given by

$$F_{p,i} = F_{drag,i} + F_{g,i} + F_{a,i} + F_{undist,i} + F_{buoyancy,i} + F_{lift,i}. \quad (14.4)$$

The drag force and gravitational body force have the same form as in Equation 14.2. The additional terms account for the forces due to added mass ($F_{a,i}$), undisturbed fluid flow ($F_{undist,i}$), buoyancy ($F_{buoyancy,i}$), and lift ($F_{lift,i}$), given respectively by

$$F_{a,i} = C_a m_a \frac{dU_i}{dt}, \quad (14.5)$$

$$F_{undist,i} = m_a \frac{d(u_i + u'_i)}{dt}, \quad (14.6)$$

$$F_{buoyancy,i} = -m_d g_i, \quad (14.7)$$

and

$$F_{lift,i} = V_p \rho C_L \epsilon_{ijk} \epsilon_{klm} U_j \frac{\partial(u_m + u'_m)}{\partial x_l}. \quad (14.8)$$

The added mass in Equations 14.5 and 14.6 is computed as

$$m_a = \frac{2}{3} \pi r_p^3 \rho, \quad (14.9)$$

Chapter 14: Discrete Phase Modeling

Parcel Equations of Motion

while the displaced mass in Equation 14.7 is

$$m_d = \frac{4}{3} \pi r_p^3 \rho = 2m_a. \quad (14.10)$$

You can activate or deactivate specific forces for solid parcels in [*parcels.in* > solid_parcel > parcel_list > parcel > force_control](#). For the added mass force, specify the coefficient C_a directly. For the drag force and the lift force, choose one of the available models for calculating the [drag coefficient](#) and the [lift coefficient](#).

14.2 Parcel Time-Step Control

CONVERGE allows you to use a variable time-step. If parcel modeling is active, the maximum time-step based on parcel motion is limited by

$$dt_{\text{parcel}} = \min\left(\frac{\Delta x}{\text{parcel_velocity}}\right) \cdot \text{mult_dt_parcel}, \quad (14.11)$$

where mult_dt_parcel is specified via [*inputs.in* > temporal_control > mult_dt.Parcel](#), Δx is the local cell size, and parcel_velocity is the parcel velocity magnitude ($|v_i|$).

The [*inputs.in* > temporal_control > mult_dt.evap](#) parameter limits the time-step based on liquid parcel evaporation. We recommend setting a very large mult_dt_evap value for all engineering simulations, which essentially disables this time-step limiter. When studying the behavior of the evaporation models themselves, a small mult_dt_evap may be useful.

If the collision mesh option is active, you may want to limit the time-step to prevent parcels from traveling through more than one collision mesh cell in a time-step. The maximum allowable time-step for the collision mesh is given by

$$dt_{\text{coll_mesh}} = \frac{\text{min_coll_size}}{\text{max_parcel_velocity}} \cdot \text{mult_dt_coll_mesh}, \quad (14.12)$$

where min_coll_size is the smallest collision mesh cell, $\text{max_parcel_velocity}$ is the maximum drop velocity in the domain, and mult_dt_coll_mesh is a multiplier specified via [*inputs.in* > temporal_control > mult_dt.coll_mesh](#).

Parcel Sub-Cycling

To improve simulation stability and accuracy with large solver time-steps, you can optionally specify a liquid parcel sub-cycle time-step limit ([*parcels.in* > liquid_parcel > subcycle_dt](#)).

Chapter 14: Discrete Phase Modeling

Parcel Time-Step Control

$sub_cycle > dt$). If the prescribed parcel sub-cycle time-step is smaller than the flow solver time-step, CONVERGE will calculate the number of parcel sub-cycles N_{ss} as

$$N_{ss} = \text{ceiling}\left(\frac{dt_{flow}}{\text{parcel_sub_cycle_dt}}\right), \quad (14.13)$$

then calculate the final parcel time-step dt_{parcel} as

$$dt_{parcel} = \frac{dt_{flow}}{N_{ss}}. \quad (14.14)$$

CONVERGE will perform N_{ss} parcel sub-iterations of time-step size dt_{parcel} to stay synchronized with the rest of the flow solver.

The parcel sub-cycling feature cannot be combined with the FGM, ECFM, or ECFM3Z combustion models.

14.3 Parcel Post-Processing Parameters

The liquid and vapor penetration lengths (LPL and VPL) are two of the properties that characterize a liquid parcel spray. CONVERGE calculates both quantities for each nozzle at each time-step.

To measure the LPL, CONVERGE first calculates the total mass of the liquid parcels from the nozzle and then multiplies this mass by the liquid penetration fraction to yield the penetrated spray mass. Starting from the center of the nozzle hole, CONVERGE sums the mass of the liquid parcels until it reaches the penetrated spray mass. This distance is the LPL. CONVERGE reports the LPL for penetration fractions of 0.90, 0.95, 0.97, and 0.99 in [liquid_parcel_ecn.out](#). Additionally, you specify a liquid penetration fraction, [parcel_introduction.in](#) > [injections](#) > [injection](#) > [penet_control](#) > [liquid_frac](#). CONVERGE writes the corresponding LPL to [liquid_parcel.out](#).

The following figure shows a schematic of the LPL with [liquid_frac](#) set to 0.90.

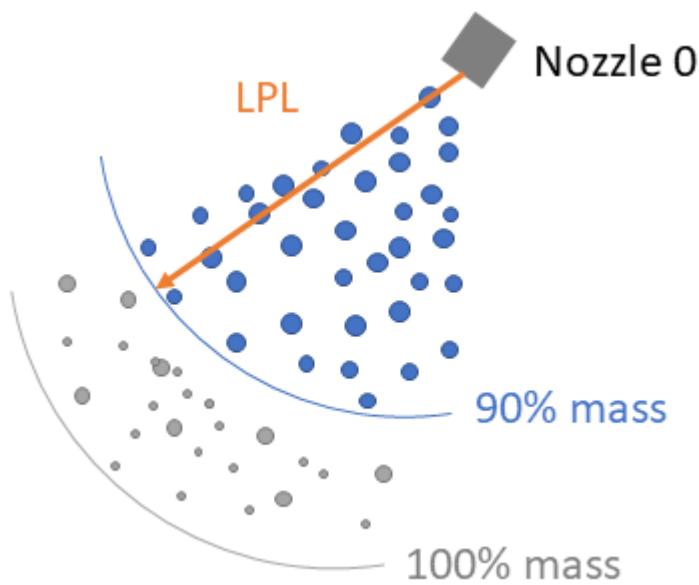


Figure 14.1: Diagram of the liquid penetration length for a penetration fraction of 0.90.

To measure the VPL, CONVERGE requires two parameters: the vapor penetration fraction, `parcel introduction.in > injections > injection > penet_control > vapor_frac`, and the penetration bin size, `parcel introduction.in > injections > injection > penet_control > vapor_bin_size`. CONVERGE identifies cells within the spray cone where the mass fraction of fuel vapor exceeds `vapor_frac`, and then calculates the distance from the nozzle hole center for each of these cells. The largest of these distances is the VPL, which CONVERGE writes to `liquid parcel enc.out`. Only cells with sizes equal to or larger than the bin size are included when calculating the VPL. If the bin size is smaller than all of the cells in the spray cone, CONVERGE uses the parent cells (instead of the embedded cells) when calculating the vapor mass fraction. We recommend a bin size of approximately twice the size of the smallest cells in the spray cone.

14.4 Parcel Settings

This section details parcel settings, which are contained primarily within `parcels.in`.

Liquid Parcels

The following sub-sections contain information about liquid parcels.

Drop Drag and Liquid/Gas Coupling

Accurate determination of drop drag coefficients is critical for accurate liquid spray modeling. If `parcels.in > liquid_parcel > parcel_list > parcel > drag_control > drag_model = NONE`, CONVERGE does not calculate drop drag. If `drag_model = SPHERE`, CONVERGE calculates the drag coefficient with the assumption that the drops are perfect spheres. If `drag_model = DYNAMIC`, CONVERGE determines the drop drag coefficient dynamically, accounting for variations in the drop shape through a drop distortion parameter y . Since values of the drop distortion parameter are determined from the [TAB model](#), a brief description of that model is given here.

The Taylor Analogy Breakup (TAB) model ([O'Rourke and Amsden, 1987](#)) is a classic method for calculating drop distortion and breakup. This method is based on Taylor's analogy between an oscillating and distorting droplet and a spring-mass system. The following table presents the analogous components.

Table 14.1: Comparison of a spring-mass system and a distorting drop.

Spring-mass system	Distorting and oscillating drop
Restoring force	Surface tension forces of spring
External force	Drop drag force
Damping force	Drop viscosity forces

The equation governing a damped, forced oscillator ([O'Rourke and Amsden, 1987](#)) is

$$F - kx - d\dot{x} = m\ddot{x}, \quad (14.15)$$

where x is the displacement of the drop equator from its spherical (undisturbed) position. The coefficients of this equation are taken from Taylor's analogy as

$$\begin{aligned} \frac{F}{m} &= C_F \frac{\rho_g |U_i|^2}{\rho_l r_o} \\ \frac{k}{m} &= C_k \frac{\sigma}{\rho_l r_o^3}, \\ \frac{d}{m} &= C_d \frac{\mu_l}{\rho_l r_o^2} \end{aligned} \quad (14.16)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

where ρ_l and ρ_g are the discrete (liquid) phase and continuous (gas) phase densities, U_i is the relative velocity of the droplet, r_0 is the undisturbed droplet radius, σ is the drop surface tension, and μ_l is the drop viscosity. The dimensionless constants C_F , C_k , and C_d are defined below. Equation 14.15 can be non-dimensionalized by setting $y = x/(C_b r_0)$ and substituting the relationships in Equation 14.16 as

$$\ddot{y} = \frac{C_F}{C_b} \frac{\rho_g U^2}{\rho_l r_0^2} - \frac{C_k \sigma}{\rho_l r_0^3} y - \frac{C_d \mu_l}{\rho_l r_0^2} \frac{dy}{dt}. \quad (14.17)$$

For under-damped drops, the equation governing y can be determined from Equation 14.17 if the relative velocity is assumed to be constant. The result is given by

$$y(t) = We_c + e^{-\frac{t}{t_d}} \left[(y(0) - We_c) \cos(\omega t) + \frac{1}{\omega} \left(\frac{dy}{dt}(0) + \frac{y(0) - We_c}{t_d} \right) \sin(\omega t) \right], \quad (14.18)$$

where

$$We_g = \frac{\rho_g U_{rel}^2 r_0}{\sigma}, \quad (14.19)$$

$$We_c = \frac{C_F}{C_k C_b} We_g, \quad (14.20)$$

$$\frac{1}{t_d} = \frac{C_d}{2} \frac{\mu_l}{\rho_l r_0^2}, \quad (14.21)$$

and

$$\omega^2 = C_k \frac{\sigma}{\rho_l r_0^3} - \frac{1}{t_d^2}. \quad (14.22)$$

In Equation 14.18, ω is the droplet oscillation frequency. In Equation 14.19, We_g is the drop Weber number, a dimensionless parameter defined as the ratio of aerodynamic forces to surface tension forces. Note the aerodynamic forces are calculated with U_{rel} , the parcel's velocity relative to the local Eulerian flow. Finally, the constants have been chosen to match experiments and theory ([Lamb, 1945](#)):

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$\begin{aligned} C_k &= 8, \\ C_F &= \frac{1}{3}, \\ C_b &= \frac{1}{2}. \end{aligned} \quad (14.23)$$

This reference provides a value of $C_d = 5.0$. We recommend setting this parameter to between 5.0 and 10.0 for best results, and CONVERGE Studio sets $C_d = 10.0$ by default.

In CONVERGE, `parcels.in` > `liquid_parcel` > `parcel_list` > `parcel` > `drag_control` > `cfocbck`, `drag_control` > `csubd`, and `drag_control` > `csubk` set the constants in the above equation. Equation 14.18 can be solved for each drop to yield its distortion parameter y .

Many drop drag models assume that a drop remains spherical throughout its lifetime. With this assumption, the drag of a spherical object is determined by (Liu *et al.*, 1993) as

$$C_{D,sphere} = \begin{cases} 0.424 & Re > 1000 \\ \frac{24}{Re} \left(1 + \frac{1}{6} Re^{2/3} \right) & Re \leq 1000 \end{cases}, \quad (14.24)$$

where Re is the drop Reynolds number based on the drop's spherical diameter, the fluid-phase density and laminar viscosity, and the relative velocity between the drop and the gas. However, as an initially spherical droplet moves through a gas, its shape will distort significantly when the Weber number is large. In the extreme case, the drop shape will approach that of a disk. The drag on a disk is significantly higher than that of a sphere. Since the drop drag coefficient is highly dependent on the drop shape, a drag model that assumes the drop is spherical can under-predict drag. The dynamic drag model accounts for the effects of drop distortion by linearly varying the drag between that of a sphere, (Equation 14.24) and a value corresponding to a disk (Liu *et al.*, 1993). The drag coefficient is given by

$$C_D = C_{D,sphere} (1 + 2.632y), \quad (14.25)$$

where y is the drop distortion, as determined from the TAB model described above. Note that in the limit of no distortion ($y = 0$), the drag coefficient of a sphere will be obtained while at maximum distortion ($y = 1$) the drag coefficient corresponding to a disk will be obtained.

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

CONVERGE uses a nearest node approach to exchange mass, momentum, energy terms of a parcel (Lagrangian particle) with the fluid-phase (Eulerian field) values of the computational node that it is closest to. A Taylor series expansion is used to calculate the gas velocity (Eulerian field) at the point of the parcel (Lagrangian particle). The use of the Taylor series expansion significantly reduces grid effects on the liquid spray.

Turbulent Dispersion

CONVERGE models the effects of the turbulent flow on spray drops by adding a fluctuating velocity u'_i to the gas velocity u_i . The following sections describe how u'_i is determined based on the type of turbulence model in use.

You can turn on turbulent dispersion via [*parcels.in* > *liquid_parcels* > *parcel_list* > *parcel* > *turb_dispersion_control* > *dispersion_model*](#). Remember that you must set [*inputs.in* > *solver_control* > *turbulence_solver* = 1](#) to activate turbulence modeling. In addition, specify a value for [*turbulence.in* > *Physics_effects* > *discrete_c_s*](#) or [*turbulence.in* > *LES_models* > *les_c_tke*](#) and a value for [*turbulence.in* > *Physics_effects* > *discrete_c_ps*](#).

RANS and Hybrid RANS/LES Turbulence Models

The [*RANS \(Reynolds-Averaged Navier-Stokes\)*](#) and [*hybrid RANS/LES*](#) turbulence models in CONVERGE include source terms to account for the depletion of turbulent kinetic energy due to work done by turbulent eddies to disperse the liquid spray droplets. The source terms S_s include the fluctuating component of the fluid-phase velocity u'_i as

$$S_s = -\frac{\sum_p N_p (F'_{drag,i} u'_i)_p}{V}, \quad (14.26)$$

where the summation is over all parcels in the cell, N_p is the number of drops in a parcel, V is the cell volume, and

$$F'_{drag,i} = \frac{F_{drag,i}}{(u_i + u'_i - v_i)} u'_i, \quad (14.27)$$

where $F_{drag,i}$ is the drag force on a drop. Note that c_s is a model constant that appears in front of S_s in the ϵ transport equation. You can specify a value for c_s via [*turbulence.in* > *Physics_effects* > *discrete_c_s*](#). Setting c_s to zero deactivates the source term in both the k and ϵ equations.

In the O'Rourke turbulent dispersion model, it is assumed that each component of u'_i follows a Gaussian distribution given by

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$G(u'_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-(u')^2}{2\sigma'^2}\right), \quad (14.28)$$

with a variance σ'^2 given by $2/3k$, where k is the turbulent kinetic energy ([Amsden et al. 1989](#)). It can be shown that the cumulative distribution function for Equation 14.28 is given by

$$\tilde{G}(u'_i) = \text{erf}\left(\frac{u'_i}{\sqrt{2\sigma'^2}}\right) = \text{erf}(\zeta) \quad 0 < \zeta < 2, \quad (14.29)$$

where

$$\zeta = \frac{u'_i}{\sqrt{(4/3)k}}. \quad (14.30)$$

Equation 14.29 is inverted numerically, using Newton's method, to obtain values of ζ for specific values of \tilde{G} . These values are calculated once at the start of the simulation and stored in a table. When a value of ζ is needed for a turbulent dispersion calculation, a random number YY between zero and one is selected which represents the chosen value of \tilde{G} . The corresponding value of ζ is found by interpolating in the table. Once a value of ζ is selected, u'_i is calculated using Equation 14.30. The sign of u'_i is determined based on the parameter $XX = 1-2YY$. Note that this process is conducted for each of the three components of u'_i .

CONVERGE chooses new values of u'_i according to the above procedure once every turbulence correlation time t_d , which is the lesser of the eddy breakup time and the time for a droplet to traverse an eddy. In other words,

$$t_d = \min\left(\frac{k}{\varepsilon}, c_{ps} \frac{k^{3/2}}{\varepsilon} \frac{1}{|u_i + u' - v_i|}\right), \quad (14.31)$$

where c_{ps} is an empirical constant and ε is the dissipation of turbulent kinetic energy. You can set the value for c_{ps} via [turbulence.in](#) > Physics_effects > discrete_c_ps. The drop needs to traverse the eddy length, l_e , which is given by

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$l_e = \frac{c_\mu^{3/4} k^{3/2}}{\varepsilon}. \quad (14.32)$$

CONVERGE also includes a tke-preserving dispersion model. This model chooses each component of u'_i such that $|u'_i| = \sqrt{2k}$. Three random numbers YY_i are first chosen and normalized such that they sum up to unity. Three additional random numbers XX_i are next chosen to determine the sign of each component. The components of the fluctuating velocity are then calculated from

$$u'_i = sign(1 - 2XX_i) \sqrt{2YY_i k} \quad (14.33)$$

where k is the cell turbulent kinetic energy. CONVERGE chooses new values of u'_i after a turbulence correlation time as in the [O'Rourke model](#).

LES Turbulence Models

When [LES \(large eddy simulation\) turbulence](#) is used, the sub-grid velocity is derived from the first term of a Taylor series expansion given by

$$u_{sub,i} = C_{les} \frac{dx^2}{24} \frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j}, \quad (14.34)$$

where d_x is a characteristic cell size given by the cube-root of the cell volume, \bar{u}_i is the resolved velocity field, and C_{les} is a scaling constant. You can specify a value for C_{les} via *turbulence.in > LES_models > les_c_tke*. A random number YY between zero and one is chosen for each of the three components, and the applied sub-grid velocity is given by

$$u'_i = 2XX u_{sub,i}, \quad (14.35)$$

where $XX = 1 - 2YY$ gives the sign of the velocity component. The factor of two is included in Equation 14.35 so that the applied sub-grid velocity of a given component is can be either positive or negative.

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

As in the RANS turbulent dispersion model, new values of u'_i are chosen once every turbulence correlation time t_d given by

$$t_d = \min \left(\frac{k_{sub}}{\varepsilon_{sub}}, c_{ps} \frac{k_{sub}^{3/2}}{\varepsilon_{sub}} \frac{1}{|u_i + u'_i - v_i|} \right), \quad (14.36)$$

where k_{sub} and ε_{sub} are derived from the sub-grid velocity field and the grid size. In other words,

$$k_{sub} = \frac{1}{2} u_{sub,i} u_{sub,i} \quad (14.37)$$

and

$$\varepsilon_{sub} = \frac{k_{sub}^{3/2}}{dx}. \quad (14.38)$$

Note that the form of Equations 14.36 through 14.38 is used so that the time scales can be written similarly to those in a RANS case. It can be shown that with the definitions given by Equations 14.37 and 14.38, the time scales are proportional to the grid size over a velocity magnitude. In other words,

$$\frac{k_{sub}}{\varepsilon_{sub}} \sim \frac{dx}{|u_{sub,i}|} \quad (14.39)$$

and

$$c_{ps} \frac{k^{3/2}}{\varepsilon} \frac{1}{|u_i + u' - v_i|} \sim \frac{dx}{|u_i + u' - v_i|}. \quad (14.40)$$

Thus, the expressions on the right-hand-sides of Equations 14.39 and 14.40 could also be used to define the two time scales. Note that c_{ps} is specified by [*turbulence.in > Physics_effects > discrete_c_ps*](#).

Spray Breakup

CONVERGE includes several models to predict liquid parcel atomization and concentrated spray breakup, including models based on the [Kelvin-Helmholtz](#) (KH) and [Rayleigh-Taylor](#) (RT) instability mechanisms, the [Taylor Analogy Breakup](#) (TAB) model, and the [LISA sheet breakup](#) model. This section describes these models. Most of the spray breakup model parameters are located in `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control`.

CONVERGE also includes an [Eulerian-Lagrangian Spray Atomization](#) (ELSA) model for predicting primary breakup of liquid sprays, which is described later in this chapter.

Spray breakup parameters are parcel-specific, which allows you to run different spray breakup models and constants for different liquid injection configurations.

Kelvin-Helmholtz Breakup Model

Set `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > breakup_model = KH` to activate the Kelvin-Helmholtz (KH) model. KH model parameters are located in the `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > kh` settings block.

The Kelvin-Helmholtz instability is based on a liquid jet stability analysis that is described in detail by [Reitz and Bracco \(1986\)](#) and is only briefly described here. The analysis considers the stability of a cylindrical, viscous, liquid jet of radius r_0 issuing from a circular orifice at a velocity U into a stagnant, incompressible, inviscid gas of density ρ_g . The liquid has a density ρ_l and viscosity μ_l and a cylindrical polar coordinate system is used which moves with the jet. An arbitrary infinitesimal axisymmetric surface displacement of the form

$$\eta = \eta_o e^{ikz+\omega t}, \quad (14.41)$$

is imposed on the initially steady motion and it is thus desired to find the dispersion relation $\omega_{KH} = \omega_{KH}(k_{KH})$, which relates the real part of the growth rate ω_{KH} to its wavenumber $k_{KH} = 2\pi / \lambda_{KH}$.

In order to determine the dispersion relation, the linearized hydrodynamic equations for the liquid are solved with wave solutions of the form

$$\begin{aligned} \phi_l &= C_1 I_0(k_{KH} r) e^{ik_{KH} z + \omega_{KH} t} \\ \psi_l &= C_2 r I_1(Lr) e^{ik_{KH} z + \omega_{KH} t}, \end{aligned} \quad (14.42)$$

where ϕ_l and ψ_l are the velocity potential and stream function, respectively, C_1 and C_2 are integration constants, I_0 and I_1 are modified Bessel functions of the first kind, $L^2 = k_{KH}^2 + \omega_{KH}^2 / \nu_l$, and ν_l is the liquid kinematic viscosity ([Reitz, 1987](#)). The liquid pressure is

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

obtained from the inviscid part of the liquid equations. In addition, the inviscid gas equations can be solved to obtain the fluctuating gas pressure at $r = r_p$:

$$p_g = -\rho_g \left(U - i \frac{\omega_{KH}}{k_{KH}} \right)^2 k_{KH} \eta \frac{K_0(k_{KH} r_p)}{K_1(k_{KH} r_p)}, \quad (14.43)$$

where K_0 and K_1 are modified Bessel functions of the second kind and U is the relative velocity between the liquid and the gas. The linearized boundary conditions are

$$\nu_l = \frac{\partial \eta}{\partial t}, \quad (14.44)$$

$$\frac{\partial u_l}{\partial r} = -\frac{\partial v_l}{\partial z} \quad (14.45)$$

and

$$-p_l + 2\mu_l \frac{\partial v_l}{\partial r} - \frac{\sigma}{r_p^2} \left(\eta + r_p^2 \frac{\partial^2 \eta}{\partial z^2} \right) + p_g = 0, \quad (14.46)$$

which are mathematical statements of the liquid kinematic free surface condition, continuity of shear stress, and continuity of normal stress, respectively. Note that u_l is the axial perturbation liquid velocity, v_l is the radial perturbation liquid velocity, and σ is the surface tension. Also note that Equation 14.45 was obtained under the assumption that $v_g = 0$.

As described by [Reitz \(1987\)](#), Equations 14.44 and 14.45 can be used to eliminate the integration constants C_1 and C_2 in Equation 14.42. Thus, when the pressure and velocity solutions are substituted into Equation 14.46, the desired dispersion relation is obtained:

$$\begin{aligned} \omega_{KH}^2 + 2\nu_l k_{KH}^2 \omega_{KH} \left[\frac{I'_1(k_{KH} r_p)}{I_0(k_{KH} r_p)} - \frac{2k_{KH} L}{k_{KH}^2 + L^2} \frac{I_1(k_{KH} r_p) I'_1(L r_p)}{I_0(k_{KH} r_p) I_1(L r_p)} \right] = \\ \frac{\sigma k_{KH}}{\rho_l r_p^2} \left(1 - k_{KH}^2 r_p^2 \right) \left(\frac{L^2 - r_p^2}{L^2 + r_p^2} \right) \frac{I_1(k_{KH} r_p)}{I_0(k_{KH} r_p)} + \\ \frac{\rho_g}{\rho_l} \left(U - i \frac{\omega_{KH}}{k_{KH}} \right)^2 \left(\frac{L^2 - r_p^2}{L^2 + r_p^2} \right) \frac{I_1(k_{KH} r_p)}{I_0(k_{KH} r_p)} \frac{K_0(k_{KH} r_p)}{K_1(k_{KH} r_p)} \end{aligned} \quad (14.47)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

As shown by [Reitz \(1987\)](#), Equation 14.47 predicts that a maximum growth rate (or most unstable wave) exists for a given set of flow conditions. Curve fits of numerical solutions to Equation 14.47 were generated for the maximum growth rate, Ω_{KH} , and the corresponding wavelength, Λ_{KH} , and are given by Reitz (1987) as

$$\frac{\Lambda_{KH}}{r_p} = 9.02 \frac{(1 + 0.45Oh^{0.5})(1 + 0.4T^{0.7})}{(1 + 0.87We_g^{1.67})^{0.6}}, \quad (14.48)$$

and

$$\Omega_{KH} \left[\frac{\rho_l r_p^3}{\sigma} \right]^{0.5} = \frac{(0.34 + 0.38We_g^{1.5})}{(1 + Oh)(1 + 1.4T^{0.6})}, \quad (14.49)$$

where $Oh = \sqrt{We_l} / Re_l$ is the Ohnesorge number and $T = Oh\sqrt{We_g}$ is the Taylor number. Furthermore, $We_l = \rho_l U^2 r_p / \sigma$ and $We_g = \rho_g U^2 r_p / \sigma$ are the liquid and gas Weber numbers, respectively, and $Re_l = Ur_p / v_l$ is the Reynolds number. Note that $U = |u_i \cdot v_i|$ where u_i is the fluid-phase velocity and v_i is the drop velocity.

In the KH model, the initial parcel diameters are set equal to the nozzle hole diameter d_0 and the atomization process of the relatively large injected blobs is modeled using the stability analysis for liquid jets as described above. The breakup of the parcels and resulting drops is calculated by assuming that the breakup drop radius r_c is proportional to the wavelength of the fastest growing unstable surface wave given by Equation 14.48. In other words,

$$r_c = B_0 \Lambda_{KH}, \quad (14.50)$$

where B_0 is a model constant typically set to 0.61 based on the work of [Reitz \(1987\)](#). Although B_0 in Equation 14.50 is typically set to 0.61, it is a user-specified parameter ([parcels.in > liquid_parcel > parcel_list > parcel > breakup_control > kh > size_const](#)) to allow adjustment if necessary. A smaller value of B_0 will result in smaller drops from breakup, while a larger value will result in larger drops.

The rate of change of drop radius in a parent parcel is given by

$$\frac{dr_p}{dt} = -\frac{(r_p - r_c)}{\tau_{KH}}, \quad (r_c \leq r_p) \quad (14.51)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

where the breakup time τ_{KH} is given by

$$\tau_{KH} = \frac{3.726 B_1 r_p}{\Lambda_{KH} \Omega_{KH}}, \quad (14.52)$$

and Λ_{KH} and Ω_{KH} are obtained from Equations 14.48 and 14.49, respectively. The breakup time constant B_1 ([parcels.in > liquid parcels > parcel_list > parcel > breakup_control > time_const](#)) is related to the initial disturbance level on the liquid jet and has been found to vary from one injector to another ([Kong et al., 1995](#)). Note that CONVERGE represents Equation 14.51 as

$$\frac{r_p^{n+1} - r_p^n}{dt} = -\frac{(r_p^{n+1} - r_c)}{\tau_{KH}}. \quad (14.53)$$

If the drop has existed for at least τ_{KH} , CONVERGE allows the drop to grow according to Equation 14.53 above. You can deactivate this enlargement step ([parcels.in > liquid parcels > parcel_list > parcel > breakup_control > no_enlarge_flag](#)) if the simulation contains excessively large drops.

You can run the KH model with or without the creation of new child parcels. When droplet breakup occurs, pieces of the original droplet fragment away. CONVERGE considers these fragmented masses of liquid to be child parcels when they accumulate a sufficient amount of mass. The mass of the fragmented liquid is given by

$$\sum_n s N^n \frac{4}{3} \pi \rho_l \left[(r_p^n)^3 - (r_p^{n+1})^3 \right], \quad (14.54)$$

where s is the shed factor ([parcels.in > liquid parcels > parcel_list > parcel > breakup_control > kh > shed_factor](#)), which represents the fraction of the parent parcel mass that contributes to the child parcel mass. When the mass of the fragmented liquid exceeds the value of [parcels.in > liquid parcels > parcel_list > parcel > breakup_control > kh > new_parcel_cutoff](#), CONVERGE creates a child parcel with a drop size of radius r_c . The *new_parcel_cutoff* parameter is typically 3 to 5 percent of the average injected parcel mass (*i.e.*, the total injected liquid mass divided by the total number of injected parcels). Note that CONVERGE conserves the liquid mass during the breakup mass accumulation process as the number of drops in a parent parcel is adjusted to a value N such that $N^{n+1} (r_p^{n+1})^3 = N^n (r_p^n)^3$.

When breakup occurs and it is determined that child droplets should be added to the computation, they are given a velocity component normal to the path of the parent drop. This normal velocity is determined by

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$v_n = C_1 \Lambda_{KH} \Omega_{KH}, \quad (14.55)$$

where C_1 is a model constant.

Alternatively, a similar model to that of [Reitz and Diwakar \(1987\)](#) can be used if you choose to not create new parcels from the KH breakup model. In the Reitz and Diwakar model, Equation 14.51 was used to model the size change of an unstable droplet with the KH breakup time given by [a later equation](#). Equation 14.52 reduces to [a later equation](#) for very high gas Weber numbers We_g and $\mu_l = 0$. In addition, Reitz and Diwakar used the so-called stripping regime breakup criterion

$$\frac{We_g}{Re_g^{0.5}} > 0.5 \quad (14.56)$$

to determine the breakup drop size given by

$$r_c = \frac{\sigma^2}{2\rho_g^2 U^3 \nu_g}. \quad (14.57)$$

Alternatively, in the limit of very high gas Weber numbers and an inviscid liquid, Equation 14.50 reduces to

$$r_c = 9.8 B_o \frac{\sigma}{\rho_g U^2}, \quad (14.58)$$

which is equivalent to the bag regime breakup criterion

$$We_g > 6.0 \quad (14.59)$$

for a B_o value of approximately 0.61. It has been noted by [Reitz \(1987\)](#) that very similar drop sizes are obtained if Equation 14.58 is always used to determine the breakup drop size. As a result, running the present KH model without the addition of new child parcels is consistent with the [Reitz and Diwakar \(1987\)](#) model for high-speed, inviscid sprays. If you wish to use the CONVERGE model that is similar to that of [Reitz and Diwakar \(1987\)](#), you need to use this KH model, not the [KH-RT model](#).

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

KH-ACT Model

Set [parcels.in](#) > [liquid_parcels](#) > [parcel_list](#) > [parcel](#) > [breakup_control](#) > [breakup_model](#) = KH-ACT to activate the KH-ACT model. KH-ACT model parameters are located in the [parcels.in](#) > [liquid_parcels](#) > [parcel_list](#) > [parcel](#) > [breakup_control](#) > [khact](#) settings block.

The KH-ACT model ([Som and Aggarwal, 2010](#)) is a modification of the [KH model](#) that includes the effects of [aerodynamics](#), [cavitation](#), and [turbulence](#) on primary breakup.

Aerodynamic-Induced Breakup

CONVERGE uses the KH model described previously to calculate the instantaneous length and time scales for every parcel as follows:

$$L_{KH} = r - r_{KH} \quad (14.60)$$

and

$$\tau_{KH} = \frac{3.276 B_1 r}{\Omega_{KH} \Lambda_{KH}}. \quad (14.61)$$

CONVERGE calculates the ratio of length and time scales for each process. As seen in Equations 14.60 and 14.63, the rate of decrease in droplet radius $\left(\frac{dr}{dt}\right)$ is proportional to the

ratio of length to time scale $\left(\frac{L_A}{\tau_A}\right)$. Thus the largest ratio determines the dominant breakup

process:

$$\frac{L_A}{\tau_A} = \max \left\{ \frac{L_{KH}(t)}{\tau_{KH}(t)}, \frac{L_{CAV}}{\tau_{CAV}}, \frac{L_T(t)}{\tau_T(t)} \right\}. \quad (14.62)$$

If the aerodynamically induced breakup process is dominant, then the KH model, as represented by Equation 14.60, is employed for primary atomization. However, if cavitation or turbulence processes dominate, then the following breakup law is used:

$$\frac{dr}{dt} = -C_{T,CAV} \frac{L_A}{\tau_A}, \quad (14.63)$$

where $C_{T,CAV}$ is the model constant (specified as [parcels.in](#) > [liquid_parcels](#) > [parcel_list](#) > [parcel](#) > [breakup_control](#) > [khact](#) > [c_tcav](#)) whose value ranges from 0.1 to 1.0.

Cavitation-Induced Breakup

Cavitation patterns generated inside the injector nozzle can reach the nozzle exit. These patterns decrease the radius of the injected parcels from the orifice radius, thereby enhancing jet atomization. The underlying assumption is that cavitation patterns are transported to the jet periphery by the turbulence velocity inside the liquid, and either burst at the periphery or collapse before reaching it. For both of these cases, a characteristic time scale is calculated, the smaller one causing breakup. Following [Bianchi and Pelloni \(1999\)](#) and [Arcoumanis and Gavaises \(1998\)](#), the characteristic cavitation time scale (τ_{CAV}) is calculated as

$$\tau_{CAV} = \min(\tau_{collapse} : \tau_{burst}). \quad (14.64)$$

All of the cavitation bubbles formed inside the orifice that reach the orifice exit are lumped together into a single artificial bubble that occupies the cumulative area of the smaller ones:

$$R_{CAV} = r_{hole} \sqrt{(1 - C_a)}, \quad (14.65)$$

where R_{CAV} is the effective radius of an equivalent bubble from the nozzle. The area reduction coefficient (C_a) is calculated from flow simulations inside the injector and r_{hole} is the exit radius of the nozzle orifice.

Bubble collapse depends on the size of the bubble. This approach of lumping together to form an effective bubble may marginally effect the breakup process. The bubble collapse time is calculated from Rayleigh Plesset theory ([Brennen, 1995](#)) as

$$\tau_{collapse} = 0.9145 R_{CAV} \sqrt{\frac{\rho_l}{p_v}}, \quad (14.66)$$

where p_v is the fuel vapor pressure and ρ_l the fuel density.

Cavitation bubbles are located along the walls of the orifice; however, the effective bubble is placed on the center of the liquid spray as it is injected. The average time required for a cavitation bubble to reach the periphery of the jet can be estimated as

$$\tau_{burst} = \frac{r_{hole} - R_{CAV}}{u_{urb}}, \quad (14.67)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

where the turbulent velocity $u'_{turb} = \sqrt{\frac{2K(t)}{3}}$ is obtained from inner nozzle flow simulations.

The length scale for the cavitation-induced breakup is calculated as

$$L_{CAV} = R_{CAV}. \quad (14.68)$$

CONVERGE will determine the length and time scales for cavitation-induced breakup in this manner. Each cavitation bubble influences the breakup only once. In other words, after the bubble collapses, it does not affect parcels during subsequent time-steps.

Turbulence-Induced Breakup

According to [Huh and Gosman \(1991\)](#), turbulent fluctuations in the liquid jet are responsible for the initial perturbations on the jet surface. These waves grow according to KH instabilities until they breakup from the surface. The relevant length and time scales ($L_T(t)$ and $\tau_t(t)$, respectively) for turbulence-induced breakup are calculated as

$$L_T(t) = \frac{C_\mu K(t)^{1.5}}{\varepsilon(t)} \quad (14.69)$$

and

$$\tau_T(t) = \frac{C_\mu K(t)}{\varepsilon(t)}, \quad (14.70)$$

where $K(t)$ and $\varepsilon(t)$ are, respectively, the instantaneous turbulent kinetic energy and dissipation rate, and C_μ and C_ε are turbulence model constants. Assuming isotropic turbulence for the liquid phase and neglecting the diffusion, convection, and production terms in the k - ε equation, the decay of $K(t)$ and $\varepsilon(t)$ for a parcel can be estimated as

$$K(t) = \left\{ \frac{(K_0)^{C_\varepsilon}}{K_0 + \varepsilon_0 t (C_\varepsilon - 1)} \right\}^{1/(C_\varepsilon - 1)} \quad (14.71)$$

and

$$\varepsilon(t) = \varepsilon_0 \left\{ \frac{K(t)}{K_0} \right\}^{C_\varepsilon}, \quad (14.72)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

where K_0 and ε_0 are the initial values at the nozzle exit at start of injection (SOI), determined from nozzle flow simulations. In the absence of detailed nozzle flow simulations and measurements, the values of K_0 and ε_0 are estimated from a force balance in a control volume, not accounting for the decay in turbulence levels.

CONVERGE allows two methods for determining the values of turbulent kinetic energy, turbulent dissipation, and the area reduction factor for use in KH-ACT computations. To use the first method, where CONVERGE calculates the values of these three values, set [parcel introduction.in > injections > injection > injection_control > khact_model > nozzle_flow_data = CONVERGE](#). The second method (*nozzle_flow_data = <filename>*) is to provide a *.in file that specifies values for these three parameters at different simulation times.

For the control volume along the nozzle downstream length, the turbulent stress and force on the nozzle orifice are estimated to be of the same order of magnitude, *i.e.*,

$$\rho_l u_l^2 \pi D L = \Delta p_{noz} \frac{\pi D^2}{4}, \quad (14.73)$$

where ρ_l and u_l are the liquid and turbulent fluctuation velocity, L , D are length and diameter of the nozzle orifice, respectively, and Δp_{noz} is the nozzle downstream pressure drop, which is obtained as

$$\Delta p_{tot} = \Delta p_{noz} + \Delta p_{form} + \Delta p_{acc}. \quad (14.74)$$

The total pressure drop, which is composed of three components, is obtained as

$$\Delta p_{tot} = \frac{1}{C_d^2} \frac{\rho_l U_{inj}^2}{2}. \quad (14.75)$$

Form loss pressure drop is estimated as

$$\Delta p_{form} = K_c \frac{\rho_l U_{inj}^2}{2}. \quad (14.76)$$

Acceleration of pressure drop is estimated as

$$\Delta p_{acc} = (1 - s^2) \frac{\rho_l U_{inj}^2}{2}. \quad (14.77)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

The initial turbulent kinetic energy and turbulent dissipation rate can be readily estimated from the above equations as

$$K_0 = \frac{U_{inj}^2}{8\left(\frac{L}{D}\right)} \left\{ \frac{1}{C_d^2} - K_c - (1-s^2) \right\} \quad (14.78)$$

and

$$\varepsilon_0 = K_\varepsilon \frac{U_{inj}^3}{2L} \left\{ \frac{1}{C_d^2} - K_c - (1-s^2) \right\}. \quad (14.79)$$

The model constants used in CONVERGE are $K_c = 0.45$, $K_\varepsilon = 0.27$, and $s = 0.01$, which are based on available literature. You can change these values by specifying `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > khact > turb_ke`, `turb_ke`, and `turb_s`, respectively. Once the initial turbulence levels K_0 and ε at the nozzle exit are known, the length and time scales can be estimated based on the Equations 14.69 and 14.70 for a $k-\varepsilon$ turbulence model.

Rayleigh-Taylor Breakup Model

Set `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > breakup_model = RT` to activate the Rayleigh-Taylor (RT) model. RT model parameters are located in the `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > rt` settings block.

In addition to the KH breakup mechanism, the Rayleigh-Taylor instability is also believed to be responsible for droplet breakup. The unstable RT waves are thought to occur due to the rapid deceleration of the drops from the magnitude of the drag force, $|F_{D,i}|$, which is given as

$$|F_{D,i}| = M_d |a_i| = M_d \frac{3}{8} C_D \frac{\rho_g |U_i|^2}{\rho_l r_o}, \quad (14.80)$$

where $|a_i|$ is the deceleration of the drop, m_d is the mass of the drop, and C_D is the drag coefficient. Typical implementations of the RT breakup model ignore both gas and liquid viscosity (Ricart et al., 1997). If viscosity is neglected, the fastest growing wavelength for the RT instabilities is given by (Xin et al., 1998)

$$\Lambda_{RT} = 2\pi \sqrt{\frac{3\sigma}{a(\rho_l - \rho_g)}} \quad (14.81)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

and the corresponding growth rate is given by

$$\Omega_{RT} = \sqrt{\frac{2}{3\sqrt{3}\sigma} \left[a(\rho_l - \rho_g) \right]^{3/2}}. \quad (14.82)$$

As described by Senecal [et al.](#) (2007), the CONVERGE spray models extend the standard RT model to include viscosity, which can have a large effect for the high decelerations typical of spray droplets. When viscosity is included, it can be shown (*e.g.*, Joseph [et al.](#) 1999) that the RT growth rate can be written as

$$\omega_{RT} = -k_{RT}^2 \left(\frac{\mu_l + \mu_g}{\rho_l + \rho_g} \right) + \sqrt{k_{RT} \left(\frac{\rho_l - \rho_g}{\rho_l + \rho_g} \right) a - \frac{k_{RT}^3 \sigma}{\rho_l + \rho_g} + k_{RT}^4 \left(\frac{\mu_l + \mu_g}{\rho_l + \rho_g} \right)^2}, \quad (14.83)$$

where k_{RT} is the wavenumber.

In CONVERGE, the wavenumber corresponding to the maximum growth rate ($K_{RT} = 2\pi / \Lambda_{RT}$) is solved for numerically using a bisection method with Equation 14.83. The value of K_{RT} is then substituted into Equation 14.83 to find the maximum growth rate, Ω_{RT} .

It is straightforward to show that Equation 14.83 reduces to the standard RT model in the limit of no viscosity. If viscosity is neglected, Equation 14.83 becomes

$$\omega_{RT} = \sqrt{k_{RT} \left(\frac{\rho_l - \rho_g}{\rho_l + \rho_g} \right) a - \frac{k_{RT}^3 \sigma}{\rho_l + \rho_g}}. \quad (14.84)$$

It can be shown that the values of k_{RT} and Ω_{RT} corresponding to Equation 14.84 are equivalent to Equations 14.81 and 14.82. To show this equivalency, set $d\omega_{RT} / dk_{RT} = 0$ with ω_{RT} defined by Equation 14.84 and substitute the resulting expression for k_{RT} into Equation 14.84.

As in the study of Xin [et al.](#) (1998), if the scaled wavelength given by $C_{RT}\Lambda_{RT}$ is calculated to be smaller than the droplet diameter, RT waves are assumed to be growing on the surface of the drop. When the RT waves have been growing for a sufficient time (*i.e.*, C_l / Ω_{RT}) where C_l is a constant), the drop is broken up according to the RT mechanism. Note that the RT size constant C_{RT} can be increased or decreased to change the size of the predicted RT breakup radius by changing the value of [`parcels.in > liquid_parcel > parcel_list > parcel >`](#)

breakup_control > rt > size_const. Similarly, the RT time constant C_t can be increased to delay RT breakup or decreased to promote faster RT breakup by changing *rt > time_const*.

KH-RT Breakup Model

Set *parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > breakup_model = KH-RT* to activate the Rayleigh-Taylor (RT) model. KH-RT model parameters are located in the *parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > kh* and *parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > rt* settings blocks.

The previous sections described two breakup models based on fundamental liquid/gas instability mechanisms. CONVERGE allows you to run these models concurrently, as is commonly done. One option for running them together is to use the so-called KH-RT breakup model. If the KH-RT breakup model is activated, an intact core or breakup length L_b (see Fig. 11.1) can be specified as

$$L_b = C_{bl} \sqrt{\frac{\rho_l}{\rho_g}} d_0. \quad (14.85)$$

As shown below in Figure 14.2, this model assumes that only KH instabilities are responsible for drop breakup inside of the characteristic breakup distance, L_b , while both KH and RT mechanisms are activated beyond the breakup length. In this case, CONVERGE first checks if the RT mechanism can break up the droplet. If not, the KH mechanism is responsible for breakup.

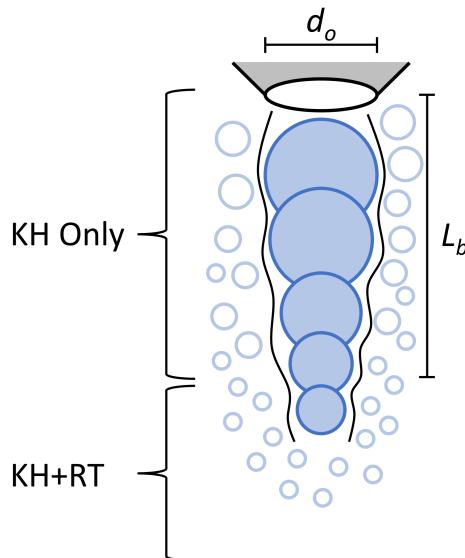


Figure 14.2: Schematic of the KH-RT spray breakup model. Note that liquid blobs are injected with a diameter equal to that of the injector nozzle. In addition, the KH breakup mechanism is applied to a droplet throughout its lifetime, while the RT mechanism is initiated only after the drop reaches a characteristic distance, L_b , from the injector.

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

As [described previously](#), the KH breakup time constant B_1 has been found to vary from one injector to another. Typical values for B_1 are in the range of 5 to 100, where small values result in faster breakup. In addition, the breakup length constant C_{bl} in Equation 14.85 can be tuned to increase or decrease spray breakup by changing `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > rt > length_const`. A methodology for tuning C_{bl} has been described by [Senecal \(2000\)](#), which showed that for very high gas Weber numbers We_g (typical of diesel sprays) and $\mu_l = 0$, the [KH model breakup time](#) reduces to

$$\tau_{KH} = \frac{B_1}{U} \sqrt{\frac{\rho_l}{\rho_g}} r_o, \quad (14.86)$$

which results in a breakup length $L_{KH} = U\tau_{KH}$ of

$$L_{KH} = B_1 \sqrt{\frac{\rho_l}{\rho_g}} r_o. \quad (14.87)$$

If Equation 14.87 is compared with Equation 14.85, it is clear that C_{bl} must equal $B_1/2$ in order for L_b and L_{KH} to be consistent. Tuning the two spray constants together as described above has been shown to result in accurate predictions of vaporizing spray penetration when compared to experiments ([Beale, 1999](#)). The breakup length can be removed from the model by setting $C_{bl} = 0$ in Equation 14.85, which results in both the KH and RT mechanisms acting on droplets as they exit the nozzle ([Patterson, 1997](#)).

Modified KH-RT Model

Set `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > breakup_model = MOD-KH-RT` to activate the modified KH-RT model. Modified KH-RT model parameters are located in the `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > kh` and `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > rt` settings blocks.

An alternative to the KH-RT breakup model allows you to run a simulation with both breakup mechanisms without the use of an ad hoc breakup length definition. In this modified KH-RT model, aerodynamic instabilities (*i.e.*, KH waves) are responsible for the primary breakup of the injected liquid blobs (also known as parents). Child drops are created during this process, and the secondary breakup of these drops is modeled by examining the competing effects of the KH and RT mechanisms.

This modified KH-RT model can be used only when the creation of child parcels is included in the KH breakup model (*i.e.*, `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > kh > new.Parcel_flag = 1`).

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

TAB Breakup Model

Set `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > breakup_model = TAB` to activate the Taylor Analogy Breakup (TAB) model. TAB model parameters are located in the `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > tab` settings block.

The Taylor Analogy Breakup model was previously briefly described in the [Drop Drag Models](#) section. That description is repeated and extended here in the context of drop breakup.

The Taylor Analogy Breakup model ([O'Rourke and Amsden, 1987](#)) is a classic method for calculating drop distortion and breakup. This method is based on Taylor's analogy between an oscillating and distorting droplet and a spring-mass system.

The equation governing a damped, forced oscillator ([O'Rourke and Amsden, 1987](#)) is

$$F - kx - d\dot{x} = m\ddot{x}, \quad (14.88)$$

where x is the displacement of the drop equator from its spherical (undisturbed) position. The coefficients of this equation are taken from Taylor's analogy as

$$\frac{F}{m} = C_F \frac{\rho_g |U_i|^2}{\rho_l r_o},$$

$$\frac{k}{m} = C_k \frac{\sigma}{\rho_l r_o^3}, \quad (14.89)$$

$$\frac{d}{m} = C_d \frac{\mu_l}{\rho_l r_o^2},$$

where ρ_l and ρ_g are the discrete (liquid) phase and continuous (gas) phase densities, U is the relative velocity of the droplet, r_0 is the undisturbed droplet radius, σ is the drop surface tension, and μ_l is the drop viscosity. The dimensionless constants C_F , C_k , and C_d are defined below. Equation 14.88 can be non-dimensionalized by setting $y = x / (C_b r_0)$ and substituting the relationships in Equation 14.89 as

$$\ddot{y} = \frac{C_F}{C_b} \frac{\rho_g}{\rho_l} \frac{U^2}{r_o^2} - \frac{C_k \sigma}{\rho_l r_o^3} y - \frac{C_d \mu_l}{\rho_l r_o^2} \frac{dy}{dt}. \quad (14.90)$$

For under-damped drops, the equation governing y can be determined from Equation 14.90 if the relative velocity is assumed to be constant. The result is given by

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$y(t) = We_c + e^{-\frac{t}{t_d}} \left[(y - We_c) \cos(\omega t) + \frac{1}{\omega} \left(\frac{dy}{dt}(0) + \frac{y(0) - We_c}{t_d} \right) \sin(\omega t) \right], \quad (14.91)$$

where

$$We_g = \frac{\rho_g U^2 r_o}{\sigma}, \quad (14.92)$$

$$We_c = \frac{C_F}{C_k C_b} We_g, \quad (14.93)$$

$$\frac{1}{t_d} = \frac{C_d}{2} \frac{\mu_l}{\rho_l r_o^2}, \quad (14.94)$$

and

$$\omega^2 = C_k \frac{\sigma}{\rho_l r_o^3} - \frac{1}{t_d^2}. \quad (14.95)$$

In Equation 14.91, ω is the droplet oscillation frequency. In Equation 14.92, We_g is the drop Weber number, which is a dimensionless parameter defined as the ratio of aerodynamic forces to surface tension forces. Finally, the constants have been chosen to match experiments and theory ([Lamb, 1945](#)), as follows:

$$\begin{aligned} C_k &= 8, \\ C_F &= \frac{1}{3}, \\ C_b &= \frac{1}{2}. \end{aligned} \quad (14.96)$$

This reference provides a value of $C_d = 5.0$. We recommend setting this parameter to between 5.0 and 10.0 for best results, and CONVERGE Studio sets $C_d = 10.0$ by default.

The numerical implementation of the TAB model for breakup calculations follows the description presented by [O'Rourke and Amsden \(1987\)](#). The drop parameters We_g , t_d and ω^2 are first calculated. If ω^2 is less than or equal to zero (implying that the drop's oscillations are

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

negligible), y and \dot{y} are both set to zero and no further breakup calculations are performed for the drop during the current time-step. If ω^2 is positive, the amplitude of the undamped oscillation is calculated as

$$A = \sqrt{(y - We_c)^2 + (\dot{y} / \omega)^2}. \quad (14.97)$$

If $A + We_c \leq 1.0$, breakup does not occur for the drop for the current time-step. In this case, y and \dot{y} are updated and no further breakup calculations are performed for the drop during the current time-step. The distortion parameters are then updated with the following expressions:

$$y^{n+1} = We_c + e^{-\frac{dt}{t_d}} \left[(y^n - We_c) \cos(\omega t) + \frac{1}{\omega} \left(\dot{y}^n + \frac{y^n - We_c}{t_d} \right) \sin(\omega t) \right] \quad (14.98)$$

and

$$\dot{y}^{n+1} = \frac{We_c - y^{n+1}}{t_d} + \omega e^{-\frac{dt}{t_d}} \left[\frac{1}{\omega} \left(\dot{y}^n + \frac{y^n - We_c}{t_d} \right) \cos(\omega t) - (y^n - We_c) \sin(\omega t) \right], \quad (14.99)$$

where the superscript $n+1$ refers to the updated values and the superscript n refers to the previous values.

Alternatively, if $A + We_c > 1.0$, then breakup is possible for the current drop. In this case, the breakup time is calculated assuming that the drop oscillation is undamped for its first period ([O'Rourke and Amsden, 1987](#)). Breakup is predicted to occur if the computational time-step is larger than the breakup time or if $y \geq 1$. If breakup occurs, y is set to 1, \dot{y} is set to that of an undamped oscillation at the breakup time, and a normal drop velocity component V_n is calculated via

$$V_n = 0.5 r_0 \dot{y}, \quad (14.100)$$

where r_0 is the drop radius prior to breakup. The direction of V_n is randomly chosen in a plane normal to the drop's relative velocity. The normal velocity component is added to the drop velocity.

The breakup drop radius r is calculated with the following expression derived by [O'Rourke and Amsden \(1987\)](#) as

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$r = \frac{r_o}{1 + \frac{8K}{20}y^2 + \frac{\rho_l r_o^3}{\sigma} \dot{y}^2 \left(\frac{6K - 5}{120} \right)}, \quad (14.101)$$

where $K = 10/3$. If a drop size distribution is not used with the TAB breakup model, the new radius is set to the value given by Equation 14.101.

If a drop size distribution is used, the radius calculated from Equation 14.101 is the Sauter mean radius of the distribution. Once the new radius is calculated, the number of drops for the parcel is updated to conserve mass. Once the drop's breakup properties have been updated, it is assumed that the drop is no longer oscillating and hence both y and \dot{y} are set to zero.

If breakup is not predicted to occur for the current time-step, y and \dot{y} are updated according to Equations 14.98 and 14.99.

LISA Breakup Model

Set `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > breakup_model = LISA` to activate the Linearized Instability Sheet Atomization (LISA) model. LISA model parameters are located in the `parcels.in > liquid_parcels > parcel_list > parcel > breakup_control > lisa` settings block.

The Linearized Instability Sheet Atomization model of [Senecal et al. \(1999\)](#) can be used to model liquid sheet breakup. The model includes two parts—a general liquid sheet breakup mechanism and a liquid injection methodology specifically for pressure-swirl atomizers. In this section, the breakup model is first described, followed by a description of the liquid injection model.

Senecal et al. [\(1999\)](#) derived the following dispersion relation for the sinuous mode for a two-dimensional, viscous, incompressible liquid sheet of thickness $2h$ moving with a velocity U through a quiescent, inviscid, incompressible gas medium:

$$\begin{aligned} \omega^2 [\tanh(kh) + Q] + \omega [4\nu_l k^2 \tanh(kh) + 2iQkU] + 4\nu_l^2 k^4 \tanh(kh) \\ - 4\nu_l^2 k^3 L \tanh(Lh) - QU^2 k^2 + \frac{\sigma k^3}{\rho_l} = 0, \end{aligned} \quad (14.102)$$

where $\omega = \omega_r + i\omega_i$ is the complex growth rate, k is the wavenumber, ρ_l is the liquid density, ρ_g is the gas density, ν_l is the liquid kinematic viscosity, σ is the surface tension, $Q = \rho_g / \rho_l$, and $L^2 = k^2 + \omega / \nu_l$. The sinuous mode solution was shown to have the property that its maximum growth rate will always be greater than or equal to the maximum growth rate of varicose waves for high velocity flows with values of Q significantly less than one ([Senecal et al., 1999](#)).

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

In order to simplify Equation 14.102 for use in multi-dimensional simulations, [Senecal et al. \(1999\)](#) performed an order of magnitude analysis using typical values from the inviscid solutions and showed that the terms of second-order in viscosity can be neglected in comparison to the other terms. With this simplification, the real part of the growth rate can be written as

$$\omega_r = -\frac{2\nu_l k^2 \tanh(kh)}{\tanh(kh) + Q} + \frac{\sqrt{4\nu_l^2 k^4 \tanh^2(kh) - Q^2 U^2 k^2 - [\tanh(kh) + Q](-QU^2 k^2 + \sigma k^3 / \rho_l)}}{\tanh(kh) + Q}. \quad (14.103)$$

Furthermore, if short waves are assumed (*i.e.*, $\tanh(kh) = 1$) and $Q \ll 1$, Equation 14.103 reduces to

$$\omega_r = -2\nu_l k^2 + \sqrt{4\nu_l^2 k^4 + QU^2 k^2 - \sigma k^3 / \rho_l}. \quad (14.104)$$

Senecal et al. analytically derived a critical Weber number of $We_g = \rho_g U^2 h / \sigma = 27/16$, below which long waves dominate the breakup process, and above which short waves dominate breakup. As the Weber number is typically well above 27/16 for sheet breakup applications of interest here (*e.g.*, pressure-swirl atomizers), it is reasonable to assume that short waves are responsible for breakup. As a result, Equation 14.104 is used in CONVERGE to predict the wave growth rate.

In CONVERGE, the physical mechanism of sheet disintegration proposed by [Dombrowski and Johns \(1963\)](#) is adopted in order to predict the drop sizes produced by the primary breakup process. In this process, disintegration occurs due to the growth of waves on the surfaces caused by the aerodynamic forces acting on the sheet. Once the waves reach a critical amplitude, fragments of the liquid are broken off which contract to form cylindrical ligaments that are believed to move normal to the ligament axis. As a result, capillary forces cause the unstable ligaments to break into drops.

In order to determine the onset of ligament formation, Senecal et al. [\(1999\)](#) made an analogy with the prediction of the breakup length of cylindrical liquid jets (*e.g.*, [Reitz and Bracco, 1986](#)). If the surface disturbance has reached a value of η_b at breakup, a breakup time τ can be evaluated via

$$\eta_b = \eta_0 \exp(\Omega_s \tau) \Rightarrow \tau = \frac{1}{\Omega_s} \ln\left(\frac{\eta_b}{\eta_0}\right), \quad (14.105)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

where Ω_s is the maximum growth rate obtained from Equation 14.104. Thus, the sheet will break up at a length given by

$$L = V\tau = \frac{V}{\Omega_s} \ln\left(\frac{\eta_b}{\eta_0}\right), \quad (14.106)$$

where the quantity $\ln(\eta_b / \eta_0)$ is typically set to 12 based on the work of [Dombrowski and Hooper \(1962\)](#). Here, V is the absolute velocity of the liquid sheet, while U in Equation 14.104 is the relative velocity between the liquid and the gas. Note that you can alter the set value for $\ln(\eta_b / \eta_0)$ by changing `parcels.in > liquid parcels > parcel_list > parcel > breakup_control > lisa > length_const`.

Once the sheet breakup length predicted by Equation 14.106 is reached, ligaments with a diameter given by

$$d_L = C_{lisa} \frac{2\pi}{K_s} \quad (14.107)$$

are formed, where C_{lisa} (`parcels.in > liquid parcels > parcel_list > parcel > breakup_control > lisa > size_const`) is a model constant and K_s is the wavenumber corresponding to the maximum growth rate Ω_s .

As in the work of Senecal et al. [\(1999\)](#), the resulting drop diameter is found from the expression

$$d_D = 1.88d_L (1 + 3Oh)^{1/6} \quad (14.108)$$

where $Oh = \mu_l / (\rho_l \sigma d_L)^{1/2}$ is the Ohnesorge number. Equation 14.108 is based on the analysis of [Weber \(1931\)](#).

In CONVERGE, the parcels representing the liquid sheet do not directly interact with the gas phase and do not undergo collision, drag, evaporation, or turbulent dispersion. Once the sheet parcels travel a distance from the injector given by Equation 14.106, the parcels are given a size predicted by Equation 14.108 and are treated as normal parcels that undergo collision, drag, evaporation, turbulent dispersion, and are coupled to the gas.

If a drop size distribution is used with the LISA model, Equation 14.108 represents the Sauter mean diameter of the distribution.

If a drop size distribution is not used, Equation 14.108 is the actual diameter of the drop. Finally, the TAB model is used to predict secondary drop breakup in the LISA model.

A methodology following the work of [Schmidt et al. \(1999\)](#) is used to initialize the size and velocity of injected sheet parcels when the LISA model is activated. This model assumes that the injector exit velocity profile is uniform and that the total velocity is related to the injection pressure Δp ([parcel_introduction.in > injections > - injection > injection_control > lisa_model > injection_pres](#)) by

$$V = k_v \sqrt{\frac{2\Delta p}{\rho_l}}, \quad (14.109)$$

where the velocity coefficient k_v is given by

$$k_v = \max \left[0.7, \frac{4\dot{m}}{\pi d_0^2 \rho_l \cos \theta} \sqrt{\frac{\rho_l}{2\Delta p}} \right]. \quad (14.110)$$

In Equation 14.110, \dot{m} and θ are the measured mass flow rate and spray angle (*i.e.*, half of [parcel_introduction.in > injections > injection > injection_control > nozzles > nozzle > spray_cone > angle](#)), respectively, and d_0 is the injector exit parameter given by [parcel_introduction.in > injections > injection > injection_control > nozzles > nozzle > geometry > diameter](#). Equation 14.109 is used to initialize the velocity of the sheet parcels.

It can be shown ([Schmidt et al., 1999](#)) that the sheet thickness at the nozzle exit h_0 is related to \dot{m} and V by conservation of mass as

$$\dot{m} = \pi \rho_l V \cos \theta h_0 (d_0 - h_0). \quad (14.111)$$

Equation 14.111 is used to calculate the initial sheet thickness (note that h_0 is the full nozzle exit thickness while h is the sheet half-thickness). The sheet parcel's radius is assumed to be equal to one-half of the thickness:

$$r_0 = 0.5h_0. \quad (14.112)$$

Vaporization

The following subsections describe how CONVERGE determines liquid [drop](#) or [film](#) vaporization.

Drop Vaporization Models

CONVERGE contains vaporization models to determine how the radius of a drop changes over time. The [Frossling correlation](#) and the [Chiang correlation](#) are described below.

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

CONVERGE also contains different methods for computing thermal transfer to a drop. The [Uniform Temperature Model](#) and the [Discretized Temperature Model](#) are described below.

1. The droplet temperature is assumed to be [uniform](#) and temperature is solved using two ordinary differential equations.
2. The droplet temperature is assumed to be [spherically symmetric](#) and temperature is solved using a partial differential equation (the 1D spherical heat equation) for the droplet temperature. This temperature solution is coupled with either the [Frossling](#) or [Chiang](#) ODEs for the boundary condition.

For all droplets whose radii are larger than the user-specified `parcels.in > liquid_parcels > parcel_list > parcel > evap_control > temp_discretization > radius`, CONVERGE will use the droplet thermal transfer using a PDE. For all droplets whose radii are smaller than or equal to the `drop_radius`, CONVERGE will use an ODE.

If you specify a value larger than any possible droplet for `parcels.in > liquid_parcels > parcel_list > parcel > evap_control > temp_discretization > radius`, CONVERGE will use only ODEs for determining droplet temperature. Using the heat equation can be computationally expensive, but is useful for generating accurate spray temperature results, as the ODE tends to under-predict the temperature and evaporation rate for large drops.

Values for some of the vaporization model input parameters are functions of the evaporating species (fuel) used in the simulation. Table 14.2 provides recommended values of `parcels.in > liquid_parcels > parcel_list > parcel > evap_control > d0_diffuse` and `n_diffuse` for different species, since the corresponding model parameters D_0 and n_0 are experimentally determined.

Table 14.2: Recommended values for `evap_control` parameters in [parcels.in](#).

Fuel name	<code>d0_diffuse</code>	<code>n_diffuse</code>
C7H16	5.94e-6	1.60
IC8H18	5.44e-6	1.60
C10H22	4.61e-6	1.60
C14H30	3.74e-6	1.60
Gasoline	7.90e-6	1.87
Jet-A	4.16e-6	1.60

Droplet Radius Change: Frossling Correlation

Once the liquid spray is injected into the computational domain, a model is needed to convert the liquid into gaseous vapor. To use the Frossling correlation to determine the time rate of change of droplet size, set `parcels.in > liquid_parcels > parcel_list > parcel > evap_control > evap_model = FROSSLING`. The Frossling correlation (Amsden *et al.*, 1989) is

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$\frac{dr_0}{dt} = -\frac{\alpha_{spray}\rho_g D}{2\rho_l r_0} B_d Sh_d, \quad (14.113)$$

where α_{spray} is the user-specified scaling factor for the mass transfer coefficient ([parcels.in > liquid_parcel > parcel_list > parcel > evap_control > scaling > mass_trans_coeff](#)), D is the mass diffusivity of liquid vapor in air. We define B_d as

$$B_d = \frac{Y_1^* - Y_1}{1 - Y_1^*}, \quad (14.114)$$

where Y_1^* is the vapor mass fraction at the drop's surface, Y_1 is the vapor mass fraction, and Sh_D is the Sherwood number given by

$$Sh_d = (2.0 + 0.6 Re_d^{1/2} Sc^{1/3}) \frac{\ln(1 + B_d)}{B_d}, \quad (14.115)$$

where

$$Re_d = \frac{\rho_{gas} |u_i + u'_i - v'_i| d}{\mu_{air}}. \quad (14.116)$$

In Equation 14.116, d is the drop diameter and μ_{air} is the air viscosity which is evaluated at the temperature \hat{T} given by ([Amsden et al., 1989](#))

$$\hat{T} = \frac{T_{gas} + 2T_d}{3}, \quad (14.117)$$

where T_{gas} is the gas temperature and T_d is the drop temperature. Furthermore, $Sc = \mu_{air}/\rho_{gas}D$ is the Schmidt number of air and D is determined from the correlation:

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$\rho_{\text{gas}} D = 1.293 D_0 \left(\bar{T} / 273 \right)^{n_0 - 1}, \quad (14.118)$$

where D_0 and n_0 are experimentally determined model constants. In addition, Y_1^* is determined from

$$Y_1^* = \frac{MW_{C_nH_{2m}}}{MW_{C_nH_{2m}} + MW_{\text{mix}} \left(\frac{p_{\text{gas}}}{p_v} - 1 \right)}, \quad (14.119)$$

where MW_{mix} is the molecular weight of the mixture (not including vapor from the liquid species), p_{gas} is the gas pressure, and p_v is the vapor pressure at the current droplet temperature.

Droplet Radius Change: Chiang Correlation

Once the liquid spray is injected into the computational domain, a model is needed to convert the liquid into gaseous vapor. To use the Chiang correlation ([Chiang et al. \(1992\)](#)) to determine the time rate of change of droplet size, set `parcels.in > liquid parcels > parcel_list > parcel > evap_control > evap_model = CHIANG`.

This option will use alternative Nu_d and Sh_d correlations of

$$Nu_d = 1.275 (1 + B_d)^{-0.678} Re_d^{0.438} Pr_d^{0.619} \quad (14.120)$$

and

$$Sh_d = 1.224 (1 + B_d)^{-0.568} Re_d^{0.385} Sc_d^{0.492}. \quad (14.121)$$

With this model, the rate of heat conduction is still given by the [Ranz-Marshall correlation](#), but the rate of change of droplet radius is given by

$$\frac{dr_0}{dt} = - \frac{\alpha_{\text{spray}} \rho_g D}{2 \rho_l r_0} B_d Sh_d (Y_1^* - Y_1), \quad (14.122)$$

where α_{spray} is the user-specified scaling factor for the mass transfer coefficient ([parcels.in > liquid_parcel > parcel_list > parcel > evap_control > scaling > mass_trans_coeff](#)).

Droplet Boiling Model

To model droplet boiling, CONVERGE includes a model for droplet radius change in the boiling regime. Equation 14.123 below gives the formula for calculating the time rate of change of the droplet radius when the droplet temperature exceeds the boiling point:

$$\frac{d(r_0)}{dt} = \frac{k_{\text{air}}}{\rho_d c_{p,\infty} r_0} \left(1 + 0.23\sqrt{Re_d}\right) \ln \left[1 + \frac{c_{p,\infty}(T_{\text{air}} - T_d)}{h_{fg}} \right], \quad (14.123)$$

where k_{air} is the thermal conductivity and $c_{p,\text{air}}$ is the specific heat capacity. Once the droplet temperature reaches the boiling temperature, this model considers the droplet temperature fixed at the boiling temperature.

The boiling temperature for each component of the liquid parcel is determined individually from [liquid.dat](#). CONVERGE uses pressure of the cell containing the parcel as the vapor pressure for the purposes of this lookup.

You can use this droplet boiling model in conjunction with either the Frossling or the Chiang correlation for droplet radius change. CONVERGE models droplet radius change with the correlation you select until the droplet temperature reaches the boiling point. At the boiling point, CONVERGE uses Equation 14.123 to model droplet radius change. To model droplet boiling, set [parcels.in > liquid_parcel > parcel_list > parcel > evap_control > drop > boiling = 1](#). Note that [parcels.in > liquid_parcel > parcel_list > parcel > evap_control > evap_model](#) must be set to *FROSSLING* or *CHIANG*.

Uniform Temperature Model

The Uniform Temperature Model, which CONVERGE uses to calculate the droplet temperature for all droplets whose radii are smaller than or equal to [parcels.in > liquid_parcel > parcel_list > parcel > evap_control > temp_discretization > radius](#), employs the following energy balance:

$$\bar{A}_d Q_d = c_l m_d^* \frac{dT_d}{dt} - \frac{dm_d}{dt} H_{\text{vap}}, \quad (14.124)$$

which states that energy conducted to a drop will either heat up the drop or supply heat for vaporization ([Amsden et al., 1989](#)). In Equation 14.124, c_l is the liquid specific heat, T_d is the drop temperature, m_d is the drop mass, and H_{vap} is the latent heat of vaporization evaluated

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

at the drop temperature. In addition, the rate of heat conduction to the drop surface per unit area is given by the Ranz-Marshall correlation ([Faeth, 1977](#)) of

$$Q_d = \frac{\beta_{spray} N u_d k_{air} (T_{gas} - T_d)}{\bar{d}_0}, \quad (14.125)$$

where β_{spray} is the user-specified scaling factor for the heat transfer coefficient ([parcels.in > liquid_parcel > parcel_list > parcel > evap_control > scaling > heat_trans_coeff](#)), k_{air} is the conductivity evaluated at \hat{T} , T_{gas} is the gas temperature, and $N u_d$ is given by

$$N u_d = (2.0 + 0.6 Re_d^{1/2} Pr_d^{1/3}) \frac{\ln(1 + B_d)}{B_d}, \quad (14.126)$$

where Pr_d is the Prandtl number. Furthermore, in the above equations \bar{A}_d and \bar{d}_0 are the average drop area and diameter, respectively, and m_d^* is an intermediate value of the drop mass. These quantities are given by

$$\begin{aligned} \bar{A}_d &= \frac{\pi(r_0^{*2} + r_0^2)}{2} \\ \bar{d}_0 &= \frac{2r_0^* + 2r_0}{2} \\ m_d^* &= \rho_l \frac{4}{3} \pi r_0^3, \end{aligned} \quad (14.127)$$

where r_0^* is an intermediate value of the drop radius, calculated from [a previous equation](#) (when [parcels.in > liquid_parcel > parcel_list > parcel > evap_control > evap_model = FROSSLING](#)) or from [another](#) (when $evap_model = CHIANG$). Once the updated value of T_d is obtained, [Equation 14.113](#) or [Equation 14.122](#) is used to solve for the updated drop radius. This change in radius is used to determine the amount of vapor to add to the computational cell.

Discretized Temperature Model

CONVERGE uses the Discretized Temperature Model to calculate the droplet temperature for all droplets whose radii are larger than [parcels.in > liquid_parcel > parcel_list > parcel > evap_control > temp_discretization > radius](#). This method, which uses partial differential equations, is summarized below.

Governing Equations

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

The heat conduction model ([Sazhin et al., 2011](#)) for a spherically symmetric, evaporating droplet with no recirculation is given by Equations 14.128 and 14.129 as

$$\rho c_p \frac{\partial T}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} \left(k r^2 \frac{\partial T}{\partial r} \right) \quad (14.128)$$

and

$$k \frac{\partial T}{\partial r} \Big|_{r=R_d} = h \left[T_g - T(R_d, t) \right] + \rho L \frac{dR_d}{dt} + q_{rad}, \quad (14.129)$$

where r is the distance from the center, h is the convection coefficient between the droplet and the surrounding gas, T_g is the temperature of the gas, $T(R_d, t)$ is the temperature at the droplet surface, $R_d(t)$ is the radius of the droplet, c_p is the droplet specific heat capacity, L is the specific heat of evaporation, ρ is the droplet density, and k is the droplet thermal conductivity. The final term q_{rad} is present only when radiation/spray coupling is active (*i.e.*, when [radiation.in > radiation_spray_coupling > rad_spray_coupling_flag = 1](#)). This term accounts for the heat flux at the droplet surface due to radiation, given by

$$q_{rad} = \varepsilon_p \left(G - 4\sigma [T(R_d, t)]^4 \right) \quad (14.130)$$

where G is the total incident radiation, σ is the Stefan-Boltzmann constant, and ε_p is the parcel emissivity (specified in [radiation.in > radiation_spray_coupling > parcel_emissivity](#)).

Because the pure conduction assumption is generally not accurate, the effects of recirculation can be simulated by replacing k with k_{eff} ([Sazhin, 2006](#)). We define k_{eff} as

$$k_{eff} = \chi k, \quad (14.131)$$

where

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

$$\chi = 1.86 = 0.86 \tanh(2.225 \log_{10}(0.03333 Pe_d)), \quad (14.132)$$

in which Pe_d is the droplet Peclet number. These equations are known as the effective thermal conductivity model ([Abramzon and Sirignano, 1989](#)).

Numerical Method: Finite Volume Derivation

Using the divergence theorem, the finite volume formulation for non-constant volumes is derived by integrating the general heat equation over a spherical shell $V = \{r \mid r \in [r_1, r_2]\}$ and then applying spherical symmetry as

$$\begin{aligned} \int_V \rho c_p \partial_t T &= \int_V \nabla \cdot (k_{\text{eff}} \nabla T) \\ &= \oint_S k_{\text{eff}} \nabla T \cdot n \\ &= 4\pi (r^2 k \partial_r T) \Big|_{r=r_1}^{r=r_2}. \end{aligned} \quad (14.133)$$

Because the volume is changing, the time derivative cannot be passed out of the integral. We assume that only the outermost boundary can change and apply both the fundamental theorem of calculus and the chain rule as follows:

$$\begin{aligned} \int_V \rho c_p \partial_t T &\approx \overline{\rho c_p} \int_V \partial_t T \\ &= 4\pi \overline{\rho c_p} \int_{r_1}^{r_2(t)} \partial_t T r^2 dr \\ &= 4\pi \overline{\rho c_p} \left(\partial_t \left(\int_{r_1}^{r_2(t)} \partial_t T r^2 dr \right) - r_2^2 T(r_2, t) \frac{dr_2}{dt} \right) \\ &= 4\pi \overline{\rho c_p} \left(\frac{1}{3} \partial_t \left((r_2^3 - r_1^3) \bar{T} \right) - r_2^2 T(r_2, t) \frac{dr_2}{dt} \right). \end{aligned} \quad (14.134)$$

Averaging, we obtain the full finite volume formulation of

$$\partial_t \left(\frac{(r_2^3 - r_1^3)}{3} \bar{T} \right) = \frac{k_{\text{eff}}}{\rho c_p} \left(r_2^2 \frac{\partial T}{\partial r} \Big|_{r=r_2} - r_1^2 \frac{\partial T}{\partial r} \Big|_{r=r_1} \right) + r_2^2 T(r_2, t) \frac{dr_2}{dt}. \quad (14.135)$$

In the present implementation, only the outermost shell, $r_2 = R_d$, is allowed to vary, meaning that $dr_2/dt = 0$ for the interior, recovering the more familiar formula.

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

Numerical Method: Linearization of Radiation Term

The radiation term q_{rad} in Equation 14.129 is a nonlinear function of temperature. When solving for the temperature at iteration $m+1$, we linearize this term about the solution T^m from the previous iteration, which yields

$$q_{rad} = \varepsilon_p \left[G + 12\sigma(T^m)^4 \right] - 16\varepsilon_p \sigma(T^m)^3 T^{m+1}. \quad (14.136)$$

The first term in Equation 14.136 is added to the right side of the system of equations, while the second term is added to the coefficient matrix.

Numerical Method: Numerical Details

- Currently, density is a constant and \bar{c}_p is computed using $c_p(\bar{T})$.
- Time discretization is fully implicit and uses the same Δt as the flow solver.
- The droplet domain is discretized into a number of shells of equal volume specified by [*parcels.in > liquid parcels > parcel_list > parcel > evap_control > temp_discretization > layers_per_drop*](#). Numerical instabilities occur when $\Delta t(dr/dt) > \delta$, where δ is the width of the outermost shell. These instabilities are resolved by merging all shells such that $r_2 < R_d(t_2)$.
- When a parcel breaks up into two or more parcels, both parcels' temperatures are uniformly set to those corresponding to the average internal energy of the original parcel. Similarly, a drop formed from two coagulating parcels is assigned an energy-averaged temperature.
- You can specify the number of FV cells in the droplet, whether to use k_{eff} (set *use_k_eff* = 0 or 1), and a threshold radius ([*parcels.in > liquid parcels > parcel_list > parcel > evap_control > temp_discretization > radius*](#)) below which the evaporation model will switch to the [Uniform Temperature Model](#).

Flash Boiling

CONVERGE applies a flash boiling model ([Price et al., 2016](#)) if you set [*parcels.in > liquid parcels > parcel_list > parcel > evap_control > evap_flag_flash_boiling*](#) = 1. The rate of change of the drop mass is the sum of the subcooled term M_{sc} and the superheat term M_{sh} . The subcooled term is calculated based on the drop density and the radius rate of change from specified correlation ([Frossling](#) or [Chiang](#)). The superheat term is calculated as

$$\frac{dM_{sh}}{dt} = \frac{4\pi r_d^2 \alpha (T - T_b)}{H_L}, \quad (14.137)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

where H_L is the latent heat of the liquid and T_b is the boiling temperature. The heat transfer coefficient α is a scaled form of the empirical relation ([Adachi et al., 1997](#)):

$$\begin{aligned}\alpha &= \beta 760(T - T_b)^{0.26} && \text{when } 0 \leq (T - T_b) \leq 5 \\ \alpha &= \beta 27(T - T_b)^{2.33} && \text{when } 5 \leq (T - T_b) \leq 25 \\ \alpha &= \beta 13800(T - T_b)^{0.39} && \text{when } (T - T_b) \geq 25.\end{aligned}\quad (14.138)$$

The scaling factor β is equal to 1 in the original formulation, and is set by adjusting [*parcels.in*](#) > *liquid_parcels* > *parcel_list* > *parcel* > *evap_control* > *evap_scale_factor_flash_boiling*.

Optionally, you can direct CONVERGE to calculate a reduced drop size based on flash boiling by setting [*parcels.in*](#) > *liquid_parcels* > *parcel_list* > *parcel* > *evap_control* > *size_flag_flash_boiling* = 1. This size reduction is calculated from the mass rate of change, assuming a constant liquid density, of

$$\frac{dr_d}{dt} = \frac{\gamma}{4\pi\rho r_d^2} \frac{dM}{dt}, \quad (14.139)$$

where γ is a scaling factor set by *size_scale_factor_flash_boiling*.

Multi-Component and Composite Species Vaporization

CONVERGE can simulate multi-component vaporization. You can have multiple parcel species evaporate into their corresponding gaseous state, or you can have multiple parcel species evaporate into a different number of gas species. Alternately, you can have multiple parcel species evaporate into a single gas species. In the latter two cases, you may have a simulation that involves liquid species for which you do not have a combustion mechanism, or you may want to simplify an existing combustion mechanism.

CONVERGE can simulate multi-component vaporization in three different ways:

1. If [*parcels.in*](#) > *liquid_parcels* > *parcel_list* > *parcel* > *evap_control* > *evap_source_flag* = 0, all of the parcels will evaporate to the species listed as [*parcels.in*](#) > *liquid_parcels* > *parcel_list* > *parcel* > *evap_control* > *evap_species*. You can list only one species as the *evap_species*. For example, a parcel consists of two liquid components, C4H8 and C14H30, and C14H30 is listed as *evap_species*. Here both C4H8 and C14H30 will evaporate to C14H30.
2. If *evap_control* > *evap_source_flag* = 1, all of the multi-component liquid species will evaporate into composite species if composite species are used. If composite species are not used, the liquid species will evaporate into the corresponding base gas species.

For example, a parcel consists of two liquid components, C4H8 and C14H30, and these two components are defined in [*composite.in*](#) as the parcel-phase composite species named

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

mixture. Another parcel species, C7H16, is defined in [*species.in*](#). In this example, the composite named *mixture* will evaporate into a gas phase species named *mixture* and C7H16 will evaporate to C7H16. All three of these parcel species must be present in [*liquid.dat*](#), and the two gas species must be present in the [thermodynamic data file](#).

The figures below show example [*composite.in*](#) and [*species.in*](#) files, and an excerpt from [*parcels.in*](#), for this option.

```
version: 3.1
---
composites:
  - composite:
    composite_name: mixture
    phase: PARCEL_PHASE
    base_species:
      C4H8: 0.50000
      C14H30: 0.50000
```

Figure 14.3: Sample [*composite.in*](#) file.

```
version: 3.1
---
parcel:
  - C14H30
  - C4H8
  - C7H16
gas:
  - C7H16
  - mixture
```

Figure 14.4: Sample [*species.in*](#) file.

```
liquid_parcels:
  combine_parcels:
    active: 0
    start_time: -9999999
    end_time: -9999999
    max_parcels_per_node: 50
  sub_cycle:
    active: 0
    dt: 1e+32
  collision_control:
    collision_mesh_flag: 0
    collision_model: NTC
    outcome_model: POST
  wall_interaction_model: REBOUND_SLIDE
  parcel_list:
    - parcel:
      parcel_name: FuelBlend_1
      parcel_species:
        C7H16: 0.7
        mixture: 0.3
```

Figure 14.5: Sample excerpt of [*parcels.in*](#).

3. If *evap_control > evap_source_flag = 2*, all of the multi-component liquid species and composite species (if present) will evaporate into the base species.

For example, a parcel consists of two liquid components, C4H8 and C14H30. These two components are defined in [*composite.in*](#) as a parcel-phase composite named *mixture*.

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

Another parcel species, C7H16, is defined in [species.in](#). In this example, the C4H8 portion of *mixture* will evaporate into C4H8, the C14H30 portion of *mixture* will evaporate into C14H30, and C7H16 will evaporate into C7H16. All three of these parcel species must be present in [liquid.dat](#), and the three gas species must be present in the [thermodynamic data file](#).

The figures below show example [composite.in](#) and [species.in](#) files, and an excerpt from [parcels.in](#), for this option.

```
version: 3.1
---

composites:
  - composite:
    composite_name: mixture
    phase: PARCEL_PHASE
    base_species:
      C4H8: 0.50000
      C14H30: 0.50000
```

Figure 14.6: Sample [composite.in](#) file.

```
version: 3.1
---

parcel:
  - C4H8
  - C14H30
  - C7H16
gas:
  - C4H8
  - C14H30
  - C7H16
```

Figure 14.7: Sample [species.in](#) file.

```
liquid_parcels:
  combine_parcels:
    active: 0
    start_time: -999999
    end_time: -999999
    max_parcels_per_node: 50
    sub_cycle:
      active: 0
      dt: 1e+32
    collision_control:
      collision_mesh_flag: 0
      collision_model: NTC
      outcome_model: POST
    wall_interaction_model: REBOUND_SLIDE
    parcel_list:
      - parcel:
        parcel_name: FuelBlend_1
        parcel_species:
          C7H16: 0.7
          mixture: 0.3
```

Figure 14.8: Sample [parcels.in](#) file excerpt.

If you have not defined any composite species, *evap_control > evap_source_flag = 1* and *evap_source_flag = 2* behave identically.

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

CONVERGE monitors each component of a multi-component fuel using the discrete multi-component model and does not operate with the assumption of a continuous blend of components. The overall composition is represented by the mole fractions of each component. CONVERGE must assume, however, that all components are well mixed. CONVERGE uses Raoult's law to correlate between the vapor mass fraction of the component over the surface and its mole fraction in the condensed phase. Properties (*e.g.*, viscosity) of the multi-component fuel are mass-weighted average values. The only exception is density, which is given as a volume-weighted average value.

For cases with the Flamelet Generated Manifold combustion model, *evap_control > evap_source_flag* must be 0. Species will evaporate to the mixture fraction and thus CONVERGE ignores the *evap_species*, but the species names in *evap_species* must match names given in the [reaction mechanism file](#).

Urea Injection

Urea injection with selective catalytic reduction (Urea/SCR) is a technique for reducing NOx by mixing exhaust gasses with urea and its products. To activate urea modeling, set [*inputs.in > feature_control > urea_model = 1*](#) and include a [*urea.in*](#) file in your case setup.

CONVERGE includes several options for urea modeling. Set [*urea.in > urea_model = 1*](#) to activate [urea/water injection](#). Set *urea_model = 2* to activate [molten solid urea decomposition](#). Set *urea_model = 3* to activate the [detailed deposition of urea model](#). In each case, the value of [*parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > urea_model*](#) should match the value of *urea.in > urea_model*.

Urea/Water Injection

If [*urea.in > urea_model = 1*](#), CONVERGE injects a urea-water solution. If you choose this option, H2O must be used as parcel in [*species.in*](#) and NH₂-CO must be included as a species in the [reaction mechanism file](#). In addition, the properties for NH₂-CO and H2O must be included in the [thermodynamic data file](#) and [*liquid.dat*](#), respectively. (Urea has the species name NH₂-CO, representing the chemical symbol (NH₂)₂CO.)

A chemical mechanism can be used with the [SAGE detailed chemical kinetics solver](#) to model the mechanism of NOx reduction by urea. For example, the sub-mechanism presented in [Golovitchev et al. \(2007\)](#) includes 17 reactions representing this process.

CONVERGE applies this model to all injectors and nozzles in the simulation.

Molten Solid Urea Decomposition

If [*urea.in > urea_model = 2*](#), CONVERGE uses the molten solid approach to model the decomposition of a urea-water solution (UWS). The [Frossling correlation](#) models the evaporation of the water in the UWS, while an Arrhenius correlation models the decomposition of the urea in the UWS. After activating the model, supply a *urea.in* file containing the prefactor *A* and the activation energy *E_a* from the Arrhenius correlation

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels

(Equation 14.141) and the change in enthalpy due to urea decomposition H_{dcmp} from the droplet heat transfer equation (Equation 14.142).

In the molten solid approach, urea decomposes to gaseous ammonia and isocyanic acid. The formula for the decomposition is



The Arrhenius correlation (given in Equation 14.141) models this decomposition by computing the time rate of change of droplet radius as a function of a prefactor, the activation energy, the droplet temperature, and the density of urea. The correlation is

$$\frac{dm_d}{dt} = 2\pi R_d A \exp\left(\frac{-E_a}{RT_d}\right), \quad (14.141)$$

where m_d is the droplet mass, R_d is the droplet radius, A is the prefactor, E_a is the activation energy, and T_d is the droplet temperature. For this model, the droplet density is equivalent to the urea density. Because the urea decomposition is significant when the droplet temperature exceeds 425 K, most of the water has evaporated and the urea remains.

The molten solid approach uses two methods to calculate droplet temperature change. For large droplets (droplet diameter greater than several hundred micrometers), CONVERGE uses the spherically symmetric heat equation with the following boundary condition for heat transfer at the surface:

$$k_d \frac{\partial T}{\partial r} \Big|_{r=R_d} = h \left[T_g - T(R_d, t) \right] + \rho_d H_{vap} \frac{dR_d}{dt} \Big|_{vap} + \rho_d H_{dcmp} \frac{dR_d}{dt} \Big|_{dcmp}, \quad (14.142)$$

where k_d is the thermal conductivity of the droplet, r is the distance from the droplet centroid, R_d is the radius of the droplet, h is the convection coefficient between the droplet and the surrounding gas, T_g is the temperature of the gas, $T(R_d, t)$ is the temperature at the droplet surface, ρ_d is the droplet density, H_{vap} is the latent heat of vaporization evaluated at the drop temperature, and H_{dcmp} is the change in enthalpy due to urea decomposition. Refer to the [Discretized Temperature Model](#) section for more details.

For small droplets (droplet diameter less than several hundred micrometers), the droplet temperature change due to heat transfer is

$$C_d m_d \frac{dT_d}{dt} = 2\pi R_d k_g N u_d (T_g - T_d) + \left. \frac{dm_d}{dt} \right|_{vap} H_{vap} + \left. \frac{dm_d}{dt} \right|_{dcmp} H_{dcmp}, \quad (14.143)$$

where C_d is specific heat capacity of the droplet and k_g is the thermal conductivity of the gas. In this case, the model assumes that the droplet temperature instantaneously increases to a uniform temperature. Note that CONVERGE uses the mass change due to urea decomposition to determine the heat loss from thermal decomposition and mass change from water evaporation to determine heat exchange from water evaporation.

Detailed Decomposition of Urea for Deposition

If `urea.in > urea_model = 3`, CONVERGE uses a detailed decomposition of urea for deposition model, which was implemented jointly by Convergent Science and IFP Energies nouvelles. This model directly predicts the deposition of solid urea decomposition byproducts based on chemical reactions in urea-water solution (UWS) droplets and films.

The detailed decomposition model consists of a multi-component evaporation model for UWS droplets and a thermal decomposition model of urea valid for automotive exhausts using SCR systems ([Ebrahimian et al., 2012](#)).

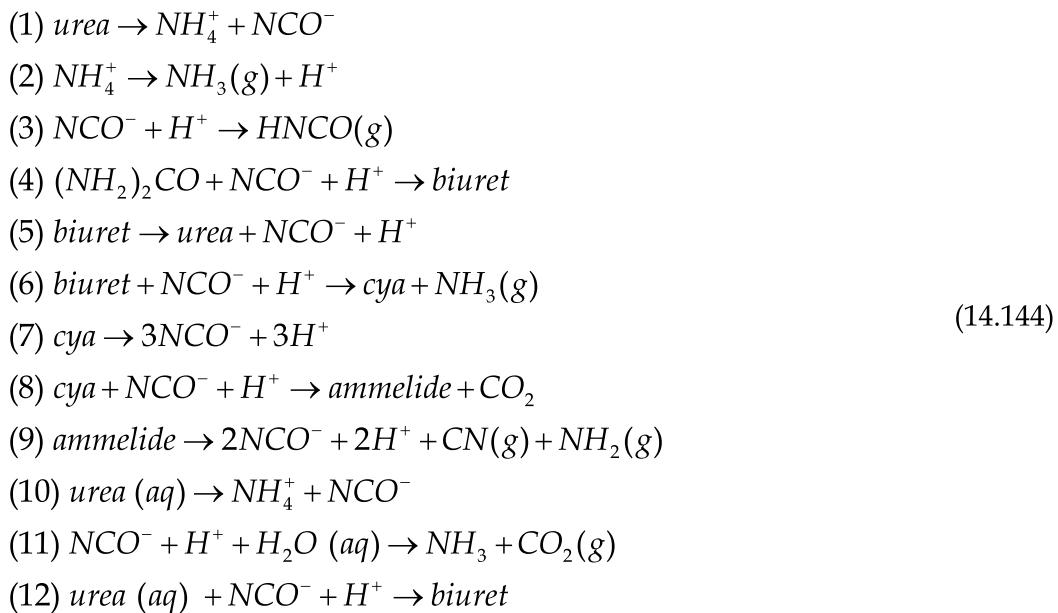
Because the vapor pressure of urea is low at atmospheric pressures, the UWS evaporation model is based on the assumption that only water evaporates from the solution. To account for the interactions between urea and water in solution, activity coefficients are calculated via the non-random two-liquid (NRTL) equation. At high urea concentrations, the effects of these interactions may be significant.

The thermal decomposition model is based on semi-detailed kinetics. It accounts for the decomposition of urea in both the solid and aqueous phases, and it can predict the polymerization of urea decomposition byproducts to form solid deposits of biuret, ammelide, and cyanuric acid ([Habchi et al., 2015](#)).

The detailed decomposition model solves the following set of equations:

Chapter 14: Discrete Phase Modeling

Parcel Settings Liquid Parcels



where *cya* is cyanuric acid. Reactions (1) through (9) describe the thermal decomposition of urea in dry media, while reactions (10) through (12) describe the thermal hydrolysis in the aqueous phase. CONVERGE solves three clipping equations on (1) through (3), with altered reaction coefficients. These clipped equations ensure realizability and boundedness of the detailed decomposition mechanism. We do not recommend altering the clipped equation reaction parameters.

This model also requires `parcels.in` > `liquid_parcels` > `parcel_list` > `parcel` > `evap_control` > `evap_source_flag = 1` and `inputs.in` > `feature_control` > `source_flag = 1`. We recommend activating either the [Kuhnke](#) or [Bai-Gosman](#) splash model (`parcel_wall_interaction.in` > `liquid_wall_interaction_control` > `default_wall_control` > `film_control` > `splash` > `model = KUHNKE` or `BAI-GOSMAN`) in conjunction with this model.

The liquid reactions for the detailed decomposition model are activated when `urea.in` > `urea_model = 3` and do not appear in the reaction mechanism file. The hydrolysis of HNCO is the only gas phase mechanism that is in the reaction mechanism file. All of the species listed in the detailed decomposition reactions must be included in the [reaction mechanism file](#) or in `species.in`. You must include CN in `species.in`. Some of the urea properties in `liquid.dat` are not used in a simulation with the detailed decomposition model. All of the water properties in `liquid.dat` are used in a simulation with this model.

Output

CONVERGE will write spray information to `spray_urea_file.out`. If `parcel_wall_interaction.in` > `liquid_wall_interaction_control` > `default_wall_control` > `wall_model = FILM`, CONVERGE will write film composition data to `film_urea_file.out`.

For 3D urea byproduct data (for post-simulation visualization), include *urea_deposit* in the *parcels* section of [*post.in*](#). CONVERGE will write the mass fractions of solid urea (*urea_solid*), urea in solution (*urea_aqueous*), biuret, cyanuric acid (*cya*), ammelide, water (*h2o*), cyanate (*nco*), ammonium (*nh4*), and hydrogen radicals (*hplus*) to the *post*.h5* files.

Parcel Consolidation for Spray

CONVERGE can combine liquid parcels within fluid cells to reduce computational time. To use this feature, set [*parcels.in*](#) > *liquid_parcels* > *combine_parcels* > *active* = 1. Note that this setting applies only to liquid spray parcels. A different setting is required to [consolidate liquid film parcels](#).

You must specify the time at which CONVERGE will begin consolidating spray parcels ([*parcels.in*](#) > *liquid_parcels* > *combine_parcels* > *start_time*), the time at which CONVERGE will stop consolidating parcels ([*parcels.in*](#) > *liquid_parcels* > *combine_parcels* > *end_time*), and the maximum number of parcels per fluid cell ([*parcels.in*](#) > *liquid_parcels* > *combine_parcels* > *max_parcels_per_node*). If the number of spray parcels in a given cell exceeds the maximum number of parcels, CONVERGE will combine parcels in that cell until the number of parcels is equal to the maximum.

Note that activating parcel consolidation will result in a different calculated spray mass, since parcel momentum, radius, and temperature will change after combining the parcels.

Solid Parcels

The following sub-sections contain information about solid parcels.

Solid Parcel Drag

There are several ways to model the drag force on solid parcels. Specify your preferred method in [*parcels.in*](#) > *solid_parcels* > *parcel_list* > *parcel* > *force_control* > *drag_control* > *drag_model*.

If you set *drag_model* = *TRACER*, CONVERGE treats the parcels as tracer particles that move with the same velocity as the surrounding fluid.

The other options for *drag_model* correspond to different methods for calculating the drag coefficient C_D in the [parcel equation of motion](#), as described below. The Reynolds number used in these correlations is defined as

$$Re_p = \frac{2\rho r_p \sqrt{U_i U_i}}{\mu}, \quad (14.145)$$

where ρ is the fluid density, μ is the fluid viscosity, r_p is the particle radius, and U_i is the fluid velocity relative to the particle, as [defined previously](#).

Chapter 14: Discrete Phase Modeling

Parcel Settings Solid Parcels

The basic correlation for spherical particles (*drag_model* = SPHERE) is given by

$$C_D = \begin{cases} \frac{24}{\text{Re}_p} \left(1 + \frac{1}{6} \text{Re}_p^{2/3} \right) & \text{Re}_p \leq 1000 \\ 0.424 & \text{Re}_p > 1000 \end{cases} \quad (14.146)$$

The correlations from [Clift et al., 1978](#) (*drag_model* = CLIFT_GRACE_WEBER), shown below in Table 14.3, are appropriate for spherical particles over a wide range of Reynolds numbers. In these expressions, $w = \log_{10} \text{Re}_p$.

Table 14.3: Clift-Grace-Weber correlations for the solid parcel drag coefficient.

Range	Correlation
$\text{Re}_p < 0.01$	$C_D = \frac{3}{16} + \frac{24}{\text{Re}_p}$
$0.01 < \text{Re}_p \leq 20$	$C_D = \frac{24}{\text{Re}_p} \left[1 + 0.1315 \text{Re}_p^{(0.82-0.05w)} \right]$
$20 < \text{Re}_p \leq 260$	$C_D = \frac{24}{\text{Re}_p} \left[1 + 0.1935 \text{Re}_p^{0.6305} \right]$
$260 < \text{Re}_p \leq 1500$	$\log_{10} C_D = 1.6435 - 1.1242w + 0.1558w^2$
$1.5 \times 10^3 < \text{Re}_p \leq 1.2 \times 10^4$	$\log_{10} C_D = -2.4571 + 2.5558w - 0.9295w^2 + 0.1049w^3$
$1.2 \times 10^4 < \text{Re}_p \leq 4.4 \times 10^4$	$\log_{10} C_D = -1.9181 + 0.6370w - 0.0636w^2$
$4.4 \times 10^4 < \text{Re}_p \leq 3.38 \times 10^5$	$\log_{10} C_D = -4.3390 + 1.5809w - 0.1546w^2$
$3.38 \times 10^5 < \text{Re}_p \leq 4 \times 10^5$	$C_D = 29.78 - 5.3w$
$4 \times 10^5 < \text{Re}_p \leq 10^6$	$C_D = 0.1w - 0.49$
$10^6 < \text{Re}_p$	$C_D = 0.19 - \frac{8 \times 10^4}{\text{Re}_p}$

The Stokes and Stokes-Cunningham correlations are appropriate for small spherical particles (*i.e.*, particles with radii smaller than 1 *micron*) and low Reynolds numbers. The Stokes correlation (*drag_model* = STOKES) is simply

Chapter 14: Discrete Phase Modeling

Parcel Settings	Solid Parcels
-----------------------	---------------

$$C_D = \frac{24}{\text{Re}_p}. \quad (14.147)$$

The Stokes-Cunningham correlation (*drag_model* = *STOKES_CUNNINGHAM*) includes a correction to Equation 14.147 to account for non-continuum effects. With this correction, the drag coefficient becomes

$$C_D = \frac{24}{\text{Re}_p C_c}, \quad (14.148)$$

where

$$C_c = 1 + \frac{\lambda}{r_p} \left(1.257 + 0.4 e^{-1.1 r_p / \lambda} \right) \quad (14.149)$$

and the mean free path λ is calculated from

$$\lambda = \frac{\mu}{P} \sqrt{\frac{\pi R T}{2 M W}}. \quad (14.150)$$

In Equation 14.150, P , T , and MW are the pressure, temperature, and molecular weight of the fluid, respectively, and R is the ideal gas constant.

The correlation from [Haider and Levenspiel, 1989](#) (*drag_model* = *HAIDER_LEVENSPIEL*) is appropriate for non-spherical particles. This correlation uses the particle sphericity ϕ to describe the particle shape. The sphericity is defined as

$$\phi = \frac{S_{\text{effective}}}{S_{\text{actual}}}, \quad (14.151)$$

where S_{actual} is the actual surface area of the particle and $S_{\text{effective}}$ is the surface area of a sphere with the same volume as the particle. Haider and Levenspiel estimate the drag coefficient as

Chapter 14: Discrete Phase Modeling

Parcel Settings Solid Parcels

$$C_D = \frac{24}{\text{Re}_p} \left(1 + 8.1716 e^{-4.0655\phi} \text{Re}_p^{0.0964+0.5565\phi} \right) + \frac{73.69 \text{Re}_p e^{-5.0748\phi}}{\text{Re}_p + 5.378 e^{6.2122\phi}}. \quad (14.152)$$

You can specify the sphericity in [*parcels.in*](#) > *solid parcels* > *parcel_list* > *parcel* > *force_control* > *drag_control* > *sphericity*.

Solid Parcel Lift

The lift force in the [*parcel equation of motion*](#) is characterized by the lift coefficient C_L . CONVERGE uses the formulation of the Saffman-Mei correlation from [Koohandaz et al., 2020](#) to calculate C_L as a function of Re_p (defined in Equation 14.145) and Re_w , given by

$$\text{Re}_w = \frac{4\rho\Omega r_p^2}{\mu}, \quad (14.153)$$

where

$$\Omega = \left| \epsilon_{ijk} \frac{\partial(u_k + u'_k)}{\partial x_j} \right|. \quad (14.154)$$

The velocity used to compute Ω is the total fluid velocity, including the fluctuating contribution from turbulence.

The Saffman-Mei correlation can be written as

$$C_L = \frac{3}{2\pi\sqrt{\text{Re}_w}} C_{ld}, \quad (14.155)$$

with

$$C_{ld} = \begin{cases} 6.46f & \text{Re}_p < 40 \\ 6.46 \times 0.037\sqrt{\text{Re}_w} & \text{Re}_p \geq 40 \end{cases}, \quad (14.156)$$

Chapter 14: Discrete Phase Modeling

Parcel Settings Solid Parcels

$$f = (1 - \alpha) e^{-0.1 \text{Re}_p} + \alpha, \quad (14.157)$$

$$\alpha = 0.3315\sqrt{\beta}, \quad (14.158)$$

and

$$\beta = 0.5 \left(\frac{\text{Re}_w}{\text{Re}_p} \right). \quad (14.159)$$

Solid Parcel Turbulent Dispersion

Two turbulent dispersion models are available for solid parcels in CONVERGE: the O'Rourke model and the tke-preserving model. These models use source terms to account for the work done by turbulent eddies to disperse the solid parcels. Both are available only for simulations with a [RANS](#) or [hybrid RANS/LES](#) turbulence model.

For implementation details, refer to the [descriptions of the O'Rourke and tke-preserving models for liquid parcels](#). The implementation is identical for solid parcels.

To activate turbulent dispersion for solid parcels, specify your choice of model in `parcels.in > solid_parcels > parcel_list > parcel > turb_dispersion_control > dispersion_model`. By default, turbulent dispersion is turned off for solid parcels.

Solid Parcel Heat Transfer

You can use a correlation of the Ranz-Marshall type to model convection heating or cooling of solid parcels by the surrounding fluid. When this model is activated, CONVERGE solves for the parcel temperature T_p from

$$m_p c_p \frac{dT_p}{dt} = -h A_p (T_p - T_f) + q_{rad}, \quad (14.160)$$

where m_p , c_p , and A_p are the particle mass, specific heat, and surface area, respectively, and T_f is the fluid temperature. The last term, q_{rad} , accounts for heat flux due to radiation and is present only when [radiation modeling](#) is active and `radiation.in > radiation_spray_coupling > rad_spray_coupling_flag = 1`.

Chapter 14: Discrete Phase Modeling

Parcel Settings Solid Parcels

CONVERGE calculates the heat transfer coefficient h from

$$h = \frac{\text{Nu} k_f}{2r_p}, \quad (14.161)$$

where k_f is the thermal conductivity of the fluid, r_p is the parcel radius, and the Nusselt number Nu is given by a Ranz-Marshall type of correlation derived by Whitaker for spherical solid particles ([Whitaker, 1972](#)),

$$\text{Nu} = 2 + \left(0.4 \text{Re}_p^{1/2} + 0.06 \text{Re}_p^{2/3} \right) \text{Pr}^{0.4} \left(\frac{\mu}{\mu_s} \right)^{1/4}. \quad (14.162)$$

In Equation 14.162, Re_p is the Reynolds number, Pr is the Prandtl number, and μ is the viscosity (evaluated at the fluid temperature). The viscosity μ_s is evaluated at the temperature at the particle surface (T_p). The length scale used to compute the Reynolds number is the particle diameter $2r_p$.

To activate the Ranz-Marshall convection model, set [*parcels.in* > solid_parcels > parcel_list > parcel > thermal_exchange_control > convection_model = RANZ_MARSHALL](#).

14.5 Parcel Introduction

This section describes how parcels are introduced to the domain. These settings are contained primarily within [*parcel_introduction.in*](#).

Nozzle Injection

In CONVERGE, concentrated parcel introduction (e.g., a fuel injector introducing liquid parcels) is called injection. An injector is a group of nozzles that have some of the same characteristics. Each injector has at least one nozzle, each with its own hole size, cone angle, position, and orientation. For this reason, injection is sometimes more specifically referred to as sometimes nozzle injection, as distinct from [boundary injection](#).

Injector and nozzle settings are described in the sections below.

Injector Inputs

Specify injector-related inputs in the *injections* settings blocks in [*parcel_introduction.in*](#).

Contraction Effects

CONVERGE accounts for the contraction effects of the nozzles in each injector based on the discharge coefficient ([*parcel_introduction.in* > injections > injection > injection_control](#) >

Chapter 14: Discrete Phase Modeling

Parcel Introduction Nozzle Injection

discharge_coeff_model > Cd), nozzle diameter ([parcel_introduction.in](#) > injections > injection > nozzles > nozzle > geometry > diameter), liquid density, injection rate-shape ([parcel_introduction.in](#) > injections > injection > injection_control > rate_shape), total injected mass ([parcel_introduction.in](#) > injections > injection > injection_control > tot_mass) and injection duration ([parcel_introduction.in](#) > injections > injection > injection_control > duration).

Discharge Coefficient Model

CONVERGE has multiple options for the discharge coefficient model ([parcel_introduction.in](#) > injections > - injection > injection_control > discharge_coeff_flag). If *discharge_coeff_flag* = 1, then CONVERGE dynamically calculates the velocity coefficient (C_v) based on the injection pressure at that time. CONVERGE then calculates the contraction coefficient (C_a) based on

$$C_a = \frac{C_d}{C_v}, \quad (14.163)$$

where the discharge coefficient (C_d) is user-specified (*injectors > injector > injector_control > discharge_coeff_model > Cd*). The effective area of the nozzle is reduced and the drop velocity magnitude is increased proportionally.

If *discharge_coeff_flag* = 2, both the discharge coefficient (C_d) and the velocity coefficient (C_v) are user-specified (in [parcel_introduction.in](#) > injections > - injection > injection_control > discharge_coeff_model > Cd and [parcel_introduction.in](#) > injections > - injection > injection_control > discharge_coeff_model > Cv, respectively). You can specify either a constant value or a [time-varying profile](#) for each parameter.

Variable RPM

You can run simulations with a [variable RPM](#). For variable RPM cases with spray injection, if the rate-shape is constant ([parcel_introduction.in](#) > injections > - injection > injection_control > rate_shape = CONSTANT), then CONVERGE automatically generates a [rateshape_inj<num>.in](#) file for each injector. For variable RPM cases, you can include [tabular profiles](#) for injection start time, injection duration, and total mass injected.

Coordinates

You can configure nozzle locations and orientations using a polar or Cartesian coordinate system. Specify the desired system via [parcel_introduction.in](#) > injections > - injection > nozzle_control > coord_sys. The options are POLAR, CARTESIAN, or POLAR-COPY (copy the attributes of the first nozzle to the remaining nozzles, in which case, CONVERGE will evenly space the nozzles around the injector axis). The POLAR-COPY option is convenient for multi-hole injectors.

If you select POLAR, you must specify the location and orientation of each injector. Use [parcel_introduction.in](#) > injections > - injection > injector_control > position to set the x, y, and z coordinates of the injector center. The angles [parcel_introduction.in](#) > injections > - injection > injector_control > angle_xy_inj and [injections > - injection > injector_control > angle_xz_inj](#)

Chapter 14: Discrete Phase Modeling

Parcel Introduction Nozzle Injection

determine the orientation of the injector, as shown below in Figures 14.9, 14.10, and 14.11. If you select the Cartesian coordinate system option, then CONVERGE does not use these injector position parameters. Instead, you must specify the location and orientation of each nozzle hole individually.

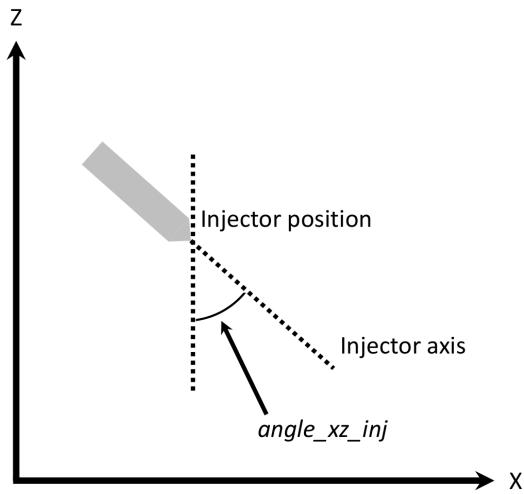


Figure 14.9: Parameters for placement and orientation of injectors.

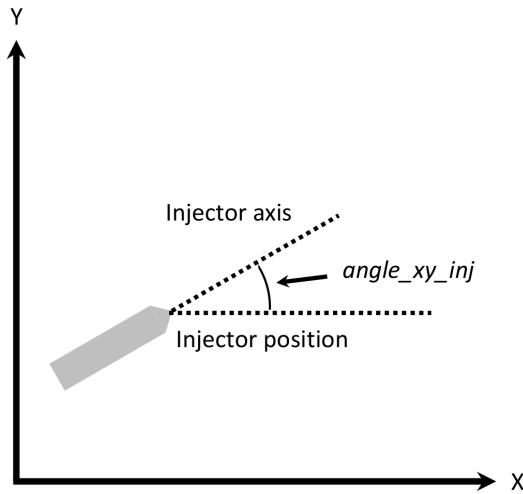


Figure 14.10: Parameters for placement and orientation of injectors.

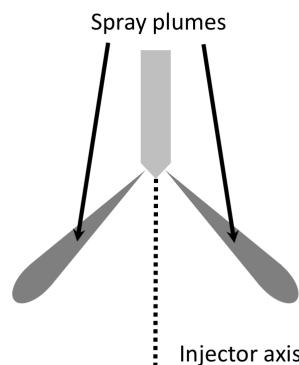


Figure 14.11: Spray plume relative to the injector axis.

The swirl fraction parameter ([*parcel_introduction.in*](#) > *injections* > - *injection* > *injector_control* > *swirl_fraction*) controls how much of the spray will move in a swirling direction. This parameter can be used only for hollow cone sprays (*i.e.*, when [*parcel_introduction.in*](#) > *injections* > - *injection* > *injector_control* > *cone_flag* = 0). The *swirl_fraction* can vary from -1 to 1. A positive *swirl_fraction* leads to clockwise (when looking from the top of the nozzle) rotational velocity while a negative value leads to counter-clockwise velocity. Figure 14.12

Chapter 14: Discrete Phase Modeling

Parcel Introduction Nozzle Injection

and the equations below it show the vectors used to define *swirl_fraction*, relative to the injector axis, and how the *swirl_fraction* is defined in terms of these vectors.

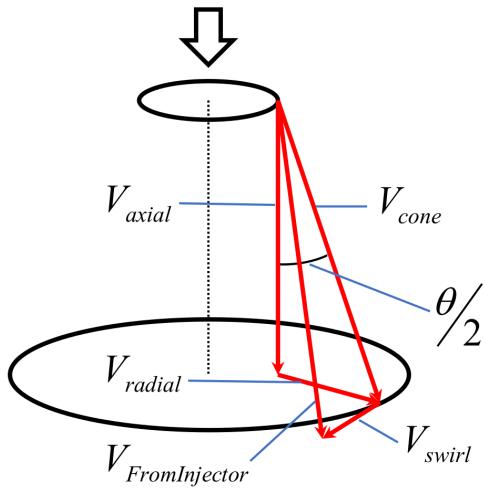


Figure 14.12: Vectors used to define the *swirl_fraction*.

The following equations define the *swirl_fraction*, relative to the injector axis, and how the *swirl_fraction* is defined in terms of these vectors:

$$\text{swirl_fraction} = \frac{V_{\text{swirl}}}{V_{\text{FromInjector}}}, \quad (14.164)$$

$$V_{\text{FromInjector}}^2 = V_{\text{Cone}}^2 + V_{\text{swirl}}^2, \quad (14.165)$$

$$V_{\text{Axial}} = \cos(\theta/2) V_{\text{Cone}}, \quad (14.166)$$

$$V_{\text{Radial}} = \sin(\theta/2) V_{\text{Cone}}, \quad (14.167)$$

and

$$\tan(\theta/2) = \frac{V_{\text{Radial}}}{V_{\text{Axial}}}. \quad (14.168)$$

Chapter 14: Discrete Phase Modeling

Parcel Introduction Nozzle Injection

You can optionally direct CONVERGE to inject parcels over an azimuthal subsection of the injector's solid or hollow spray cone. The optional inputs [parcel_introduction.in > injections > -injection > injector_control > azimuth_angle_start](#) and [azimuth_angle_end](#) control this feature. If not provided, CONVERGE defaults to `azimuth_angle_start = 0` and `azimuth_angle_end = 360` (i.e., a full cone). Note that this setting will equally affect every nozzle on the injector. These angles are defined such that the nozzle tangent vector is at an azimuthal angle of zero. CONVERGE increments the azimuth angle from the tangent vector toward the increment vector.

Table 14.4 presents tangent vectors and increment vectors for polar-coordinate injectors which are aligned with a coordinate axis, and Table 14.5 presents these vectors for axis-aligned injectors in Cartesian coordinates. We recommend using CONVERGE Studio to set up injectors that are not aligned with the coordinate axes.

Table 14.4: Tangent and increment vectors for polar-coordinate injectors.

Nozzle direction	Injector		Nozzle		Nozzle axis vector	Nozzle tangent vector	Nozzle increment vector
	<code>angle_xz_i</code> <code>nj</code>	<code>angle_xy_i</code> <code>nj</code>	<code>angle_xz</code>	<code>angle_xy</code>			
+x	-90	0	180	90	(1,0,0)	(0,-1,0)	(0,0,-1)
-x	-90	180	180	90	(-1,0,0)	(0,-1,0)	(0,0,-1)
+y	-90	90	180	90	(0,1,0)	(1,0,0)	(0,0,-1)
-y	-90	-90	180	90	(0,-1,0)	(-1,0,0)	(0,0,-1)
+z	0	90	180	90	(0,0,1)	(1,0,0)	(0,1,0)
-z	-180	90	180	90	(0,0,-1)	(1,0,0)	(0,-1,0)

Table 14.5: Tangent and increment vectors for Cartesian-coordinate injectors.

Nozzle direction	<code>noz_xx_vec</code>	<code>noz_yy_vec</code>	<code>noz_zz_vec</code>	Nozzle axis vector	Nozzle tangent vector	Nozzle increment vector
+x	1	0	0	(1,0,0)	(0,0,1)	(0,-1,0)
-x	-1	0	0	(-1,0,0)	(0,0,-1)	(0,1,0)
+y	0	1	0	(0,1,0)	(0,0,-1)	(-1,0,0)
-y	0	-1	0	(0,-1,0)	(0,0,1)	(-1,0,0)
+z	0	0	1	(0,0,1)	(0,1,0)	(-1,0,0)
-z	0	0	-1	(0,0,-1)	(0,-1,0)	(-1,0,0)

CONVERGE does not normalize the injected mass based on the active fraction of the azimuth. The entire specified mass will be injected through the active sector.

Nozzle Inputs

Specify nozzle-related inputs in the *nozzle* settings blocks in [*parcel_introduction.in*](#).

Each injector has at least one nozzle.

When [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzle_control* > *coord_sys* = POLAR or POLAR-COPY, you must define the location and orientation of the nozzles using [*parcel_introduction.in*](#) > *injections* > *nozzles* > *nozzle* > *geometry* > *radial_dist*, *axial_dist*, *theta*, *angle_xy*, and *angle_xz*. These parameters are illustrated below in Figures 14.13 and 14.14. The nozzle locations and orientations are relative to the injector center and axis. Figures 14.13 and 14.14 below are drawn with the assumption that the injector center is at the origin and its axis is lined up with the z axis, with the injector axis in the negative z direction.

When [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzle_control* > *coord_sys* = CARTESIAN, the nozzle locations are unrelated to each other and not linked to an injector center or axis. You must then define the location of a nozzle with two vectors: *position* and *orientation*.

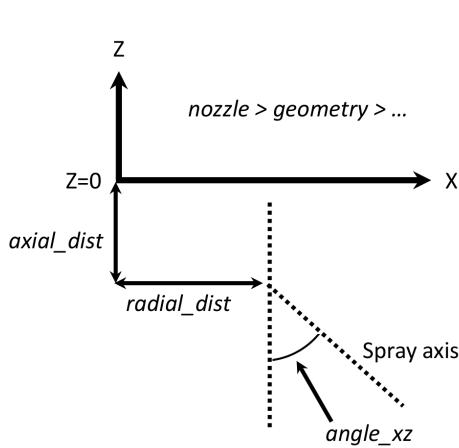


Figure 14.13: Parameters for placement and orientation of nozzles. These parameters are used only when [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzle_control* > *coord_sys* = POLAR or POLAR-COPY. This figure is drawn with the assumption that the injector center is at the origin and its axis is lined up with the z axis, with the injector axis in the negative z direction.

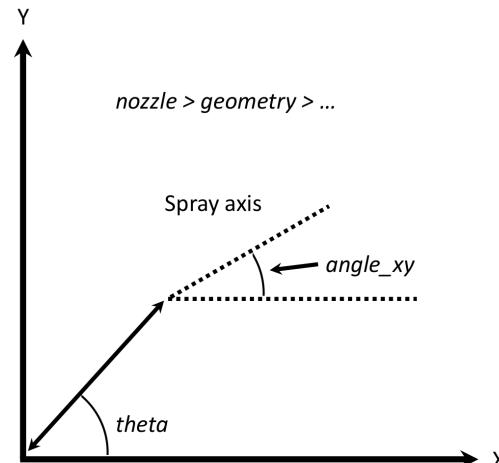


Figure 14.14: Parameters for placement and orientation of nozzles. These parameters are used only when [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzle_control* > *coord_sys* = POLAR or POLAR-COPY. This figure is drawn with the assumption that the injector center is at the origin and its axis is lined up with the z axis, with the injector axis in the negative z direction.

Dynamic Spray Cone Angle

To capture both the transient and steady-state behavior of the spray cone, you can optionally use a dynamic spray cone angle, as shown below in Figure 14.15. CONVERGE uses the spray cone angle correlation of [Tang et al. \(2018\)](#) to calculate the spray cone angle at each time-step between the start of injection (SOI) and the end of injection (EOI). This correlation is based on experimental data obtained from fuel sprays under diesel engine conditions.

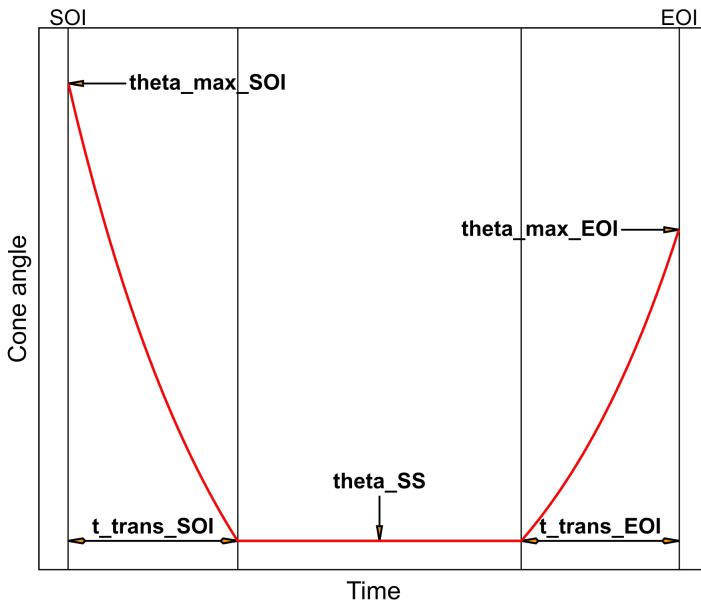


Figure 14.15: Spray cone angle profile from [Tang et al. \(2018\)](#).

To specify a dynamic spray cone angle, set `parcel_introduction.in > injections > injection > dynamic_spray_cone_angle_flag = 1` and include a `dynamic_spray_cone_angle.in` file in your case setup. This file includes parameters for the fuel type, ambient gas density, and injection pressure. CONVERGE uses these parameters to calculate the quantities listed below in Table 14.6, which are then used to construct the spray cone angle profile.

Table 14.6: Parameters defined in the spray cone angle correlation of [Tang et al. \(2018\)](#).

Parameter	Description
<code>theta_max_SOI</code>	Maximum angle at the start of injection (in degrees).
<code>theta_max_EOI</code>	Maximum angle at the end of injection (in degrees).
<code>theta_SS</code>	Angle at quasi-steady-state conditions (in degrees).
<code>t_trans_SOI</code>	Transition time from start of injection to quasi-steady state (in seconds if <code>inputs.in > simulation_control > crank_flag = 0</code> or in crank angle degrees if <code>crank_flag</code> is non-zero).
<code>t_trans_EOI</code>	Transition time from quasi-steady state to end of injection (in seconds if <code>inputs.in > simulation_control > crank_flag = 0</code> or in crank angle degrees if <code>crank_flag</code> is non-zero).

Chapter 14: Discrete Phase Modeling

Parcel Introduction Nozzle Injection

When you run a simulation with a dynamic spray cone angle, at each time-step CONVERGE writes a line such as the following to the [log file](#):

```
time vs angle: 7.00337e+02 1.93514e+01 2.08000e+01 1.72389e+01  
9.95978e+00 1.05438e-03 6.65840e-04 2.00000e+01
```

These values correspond to the time-step (in *seconds* if [inputs.in](#) > *simulation_control* > *crank_flag* = 0 or in *crank angle degrees* if *crank_flag* is non-zero), the current value of the spray cone angle (in *degrees*), *theta_max_SOI*, *theta_max_EOI*, *theta_SS*, *t_trans_SOI*, *t_trans_EOI*, and the ambient gas density (in kg/m^3), respectively.

Injection Size Distributions

CONVERGE includes several options for the injection size distribution. These distributions are specified via [parcel_introduction.in](#) > *injections* > *injection* > *injector_control* > *inject_distribution*. As an alternative to the three standard options ([blob](#), [chi_squared](#), [Rosin-Rammler](#), and [constant_injected_radius](#) distributions), you can enter a file name (e.g., [injdist.in](#)) to specify an injection profile.

Blob Injection Model

Set [parcel_introduction.in](#) > *injections* > *injection* > *injector_control* > *inject_distribution* = *BLOB* to direct CONVERGE to set injected drop sizes equal to the nozzle diameter or effective diameter when the discharge coefficient model is on (i.e., when *injectors* > *injector* > *injector_control* > *discharge_coeff_flag* is non-zero).

Chi Squared Distribution

Set [parcel_introduction.in](#) > *injections* > *injection* > *injector_control* > *inject_distribution* = *CHI-SQUARED* to direct CONVERGE to obtain injected drop sizes from the χ^2 distribution. This distribution is given by

$$C(r) = \frac{1}{r} \exp\left(-\frac{r}{\bar{r}}\right), \quad (14.169)$$

where r is the drop radius and \bar{r} is the number-averaged drop radius given by

$$\bar{r} = \frac{1}{3} r_{32}, \quad (14.170)$$

where r_{32} is the Sauter mean radius, defined as

Chapter 14: Discrete Phase Modeling

Parcel Introduction Nozzle Injection

$$r_{32} = \frac{\sum_{i=1}^{N_{tot}} N_i r_i^3}{\sum_{i=1}^{N_{tot}} N_i r_i^2}, \quad (14.171)$$

where N_{tot} is the total number of drops, N_i is the number of drops of size i , and r_i is the radius of size i .

Following the approach of Amsden *et al.* (1989), to obtain the best resolution of the size distribution where the most mass is located, the distribution should be proportional to the mass distribution given by $r^3 C(r)$. In order to create a probability density function of this mass distribution, the normalizing constant N_c must be found. This constant can be determined via

$$N_c = \int_0^\infty r^3 C(r) dr = \int_0^\infty \frac{r^3}{\bar{r}} \exp\left(-\frac{r}{\bar{r}}\right) dr = 6\bar{r}^3, \quad (14.172)$$

and the new distribution is given by

$$M(r) = \frac{r^3 C(r)}{N_c} = \frac{r^3}{6\bar{r}^4} \exp\left(-\frac{r}{\bar{r}}\right). \quad (14.173)$$

The cumulative probability function for this distribution is then given by

$$M(r) = 1 - \left(1 + \zeta + \frac{1}{2}\zeta^2 + \frac{1}{6}\zeta^3\right) \exp(-\zeta), \quad 0 < \zeta < 12 \quad (14.174)$$

where

$$\zeta = \frac{r}{\bar{r}}. \quad (14.175)$$

Once a value of ζ is selected, the injected drop radius is determined from

$$r = \bar{r}\zeta = \frac{1}{3}r_{32}\zeta, \quad (14.176)$$

Chapter 14: Discrete Phase Modeling

Parcel Introduction Nozzle Injection

where r_{32} is the Sauter mean radius, which is one-half of *injectors > injector > nozzles > nozzle > smd*.

Rosin-Rammler Distribution

Set [*parcel_introduction.in*](#) > *injections* > *- injection* > *injector_control* > *inject_distribution* = **ROSIN-RAMMLER** to use the Rosin-Rammler distribution to obtain injected drop sizes. The cumulative probability function for the Rosin-Rammler distribution is calculated with 100 bins and is given by

$$\tilde{R}(r) = 1 - \exp[-\zeta^q], \quad 0 < \zeta < \zeta_{\max} \quad (14.177)$$

where

$$\zeta = \frac{r}{\bar{r}}, \quad (14.178)$$

$$\zeta_{\max} = \ln(1000)^{\frac{1}{q}}, \quad (14.179)$$

q is an empirical constant ([*parcel_introduction.in*](#) > *injections* > *injection* > *injector_control* > *q_rr*), and

$$\bar{r} = \Gamma(1 - q^{-1}) r_{32}, \quad (14.180)$$

where Γ is the gamma function and r_{32} is the Sauter mean radius (one-half of [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzles* > *nozzle* > *smd*). Once a value of ζ is selected, the injected drop radius is determined from

$$r = \bar{r} \zeta = \Gamma(1 - q^{-1}) r_{32} \zeta. \quad (14.181)$$

Constant Injection Radius Distribution

Set [*parcel_introduction.in*](#) > *injections* > *injection* > *injector_control* > *inject_distribution* = **UNIFORM** to make the radius of injected drops be equal to one-half of *injectors > injector > nozzles > nozzle > smd*. This setting allows for a constant injected radius that is independent of the nozzle diameter.

Boundary Injection

You can use boundary injection to introduce parcels into the domain from a WALL or INFLOW boundary. For example, you can add solid particles to the flow at an INFLOW boundary to aid in [*erosion modeling*](#).

Chapter 14: Discrete Phase Modeling

Parcel Introduction Boundary Injection

Boundary injection is available for both liquid parcels and solid parcels. If you use boundary injection for liquid parcels, note that the [Kelvin-Helmholtz](#) and [LISA](#) breakup models are unavailable for parcels injected from a boundary.

To set up boundary injection, add the *boundary_injections* settings block to [*parcel_introduction.in*](#). For each boundary injection, specify the boundaries from which parcels enter the domain, the timing of the injection, and the properties of the injected parcels. You can specify the injection velocity directly or, for INFLOW boundaries, inject parcels with the same velocity as the local flow.

The amount of injected mass is controlled by the injection ratio ϕ , which can be defined in one of three ways. If [*parcel_introduction.in*](#) > *boundary_injections* > *boundary_injection* > *injection_control* > *injection_velocity_control* > *injection_ratio* > *type* = MASS, then ϕ is the parcel mass fraction given by

$$\phi = \frac{m_p}{m_f + m_p}, \quad (11.1)$$

where m_p is the parcel mass and m_f is the fluid mass. For each cell adjacent to the boundary, CONVERGE calculates the total parcel mass injected at each time-step as

$$m_{inj} = \left(\frac{\phi}{1 - \phi} \right) \rho_f A u_{inj} dt, \quad (11.2)$$

where ρ_f is the fluid density, A is the boundary surface area adjacent to the cell, and u_{inj} is the injection velocity (the latter is equal to the local fluid velocity if [*parcel_introduction.in*](#) > *boundary_injections* > *boundary_injection* > *injection_control* > *injection_velocity_control* > *method* = INJECT_WITH_FLOW).

If [*parcel_introduction.in*](#) > *boundary_injections* > *boundary_injection* > *injection_control* > *injection_velocity_control* > *injection_ratio* > *type* = VOLUME, then ϕ is the parcel volume fraction given by

$$\phi = \frac{V_p}{V_f + V_p}, \quad (11.3)$$

where V_p is the parcel volume and V_f is the fluid volume, and the parcel mass injected at each time-step is

Chapter 14: Discrete Phase Modeling

Parcel Introduction Boundary Injection

$$m_{inj} = \left(\frac{\phi}{1-\phi} \right) \rho_p A u_{inj} dt, \quad (11.4)$$

where ρ_p is the parcel density.

If `parcel_introduction.in > boundary_injections > boundary_injection > injection_control > injection_velocity_control > injection_ratio > type = AREA`, then ϕ is defined as a multiplier for the area A , and the parcel mass injected at each time-step is

$$m_{inj} = \phi \rho_p A u_{inj} dt. \quad (11.5)$$

The MASS and VOLUME injection ratio types are available only when `parcel_introduction.in > boundary_injections > boundary_injection > injection_control > injection_velocity_control > method = INJECT_WITH_FLOW`.

For all injection ratio types, the number of parcels injected at each time-step is m_{inj}/m_{1p} , where m_{1p} is the mass per parcel specified in `parcel_introduction.in > boundary_injections > boundary_injection > injection_control > mass_per_parcel`.

14.6 Parcel-Parcel Interaction

This section details how parcels interact with each other. These settings are contained primarily in [`parcels.in`](#).

Adaptive Collision Mesh

Lagrangian collision calculations can be highly grid-sensitive when an under-resolved fluid-phase mesh is used. To help alleviate this issue, CONVERGE includes an adaptive collision mesh option. This implementation is based on the concept proposed by [Hou \(2005\)](#).

In a simulation without a collision mesh, parcels can collide only with parcels in the same grid cell. This results in grid-related artifacts when parcels are dense relative to the grid spacing, even at grid spacings that are otherwise excessively fine.

Using a collision mesh can eliminate both the grid artifacts and the need for otherwise excessive resolution. Simulations with a collision mesh much more accurately represent parcel dispersion by eliminating grid effects. Figures 14.16 and 14.17 illustrate spray results from a simulation without a collision mesh and with a collision mesh, respectively.

Chapter 14: Discrete Phase Modeling

Parcel-Parcel Interaction

..... Adaptive Collision Mesh

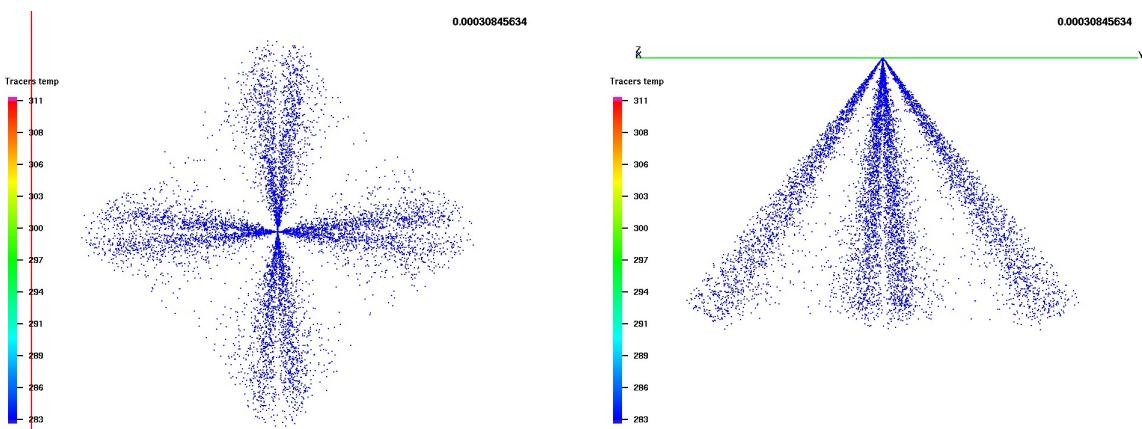


Figure 14.16: Parcels from a four-nozzle spray simulation without collision mesh. Notice the split in the spray plumes along the x and y axes in the left image. The right image shows how this split propagates along the z axis.

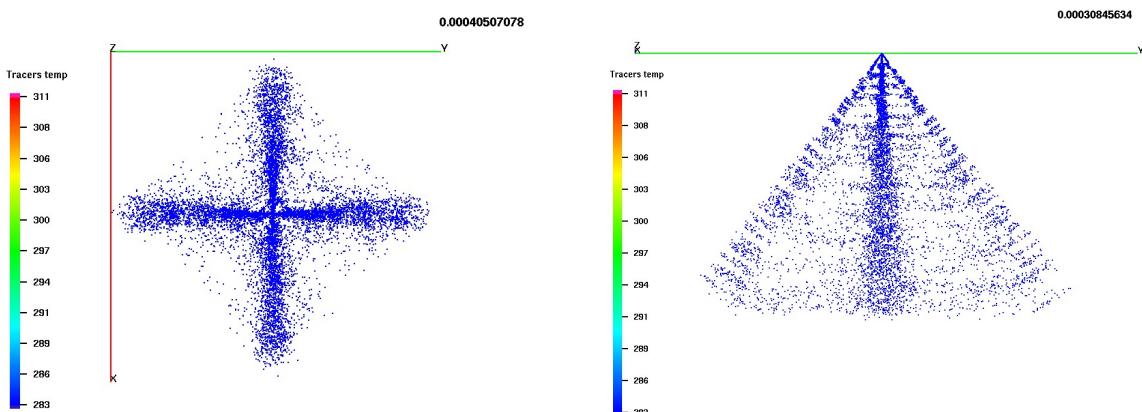


Figure 14.17: Spray parcels from a four-nozzle spray simulation with collision mesh. There is no grid effect-induced split along the horizontal and vertical axes in the left image. The right image shows the spray waves that more accurately mimic a physical spray.

The collision mesh is a uniform grid, used only for parcel collision, that rotates about a random axis at each time-step. The embed level of the collision mesh (set by the `coll_scale` parameter) is the number of levels below the base mesh size, as shown in Equation 14.182 below.

This mesh is completely independent of the fluid-phase mesh and is used only for collision calculations. The algorithm for creating the collision mesh is based on randomly selecting a coordinate system and creating a collision mesh at each time-step. The collision mesh cell size is based on `parcels.in > collision_mesh_embed_level` and is given by

$$dx_{coll} = \frac{dx_base}{2^{collision_mesh_embed_level}}, \quad (14.182)$$

where dx_base is the base cell size ([inputs.in](#) > *grid_control* > *base_grid*). Once the mesh is created, the parcels are placed in the appropriate collision mesh cell and the collision calculation proceeds as usual.

The cost of calculating the collisions grows proportional to the square of the number of droplets in a cell, so assigning the typical value of 3 to 5 for *collision_mesh_embed_level* may significantly reduce overall computational time of the simulation. Values in this range for *collision_mesh_embed_level* results in 512 to 32,768 collision mesh cells for every fluid base mesh cell, and thus drastically decreases the number of spray parcels per collision mesh cell. We recommend specifying a collision mesh to increase both the accuracy and the computational efficiency of a dense parcel simulation.

However, you should use caution when decreasing the collision mesh cell size. If the collision mesh is too fine, parcels can pass through a collision mesh cell without having the chance to collide. To alleviate this issue, you can activate the following time-step constraint:

$$dt_{mesh} = mult_dt_coll_mesh \cdot \frac{dx_{coll}}{V_{drop,max}}. \quad (14.183)$$

In Equation 14.183, dx_{coll} is the collision mesh cell size, $V_{drop,max}$ is the maximum drop velocity in the entire domain, and [inputs.in](#) > *temporal_control* > *mult_dt_coll_mesh* is a scaling constant.

Collision Models

This section provides an overview of the collision models available for liquid and solid parcels.

O'Rourke Numerical Scheme

The widely used [O'Rourke \(1981\)](#) parcel collision model is available for liquid and solid parcels. This model is designed to estimate the number of physical collisions and their outcomes in a relatively computationally efficient manner, which is made possible by the ensemble parcel concept. It is easy to see that for N drops or particles, each having $N-1$ possible collision partners, the number of possible collision pairs is approximately $(1/2)N^2$. Without the parcel concept this N^2 -dependence would render the collision calculation computationally prohibitive for the millions of drops or particles that may exist in a simulation.

Chapter 14: Discrete Phase Modeling

Parcel-Parcel Interaction

Collision Models

Because a parcel can represent hundreds or thousands of drops or particles, the cost of the collision calculation is significantly reduced. In order to reduce the computational cost even further, the algorithm of [O'Rourke \(1981\)](#) uses a stochastic (randomly determined) estimate of collisions and assumes that parcels can collide only if they are located in the same fluid-phase cell.

As described by [O'Rourke \(1981\)](#), CONVERGE performs a collision calculation for a pair of parcels. In O'Rourke's description, the parcel containing particles of larger radius is known as the collector (identified by subscript 1 below), while the parcel containing particles of smaller radius is known as the droplet (identified by subscript 2). Although this terminology is based on liquid parcels, CONVERGE performs the same calculation for solid parcels.

The collision frequency of a collector drop with all of the droplets is given by [O'Rourke \(1981\)](#) as

$$\nu_{coll} = \frac{N_2 \pi (r_1 + r_2)^2 V_{12}}{\mathcal{V}}, \quad (14.184)$$

where N_2 is the number of drops in the droplet parcel, $V_{12} = |v_{i1} - v_{i2}|$ is the relative velocity between the collector and droplet parcels, r_1 and r_2 are the radii of the collector and droplet, respectively, and \mathcal{V} is the volume of the fluid-phase cell that includes the two parcels. The probability that the collector collides n times with drops is assumed to follow a Poisson distribution given by

$$P_n = e^{-\bar{n}} \frac{\bar{n}^n}{n!}, \quad (14.185)$$

with a mean value given by

$$\bar{n} = \nu_{coll} dt = \frac{N_2 \pi (r_1 + r_2)^2 V_{12} dt}{\mathcal{V}}, \quad (14.186)$$

where dt is the computational time-step. In Equation 14.186 the quantity $\pi(r_1 + r_2)^2 V_{12} dt$ is the collision volume, which is simply the collision area $\pi(r_1 + r_2)^2$ multiplied by the distance traveled by a droplet in one time-step. Since there is a uniform probability that a droplet will be anywhere in the fluid-phase cell, then the probability of the droplet being in the collision volume is the ratio of the collision volume to the fluid-phase cell volume. Generalizing to parcels results in the multiplication of N_2 in Equation 14.186.

Chapter 14: Discrete Phase Modeling

Parcel-Parcel Interaction Collision Models

The probability of no collisions is given by

$$P_0 = e^{-\bar{n}}. \quad (14.187)$$

In CONVERGE, a random number between zero and one is chosen to determine if collision occurs. If the value of the random number is less than P_0 , no collision occurs. If the value of the random number is greater than or equal to P_0 , collision occurs.

In the event that a collision occurs, the next step is to determine the outcome of the collision. For liquid parcels, you can use [O'Rourke collision outcomes](#) or [Post collision outcomes](#). Both models utilize the critical impact parameter b_{crit} , calculated from

$$b_{crit} = (r_1 + r_2) \min \left(1.0, \frac{2.4f}{We_{coll}} \right), \quad (14.188)$$

where f is given by

$$f = \left(\frac{r_1}{r_2} \right)^3 - 2.4 \left(\frac{r_1}{r_2} \right)^2 + 2.7 \left(\frac{r_1}{r_2} \right), \quad (14.189)$$

and the collision Weber number We_{coll} is given by

$$We_{coll} = \frac{\rho_l V_{12}^2 r_2}{\sigma}. \quad (14.190)$$

The actual collision impact parameter is given by $b = (r_1 + r_2)\sqrt{Y}$, where Y is a random number between zero and one. In the [O'Rourke collision outcome model](#), the outcome is coalescence if $b < b_{crit}$ and otherwise the outcome is grazing collision. In the [Post collision outcome model](#), the outcome can be bouncing, stretching separation, reflexive separation, or coalescence.

For solid parcels, the [elastic and stick collision outcome models](#) are available.

NTC Numerical Scheme

The NTC method of [Schmidt and Rutland \(2000\)](#) is based on techniques used in gas dynamics for direct simulation Monte Carlo calculations. This model has been shown to be faster and more accurate than O'Rourke's model under certain conditions. Although this model was originally formulated for liquid parcels, it is also available for solid parcels in CONVERGE. The description below uses the terminology of liquid parcels.

The NTC method involves stochastic (randomly determined) sub-sampling of the parcels within each cell. This sampling potentially results in much faster collision calculations. Unlike O'Rourke's method, which incurs an additional computational cost that increases with the square of the number of parcels, the computational cost of the NTC method is linear with respect to the number of parcels. O'Rourke's method assumes that multiple collisions can occur between parcels and that this process is governed by a Poisson distribution. However, the Poisson distribution is not correct unless collisions have no consequences for the parcels. Since collisions change parcels' velocities, size, and number, the method of repeated sampling used by the NTC method generates more accurate results ([Schmidt and Rutland, 2000](#)).

The NTC method is derived, without assumptions, from the basic probability model for stochastic collision. The basic probability model requires that the cell size is sufficiently small such that spatial variations in spray quantities can be neglected. These assumptions are a subset of those required for deriving the O'Rourke collision model. For a detailed derivation, see [Schmidt and Rutland \(2000\)](#). Only a simple description of the actual implementation will be given here.

The NTC method first sorts the parcels into groups that reside in the same cell. This procedure requires only $2N$ operations, where N is the number of droplets in a cell. Next, the NTC method picks a stochastic subsample from all of the possible pairs in a cell. The number of picked pairs does not affect the final average answer, as long as the number meets constraints derived in [Schmidt and Rutland \(2000\)](#). The probabilities for the sub-sample pairs are multiplied by the reciprocal of this fraction, increasing the probability of collision. Sampling is performed with replacement so that multiple collisions for a pair can be correctly calculated. The resulting method incurs a cost that is linearly proportional to the number of parcels, as opposed to the N -squared cost of many existing methods.

If a cell contains N droplets which have a collision cross section given by $\sigma_{ij} = \pi(r_i + r_j)^2$, then the expected number of collisions in the cell over a time interval of Δt is given by summing the probability of all possible collisions as

$$M_{coll} = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \frac{V_{i,j} \sigma_{i,j} \Delta t}{\nabla}. \quad (14.191)$$

Chapter 14: Discrete Phase Modeling

Parcel-Parcel Interaction

Collision Models

The factor of one-half is a result of symmetry. If we group the individual droplets into parcels having identical properties, then the double summation becomes

$$M_{coll} = \frac{1}{2} \sum_{i=1}^{N_p} q_i \sum_{j=1}^{N_p} q_j \frac{V_{i,j} \sigma_{i,j} \Delta t}{\nabla}, \quad (14.192)$$

where N_p is the number of parcels in the cell and q is the number of droplets in a parcel. Evaluating this summation directly would be as expensive as the O'Rourke method, with the cost on the order of N_p^2 . However, this summation can be modified by pulling a constant factor outside of the summation so that

$$M_{coll} = \frac{(qV\sigma)_{max} \Delta t}{2\nabla} \sum_{i=1}^{N_p} q_i \sum_{j=1}^{N_p} \frac{q_j V_{i,j} \sigma_{i,j}}{(qV\sigma)_{max}}. \quad (14.193)$$

The value of $(qV\sigma)_{max}$ is used for scaling the selection probability of a collision. The value chosen must be sufficiently large so that the following restriction holds:

$$\frac{q_j V_{i,j} \sigma_{i,j}}{(qV\sigma)_{max}} < 1. \quad (14.194)$$

We next assume that a representative sub-sample of parcels may be randomly selected from the set of parcels in the cell, such that

$$\sum_{i=1}^{aN_p} x_i = a \sum_{i=1}^{N_p} x_i, \quad (14.195)$$

where $a < 1$ and x_i is a characteristic of each parcel. Hence, a subset of parcels is used to represent the larger population. This statistical approximation allows a constant multiplier to reduce the limits of summation. Using this relationship, the limits of the summations in Equation 14.193 are both reduced to

$$M_{coll} = \sum_{i=1}^{N_p \sqrt{\frac{(qV\sigma)_{max} \Delta t}{2\nabla}}} q_i \sum_{j=1}^{N_p \sqrt{\frac{(qV\sigma)_{max} \Delta t}{2\nabla}}} \frac{q_j V_{i,j} \sigma_{i,j}}{(qV\sigma)_{max}}, \quad (14.196)$$

This equation is written more succinctly by defining the quantity M_{cand} as

$$M_{cand} = \frac{N_p^2 (qV\sigma)_{\max} \Delta t}{2V}. \quad (14.197)$$

This definition is used in the limits of the summations of Equation 14.196 as

$$M_{coll} = \sum_{i=1}^{\sqrt{M_{cand}}} q_i \sum_{j=1}^{\sqrt{M_{cand}}} \frac{q_j V_{i,j} \sigma_{i,j}}{(qV\sigma)_{\max}}. \quad (14.198)$$

This equation is the final expression of the NTC method for application to parcels representing varying numbers of particles. In the limit of constant cross section and constant q , this equation reduces to the expression of [Alexander and Garcia \(1997\)](#). The overall cost is proportional to the product of the limits of the summation, namely M_{cand} . The value of M_{cand} is linearly proportional to N_p , because q goes as $1/N_p$. When using this model, you must make a sensible choice for $(qV\sigma)_{\max}$ for the algorithm to be efficient. If the spray is so dense that $M_{cand} > N_p^2 / 2$, then direct calculation of collisions may be more efficient than the NTC algorithm for this cell.

The double summation of Equation 14.198 is evaluated using an acceptance-rejection scheme. The number of candidate pairs given by M_{cand} is selected with replacement from the cell population. Because the parcels are selected with replacement, multiple collisions may occur between parcels. [O'Rourke \(1981\)](#) has observed that the consideration of multiple collisions is required for accurate results with large time-steps in dense sprays.

After a pair has been selected, a uniform deviate from [0,1) is used to determine if the candidate pair actually collides. A collision takes place between parcel i and j if the deviate, r , satisfies

$$r < \frac{q_g V_{i,j} \sigma_{i,j}}{(qV\sigma)_{\max}}. \quad (14.199)$$

The parameter q_g represents the greater number of particles between q_i and q_j . If the collision is accepted, then q_l , the lesser number of particles, actually participate in the collision. This distinction is important in the case of particle coalescence, where one parcel of particles absorbs the other. The differentiation of the parcels by the larger and smaller values of q does not change the expected outcome of the scheme.

Outcome Models

This section provides an overview of the collision outcome models available for liquid and solid parcels.

O'Rourke Collision Outcomes

For liquid parcels, the O'Rourke collision scheme can result in [grazing collisions](#) or in [coalescence](#).

In the case of a grazing collision, the new droplet velocities are calculated based on conservation of momentum and kinetic energy. It is assumed that some fraction of the kinetic energy of the drops is lost to viscous dissipation and angular momentum generation. Using assumed forms for the energy and angular momentum losses, [O'Rourke \(1981\)](#) derived the following expressions for the new velocities of the collector and droplet:

$$v_{i,1}^* = \frac{m_1 v_{i,1} + m_2 v_{i,2} + m_2 (v_{i,1} - v_{i,2}) \sqrt{1 - f_E}}{m_1 + m_2} \quad (14.200)$$

and

$$v_{i,2}^* = \frac{m_1 v_{i,1} + m_2 v_{i,2} + m_1 (v_{i,2} - v_{i,1}) \sqrt{1 - f_E}}{m_1 + m_2}, \quad (14.201)$$

where the $*$ superscript indicates the post-grazing collision velocity values and

$$1 - f_E = \frac{(b - b_{crit})^2}{(r_1 + r_2 - b_{crit})^2}, \quad (14.202)$$

where f_E is the fraction of energy dissipated in the collision.

In addition, grazing collisions only take place between the smaller number of drops between parcel 1 and parcel 2. For example, if parcel 1 has fewer drops than parcel 2, the updated velocities are given by

$$v_{i,1}^{n+1} = v_{i,1}^* \quad (14.203)$$

Chapter 14: Discrete Phase Modeling

Parcel-Parcel Interaction Outcome Models

and

$$v_{i,2}^{n+1} = \frac{N_1 v_{i,2}^* + (N_2 - N_1) v_{i,2}^n}{N_2}, \quad (14.204)$$

where the superscript n refers to the previous time-step value and the superscript $n+1$ refers to the updated value.

In the event that the outcome is coalescence, a [Poisson distribution](#) determines the number of droplets that coalesce with a collector drop. The properties of the coalesced drops are found using the basic conservation laws.

Post Collision Outcomes

In addition to the grazing collision and coalescence outcomes, [Post and Abraham \(2002\)](#) included both stretching separation and reflexive separation in their model based on experimental results of hydrocarbon drops. Post collision outcomes are available only for liquid parcels.

In this model, the collision Weber number based on diameter (*i.e.*, $2We_{coll}$), is first compared with a bouncing parameter given by

$$We_{Bounce} = \frac{\Delta_p (1 + \Delta_p^2)(4\phi' - 12)}{\chi_1 [\cos(\arcsin B)]^2}, \quad (14.205)$$

where

$$\Delta_p = \frac{r_1}{r_2}, \quad (14.206)$$

with $r_2 > r_1$,

$$\phi' = \phi_0 (\rho_a / \rho_0)^{2/3}, \quad (14.207)$$

where $\phi_0' = 3.351$, $\rho_0 = 1.16 \text{ kg/m}^3$, ρ_a is the gas density, and

Chapter 14: Discrete Phase Modeling

Parcel-Parcel Interaction Outcome Models

$$\begin{aligned}\chi_1 &= 1 - 0.25(2 - \tau)^2(1 + \tau) & \tau > 1.0 \\ \chi_1 &= 0.25\tau^2(3 - \tau) & \tau \leq 1.0,\end{aligned}\quad (14.208)$$

where

$$\tau = \frac{1 - B}{1 + \Delta_p}. \quad (14.209)$$

In addition, B in Equations 14.205 and 14.209 is given by

$$B = \frac{b}{r_1 + r_2}. \quad (14.210)$$

If $2We_{coll} < We_{bounce}$, it is assumed that the two parcels bounce. As in the work of [Hou \(2005\)](#), it is assumed that the post bounce velocities are given by Equations [14.200](#) and [14.201](#) with $f_E = 0$ as follows:

$$v_{i,1}^* = \frac{m_1 v_{i,1} + m_2 v_{i,2} + m_2 (v_{i,1} - v_{i,2})}{m_1 + m_2} \quad (14.211)$$

and

$$v_{i,2}^* = \frac{m_1 v_{i,1} + m_2 v_{i,2} + m_1 (v_{i,2} - v_{i,1})}{m_1 + m_2}. \quad (14.212)$$

If $2We_{coll} \geq We_{bounce}$, then either permanent coalescence, stretching separation, or reflexive separation take place. To determine if a separation has occurred, CONVERGE checks two other criteria. If $b > b_{crit}$, stretching separation may take place. If

$$2We_{coll} > 3 \left[7(1 + \Delta_p^3)^{2/3} - 4(1 + \Delta_p^2) \right] \frac{\Delta_p (1 + \Delta_p^3)^2}{\Delta_p^6 \eta_1 + \eta_2}, \quad (14.213)$$

Chapter 14: Discrete Phase Modeling

Parcel-Parcel Interaction Outcome Models

there is also a possibility of reflexive separation. Equation 14.213 is the reflexive separation criterion proposed by [Ashgriz and Poo \(1990\)](#), with

$$\eta_1 = 2(1 - \xi)^2 (1 - \xi^2)^{1/2} - 1, \quad (14.214)$$

$$\eta_2 = 2(\Delta_p - \xi)^2 (\Delta_p^2 - \xi^2)^{1/2} - \Delta_p^3, \quad (14.215)$$

and

$$\xi = 0.5B(1 + \Delta_p) \quad (14.216)$$

where $B = b / (r_1 + r_2)$.

If only stretching separation is possible, the post-collision velocities are calculated using Equations [14.200](#) and [14.201](#) with ([Hou, 2005](#))

$$1 - f_E = \left(\frac{b - b_{crit}}{1 - b_{crit}} \right)^2. \quad (14.217)$$

Alternatively, if only reflexive separation is possible, the post-collision velocities are calculated using Equations [14.200](#) and [14.201](#) with

$$1 - f_E = 1 - \frac{3 \left[7(1 + \Delta_p^3)^{2/3} - 4(1 + \Delta_p^2) \right] \Delta_p (1 + \Delta_p^3)^2}{2We_{coll} (\Delta_p^6 \eta_1 + \eta_2)}, \quad (14.218)$$

as in the work of [Hou \(2005\)](#). If $2We_{coll} \geq We_{bounce}$ and neither stretching separation or reflexive separation are possible, coalescence is assumed to occur.

Solid Parcel Collision Outcomes

CONVERGE includes two options for modeling collision outcomes for solid parcels.

When `parcels.in > solid_parcels > collision_control > collision_outcome_control > outcome_model = ELASTIC`, CONVERGE treats all solid parcel collisions as elastic. The final velocities of the two parcels are calculated based on conservation of kinetic energy and momentum, taking the coefficient of restitution into account.

When `parcels.in > solid_parcels > collision_control > collision_outcome_control > outcome_model = STICK`, CONVERGE generates a random number between zero and one for each colliding pair. If the random number is smaller than `parcels.in > solid_parcels > collision_control > collision_outcome_control > stick_percentage`, the parcels stick together. Otherwise, the collision is elastic.

When two parcels stick together, CONVERGE uses mass averaging to update the properties of the larger parcel. The properties of the smaller parcel are set to zero. For example, if r_1 and r_2 are the radii of two colliding parcels, with $r_1 > r_2$, then

$$r_1^{new} = \frac{m_1 r_1 + m_2 r_2}{m_1 + m_2} \quad (14.219)$$

and

$$r_2^{new} = 0. \quad (14.220)$$

To ensure mass conservation, CONVERGE recalculates the number of particles in the larger parcel using r_1^{new} and the total mass ($m_1 + m_2$), assuming that the density of the larger parcel is constant.

The temperature, velocity, and species mass fractions of the larger parcel are calculated from expressions analogous to Equation 14.219.

14.7 Parcel-Wall Interaction

This section details how parcels interact with WALL boundaries.

Liquid Parcels

CONVERGE offers three options for modeling the interaction of liquid parcels with boundaries: the [rebound/slide model](#), the [wall film model](#), and the [vanish model](#).

Rebound/Slide Model

You can use the wall impingement model of [Naber and Reitz \(1988\)](#), which was later improved by [Gonzalez et al. \(1991\)](#), to model the interaction of liquid spray parcels with solid surfaces. This model includes two impingement regimes, rebound and slide, that are based on the Weber number

$$We_i = \frac{\rho_1 V_n^2 d_0}{\sigma}, \quad (14.221)$$

of the incoming drop at impact, where V_n is the velocity component normal to the surface. If We_i is less than 80 (rebound regime), the drop rebounds elastically with a normal velocity given by [Gonzalez et al. \(1991\)](#) as

$$V_{n,o} = V_{n,i} \sqrt{\frac{We_o}{We_i}}, \quad (14.222)$$

where the Weber number of the outgoing drop We_o is obtained from

$$We_o = 0.678 We_i \exp(-0.04415 We_i). \quad (14.223)$$

Equation 14.223 is a numerical fit of the experimental observations of [Wachters and Westerling \(1966\)](#) for water drops impinging on a metal surface.

If We_i is greater than 80, the jet model of [Naber and Reitz \(1988\)](#) is used to update the drop velocity. In this model, the sheet thickness produced from an impinging liquid jet is assumed to be given by

$$H(\psi) = H_\pi \exp[\beta(1 - \psi/\pi)], \quad (14.224)$$

where H_π is the sheet height at $\psi = \pi$ and β is a parameter that can be determined from mass and momentum conservation. Naber and Reitz showed that β can be calculated from

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$\sin \alpha = \left(\frac{\exp \beta + 1}{\exp \beta - 1} \right) \frac{1}{1 + (\pi / \beta)^2}, \quad (14.225)$$

where α is the jet (or parcel) inclination angle. If we assume that $[(\exp \beta + 1) / (\exp \beta - 1)] \approx 1$, which is true for values of β greater than approximately 3, Equation 14.225 can be simplified to give

$$\beta = \pi \sqrt{\frac{\sin \alpha}{1 - \sin \alpha}}. \quad (14.226)$$

Naber and Reitz also showed that the angle ψ at which an impinging parcel leaves the surface is given by

$$\psi = -\frac{\pi}{\beta} \ln \{1 - YY[1 - \exp(-\beta)]\}, \quad (14.227)$$

where YY is a random number between zero and one.

Wall Film Model

CONVERGE offers a particle-based wall film for modeling the interaction of liquid drops with solid surfaces. The model uses a hybrid approach to film modeling: some calculations assume individual particle-based quantities, while other calculations assume film-based quantities. For example, the thickness of the film on wall face α , which is used throughout the film model, is given by

$$h_\alpha = \frac{\sum_p V_p}{|A_{\alpha,i}|}, \quad (14.228)$$

where V_p is the volume of parcel p , $A_{\alpha,i}$ is the area projection vector of face α , and the summation is over all particles located on face α . Details of the various aspects of the film model are given in the sections below.

Film Initialization

When `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > init_wall_film = 1`, CONVERGE reads `film_init.in`. This file defines how the wall film is to be initialized. A wall film can be initialized on an entire boundary or in a circular or rectangular

shape. Projections of circular or rectangular shapes are taken on wall boundaries as shown in Figure 14.18.

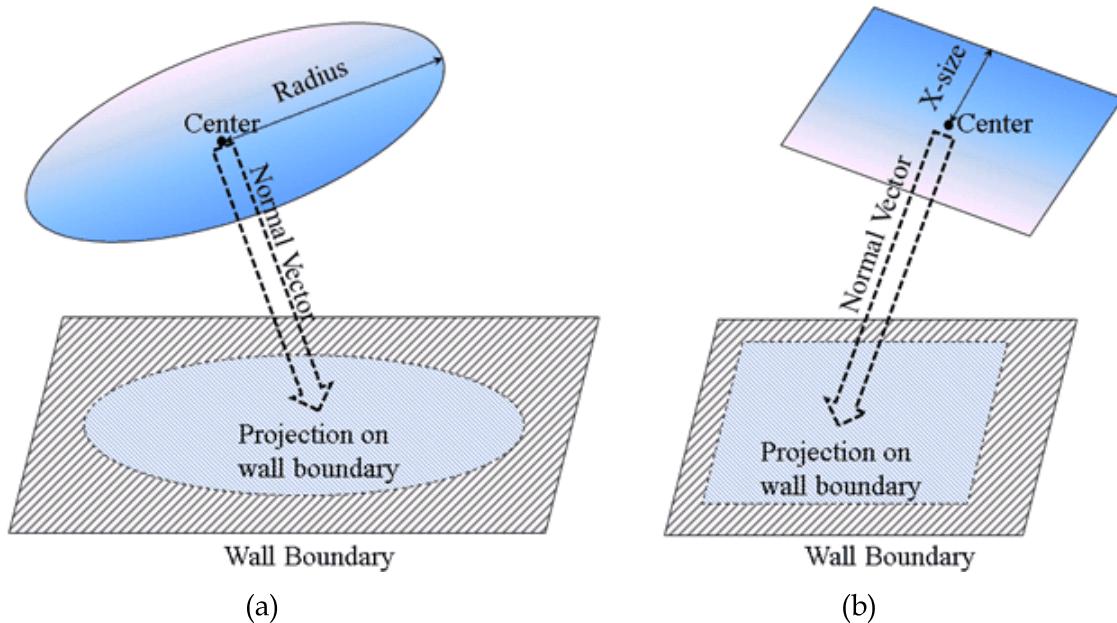


Figure 14.18: (a) Circular wall-film initialization, (b) Rectangular wall-film initialization.

Film Momentum Equation

The film momentum equation used to model liquid film transport is a modified version of the one proposed by [O'Rourke and Amsden \(2000\)](#), wherein momentum transfer from impinging spray parcels and inter-particle momentum transfer is accounted for via a drag force term. The following equation is solved for each particle of the wall film to update its velocity:

$$u_{p,i}^{n+1} = \frac{\frac{\rho_l h_\alpha}{dt} u_{p,i}^n + \tau_{w,\alpha} t_{\alpha,i} + \frac{2\mu_l}{h_\alpha} u_{wall,i} + S_{\alpha,i} - \Xi_\alpha n_{\alpha,i} + \frac{\beta\mu_l}{XM_{tot}} \sum_{k=1}^{N_\alpha} M_k u_{k,1}^n}{\frac{\rho_l h_\alpha}{dt} + \frac{2\mu_l}{h_\alpha} + \frac{\beta\mu_l}{XM_{tot}} \sum_{k=1}^{N_\alpha} M_k}, \quad (14.229)$$

where

$$\Xi_\alpha = \frac{2\mu_l}{h_\alpha} (u_{p,i}^n - u_{wall,i}) n_{\alpha,i} + S_{\alpha,i} n_{\alpha,i}, \quad (14.230)$$

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

and where α is the wall face on which the particle is located, ρ_l is the liquid density, h_α is the film thickness, $u_{p,i}^n$ is the film particle's i^{th} component of velocity from the previous time step, $\tau_{w,\alpha}$ is the shear stress on the gas-side of the wall film, $t_{\alpha,i}$ is the unit vector in the direction of gas motion, μ_l is the liquid viscosity calculated at the local mass-averaged film temperature, $u_{wall,i}$ is the boundary velocity, $n_{\alpha,i}$ is the unit vector normal to the boundary, β is a scaling factor used to modify the inter-particle drag (default value is 1.0), X is the average distance between two particles for a uniform distribution of particles over a plane area

(approximated to be $0.5\sqrt{A_\alpha}$ where A_α is the area of the boundary face) and M_{tot} is the total mass of all film particles on the α wall face. The summation in Equation 14.229 is over all N_α particles belonging to face α excluding the p^{th} particle. The term $S_{\alpha,i}$ is given by

$$S_{\alpha,i} = h_\alpha \left(\frac{\partial p_f}{\partial x_s} \right)_{\alpha,i} + \rho_l h_\alpha g_i, \quad (14.231)$$

where g_i is the acceleration due to gravity, and the film pressure gradient is approximated by

$$\left(\frac{\partial p_f}{\partial x_s} \right)_{\alpha,i} = \frac{1}{|A_{\alpha,i}|} \sum_m (p_f)_m t_{m,i} |x_{m+1,i} - x_{m,i}|, \quad (14.232)$$

where the sum is over the edges of face α , $(p_f)_m$ is the average of the film pressures on the two faces on either side of edge m , $x_{m+1,i}$ and $x_{m,i}$ are the vertex locations at the ends of edge m , and $t_{m,i}$ is the unit tangent to the surface in the direction normal to edge m , which is approximated as

$$t_{m,i} = \frac{\varepsilon_{ijk} (x_{m+1,i} - x_{m,i}) A_{m,j}}{|\varepsilon_{ijk} (x_{m+1,i} - x_{m,i}) A_{m,j}|}, \quad (14.233)$$

where $A_{m,j}$ is the average of the area projection vectors of the two faces on either side of edge m ([O'Rourke and Amsden, 2000](#)), and

$$\varepsilon_{ijk} = \begin{cases} 0 & \text{if any two indices are the same} \\ +1 & \text{if } ijk = 123, 231 \text{ or } 312 \\ -1 & \text{if } ijk = 132, 213 \text{ or } 321. \end{cases} \quad (14.234)$$

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

Note that the summation in Equation 14.232 is only over particles that impinge on the face α in the current time-step. The film pressure in Equation 14.232 is given by

$$p_{f,\alpha} = \dot{M}_{imp,\alpha} u_{wall,i} n_{\alpha,i} - \dot{P}_{imp,\alpha,i} n_{\alpha,i}, \quad (14.235)$$

where $\dot{M}_{imp,\alpha}$ is the mass impingement source given by

$$\dot{M}_{imp,\alpha} = \frac{\sum_p \rho_l V_p}{|A_{\alpha,i}| dt}, \quad (14.236)$$

and $\dot{P}_{imp,\alpha,i}$ is the momentum impingement source given by

$$\dot{P}_{imp,\alpha,i} = \frac{\sum_p \rho_l V_p (u_{p,i}^{n+1} - u_{p,i}^n)}{|A_{\alpha,i}| dt}. \quad (14.237)$$

In Equations 14.236 and 14.237, the summation is over only particles that stick to the face α in the current time-step.

The film jet model described in the [rebound/slide model](#) section is used to update the velocity of the particles that hit the wall. Once these particles have a new velocity tangential to the wall, their velocity will be further updated by Equation 14.229.

With the legacy approach, which can be activated by setting `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > wall_model = FILM_LEGACY`, the transfer of momentum from impinging spray to film is handled by mass and momentum sources appearing in the velocity equation as follows ([O'Rourke and Amsden, 2000](#)):

$$u_{p,i}^{n+1} = \frac{\frac{\rho_l h_\alpha}{dt} u_{p,i}^n + \tau_{w,\alpha} t_{\alpha,i} + \frac{2\mu_l}{h_\alpha} u_{wall,i} + S_{\alpha,i} - \Xi_\alpha n_{\alpha,i}}{\frac{\rho_l h_\alpha}{dt} + \frac{2\mu_l}{h_\alpha} + \dot{M}_{imp,\alpha}}, \quad (14.238)$$

where

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$\Xi_{\alpha} = \frac{2\mu_l}{h_{\alpha}} (u_{p,i}^n - u_{wall,i}) n_{\alpha,i} + S_{\alpha,i} n_{\alpha,i} - \dot{M}_{imp,\alpha} u_{wall,i} n_{\alpha,i}. \quad (14.239)$$

Next, the term $S_{\alpha,i}$ is given by

$$S_{\alpha,i} = \dot{P}_{imp,\alpha,i} - h_{\alpha} \left(\frac{\partial p_f}{\partial x_s} \right)_{\alpha,i} + \rho_l h_{\alpha} g_i. \quad (14.240)$$

In the legacy approach, Equation 14.238 is used to update velocities of particles that impinged on a solid surface in a previous time-step. If a particle has just impinged on a wall in the current time-step, CONVERGE calculates its velocity with the jet model described in the [rebound/slide model](#) section earlier in this section.

Drop/Film Rebounding

Liquid parcels with low Weber numbers may rebound off of a solid surface. In the present model, if

$$We_i < We_{rebound}, \quad (14.241)$$

then the parcel is assumed to rebound. $We_{rebound}$ is specified as `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > rebound > weber_no`. In Equation 14.241, We_i is given by

$$We_i = \frac{\rho_l V_n^2 d}{\sigma}, \quad (14.242)$$

where ρ_l is the liquid density, V_n is the parcel velocity component normal to the surface, d is the parcel diameter, and σ is the liquid surface tension.

Drop/Film Splashing: O'Rourke Model

To activate the O'Rourke film splash model, set `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > wall_model = FILM` and `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > splash > model = OROURKE`. Within the O'Rourke splash model, there are two different ways to determine the criterion for splash. The first option is based on the critical value of E_{crit}^2 , according to Equation 14.243 below (set `wall_control > film_control > splash > criterion_flag = 0`). The second option is based on the critical value of the Weber number, We (set `criterion_flag = 1`).

When `criterion_flag = 0`, the O'Rourke film splash technique follows the approach of [O'Rourke and Amsden \(2000\)](#). In this approach, the criterion for splash is given by

$$E^2 = \frac{We_i}{\min\left(\frac{h_\alpha}{d}, 1\right) + \frac{\delta_{bl}}{d}} > E_{crit}^2, \quad (14.243)$$

where We_i is given by Equation 14.242, h_α is the local film thickness, d is the impinging drop diameter, and the boundary layer thickness is given by

$$\delta = \frac{d}{\sqrt{Re_d}}, \quad (14.244)$$

where

$$Re_d = \frac{\rho_l V_n d}{\mu_l}. \quad (14.245)$$

In Equation 14.245, ρ_l is the liquid density, μ_l is the liquid viscosity, and V_n is the drop velocity component normal to the surface. [O'Rourke and Amsden \(2000\)](#) suggest a value of 3330 for E_{crit}^2 , based on the experimental work of [Mundo et al., \(1995\)](#). Specify E_{crit}^2 via `wall_control > film_control > splash > critical_value`.

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

When `parcel_wall_interaction.in` > `liquid_wall_interaction_control` > `default_wall_control` > `film_control` > `splash` > `criterion_flag = 1`, CONVERGE will employ the second (and simpler) approach for determining if a drop splashes by comparing the impinging drop Weber number, given by Equation 14.242, to a critical splash Weber number. In other words, if

$$We_i > We_{splash}, \quad (14.246)$$

then the drop splashes.

If it is determined that splashing is possible, a fraction of the impinging drop's mass, given by f_{splash} (`parcel_wall_interaction.in` > `liquid_wall_interaction_control` > `default_wall_control` > `film_control` > `splash` > `fraction`) is used to create a new parcel that splashes. The remaining fraction of mass, given by $(1 - f_{splash})$, is incorporated into the wall film.

The splashed parcel properties are initialized following the approach of [O'Rourke and Amsden \(2000\)](#). The drop velocity is calculated with the expression

$$u_{p,splash,i} = w'n_i + (0.12V_n + v')(\cos\psi e_{t,i} + \sin\psi e_{p,i}) + 0.8V_t e_{t,i}, \quad (14.247)$$

where n_i is the unit normal to the boundary, $e_{t,i}$ is the unit vector tangent to the surface and in the plane of both n_i and the incident drop velocity, and $e_{p,i} = \epsilon_{ijk}n_j e_{t,j}$. In addition, V_n is the normal component of the incident velocity and V_t is the component of the incident velocity that is tangent to the surface. Furthermore, w' is the normal component of the secondary droplet velocity which is chosen from the distribution ([O'Rourke and Amsden, 2000](#))

$$P(w') = \frac{4}{\sqrt{\pi}} \frac{(w')^2}{w_{\max}^3} \exp\left[-\left(\frac{w'}{w_{\max}}\right)^2\right], \quad (14.248)$$

where

$$w_{\max} = 0.2V_n. \quad (14.249)$$

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

The cumulative probability function for the above distribution can be obtained by integrating Equation 14.248 with respect to w' :

$$\tilde{P}(w') = \operatorname{erf}(\zeta) - \frac{2}{\sqrt{\pi}} \zeta \exp(-\zeta^2) \quad 0 < \zeta < 3, \quad (14.250)$$

where

$$\zeta = \frac{w'}{w_{\max}}. \quad (14.251)$$

Equation 14.250 is inverted numerically, using Newton's method, to obtain values of ζ for specific values of \tilde{P} . These values are calculated once at the start of the simulation and stored in a table. When a value of ζ is needed for a splashing calculation, a random number YY between zero and one is selected which represents the chosen value of \tilde{P} . The corresponding value of ζ is found by interpolating in the table. Once a value of ζ is selected, w' is calculated using Equation 14.251.

The fluctuating component of the secondary droplet tangential velocity, v' , is chosen from a Gaussian distribution ([O'Rourke and Amsden, 2000](#))

$$G(v') = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(v')^2}{2\sigma^2}\right], \quad (14.252)$$

where $\sigma^2 = 0.01V_n^2$ is the variance. It can be shown that the cumulative probability function for Equation 14.252 is given by

$$\tilde{G}(u_i') = \operatorname{erf}\left(\frac{v'}{\sqrt{2\sigma^2}}\right) = \operatorname{erf}(\zeta) \quad 0 < \zeta < 2, \quad (14.253)$$

where

$$\zeta = \frac{v'}{\sqrt{0.02V_n}}. \quad (14.254)$$

Equation 14.253 is inverted numerically, using Newton's method, to obtain values of ζ for specific values of \tilde{G} . These values are calculated once at the start of the simulation and stored in a table. When a value of ζ is needed for a velocity calculation, a random number YY between zero and one is selected which represents the chosen value of \tilde{G} . The corresponding value of ζ is found by interpolating in the table. Once a value of ζ is selected, v' is calculated using Equation 14.254. The sign of v' is determined based on the parameter XX = 1 - 2YY.

The final parameter needed in the splashed drop velocity expression, Equation 14.247, is the random angle φ . This angle is calculated in a similar way as in the jet model described previously in the [rebound/slide model](#) section.

Splashed drop radii are obtained from the following distribution ([O'Rourke and Amsden, 2000](#)):

$$F(r) = \frac{4}{\sqrt{\pi}} \frac{r^2}{r_{\max}^3} \exp\left[-\left(\frac{r}{r_{\max}}\right)^2\right]. \quad (14.255)$$

Following the approach of [Amsden et al. \(1989\)](#), in order to obtain the best resolution of the size distribution where the most mass is located, the distribution should be proportional to the mass distribution given by $r^3 F(r)$. In order to create a probability density function of this mass distribution, the normalizing constant N_c must be found. This constant can be determined via

$$N_c = \int_0^\infty r^3 F(r) dr = \int_0^\infty \frac{4}{\sqrt{\pi}} \frac{r^5}{r_{\max}^3} \exp\left[-\left(\frac{r}{r_{\max}}\right)^2\right] dr = \frac{4}{\sqrt{\pi}} r_{\max}^3, \quad (14.256)$$

and the new distribution is given by

$$M(r) = r^3 F(r) / N_c = \frac{r^5}{r_{\max}^6} \exp\left[-\left(\frac{r}{r_{\max}}\right)^2\right]. \quad (14.257)$$

The cumulative probability function for this distribution is given by

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$\tilde{M}(r) = 1 - \left[1 + \zeta^2 + \frac{1}{2} \zeta^4 \right] \exp[-\zeta^2] \quad 0 < \zeta < 3, \quad (14.258)$$

where

$$\zeta = \frac{r}{r_{\max}}. \quad (14.259)$$

Once a value of ζ is obtained, r is calculated from ([O'Rourke and Amsden, 2000](#))

$$r = \zeta \max\left(\frac{E_{crit}^2}{E^2}, \frac{6.4}{We_i}, 0.06\right) r_0, \quad (14.260)$$

where r_0 is the incident drop radius.

Drop/Film Splashing: Kuhnke Model

To activate the Kuhnke film splash model ([Kuhnke, 2004](#)), set `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > wall_model = FILM` and `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > splash > model = KUHNKE`. The Kuhnke splash model considers the temperature and the wetness (wall film thickness) of the wall to determine the extent of drop/film splashing.

The Kuhnke film splash model operates by comparing the K number of the liquid parcel (as defined below in Equations 14.261 and 14.262) to the splash critical K number, as shown later in this section in Figures 14.19, 14.20, and 14.21.

$$K = \frac{(\rho d)^{3/4} U^{5/4}}{\sigma^{1/2} \mu^{1/4}} \quad (14.261)$$

Alternatively, we can write

$$K = We^{5/8} La^{1/8}, \quad (14.262)$$

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

where We is the Weber number and La is the Laplace number, defined as

$$La = \frac{\sigma \rho L}{\mu^2}, \quad (14.263)$$

where σ is the surface tension, ρ is the density, L is the diameter of the parcel, and μ is the liquid viscosity. The K number accounts for the effects of both the kinematic condition and the size of the spray droplets.

When the K number of the liquid parcel is higher than the splash critical K number for the specific conditions of wall (based on T^* , which is the ratio of T_{wall} to T_{boil}) the drop will splash, thermally break up, or some combination of the two. Figure 14.264 below offers a conceptual scheme of the conditions under which the parcel will undergo each type of drop/wall interaction.

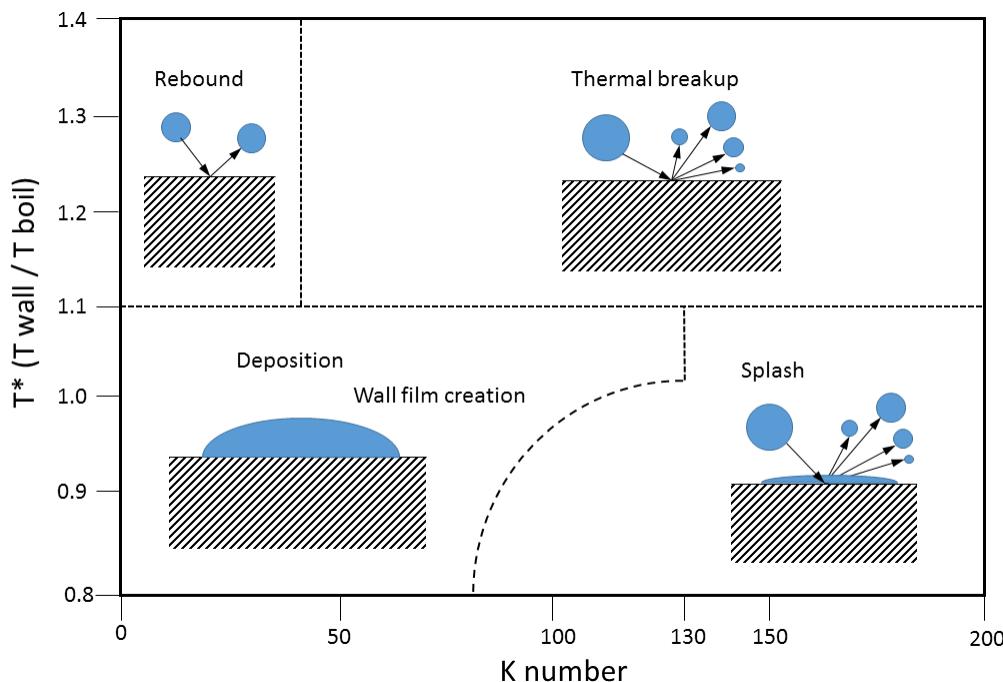


Figure 14.19: A conceptual representation of the four parcel/wall interaction outcomes for a dry wall available in the Kuhnke film splashing model. This map is for $T^*_{crit} = 1.1$.

The four outcomes shown above are as follows:

- When the K value of the parcel and wall temperature are both low, wall film will be created.
- When the K value is low but the wall temperature is high, the parcel will rebound.
- When the K value is high but the wall temperature is low, the parcel will splash.

- When the K value is high and the wall temperature is high, the parcel will thermally break up.

Dry Wall Splash Criteria

Equation 14.264 defines the critical K value above which a drop will splash upon impact with the wall as

$$K_{crit}(T^*) = (1 - \lambda)K_{cold} + \lambda K_{hot}, \quad (14.264)$$

where

$$\begin{aligned} K_{cold}(T^*, \varepsilon_a) &= K_{cold,0} - 12.75(\ln[\varepsilon_a] - X) + 12.75\sqrt{(\ln[\varepsilon_a] - X)^2 + 5}, \quad \text{where} \\ K_{cold,0} &= \begin{cases} 54 + 76e^{13(T^* - 1.0)} & T^* \leq 1.0 \\ 130 & 1.0 < T^* \leq T_{crit} \end{cases} \quad \text{and} \\ X &= -\frac{K_{cold,0} + 43.6}{25.5}, \end{aligned} \quad (14.265)$$

and

$$K_{hot}(T^*) = \text{unif}(20, 40) \quad T^* > T_{crit}^* + 0.06 \quad (14.266)$$

is a uniform distribution between 20 and 40. ε_a is the user-specified wall roughness, with a minimum value of 1.0e-6 m. The transition function, λ , connects K_{cold} and K_{hot} , and is defined by

$$\lambda(T^*) = \arctan\left[\omega\left(\frac{T^* - T_{mean}^*}{0.06}\right)\right] / \pi + \frac{1}{2}, \quad (14.267)$$

where $T_{mean}^* = [T_{crit}^* + (T_{crit}^* + 0.06)]/2$ and $\omega = 100$ is a tuned parameter that yields smooth transition within region $T^* \in (T_{crit}^*, T_{crit}^* + 0.06)$.

Figure 14.20 below shows critical K values versus the ratio of the wall temperature to the liquid boiling temperature for a dry wall condition, defined by Equation 14.264. The figure

shows that spray droplets splash/break up easily on dry walls when wall temperatures are significantly higher than the boiling temperature of the liquid.

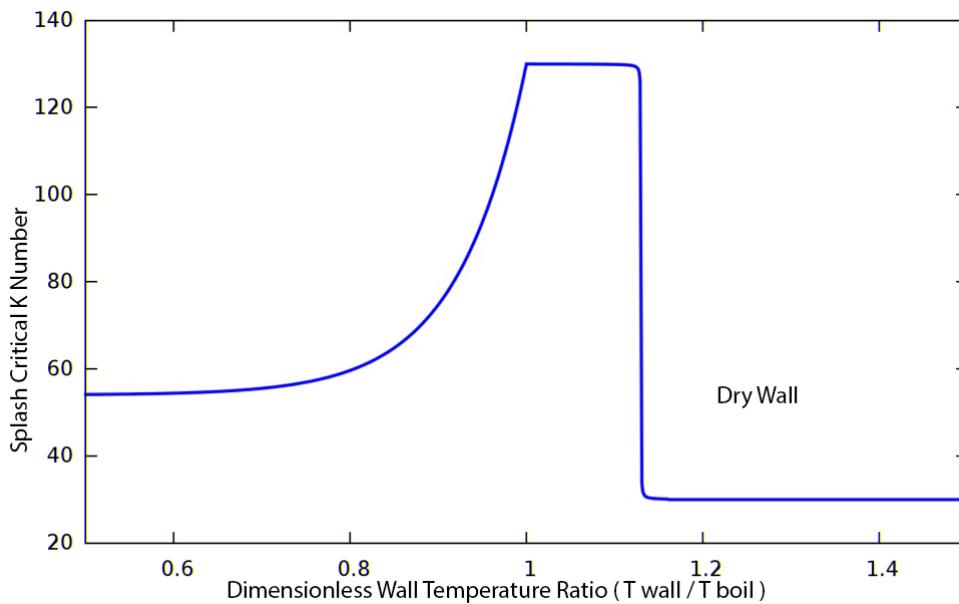


Figure 14.20: Critical K number values used in the Kuhnke film splash model for a dry wall at different (T_{wall}/T_{boil}) temperature ratios.

Wet Wall Splash Criteria

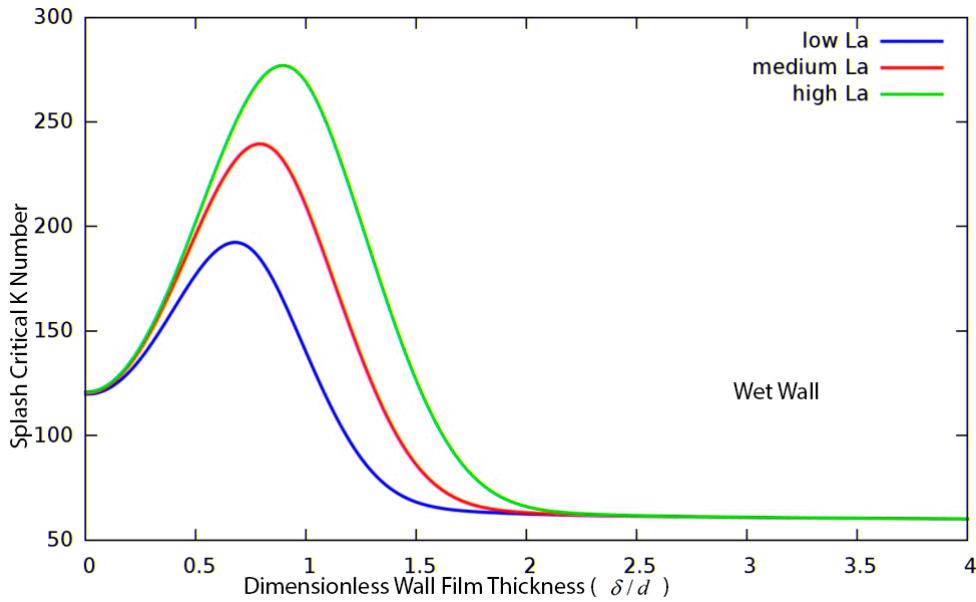


Figure 14.21: Data used in the Kuhnke film splashing model: K number values for a wet wall at different dimensionless wall film thicknesses.

Figure 14.21 above shows critical K values versus the relative thickness of existing wall film. For wet walls, the critical K value does not depend on the wall temperature and is given by

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$K_{crit}(\delta, La) = \phi(\delta, \delta_0, 125, 58.7, -100) + (408.4\delta_0 - 283.6) \cdot Weibull(\delta, \delta_0, 3), \quad (14.268)$$

where ϕ is a function defined as

$$\phi(x, x_0, y_l, y_r, s) = \frac{y_r - y_l}{\pi} \arctan \left(\pi s \frac{x - x_0}{y_r - y_l} \right) + \frac{y_r + y_l}{2}, \quad (14.269)$$

δ_0 is defined by $\delta_0 = \phi(La, 600, 0.85, 1.08, 0.003)$, and $Weibull()$ is the Weibull (Rosin-Rammler) distribution PDF function defined as

$$Weibull(x, \bar{x}, q) = \left(\frac{x}{\bar{x}} \right)^{q-1} \frac{q}{\bar{x}} e^{-\left(\frac{x}{\bar{x}} \right)^q}, \quad (14.270)$$

which is used to model the humps in Figure 14.21. This figure shows that, in the Kuhnke model, spray droplets will deposit into the existing film when the wall film is thin and they will splash into secondary droplets when the wall film thickness, δ , is thick (relative to the droplet diameter, d).

Secondary Droplet Properties

The Kuhnke model will automatically determine the properties of secondary droplets generated from slash/breakup based on experimental data and conservation laws.

The mass fraction of splashed secondary droplets is calculated according to Equation 14.271 below. Note that this model also allows existing wall film to be splashed into secondary droplets.

$$v_m = \begin{cases} \min \left\{ 1, \frac{T^* - 0.8}{T_{crit}^* - 0.8} (1 - B) + B \right\}, & B = 0.2 + 0.6p \text{ dry} \\ \min \left\{ 1 + \frac{m_{wf}}{m_d}, \frac{T^* - 0.8}{T_{crit}^* - 0.8} (1 - B) + B \right\}, & B = 0.2 + 0.9p \text{ wet} \end{cases} \quad p = rand(0, 1) \quad (14.271)$$

The diameter of the secondary droplets is calculated as

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$\gamma_{10} = \frac{d_{10}}{d_0} = \begin{cases} 3.3e^{3.6\left(\frac{\alpha}{\pi}\right)^2} We^{-0.65} & \text{dry} \\ 2.2e^{3.6\left(\frac{\alpha}{\pi}\right)^2} We^{-0.36} & \text{wet.} \end{cases} \quad (14.272)$$

Then CONVERGE determines the number of secondary droplets based on mass conservation.

The velocity magnitude, U_{a1} , is calculated from the Weber number of the secondary droplets, We_{a1} . Equation 14.273 below, which is derived with energy conservation in consideration, provides an estimate of We_{a1} as

$$We_{a1} = \begin{cases} \gamma_{10} \left[We_{a0} \left(1 - 0.85 \sin^2 \alpha \right) + 12 \right] - \frac{12}{\nu_{32}} & \text{dry} \\ \max \left\{ 51 - 7.1e^{3.4\frac{\alpha}{\pi}}, We_{a0} \left[-0.378 \left(\frac{\alpha}{\pi} \right)^2 - 0.123 \left(\frac{\alpha}{\pi} \right) + 0.156 \right] \right\} & \text{wet} \end{cases} \quad (14.273)$$

$$\Rightarrow U_{a1} = \sqrt{\frac{\sigma We_{a1}}{\rho d_{10}}}.$$

The directions of the splashed velocity are described using an ejection angle, β , and a deviation angle, ψ , which are calculated using Equations 14.274 and 14.276, respectively. In Equation 14.274, it is assumed that the ejection angle follows a logistic distribution with a standard deviation of 4 and a mean of $\bar{\beta}$, calculated using Equation 14.275.

$$\beta = 4 \log \frac{\xi}{1-\xi} + \bar{\beta}, \quad \xi = rand(0,1), \quad (14.274)$$

where

$$\bar{\beta} = \begin{cases} 9.3 + \ln(\varepsilon_a)(2.7 - 0.03\alpha) + 0.22\alpha & \text{cold, dry} \\ 0.225\alpha e^{(0.017\alpha - 0.937)^2} & \text{cold, wet} \\ \alpha 0.96 e^{-0.0045 We} & \text{hot.} \end{cases} \quad (14.275)$$

The deviation angle is given by

$$\psi = -\frac{\pi}{\omega} \ln \left[1 - p \left(1 - e^{-\omega} \right) \right], \quad (14.276)$$

where

$$\omega = \begin{cases} \sqrt{\frac{1 + 8.872 \cos(1.152\alpha)}{1 - \cos \alpha}} & \alpha \leq 80 \\ \frac{\pi^2}{2} \cos \alpha & \alpha > 80 \end{cases} \quad p = \text{rand}(0,1). \quad (14.277)$$

Dense Spray Consideration

The discussion above considers only a single drop impinging on a wall. To model the real spray-wall interaction, multiple drop effects need to be taken into account. It is observed in experiments that for dense spray the critical K value is much lower than for single drop interaction. The ejection angles of the secondary droplets in dense spray are generally larger than for single drop interaction. The dense spray splashed droplet diameter is more uniform. Good estimates of the critical K , secondary droplet diameter, γ , and ejection angle, β , for multiple drop interaction are

$$K_{MD} \approx 25, \quad (14.278)$$

$$\gamma_{MD} = \begin{cases} 0.5 & dry \\ 1.0 & wet, \end{cases} \quad (14.279)$$

and

$$\beta_{MD} = 1.5\beta_{SD}. \quad (14.280)$$

Kuhnke introduces a blending function to allow smooth transition between single and multiple drop interaction regimes, *i.e.*,

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$\begin{aligned} K_{crit} &= K_{SD} + \lambda_{MD}(K_{MD} - K_{SD}) \\ \gamma &= \gamma_{SD} + \lambda_{MD}(\gamma_{MD} - \gamma_{SD}) \\ \beta &= \beta_{SD} + \lambda_{MD}(\beta_{MD} - \beta_{SD}), \end{aligned} \quad (14.281)$$

where λ_{MD} is the blending function. For a dry wall condition,

$$\lambda_{MD} = \max \left\{ \frac{D_l^* - \kappa}{D_l^* - 1}, 0 \right\} \quad \kappa \geq 1, \quad (14.282)$$

where $D_l^* = (0.28We + 1)^{0.39}$. For a wet wall condition,

$$\lambda_{MD} = e^{\frac{1-\kappa}{c_{md}}}. \quad (14.283)$$

In Equations 14.282 and 14.283, κ represents the average distance between spray drops relative to the drop diameter. $\kappa = 1$ means that the drops are next to each other in the spray; $\kappa = 100$ means that the distance between drops is 100x the drop diameter. κ cannot be calculated exactly, but can be estimated as

$$\kappa = \left(\frac{A_{ref} dt}{\sum_i N_i d_i^2 \tau_{exp_i}} \right)^{\frac{1}{2}}, \quad (14.284)$$

where A_{ref} is a reference area upon which the wall film spreads, dt is the time-step, N_i is the number of drop that impact area A_{ref} within dt , d_i is the diameter of the impinging drop, and τ_{exp_i} is the lamella spreading time, which is estimated as

$$\tau_{exp} = 0.4\pi \sqrt{\frac{\rho d^3}{16\sigma}}. \quad (14.285)$$

Drop/Film Splashing: Bai-Gosman Model

To activate the Bai-Gosman film splash model ([Bai and Gosman, 1995](#); [Smith et al., 2015](#)), set `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > wall_model = FILM` and `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > splash > model = BAI-GOSMAN`.

As with the [Kuhnke model](#), the Bai-Gosman model considers the wall temperature, the Weber numbers of the impinging drops, and the wall condition (*i.e.*, wet or dry) to determine the extent of drop/film splashing. The Bai-Gosman model in CONVERGE has been customized for simulations that involve urea injection with selective catalytic reduction (*i.e.*, `inputs.in > feature_control > urea_flag = 1`). This model should be used with caution for non-urea applications.

The Bai-Gosman model operates by comparing the Weber number of an impinging drop to a set of splash critical Weber numbers. These critical limits are slightly different for dry wall and wet wall conditions. In addition, the wall temperature and wall condition (*i.e.*, wet or dry) will determine the eventual outcome of the impingement event (whether the drop deposits, splashes, rebounds, or goes through thermal breakup). Figures 14.22 and 14.23 illustrate the different drop-wall interaction regimes, similar to those presented by [Smith et al. \(2015\)](#).

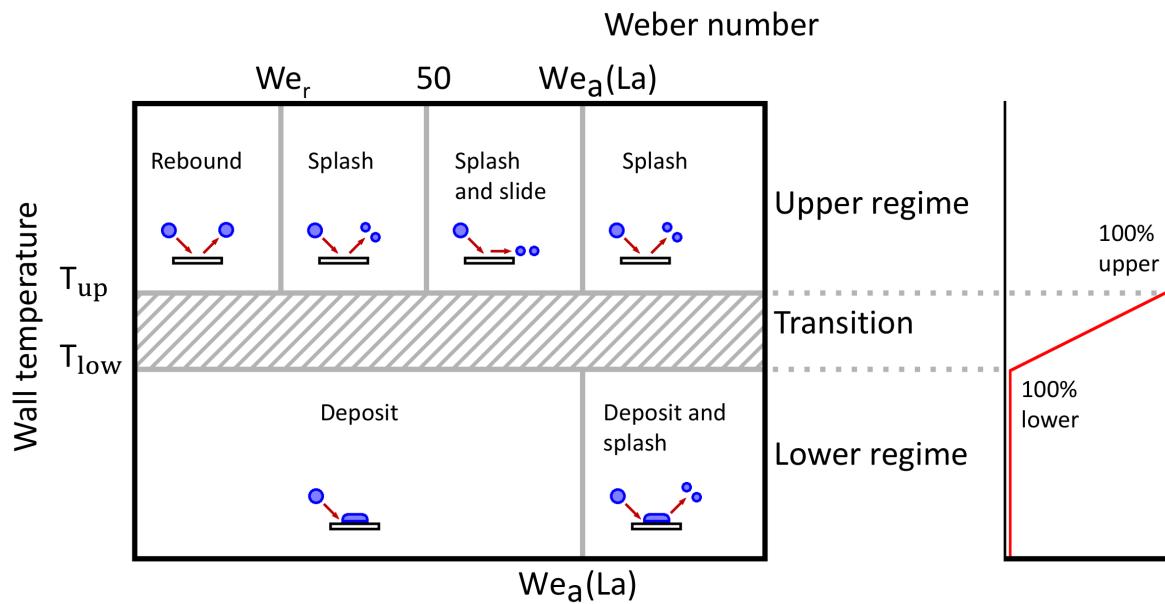


Figure 14.22: Bai-Gosman model regime diagram for dry wall conditions.

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

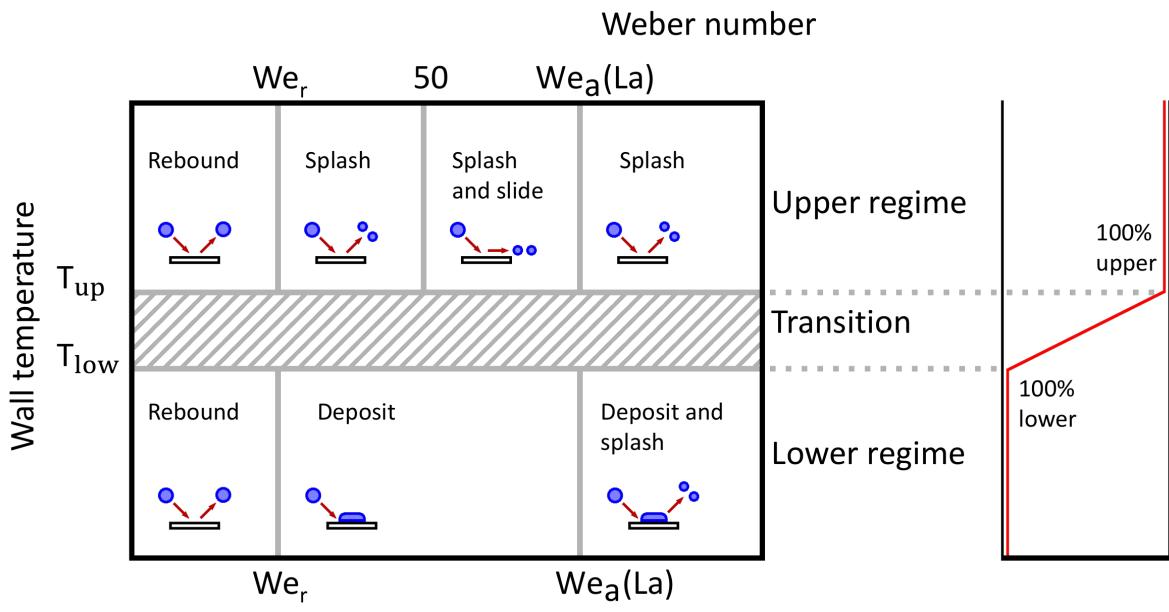


Figure 14.23: Bai-Gosman model regime diagram for wet wall conditions.

Note the existence of a transition region between the upper and lower halves of the regime diagrams, reflecting the lack of a sharp demarcation boundary for the deposition temperature. T_{upper} and T_{lower} denote the upper and lower bounds of the transition region, calculated as

$$\begin{aligned} T_{upper} &= T_{boil} + \delta T_{upper} \\ T_{lower} &= T_{boil} + \delta T_{lower}, \end{aligned} \quad (14.286)$$

where empirical studies suggest the following temperature offsets:

$$\begin{aligned} \delta T_{upper} &= 139 \\ \delta T_{lower} &= 38. \end{aligned} \quad (14.287)$$

Impinging drops that experience wall temperatures between T_{upper} and T_{lower} are shifted to either the upper or the lower half of the regime, based on the linear probability distribution shown alongside the regime diagrams.

There are several outcomes for dry walls.

- When the Weber number and the wall temperature are both low, the parcel will deposit on the wall, forming a wall film.

- When the Weber number is low and the wall temperature is high, the parcel will rebound.
- When the Weber number is high and the wall temperature is low, the parcel will undergo thermal breakup.
- When the Weber number and the wall temperature are both high, the parcel will splash, and, under certain conditions, slide just above the wall.

The outcomes for the wet walls are similar to those for dry walls, except that the critical Weber numbers for the regime boundaries are different, and an additional rebound regime is observed when both the Weber number and wall temperature are low. The splash critical Weber number We_a is a function of the Laplace number La and the actual surface roughness r_s ([Bai and Gosman, 1995](#); [Smith et al., 2015](#)). The rebound Weber number We_r is set to 5 in the original formulation, although in CONVERGE it can be any positive value less than 50 ([*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *default_wall_control* > *film_control* > *rebound* > *weber_no*).

In the rebound regime, the rebound angle of the parcel equals the incident angle, and it undergoes no other changes in its physical properties. In the splash regime, each incident parcel breaks up into two outgoing parcels, and the criteria used for determining the mass ratios, radii, and velocities of the secondary droplets is below. CONVERGE closely follow the methodology described in [Bai and Gosman \(1995\)](#).

Each incident splashing parcel (of mass m_I) breaks into two secondary parcels with equal mass ($m_s/2$). The secondary-to-incident mass ratios for dry and wet walls are given by

$$\begin{aligned} \text{For dry walls: } \frac{m_s}{m_I} &= 0.2 + 0.6\alpha \\ \text{For wet walls: } \frac{m_s}{m_I} &= 0.2 + 0.9\alpha, \end{aligned} \tag{14.288}$$

where α is a random number distributed uniformly in the interval (0,1). Note that, for wet walls, the ratio on the left side of Equation 14.288 can be greater than 1, since a part of the existing wall film might get splashed by the incident parcel.

For the secondary droplet sizes, mass conservation requires

$$N_1 d_1^3 + N_2 d_2^3 = \frac{m_s}{m_I} d_I^3, \tag{14.289}$$

where N_1 and N_2 are the number of droplets of diameters d_1 and d_2 , respectively. The total number of secondary droplets per splash ($N = N_1 + N_2$) is obtained from

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$N = a_0 \left(\frac{We}{We_{crit}} - 1 \right) \quad (14.290)$$

where a_0 is approximately 5, We is the Weber number, and We_{crit} is the critical Weber number. The value of N_1 is chosen randomly such that $1 \leq N_1 \leq N$, and thus $N_2 = N - N_1$. The secondary diameters can then be determined from the requirements

$$N_1 d_I^3 = \frac{m_s}{m_I} \frac{d_I^3}{2} \quad \text{and} \quad N_2 d_I^3 = \frac{m_s}{m_I} \frac{d_I^3}{2}, \quad (14.291)$$

where d_I is the diameter of the each droplet in the incident parcel.

The velocity magnitudes (U_1 and U_2) of the secondary droplets must satisfy the overall energy conservation equation

$$\frac{m_s}{4} (U_1^2 + U_2^2) + \pi \sigma (N_1 d_I^2 + N_2 d_I^2) = \frac{m_I}{2} V_{I,n}^2 - \frac{\pi}{12} \sigma \left(\frac{m_s}{m_I} \right) We d_I^2 N_I, \quad (14.292)$$

where σ is the surface tension and $V_{I,n}$ is the incident normal velocity.

The kinetic energy of the incident parcel should be large enough to overcome the critical splashing energy (which is the final term in Equation 14.292, and the excess energy is then used for the creation and motion of the secondary drops. An additional equation for the two secondary velocity magnitudes is obtained from the size-velocity correlation

$$\frac{U_1}{U_2} \approx \frac{\ln(d_I/d_I)}{\ln(d_2/d_I)}. \quad (14.293)$$

The ejection angles of the secondary drops are assumed to fall within a cone for both normal and oblique impingement. The angle for one secondary parcel is randomly selected in the range from 5 to 50 *degrees*, and the second angle can then be determined using the conservation of normal (tangential) momentum and the velocity magnitudes from Equations 14.292 and 14.293.

For the special case in which the drop slides just above the wall (described in #4 in the list of outcomes above), CONVERGE determines the parcel velocities from the slide regime in the [rebound/slide model](#).

Spray-Wall Heat Transfer: Wruck Model

You can use the Wruck model in conjunction with either the [Kuhnke](#) or the [Bai-Gosman](#) film splash model. To activate the Wruck model, first verify that either the [Kuhnke](#) or the [Bai-Gosman](#) film splash model has been activated and then set `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > splash > wruck_model_flag = 1.`

The Wruck model accounts for the thermal energy transfer from heated walls to the spray droplets that rebound or splash off of the walls ([Wruck and Renz, 2000](#)). The resultant heating of the droplets (and cooling of the walls) is an important phenomenon in many industrial applications (e.g., urea aftertreatment systems). In CONVERGE, the model has been augmented to allow for the Leidenfrost effect, which effectively shields droplets from direct contact with the wall (for at least a part of the total duration of contact), thus reducing the energy exchanged ([Birkhold, 2007](#)). Note that the Wruck model is active only when spray droplets impinge on temperature-coupled boundaries.

The model considers both the wall and the impinging parcel to be semi-infinite bodies with the exchange of energy taking place across an effective area and over an effective time of contact. The droplet deforms during its interaction with the wall (thus affecting the effective area of contact), which is also taken into account. When a droplet (at temperature T_d) is in direct contact (over a duration of direct contact t_{dc}) with a wall (at temperature T_w), the model approximates the total heat transfer Q_{w-d} as

$$Q_{w-d} = A_{cont} \frac{2\sqrt{t_{dc}}}{\sqrt{\pi}} \frac{b_w b_d}{b_w + b_d} (T_w - T_d), \quad (14.294)$$

where b_d and b_w are the thermal effusivities of the droplet and the wall, respectively. Each effusivity is a function of its thermal conductivity λ and heat capacity c_p . Equation 14.295 below describes the equation for the effusivities, where i represents either d (droplet) or w (wall), as

$$b_i = \sqrt{(\lambda \rho c_p)}. \quad (14.295)$$

The effective area of contact between the droplet and the wall (A_{cont}) is evaluated as

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$A_{cont} = \frac{1}{t_{dc}} \int_0^{t_{dc}} \frac{\pi}{4} d^2(t) dt, \quad (14.296)$$

where $d(t)$ represents the droplet diameter as it deforms during the duration of direct contact t_{dc} . An assumption for this formulation is that $d(t)$ is sinusoidal in nature, and its value depends on the kinetics of the impact (*i.e.*, whether the droplet rebounds or splashes, which is determined by the selected film splash model).

For rebounding droplets, $d(t)$ is

$$d(t) = d_{max} \sin\left(\frac{t}{t_{cont}}\pi\right). \quad (14.297)$$

For splashing droplets, $d(t)$ is

$$d(t) = d_{max} \sin\left(\frac{t}{t_{cont}}\frac{\pi}{2}\right). \quad (14.298)$$

The variable t_{cont} is the total duration of contact between the droplet and the wall. In Equations 14.297 and 14.298 above, d_{max} is the diameter of the droplet at the point of maximum deformation, which is obtained from the Akao correlation. This correlation is

$$\frac{d_{max}}{d} = 0.61 We^{0.38}, \quad (14.299)$$

where d is the diameter of the impinging drop. The total duration of contact depends also on the kinetics of the impact. For rebounding drops, the total duration of contact is

$$t_{cont} = \frac{\pi}{4} \sqrt{\frac{\rho d^3}{\sigma}}. \quad (14.300)$$

For splashing drops, the total duration of contact is

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$t_{cont} = \sqrt{\frac{\pi}{2}} \left[\frac{\rho d^5}{\sigma u_n^2} \right]^{0.25}, \quad (14.301)$$

where u represents the wall normal component of the impinging drop's velocity.

Note that the action of the Leidenfrost effect can cause the total duration of contact (t_{cont}) to differ from the duration of direct contact (t_{dc}), since the thin layer of vapor breaks direct contact between the drop and the wall. Figure 14.24 below offers a schematic of this effect.

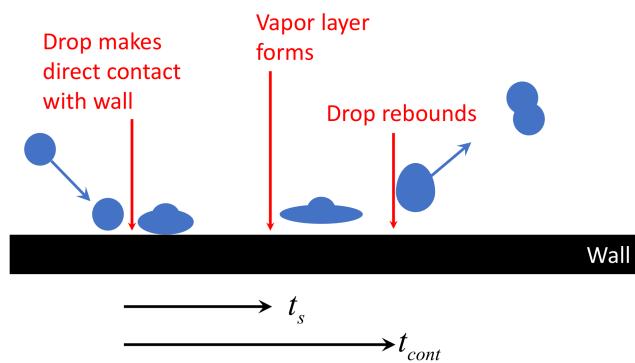


Figure 14.24: Timelapse schematic of a rebounding drop that demonstrates the initiation of the Leidenfrost effect.

Experimental observations show that drops need to be in direct contact with the wall for a certain minimum duration (t_s) before the Leidenfrost vapor lifts the drop and breaks direct contact. In the temperature range of 520–670 K, this minimum duration is given as

$$t_s = -1.687 \cdot 10^{-7} T_w + 1.376 \cdot 10^{-4}. \quad (14.302)$$

The duration of direct contact t_{dc} is thus

$$t_{dc} = \min(t_{cont}, t_s). \quad (14.303)$$

Film Separation

Film separation can occur if wall film particles flow over a sharp corner. The separation criterion of [O'Rourke and Amsden \(2000\)](#) is used to determine if film separation takes place. The criterion is given as

$$c_{sep} \frac{\rho_l \left[(u_{p,i} - u_{wall,i}) t_i \right]^2 \sin \theta}{1 + \cos \theta} > p_{gas}, \quad (14.304)$$

where c_{sep} is a constant (typically 3, and specified as `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > separation > const`), p_{gas} is the local gas pressure, and t_i and θ are defined in Figure 14.25.

If the inequality in Equation 14.304 is satisfied, the film parcels are converted to spray parcels with a diameter equal to the film thickness.

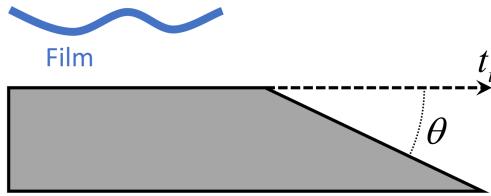


Figure 14.25: Schematic of the film separation criterion, with flow left to right.

Film Stripping

CONVERGE accounts for wall film stripping when `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > strip > active = 1`. The mechanism of wall film stripping is the same as the LISA breakup model of liquid sheet parcels, as discussed in the [LISA Breakup Model](#) section earlier in this chapter. Film stripping takes place due to the growth of waves on the surfaces caused by the aerodynamic forces acting on the film. Once the waves reach a critical amplitude, fragments of the liquid are broken off which contract to form cylindrical ligaments that are believed to move normal to the ligament axis. As a result, capillary forces cause the unstable ligaments to break into drops.

The maximum growth rate of the waves, Ω_s is obtained from

$$\omega_r = -2\nu_l k^2 + \sqrt{4\nu_l^2 k^4 + QU^2 k^2 - \sigma k^3 / \rho_l}. \quad (14.305)$$

The film stripping time, τ_{film_strip} is given by

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$\tau_{film_strip} = \frac{C_{time}}{\Omega_s}, \quad (14.306)$$

where C_{time} is the film strip time constant, which is given by `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > strip > time_const.`

The ligaments formed due to film stripping have a diameter given by

$$d_L = C_{size} \frac{2\pi}{K_s}, \quad (14.307)$$

where C_{size} is the film strip size constant and K_s is the wavenumber corresponding to the maximum growth rate Ω_s . The film strip size constant C_{size} is given by `parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > film_control > strip > size_const.`

Film Vaporization

Uniform Temperature Model

The film vaporization model in CONVERGE performs an energy balance between four energy terms: convection to the gas, vaporization of the film mass, conduction from the wall, and boiling of the film. A schematic of the model is shown in Figure 14.26.

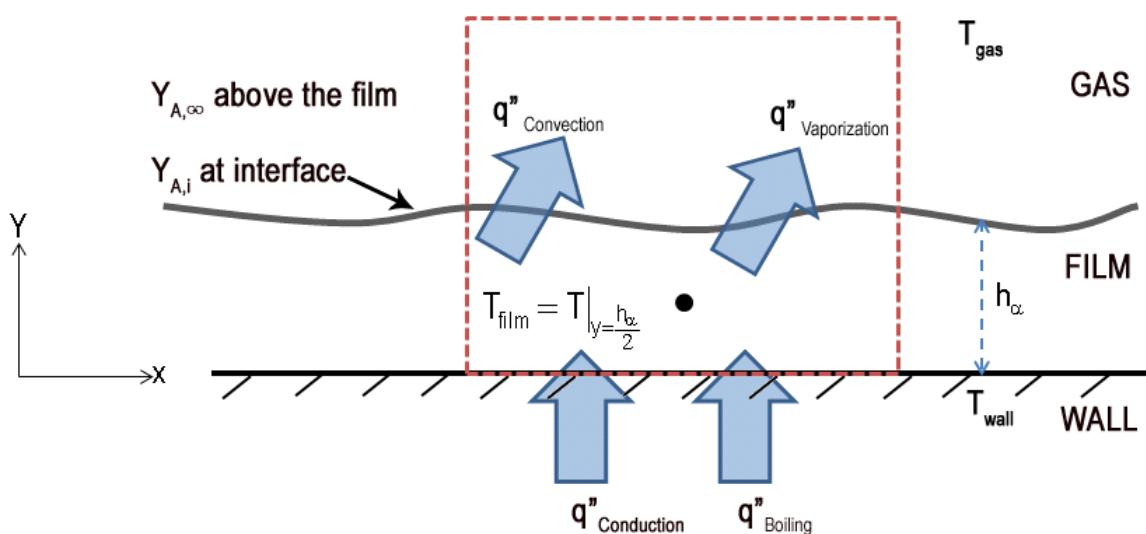


Figure 14.26: Schematic illustrating the film vaporization model in CONVERGE.

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

The thickness h_α of the film on the wall face α is calculated as

$$h_\alpha = \frac{\sum_p V_p}{|A_\alpha|}, \quad (14.308)$$

where V_p is the volume of parcel p and the summation is over all parcels located on face α , A_α is the area of face α .

The modeled film parcel temperature T_{film} is represented at the film half-height, as seen in Figure 14.26. It is solved by the following energy balance on the film mass:

$$\frac{(m_{film}^{n+1} C_{p,film}^{n+1} T_{film}^{n+1} - m_{film}^n C_{p,film}^n T_{film}^n)}{A_{film} (t^{n+1} - t^n)} = q''_{Convection} + q''_{Vaporization} + q''_{Conduction} + q''_{Boiling}, \quad (14.309)$$

where the superscript n represents the previous time-step, $n+1$ represents the current time-step, and t is the time. Area and heat capacity of the film are denoted by A_{film} and $C_{p,film}$, respectively. Heat fluxes due to convection, vaporization, conduction, and boiling are denoted by $q''_{Convection}$, $q''_{Vaporization}$, $q''_{Conduction}$, and $q''_{Boiling}$, respectively. The film mass m_{film} is obtained by summing over the masses for each drop in the parcel. $C_{p,film}$ is calculated by summing the specific heat capacities for each component in the film multiplied by their corresponding mass fractions. The functional forms of the heat flux terms are explained in the following paragraphs.

The convective heat transfer coefficient h is based on the cell and film temperatures and the thermal conductivity of the gas. The heat flux due to convection is given by

$$q''_{Convection} = \beta_{film} h (T_{gas} - T_{film}), \quad (14.310)$$

where β_{film} is the scaling factor for the heat transfer coefficient ([parcels.in > liquid_parcels > parcel_list > parcel > evap_control > scaling > heat_trans_coeff](#) optionally scaled by [parcel_wall_interaction.in > \[*_wall_control\] > film_control > evap_control > heat_trans_coeff](#)), T_{gas} is the temperature of the gas and T_{film} is the film temperature.

The mass flux due to vaporization for component A is given by

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$m''_A = \alpha_{film} \rho h_m \ln \left[\frac{1 - Y_{A,\infty}}{1 - Y_{A,i}} \right], \quad (14.311)$$

where α_{film} is the scaling factor for the mass transfer coefficient ([parcels.in > liquid parcels > parcel_list > parcel > evap_control > scaling > mass_trans_coeff](#) optionally scaled by [parcel_wall_interaction.in > \[*_wall_control\] > film_control > evap_control > mass_trans_coeff](#)), ρ is the density, h_m is the mass transfer coefficient, and $Y_{A,\infty}$ is the mass fraction of component A in the gas above the film. The model assumes equilibrium conditions at the gas-film interface, so the mass fraction at the interface $Y_{A,i}$ is given by

$$Y_{A,i} = \frac{P_{sat,A}|_{T_{film}}}{P} \frac{MW_A}{MW_{mix}}, \quad (14.312)$$

where $P_{sat,A}$ is the saturation vapor pressure of component A evaluated at the film temperature T_{film} , P is the gas pressure, MW_A is the molecular weight of component A, and MW_{mix} is the molecular weight of the mixture (not including vapor from the liquid species) at the liquid/gas interface. Equation 14.312 also assumes that vapor A is an ideal gas and that the liquid mixture is ideal. In Equation 14.311 h_m is calculated with the Reynolds-Colburn analogy of heat and mass transfer. It is given by

$$h_m = \frac{h}{\rho C_p} \left[\frac{Pr}{Sc} \right]^{2/3}, \quad (14.313)$$

where C_p is the specific heat capacity of the fuel mixture, Pr is the Prandtl number and Sc is the Schmidt number. The heat flux due to vaporization of component A is given by

$$q''_{A,Vaporization} = h_{fg,A}|_{T_{film}} m''_A, \quad (14.314)$$

where $h_{fg,A}$ is the heat of vaporization of component A at the film temperature T_{film} . We sum over all the liquid components to obtain the total heat flux due to vaporization:

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Liquid Parcels

$$q''_{Vaporization} = \sum_{components} q''_{A,Vaporization}. \quad (14.315)$$

Heat flux due to conduction in the film vaporization model is given by

$$q''_{Conduction} = k \frac{T_{wall} - T_{film}}{\frac{h_a}{2}} \approx k \frac{\partial T}{\partial y}, \quad (14.316)$$

where k is the thermal conductivity of the liquid mixture and T_{wall} is the wall temperature.

At each time-step, CONVERGE compares the boiling points of every component in the liquid mixture and identifies the lowest boiling point, $T_{Boil,min}$. If the film temperature T_{film} is found to be greater than $T_{Boil,min}$, then the component associated with that boiling point is considered to be boiling in that time-step. Only the component with the lowest boiling point is allowed to boil. If T_{film} is found to be less than $T_{Boil,min}$, boiling does not occur and the associated heat flux term is set to zero in Equation 14.309.

Heat flux due to boiling in this model is given by the Rohsenow Correlation ([Rohsenow, 1952](#)),

$$q''_{A,Boiling} = \mu_A h_{fg,A} \sqrt{\frac{g(\rho_A - \rho_{gas})}{\sigma_A}} \left(\frac{C_{p,A}(T_{wall} - T_{boil,A})}{0.006h_{fg,A} Pr_A} \right)^3, \quad (14.317)$$

where μ_A is the viscosity, $h_{fg,A}$ is the heat of vaporization, ρ_A is the density, σ_A is the surface tension, $C_{p,A}$ is the specific heat capacity, $T_{boil,A}$ is the boiling point, Pr_A is the Prandtl number for component A , and ρ_{gas} is the density of the gas. The parameters, μ_A , $h_{fg,A}$, ρ_A , ρ_{gas} , σ_A , and $C_{p,A}$ are evaluated at the wall temperature, T_{wall} . $q''_{Boiling}$ does not decrease in the model at increasing excess temperature, $(T_{wall} - T_{boil,A})$ once it reaches the maximum value, $q''_{Boil,max}$. Equation 14.318 gives $q''_{Boil,max}$ as

$$q''_{Boil,max} = 0.149 h_{fg,A} \rho_{gas} \left[\frac{\sigma_A g (\rho_A - \rho_{gas})}{\rho_{gas}^2} \right]^{1/4}. \quad (14.318)$$

Discretized Temperature Model

The discretized temperature model improves the accuracy of the temperature calculation within wall films. This model accounts for temperature gradients within the film during evaporation and is controlled via parameters in [*parcel_wall_interaction.in*](#). Use [*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *default_wall_control* > *film_control* > *evap_control* > *temp_discretization* > *film_height* to specify a film height (in meters) above which CONVERGE employs the discretized temperature model. Below this film height, CONVERGE employs the previously described uniform temperature model. Also, specify the number of finite volume cells in the wall film for [*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *default_wall_control* > *film_control* > *evap_control* > *temp_discretization* > *layers_per_film*.

Governing Equations

The governing equation for the temperature across the film is

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right), \quad (14.319)$$

where x is the wall normal direction, ρ is the film density, c_p is the specific heat of the film, and k is the film thermal conductivity. CONVERGE enforces the following temperature boundary conditions at the boundaries of the film. At $x = H$ (i.e., the maximum height of the film in the direction normal to the wall), the boundary condition is

$$k \frac{\partial T}{\partial t} = h (T_{gas} - T_{surf}) + \rho L \frac{dH}{dt}, \quad (14.320)$$

where T_{gas} is the gas temperature, T_{surf} is the surface temperature of the film, and L is the latent heat of the film. At the wall ($x = 0$), the boundary condition is $T = T_{wall}$. If the boiling model is active, a boiling flux is added to the conduction flux in a manner similar to that for the uniform temperature model.

Numerical Method: Finite Volume Derivation

The derivation of the discrete equations follows the method for the drop vaporization model described in [*Discretized Temperature Model*](#). As with the drop vaporization procedure, the derivation for film vaporization integrates the governing equation over the film volume and applies the divergence theorem to convert the volume integral to a surface integral. The derivation is, however, performed in Cartesian coordinates instead of spherical coordinates.

Applying Reynold's transport theorem allows the time derivative to be removed from the volume integral. The resulting discrete equation is

$$\frac{d}{dt} \left(\int_{x=0}^{H(t)} T \, dx \right) - T_{x=H} \left(\frac{dH}{dt} \right) = \frac{k}{\rho c_p} \frac{\partial T}{\partial x} \Big|_{x=H} - \frac{k}{\rho c_p} \frac{\partial T}{\partial x} \Big|_{x=0}. \quad (14.321)$$

Numerical Details

- Currently, density is a constant and \bar{c}_p is computed using $c_p(\bar{T})$.
- Time discretization is fully implicit and uses the same Δt as the flow solver.

Film Splash Considerations

If a parcel detaches from the film due to splash, CONVERGE initializes the parcel with a uniform temperature. To maintain conservation of energy, CONVERGE determines the parcel temperature such that the energy of the splashed mass in the film and spray remains constant.

Parcel Consolidation in Wall Films

For simulations that contain wall films, CONVERGE can combine parcels at boundary cells (set [*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *combine_parcels* > *active* = 1) to reduce computational time.

You must specify the time at which CONVERGE will begin consolidating film parcels ([*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *combine_parcels* > *start_time*), will stop consolidating film parcels ([*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *combine_parcels* > *end_time*), and the maximum number of film parcels per uncut boundary cell ([*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *combine_parcels* > *max_film_parcels_per_boundary_node*). CONVERGE will consolidate film parcels in boundary cells until either the maximum number of parcels or the end time is reached.

Note that activating parcel consolidation will result in a different calculated film mass, since momentum, parcel radius, and temperature will change after combining the parcels.

Adaptive Film Mesh

Parcel-based wall films are sensitive to grid alignment, and error in wall film height calculations can concentrate on grid boundaries. CONVERGE includes an adaptive film mesh model to minimize these grid effects. By introducing randomized grid boundary motion, film-height error is prevented from accumulating on a static grid face. This approach is analogous to the [*adaptive collision mesh*](#) discussed above.

The film mesh is completely independent of the fluid-phase mesh and is used only for film calculations. The algorithm for creating the film mesh is based on randomly selecting a coordinate system and creating a film mesh at each time-step. To activate the adaptive film mesh model, set [*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *default_wall_control* > *film_control* > *film_mesh* > *active* = 1. The film mesh cell size ([*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *default_wall_control* > *film_control* > *film_mesh* > *embedding_scale*) is given by

$$dx_{film} = \frac{dx_base}{2^{film_mesh_scale}}, \quad (14.322)$$

where dx_base is the base cell size specified via [*inputs.in* > *grid_control* > *base_grid*](#). Once the mesh is created, the parcels are placed in the appropriate film mesh cell and the film calculation proceeds as usual. There is no runtime difference when using the adaptive film mesh model.

CONVERGE [calculates film height](#) using the number of parcels in the near-wall cells. Thus, grid size and orientation affects the calculated wall film height. By activating the adaptive film mesh feature, the film mesh is oriented randomly at every time-step, and the parcels are effectively regrouped in the new mesh. The resulting calculated film height is more independent of the grid orientation.

You should set a value of [*parcel_wall_interaction.in* > *liquid_wall_interaction_control* > *default_wall_control* > *film_mesh* > *embedding_scale*](#) that will result in film mesh spacing approximately equal to, or slightly coarser than, the fluid mesh spacing (*i.e.*, set [*embedded.in* > *embedding* > *embed_scale*](#) to be equal to or less than 1). An overly coarse film mesh will result in a featureless and unresolved wall film. An overly fine film mesh (*i.e.*, $dx_{film} < dx_{local}$) will result in too few parcels being located in each film mesh cell, and physical film effects will not be resolved. If your case will generate fluid films on multiple boundaries, you should consider using an identical embedding level on each boundary.

Drop Vanish Model

To activate the liquid parcel vanish model, set [*parcel_wall_interaction* > *liquid_wall_interaction_control* > *default_wall_control* > *wall_model* = VANISH](#). In this model, the parcels vanish as they impinge on the wall boundary. The liquid parcel mass disappears from the simulation in such an instance. Note that this process is not the same as [drop vaporization](#).

Solid Parcels

For solid parcels, CONVERGE offers a set of parcel-wall interaction models and a set of erosion models.

Parcel-wall interaction models control the motion of solid parcels after they strike a WALL boundary. Options include the [rebound](#) model, the [slide](#) model, the [stick](#) model, and the [vanish](#) model.

[Erosion models](#) track the solid parcels that strike a WALL boundary and use that information to calculate an erosion ratio. These models have no impact on the motion of a parcel after it strikes the wall. Thus, when erosion modeling is active, your choice of parcel-wall interaction model is independent of your choice of erosion model.

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Solid Parcels

Rebound Model

To activate the solid parcel rebound model, set [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *wall_interaction_model* = REBOUND.

In this model, the component of parcel velocity normal to the wall (v_{\perp}) reverses direction after the parcel strikes the wall. The magnitude of the normal velocity is updated according to

$$|v_{\perp}^*| = C_{normal} |v_{\perp}|, \quad (14.323)$$

where C_{normal} is the coefficient of restitution specified in [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *coefficient_of_restitution* > *normal_direction*.

The tangential components of velocity remain in the same direction, with their magnitudes updated according to

$$|v_{\parallel,i}^*| = C_{tang} |v_{\parallel,i}|, \quad (14.324)$$

where C_{tang} is specified in [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *coefficient_of_restitution* > *tangential_direction*.

Slide Model

To activate the solid parcel slide model, set [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *wall_interaction_model* = SLIDE.

In this model, the component of parcel velocity normal to the wall (v_{\perp}) becomes zero after the parcel strikes the wall. The tangential components of velocity are updated according to

$$v_{\parallel,i}^* = C_{tang} v_{\parallel,i}. \quad (14.325)$$

The coefficient of restitution C_{tang} is specified in [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *coefficient_of_restitution* > *tangential_direction*.

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Solid Parcels

Stick Model

To activate the solid parcel stick model, set [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *wall_interaction_model* = *STICK*.

In this model, CONVERGE generates a random number between zero and one for each parcel that strikes the wall. If the random number is smaller than [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *stick_percentage*, the parcel sticks to the wall and the parcel velocity is updated to be the same as the wall velocity. If the random number is equal to or larger than *stick_percentage*, the parcel velocity is updated in the same way as in the [*rebound model*](#).

Vanish Model

To activate the solid parcel vanish model, set [*parcels.in*](#) > *solid_parcels* > *wall_interaction_control* > *wall_interaction_model* = *VANISH*. In this model, the parcels vanish as they impinge on the wall boundary. The parcel mass disappears from the simulation in such an instance.

Erosion Modeling

CONVERGE offers several models to predict the erosion of boundaries due to impinging solid particles (e.g., the erosion of gate valves due to sand in an oil or gas pipeline). These models provide a phenomenological description of the eroded mass (m_{er}) as a function of the mass and velocity of an impinging particle (m_{im} and v , respectively) and the angle of impingement (θ), as depicted in the figure below. CONVERGE uses the solid parcel framework to calculate the mass eroded by each parcel at a given time-step, and then adds the contributions from all parcels to compute the total mass eroded at that time-step.

The main quantity of interest for each model is the erosion ratio ER , which has the general form

$$ER = \frac{m_{er}}{m_{im}} \propto v^n F_s f(\theta), \quad (14.326)$$

where n is the velocity exponent, F_s is the particle shape factor, and $f(\theta)$ is the angle factor. The various models predict different exponents, different proportionality constants, and different functional forms for the angle factor. In CONVERGE, the shape factor F_s , which provides a measure of the sharpness of the impinging particles, is defined separately for each solid parcel species.

Erosion modeling is available only for solid parcels ([*inputs.in*](#) > *feature_control* > *parcel_mode* > *solid_parcel* = 1). To activate an erosion model, set [*inputs.in*](#) > *feature_control* > *erosion_flag* = 1 and include an [*erosion.in*](#) file in your case setup. In [*erosion.in*](#), you configure the parameters

for each available erosion model and define one or more erosion model groups. Each erosion model group contains a set of boundaries for which a specified erosion model will be used.

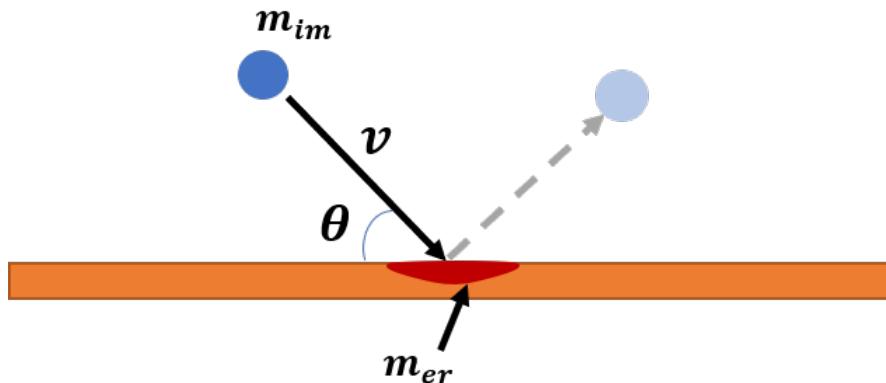


Figure 14.27: Erosion of a boundary due to an impinging solid particle.

You can activate erosion modeling in conjunction with the [fixed flow method](#) by setting [`inputs.in`](#) > `feature_control > fixed_flow_flag =1`. The drag on the solid parcels will be averaged throughout the fixed flow cycle and used as a source to further develop the velocity field during the next non-fixed flow cycle. This decreases the total simulation time. You can monitor the parcel drag source source by including `cells > general > parcel_source` in [`post.in`](#)

The sections below contain brief descriptions of the available erosion models.

Generic Erosion Model

The generic erosion model in CONVERGE calculates the erosion ratio as

$$ER = C v^n F_s f(\theta), \quad (14.327)$$

where C is the erosion constant. You can specify the angle factor $f(\theta)$ in a file (e.g., `angle_factor.dat`) formatted as in the example below. For values of θ not listed in this file, CONVERGE linearly interpolates between the values of $f(\theta)$ for the two angles closest to θ .

```
angle ftheta
0 0
0.1011777777777778 1.469168335506173e-05
0.2337555555555555 2.359193008691359e-05
0.3000444444444444 2.397081827353324e-05
0.3872666666666668 2.398171307212347e-05
0.4797222222222223 2.363119329054104e-05
0.5983444444444446 2.265272983859072e-05
0.6611444444444444 2.191535147741344e-05
0.7727888888888891 2.029428836323629e-05
0.8286111111111112 1.936711497578101e-05
0.9315333333333334 1.753198785901043e-05
1.0728333333333338 1.495273878188454e-05
1.1338888888888894 1.389424646417193e-05
1.1722666666666679 1.326470342233827e-05
1.2699555555555562 1.183573890960862e-05
1.3868333333333335 1.055647356880903e-05
1.4426555555555565 1.014533267133515e-05
1.5682555555555562 9.750156933058643e-06
```

Figure 14.28: Example of angle factor data file for the generic erosion model. Angles are specified in radians.

Arabnejad Erosion Model

The Arabnejad model ([Arabnejad et al., 2015](#)) considers the contributions due to cutting erosion, which dominates at oblique impact (small θ), and deformation erosion, which dominates at normal impact ($\theta \approx 90^\circ$). The erosion ratio is expressed as

$$ER = C_1 v^n F_s f(\theta) \quad \text{for } v \sin \theta \leq u_{tsh}, \\ = C_1 v^n F_s f(\theta) + C_2 F_s (v \sin \theta - u_{tsh})^2 \quad \text{for } v \sin \theta > u_{tsh}, \quad (14.328)$$

where C_1 and C_2 are the erosion constants for cutting erosion and deformation erosion, respectively. The latter is negligible when the normal component of the velocity is smaller than a threshold velocity u_{tsh} . For this model, the velocity exponent n is typically set to 2.41 and the angle factor is given by

$$f(\theta) = \frac{\sin \theta (2K \cos \theta - \sin \theta)}{2K^2} \quad \text{for } \theta \leq \arctan K, \\ = \frac{\cos^2 \theta}{2} \quad \text{for } \theta > \arctan K, \quad (14.329)$$

where the angle K depends on the material and particle properties.

Zhang Erosion Model

The Zhang model ([Zhang et al., 2007](#)) calculates the erosion ratio as

Chapter 14: Discrete Phase Modeling

Parcel-Wall Interaction Solid Parcels

$$ER = C(BH)^m v^n F_s f(\theta), \quad (14.330)$$

where C is the erosion constant, BH is the Brinell hardness of the boundary material, and the exponent m is typically set to -0.59. The angle factor is given by

$$f(\theta) = \sum_{i=1}^N A_i \theta^i, \quad (14.331)$$

where the polynomial coefficients A_i are empirically determined.

Generalized E/CRC Erosion Model

The Generalized E/CRC model ([Mansouri, 2016](#)) is a generalization of the [Arabnejad model](#) that can be applied to gas-sand, water-sand, and viscous-liquid-sand flows. The erosion ratio can be written as

$$\begin{aligned} ER &= C_1 v^n F_s f(\theta) && \text{for } v \sin \theta \leq u_{tsh}, \\ &= C_1 v^n F_s f(\theta) + C_2 F_s (v \sin \theta - u_{tsh})^2 && \text{for } v \sin \theta > u_{tsh}, \end{aligned} \quad (14.332)$$

where the angle factor is given by

$$\begin{aligned} f(\theta) &= \frac{\sin \theta (2K \cos \theta - \sin \theta)}{2K^2} && \text{for } \theta \leq \arctan K, \\ &= \cos^2 \theta - \frac{F(\text{St}) \sin^2 \theta}{2} && \text{for } \theta > \arctan K. \end{aligned} \quad (14.333)$$

The parameters C_1 , C_2 , u_{tsh} , and K have the same definitions as in the [Arabnejad model](#), while the dependence on the Stokes factor $F(\text{St})$ at large θ captures the differences among gas-sand, water-sand, and viscous-liquid-sand flows. The Stokes factor is defined as

$$F(\text{St}) = C_{st} \text{St}^x, \quad (14.334)$$

where the constant C_{st} and the exponent x are typically set to 0.24 and -0.503, respectively. The Stokes number can be expressed as

$$St = \frac{\tau_p u_f}{D}, \quad (14.335)$$

where τ_p is the particle response time, u_f is the velocity of the flow, and D is the diameter of the pipe. In CONVERGE, the diameter D is equal to the nozzle diameter (for nozzle injectors) or the diameter of a circle with the same area as the total area of the injecting boundaries (for boundary injectors). CONVERGE calculates the particle response time from

$$\tau_p = \frac{d_p^2 \rho_p}{18 \mu_f}, \quad (14.336)$$

where d_p is the particle diameter, ρ_p is the particle density, and μ_f is the fluid viscosity.

Oka Erosion Model

The Oka model ([Oka et al., 2005](#)) calculates the erosion ratio as

$$ER = K_p (\text{Hv})^{k_1} \left(\frac{v}{v_p} \right)^{k_2} \left(\frac{r_p}{D_p} \right)^{k_3} F_s f(\theta). \quad (14.337)$$

This expression depends explicitly on the impinging particle radius r_p in addition to the velocity v and angle of impact θ . Model parameters include the particle property factor K_p , the initial Vickers material hardness pyramid number Hv, the reference particle velocity v_p , the reference particle diameter D_p , and the exponents k_1 and k_3 . CONVERGE calculates the exponent k_2 from

$$k_2 = 2.3 (\text{Hv})^{0.038}, \quad (14.338)$$

and the angle factor is given by

$$f(\theta) = (\sin \theta)^{n_1} [1 + \text{Hv}(1 - \sin \theta)]^{n_2}. \quad (14.339)$$

The exponents n_1 and n_2 depend on the initial material hardness via the general expression

$$n_i = s_i (\text{Hv})^{q_i}, \quad (14.340)$$

where the model parameters s_i and q_i ($i=1,2$) are determined by the particle properties.

Finnie Erosion Model

The Finnie model ([Finnie, 1960](#)) considers a rigid abrasive particle that strikes a ductile material and cuts away part of the surface. The erosion ratio can be written as

$$ER = \frac{v^n}{p\psi K} F_s f(\theta), \quad (14.341)$$

where p is the plastic flow stress exerted by the material on the particle, ψ is the ratio of the depth of contact to the depth of cut, and K is the ratio of the vertical (normal) component of the force on the particle face to the horizontal component. The velocity exponent n is typically set to 2, and the angle factor is given by

$$\begin{aligned} f(\theta) &= \sin(2\theta) - \frac{6}{K} \sin^2 \theta && \text{for } \theta < \arctan\left(\frac{K}{6}\right), \\ &= \frac{K \cos^2 \theta}{6} && \text{for } \theta \geq \arctan\left(\frac{K}{6}\right). \end{aligned} \quad (14.342)$$

McLaury Erosion Model

The McLaury model ([McLaury et al., 1997](#)) focuses on the erosion caused by sand particles in the elbow fittings of a pipe. The erosion ratio can be expressed as

$$ER = (CR) ER_{std}, \quad (14.343)$$

where ER_{std} is the erosion ratio for a standard elbow and CR is a correction ratio that accounts for the effects of different elbow curvatures, carrier fluids, and particle sizes. The expression for the standard elbow is

$$ER_{std} = AF_s F_p \left(\frac{q_{sd} \rho_p v^n}{BH^m d} \right), \quad (14.344)$$

where A is a material-dependent coefficient, F_s is the sand sharpness factor, F_p is a penetration factor, q_{sd} is the sand production rate, ρ_p is the particle density, BH is the Brinell hardness of the pipe, and d is the pipe diameter. The exponents are typically set to $n = 1.73$ and $m = 0.59$. The correction ratio is given by

$$CR = \exp \left[- \left(\frac{0.215 \rho_f^{0.4} \mu_f^{0.65}}{d_p^{0.3}} + 0.03 \rho_f^{0.25} + 0.12 \right) \left(\frac{r}{d} - C_{std} \right) \right], \quad (14.345)$$

where ρ_f and μ_f are the fluid density and viscosity, respectively, d_p is the particle diameter, r is the elbow radius of curvature, and C_{std} is the r/d ratio for a standard elbow. CONVERGE sets $r/d = 1$ in Equation 14.345, while C_{std} is user-specified.

14.8 Eulerian-Lagrangian Spray Atomization

The Eulerian-Lagrangian Spray Atomization (ELSA) model is a spray injection model that combines Eulerian multi-phase modeling and Lagrangian particle tracking methods. The Eulerian multi-phase technique is used to model the liquid fuel within the fuel injector. This approach accounts for the effects of nozzle geometry on the injector flow field. CONVERGE tracks the density of the injected fluid as it enters the combustion chamber. When the liquid fuel is sufficiently dilute, it is converted into Lagrangian parcels, which are then subject to the spray modeling techniques described within this chapter. Figure 14.29 below illustrates the ELSA methodology.

Chapter 14: Discrete Phase Modeling

Eulerian-Lagrangian Spray Atomization

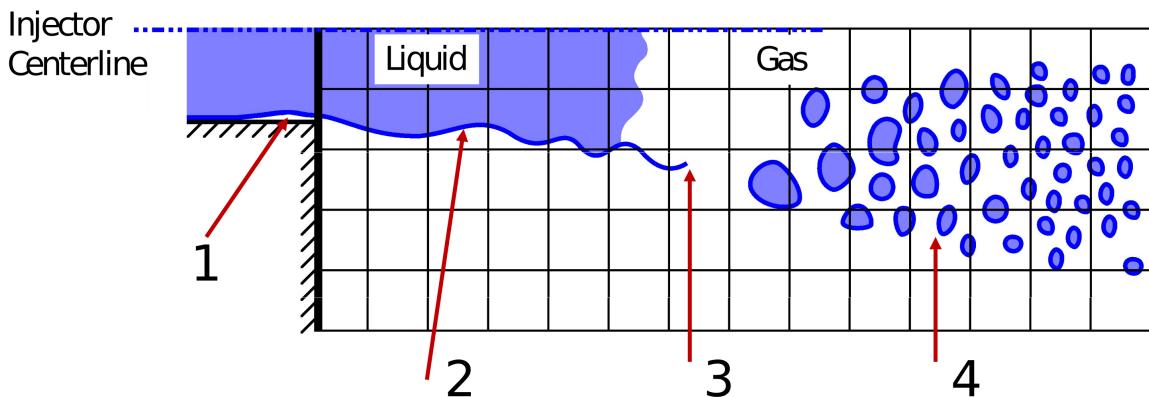


Figure 14.29: Schematic of ELSA modeling. (1) The Eulerian (VOF) simulation of internal nozzle dynamics is conducted. (2) The Eulerian fluid interface is stretching and the cell void fraction is increasing. (3) The surface area density and cell void fraction are reaching the ELSA transition criteria. (4) The injected fuel has been converted to Lagrangian parcels.

The Eulerian liquid fuel is transported according to the methods detailed in [Chapter 15 - Volume of Fluid Modeling](#). You can use any of the advanced modeling techniques available for VOF calculations, such as [wall adhesion](#) and [cavitation modeling](#).

For ELSA, in the Eulerian calculation, CONVERGE solves an additional transport equation for liquid surface area density. CONVERGE uses this surface area density to determine the size of the droplets formed when the fuel is converted to Lagrangian parcels. Transition is controlled by two criteria. The first criterion is the gas volume fraction in a cell, controlled via `elsa.in > stream > elsa_transition_criteria > else_transition_alpha`. If the gas volume fraction in a cell is greater than the value of `elsa_transition_alpha`, this criterion is satisfied. The second criterion is the fluid surface area in a cell, controlled via `elsa.in > stream > elsa_transition_criteria > else_transition_area_ratio`. If the liquid surface area in a cell is greater than the cell minimum value (defined in Equation 14.354) times `elsa_transition_area_ratio`, this criterion is satisfied.

If both transition criteria in a cell are satisfied, the liquid mass in that cell is converted to parcels. CONVERGE treats these parcels in the same way as parcels injected from a nozzle.

The model formulation is taken from [Blanco \(2016\)](#). The liquid surface area density is denoted Σ , and the associated transport equation is written

$$\frac{\partial \tilde{\Sigma}}{\partial t} + \frac{\partial \tilde{u}_j \tilde{\Sigma}}{\partial x_j} - \frac{\partial}{\partial x_j} \left(D_\Sigma \frac{\partial \tilde{\Sigma}}{\partial x_j} \right) - C_\Sigma \tilde{\Sigma} \left(1 - \frac{\tilde{\Sigma}}{\Sigma_{eq}} \right) - S_{\Sigma_{evap}} - S_{\Sigma_{init}} = 0, \quad (14.346)$$

with the component terms defined as

Chapter 14: Discrete Phase Modeling

Eulerian-Lagrangian Spray Atomization

$$\overline{\Sigma}_{eq} = \frac{3\bar{\rho}Y}{\rho_l r_{eq}}, \quad (14.347)$$

$$S_{\Sigma_{evap}} = \frac{2\Sigma}{3\bar{Y}} S_{evap}, \quad (14.348)$$

$$C_\Sigma = \alpha_1 \frac{\tilde{\varepsilon}}{\tilde{k}}, \quad (14.349)$$

$$r_{eq} = \alpha_2 \frac{\sigma^{3/5}}{\varepsilon^{2/5}} \frac{(\bar{\rho}\tilde{Y})^{2/15}}{\rho_l^{11/15}}, \quad (14.350)$$

and

$$D_\Sigma = \frac{V_T}{Sc_\Sigma}, \quad (14.351)$$

where D_Σ is the diffusion coefficient, Y is the liquid volume fraction, $S_{\Sigma_{evap}}$ is the evaporation source term, $S_{\Sigma_{ini}}$ is the initialization value, r_{eq} is the equilibrium droplet size, and α_1 and α_2 are tunable parameters. After calculating this transport equation, the surface area density in each cell is required to be at least as large as the minimum surface area that can contain the liquid volume in that cell. When the fuel is converted to parcels, the droplets are formed according to

$$\Sigma = \frac{n\pi D^2}{4V}, \quad (14.352)$$

and

Chapter 14: Discrete Phase Modeling

Eulerian-Lagrangian Spray Atomization

$$\frac{m_l}{\rho_l} = n \frac{\pi}{6} D^3, \quad (14.353)$$

where n is the number of droplets and D is the droplet diameter. The model compares the liquid surface area density to a nominal minimum fluid surface area density Σ_{min} , defined as

$$\Sigma_{min} = \sqrt{\alpha(1-\alpha)} V^{-\frac{1}{3}}, \quad (14.354)$$

where V is the volume of the cell and α is the void fraction in the cell. The initialization value $S_{\Sigma_{init}}$ is calculated from the minimum fluid surface area in the cell:

$$S_{\Sigma_{init}} = \frac{\Sigma_{min} - \Sigma}{\Delta t} pos(\Sigma_{min} - \Sigma), \quad (14.355)$$

where

$$pos(\psi) = \begin{cases} 1 & \text{if } \Psi > 0 \\ 0 & \text{if } \Psi \leq 0. \end{cases} \quad (14.356)$$

Most of the advanced physical models for liquid parcels can be used with ELSA. However, you cannot use the [KH-RT breakup model](#) or the [LISA breakup model](#) for liquid parcels in an ELSA simulation. Both of these models incorporate a mechanism switching behavior based on a breakup length, which ELSA models directly.

To activate ELSA, set `inputs.in` > `feature_control` > `parcel_control` > `liquid_parcel` = 1 and `feature_control` > `vof_flag` = 1 and include an `elsa.in` file in your case setup. The `elsa.in` file contains ELSA model parameters and a list of ELSA injectors and nozzles. For each ELSA nozzle, you must specify a passive for the injected liquid (`elsa > stream > elsa_injectors > injectors > injector > nozzles > nozzle > elsa_passive_liquid`) and the region ID(s) of the region(s) in which the Eulerian-to-Lagrangian transition occurs (`elsa > stream > elsa_injectors > injectors > injector > nozzles > nozzle > elsa_transition_region_ids`). In `initialize.in`, the ELSA passive must be initialized to 1 in the region containing the center of the nozzle (as defined in `parcel_introduction.in` > `injections` > `injection` > `nozzles` > `nozzle` > `position`) and to 0 in the ELSA transition region(s). This setup requires that the center of the nozzle is located outside of the ELSA transition region(s), as shown below in Figure 14.30.

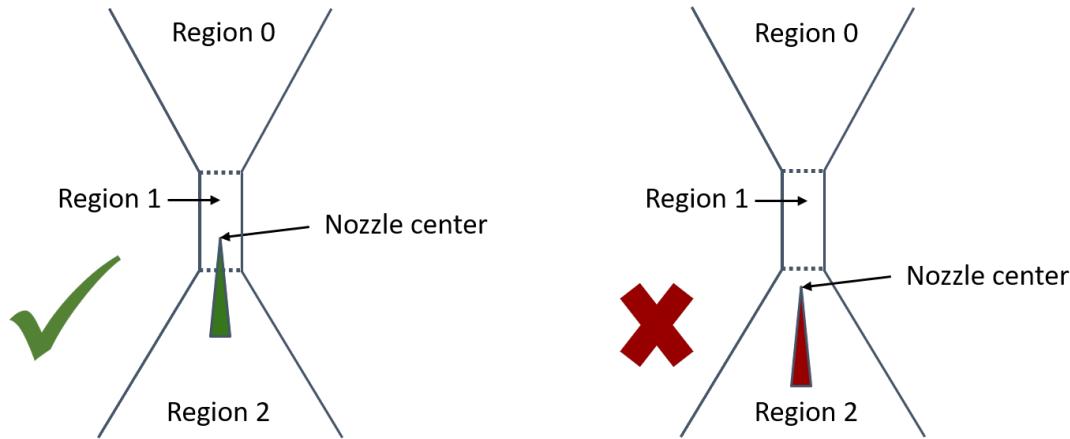


Figure 14.30: Examples of correct and incorrect ELSA nozzle setup. The correct setup, on the left, places the center of the nozzle outside of the ELSA transition region (Region 2). The ELSA passive is initialized to 1 in Region 1 and to 0 in Region 2. The incorrect setup, on the right, places the center of the nozzle inside the ELSA transition region. The ELSA passive cannot be properly initialized in this configuration.

Bidirectional VOF-DPM-VOF

CONVERGE offers an extension of the ELSA model that also allows liquid mass to be converted from Lagrangian parcels to an Eulerian VOF representation (e.g., a spray impinging on a pool of liquid). A parcel within a cell is a candidate for conversion if the local void fraction is less than a specified value and if the parcel is moving toward the liquid interface. Conservation of mass, momentum, energy, and composition during conversion is governed by the same relations that control the standard ELSA model.

CONVERGE limits parcel conversion so that the Eulerian density in any cell cannot increase by more than 5% during each time-step. CONVERGE iterates on the internal parcel ID number, converting candidate parcels until the 5% limit is reached. Parcels that otherwise meet the conversion criteria will be available for conversion during successive time-steps.

To activate this feature, set `elsa.in > stream > dpm_to_vof_transition_criteria > dpm_to_vof_transition_alpha` to a non-zero value. CONVERGE will use this value as the maximum local void fraction to control Lagrangian-to-Eulerian conversion.

Chapter



15

Volume of Fluid Modeling

15 Volume of Fluid Modeling

CONVERGE can simulate multi-phase flows with the volume of fluid (VOF) method. The VOF method locates and tracks the free surface in a liquid-gas flow or the interface in a liquid-liquid flow. It is an Eulerian method that reconstructs the interface location and orientation within each fluid cell. CONVERGE incorporates several models for interface reconstruction, which are suitable for a range of multi-phase flows. Coupled with [Adaptive Mesh Refinement](#), CONVERGE tracks these interfaces accurately and efficiently.

The VOF method can be used with compressible or incompressible fluids. In low-pressure applications, liquids are incompressible and gases are compressible. However, there are some high-pressure applications for which the compressibility of engineering liquids is important.

The two fluids share a joint momentum and energy equation. They are coupled through the interface. As the name implies, the VOF method tracks the volume of fluid within each cell. The volume of fluid is represented by the void fraction α , which is the fraction of the cell's volume that does not contain fluid:

$$\begin{aligned}\alpha = 0 & \quad (\text{the cell contains only liquid}) \\ 0 < \alpha < 1 & \quad (\text{the cell contains both liquid and gas}) \\ \alpha = 1 & \quad (\text{the cell contains only gas}).\end{aligned}\tag{15.1}$$

The value of the void fraction represents one of the following cell compositions, as illustrated below.

$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$
$\alpha = 1$	$\alpha = 1$	$\alpha \sim 0.8$	$\alpha \sim 0.3$	$\alpha \sim 0.3$	$\alpha \sim 0.8$	$\alpha = 1$	$\alpha = 1$
$\alpha = 1$	$\alpha = 1$	$\alpha \sim 0.3$	$\alpha = 0$	$\alpha = 0$	$\alpha \sim 0.3$	$\alpha = 1$	$\alpha = 1$
$\alpha = 1$	$\alpha = 1$	$\alpha \sim 0.3$	$\alpha = 0$	$\alpha = 0$	$\alpha \sim 0.3$	$\alpha = 1$	$\alpha = 1$
$\alpha = 1$	$\alpha = 1$	$\alpha \sim 0.8$	$\alpha \sim 0.3$	$\alpha \sim 0.3$	$\alpha \sim 0.8$	$\alpha = 1$	$\alpha = 1$
$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$

Figure 15.1: Void fraction (α) values. The blue circle represents a liquid droplet.

CONVERGE offers two solution methods for a VOF simulation.

The [void fraction solution \(VFS\) method](#) directly transports the void fraction α , in addition to the usual mass, momentum, and energy. This additional constraint allows the VFS method to resolve sharp fluid interfaces. It can also be used for miscible fluids. However, because volume is not generally a conserved quantity, transportation of volume implies that the fluids must be incompressible.

The [individual species solution \(ISS\) method](#) transports species, momentum, and energy, calculating the void fraction or species volume fractions in each cell from the transported species. For the species solver, you can choose to transport species mass fractions or species densities. The species mass fraction solver is suitable only for simulations with at least one compressible fluid. The species density solver can be used for both compressible and incompressible fluids.

The local value of the void fraction does not contain any information about the shape or location of any interface within the cell. For applications that require interface tracking, several [front capturing schemes](#) are available. A front capturing scheme is always required when running the VFS method, while the ISS method can be run with or without a front capturing scheme.

Additional models can be activated to study phenomena such as [surface tension](#), [cavitation](#), and [dissolved gas](#).

Chapter 15: Volume of Fluid Modeling

Void Fraction Solution Method

15.1 Void Fraction Solution Method

The continuity equation for an incompressible two-phase system is the requirement that the vector field be divergence-free:

$$\nabla \cdot u = 0 \quad (15.2)$$

In Equations 15.4 through 15.9 below, α represents the void fraction in a cell. The momentum conservation equation takes the following form:

$$\rho \left[\frac{\partial u}{\partial t} + u \cdot \nabla u \right] = -\nabla p + \nabla \cdot [\mu \nabla u + \mu (\nabla u)^T] + \sigma \kappa \delta_s n \quad (15.3)$$

$$\frac{\partial \alpha}{\partial t} + u \cdot \nabla \alpha = 0 \quad (15.4)$$

$$\rho = \rho_1 \alpha + \rho_2 (1 - \alpha) \quad (15.5)$$

$$\mu = \mu_1 \alpha + \mu_2 (1 - \alpha). \quad (15.6)$$

The void fraction at any given time is calculated as

$$\alpha^{n+1} = \alpha^n - \frac{\Delta t}{V} \int_A \alpha^n u \cdot dA + \frac{\Delta t}{V} \int_V (\nabla \cdot u) \alpha^n dV, \quad (15.7)$$

where

$$\alpha^{n+1} = \alpha^n + \sum \text{fluxes+sources}. \quad (15.8)$$

CONVERGE geometrically calculates the fluxes and sources in Equations 15.7 and 15.8 using either the [PLIC scheme](#) or the [HRIC scheme](#).

Setup for the VFS Method

You can specify a front capturing scheme for VFS. The [HRIC scheme](#) allows CONVERGE to stably iterate with a larger time-step than the [PLIC scheme](#), but HRIC is more expensive and it cannot resolve the fluid-fluid interface as sharply. We recommend using HRIC for simulations in which:

- You are solving the energy and/or turbulence equations.
- Your geometry is complex or complex and/or moving geometry.

Chapter 15: Volume of Fluid Modeling

Void Fraction Solution Method

To run a simulation with the VFS method, complete the following steps:

- Set `inputs.in > solver_control > steady_solver = 0`.
- Set `inputs.in > property_control > gas_compressible_flag = 0` and `property_control > liquid_compressible_flag = 0`.
- Set `inputs.in > solver_control > species_solver = 0`.
- Set `inputs.in > feature_control > vof_flag = 1`.
- Set `vof.in > front_capture_model = PLIC or HRIC`.
- Set `solver.in > Numerical_Schemes > conserve = 0`.
- If `vof.in > front_capture_model = HRIC`, set `solver.in > transport > species > itmax` to the maximum number of iterations used to solve the transport equation for α . If the value of `itmax` is smaller than 25, CONVERGE uses a default value of 25.
- Set the VFS convergence tolerance values. The values presented in Table 15.1 represent a good balance between simulation speed and solution accuracy.

Table 15.1: Recommended VFS convergence tolerance values in `solver.in`.

Quantity	Parameter	Value
PISO/SIMPLE tolerance	<code>PISO > piso_tol</code> or <code>SIMPLE > simple_tol</code>	1.0e-3
PISO/SIMPLE tolerance scale	<code>PISO > tol_scale</code> or <code>SIMPLE > tol_scale</code>	100
Momentum tolerance	<code>Transport > mom > tol</code>	1.0e-5
Energy tolerance	<code>Transport > energy > tol</code>	1.0e-4
Pressure tolerance	<code>Transport > pres > tol</code>	1.0e-8

- Set [Adaptive Mesh Refinement](#) parameters to automatically refine the grid in order to more accurately capture the VOF interface:
 - Set `amr.in > amr_groups > amr_group > amr_void > active = 1`,
 - Set `amr.in > amr_groups > amr_group > amr_void > embed_scale` to the integer maximum number of levels you want the grid to be refined in the fluid-fluid interface area,
 - Set `amr.in > amr_groups > amr_group > amr_void > sgs_embed` to be the sub-grid value of α that results in grid refinement,
 - Set `amr.in > amr_groups > amr_group > amr_void > temporal_type` to PERMANENT, CYCLIC (VOF AMR repeats according to the specified period), or SEQUENTIAL (VOF AMR does not repeat), depending on your simulation, and
 - Set `amr.in > amr_groups > amr_group > amr_void > start_time` and `amr_groups > amr_group > amr_void > end_time` for VOF AMR (in seconds if `inputs.in > simulation_control > crank_flag = 0` or in crank angle degrees if `crank_flag` is non-zero).

The fluid properties specified in `liquid.dat` apply to the $\alpha = 0$ fluid. The properties specified in `gas.dat` apply to the $\alpha = 1$ fluid, even though this fluid must be incompressible.

Chapter 15: Volume of Fluid Modeling

Void Fraction Solution Method

You can initialize the void fraction α via a user-defined function (UDF). Consult the CONVERGE 3.1 UDF Manual for more information about initialization UDFs.

15.2 Individual Species Solution Method

In the individual species solution (ISS) method, CONVERGE does not transport the void fraction directly as in Equation 15.4. Instead, CONVERGE transports species and then calculates the void fraction or species volume fractions.

The general form of the transport equation is given by

$$\frac{\partial(Y_m\rho)}{\partial t} + \frac{\partial(Y_m\rho u_i)}{\partial x_i} = \frac{\partial}{\partial x_i}\left(\rho D \frac{\partial Y_m}{\partial x_i}\right) + s_m, \quad (15.9)$$

where Y_m is the mass fraction of species m , ρ is the mixture density, u_i is the mixture velocity in direction i , D is the molecular diffusion coefficient, and s_m is a source term for species m .

For VOF simulations, you can solve Equation 15.9 either by transporting the species mass fraction or by transporting the species density. If you transport the species mass fraction ([*inputs.in* > solver_control > species_solver = 1](#)), CONVERGE first solves the [mass transport equation](#) for the mixture density ρ and then solves Equation 15.9 for the mass fraction Y_m . The gas and liquid mass fractions are computed from

$$m_g = \sum_{i=1}^{n_g} Y_i \quad (15.10)$$
$$m_l = 1 - m_g,$$

where m_g is the total gas mass fraction, m_l is the total liquid mass fraction, and n_g is the number of gas species. The void fraction is then obtained from

$$\alpha = \frac{m_g}{\rho_g} \left(\frac{m_g}{\rho_g} + \frac{m_l}{\rho_l} \right)^{-1}. \quad (15.11)$$

If you transport the species density ([*inputs.in* > solver_control > species_solver = 2](#)), CONVERGE first solves Equation 15.9 for the transported species density ρ_m^{tr} , defined as

$$\rho_m^{tr} = Y_m \rho, \quad (15.12)$$

Chapter 15: Volume of Fluid Modeling

Individual Species Solution Method

ignoring the effects of species diffusion (*i.e.*, $D = 0$). Then, CONVERGE calculates the mixture density from

$$\rho = \sum_m \rho_m^{tr} \quad (15.13)$$

and the species mass fractions from $Y_m = \rho_m^{tr}/\rho$. The volume fraction for each species is given by

$$\alpha_m = \left(\frac{Y_m}{\rho_m} \right) \rho, \quad (15.14)$$

where ρ_m is the physical density of species m as determined by the physical properties.

Note that this quantity is not identical to the transported species density ρ_m^{tr} defined in Equation 15.12.

When running the ISS method for compressible fluids (*i.e.*, any species in the simulation is compressible), you can use either the species mass fraction solver or the species density solver (*species_solver* = 1 or 2). When running the ISS method for incompressible fluids (*i.e.*, all species are incompressible), you must use the the species density solver (*species_solver* = 2). Additionally, you must use the species density solver (*species_solver* = 2) for any simulation with more than one liquid species.

Setup for the ISS Method

You can run the ISS method with or without a front capturing scheme. Activate a front capturing scheme in [*vof.in*](#) > *front_capture_model* if you need to track the location of the fluid-fluid interface. For the species mass fraction solver ([*inputs.in*](#) > *solver_control* > *species_solver* = 1), only the [HRIC scheme](#) is available. For the species density solver ([*inputs.in*](#) > *solver_control* > *species_solver* = 2), the [HRIC scheme](#) and [FCT scheme](#) are available. If your application requires a sharp interface, we recommend using the species density solver with HRIC.

In [*solver.in*](#), we recommend setting the convergence tolerance values listed below in Table 15.2.

Table 15.2: ISS convergence tolerance values in [*solver.in*](#).

Quantity	Parameter	Value
PISO/SIMPLE tolerance	<i>PISO</i> > <i>piso_tol</i> or <i>SIMPLE</i> > <i>simple_tol</i>	1.0e-3
PISO/SIMPLE tolerance scale	<i>PISO</i> > <i>tol_scale</i> or <i>SIMPLE</i> > <i>tol_scale</i>	20 for compressible cases, 100 for incompressible cases

Chapter 15: Volume of Fluid Modeling

Individual Species Solution Method

Quantity	Parameter	Value
Momentum tolerance	<i>Transport > mom > tol</i>	1.0e-5
Energy tolerance	<i>Transport > energy > tol</i>	1.0e-4
Pressure tolerance	<i>Transport > pres > tol</i>	1.0e-8
Density tolerance	<i>Transport > density > tol</i>	1.0e-4
Species tolerance	<i>Transport > species > tol</i>	1.0e-4

Species Mass Fraction Solver

For the species mass fraction solver, complete the following steps:

- Set *inputs.in > property_control > gas_compressible_flag = 1* and *property_control > liquid_compressible_flag = 0 or 1*.
- Set *inputs.in > solver_control > species_solver = 1*.
- Set *inputs.in > feature_control > vof_flag = 1*.
- Set *vof.in > front_capture_model = NO_FRONT_CAPTURE or HRIC*.

When *vof.in > front_capture_model = HRIC*, we recommend the solver settings listed below:

- Set *inputs.in > temporal_control > max_cfl_u = 0.5*.
- Set *solver.in > Numerical_Schemes > fv_upwind_factor > mom = 0.5* and *Numerical_Schemes > fv_upwind_factor > global = 0.5*. If the simulation recovers frequently, set these factors to 1.

Species Density Solver

For the species density solver, complete the following steps:

- Set *inputs.in > property_control > gas_compressible_flag = 0 or 1* and *property_control > liquid_compressible_flag = 0 or 1*.
- Set *inputs.in > solver_control > species_solver = 2*.
- Set *inputs.in > feature_control > vof_flag = 1*.
- Set *vof.in > front_capture_model = NO_FRONT_CAPTURE, HRIC, or FCT*.

15.3 Front Capturing Schemes

CONVERGE offers three front capturing schemes for tracking the fluid-fluid interface in a VOF simulation. You can specify a front capturing scheme in *vof.in > front_capture_model*.

- The [piecewise linear interface calculation \(PLIC\) scheme](#) is available only for the [VFS method](#) (*inputs.in > solver_control > species_solver = 0*).
- The [high resolution interface capturing \(HRIC\) scheme](#) is available for all methods (*species_solver = 0, 1, and 2*).
- The [flux-corrected transport \(FCT\) scheme](#) is available only for the [ISS method](#) with the species density solver (*species_solver = 2*).

Piecewise Linear Interface Calculation (PLIC) Scheme

Chapter 15: Volume of Fluid Modeling

Front Capturing Schemes

The piecewise linear interface calculation (PLIC) scheme ([Aulisa et al., 2007](#); [Gueyffier et al., 1999](#); [Rider and Kothe, 1998](#); [Scardovelli and Zaleski, 1999](#); and [Tryggvason and Scardovelli, 2011](#)) constructs the interface separating the fluids geometrically, using a planar shape in each cell (see Figure 15.2 below). This approach maintains a sharper interface than the HRIC scheme.

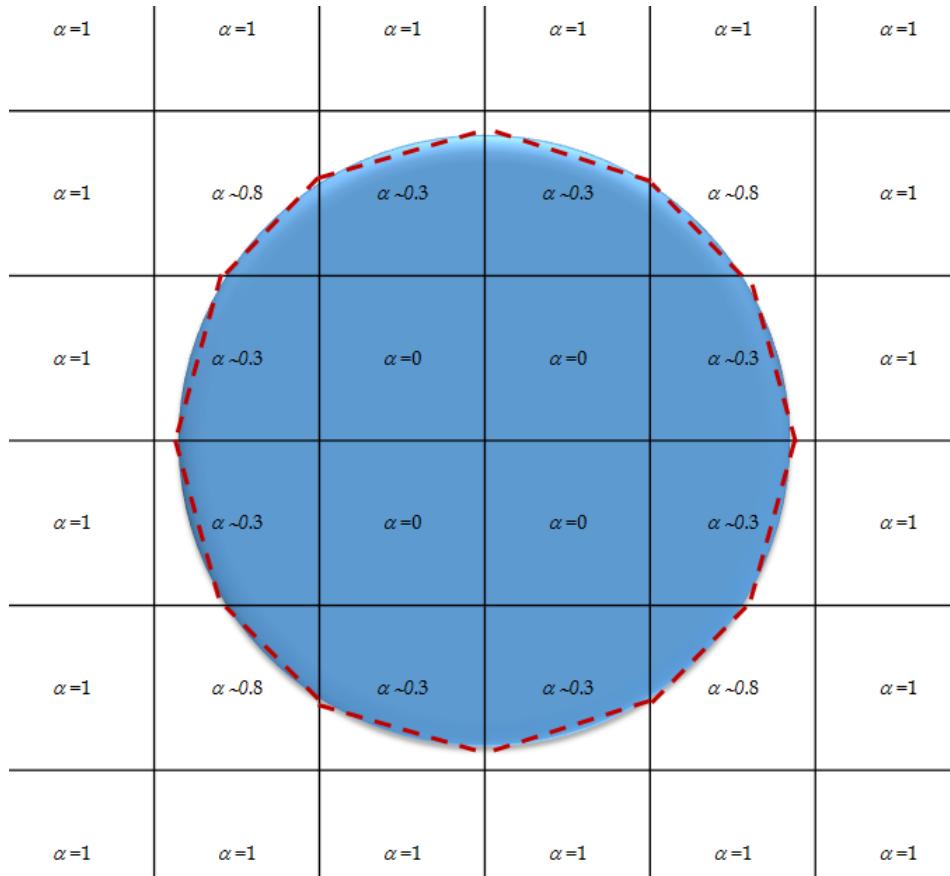


Figure 15.2: The red dotted lines represent the planar shapes that the PLIC method uses to construct the fluid-fluid interface geometrically.

In CONVERGE, the PLIC method follows four steps to geometrically calculate the fluxes and source terms in the void fraction transport equation:

- Estimate the normal orientation, $m = \nabla \alpha$ (see Figure 15.3 below),
- Construct the surface with the plane equation: $m_x x + m_y y + m_z z + d = 0$, where the normals in the x , y , and z directions are m_x , m_y , and m_z
- Determine d uniquely from the void fraction, α , and the interface normal m , and
- Translate the interface according to the local fluid velocity (estimate the fluxes).

Chapter 15: Volume of Fluid Modeling

Front Capturing Schemes

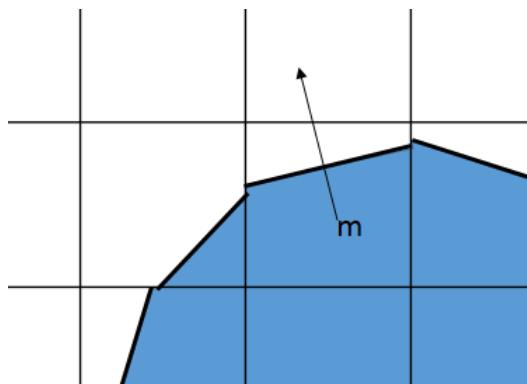


Figure 15.3: Planar shapes used to estimate the normal vectors for the fluid-fluid interface in each cell.

High Resolution Interface Capturing (HRIC) Scheme

Capturing the fluid interface position requires careful discretization of convective terms. Commonly used schemes such as upwinding and central differencing schemes introduce effects of artificial diffusion or dispersion, respectively. Other higher-order schemes cause local oscillations of the void fraction. To avoid these artificial effects, CONVERGE incorporates a high resolution interface capturing (HRIC) scheme ([Waclawczyk and Kornowicz, 2006](#)).

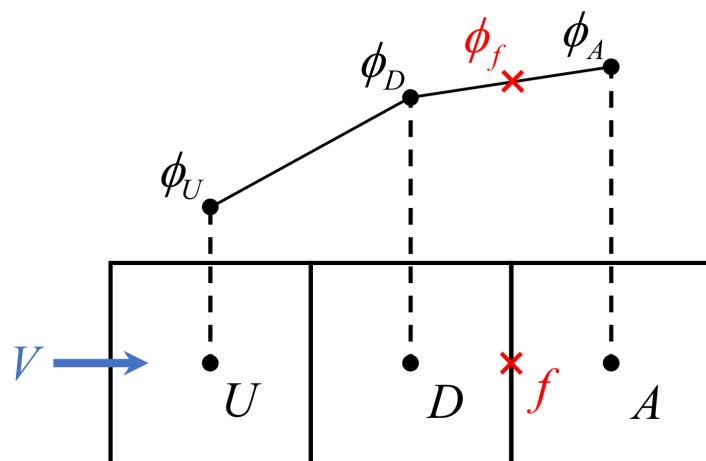


Figure 15.4: A schematic showing the convective boundedness criterion at face f . U represents the upwind cell, D the donor cell, and A the acceptor cell.

The HRIC method is based on the convective boundedness criterion, which states that the variable distribution between the centers of the neighborhood control volumes (for example, D and A in Figure 15.3 above) should remain smooth when $\phi_D \leq \phi_f \leq \phi_A$. (Here, the generic variable ϕ corresponds to the void fraction.) Using this constraint, as well as information

Chapter 15: Volume of Fluid Modeling

Front Capturing Schemes

about the value of the variable in the upwind control volume, ϕ_U , we define the normalized variables

$$\tilde{\phi}_f = \frac{\phi_f - \phi_U}{\phi_A - \phi_U} \quad (15.15)$$

and

$$\tilde{\phi}_D = \frac{\phi_D - \phi_U}{\phi_A - \phi_U}. \quad (15.16)$$

When using the equations above, the value of the void fraction at the control volume, ϕ_f , takes the form

$$\phi_f = (1 - \tilde{\beta})\phi_D + \tilde{\beta}\phi_A, \quad (15.17)$$

where

$$\tilde{\beta} = \frac{\tilde{\phi}_f - \tilde{\phi}_D}{1 - \tilde{\phi}_D}. \quad (15.18)$$

There are three steps in the application of the HRIC scheme. First, the normalized cell face value is estimated from a scheme that continuously connects upwind and downwind schemes:

$$\tilde{\phi}_f = \begin{cases} \tilde{\phi}_D & \text{if } \tilde{\phi}_D < 0, \tilde{\phi}_D > 1 \quad (\text{first-order upwind}), \\ 2\tilde{\phi}_D & \text{if } 0 \leq \tilde{\phi}_D \leq 0.5 \quad (\text{second-order upwind}), \\ 1 & \text{if } 0.5 \leq \tilde{\phi}_D \leq 1 \quad (\text{first-order downwind}). \end{cases} \quad (15.19)$$

Second, since a downwind differencing scheme can cause unphysical alignment of the interface with the mesh, another scheme must be used to satisfy the convective boundedness criterion. For this process, CONVERGE uses an upwind differencing scheme. The blending factor, γ_f , connected with the angle θ_f (shown below in Figure 15.5), yields smooth transitions between schemes:

$$\tilde{\phi}_f^* = \gamma_f \tilde{\phi}_f + (1 - \gamma_f) \tilde{\phi}_D, \quad (15.20)$$

Chapter 15: Volume of Fluid Modeling

Front Capturing Schemes

where

$$\gamma_f = \sqrt{\cos(\theta_f)} \quad (15.21)$$

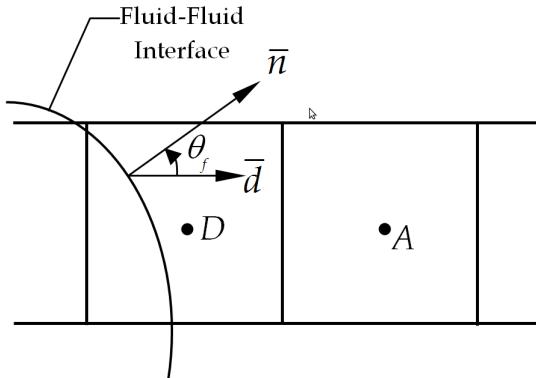


Figure 15.5: A schematic showing the definition of the angle θ_f . θ_f is defined as the angle between the vector normal to the fluid-fluid interface in a cell and the direction vector from the center of fluid-fluid interface cell, D , to the center of the neighboring cell, A .

Dynamic blending of the upwind and downwind differencing schemes accounts for local distribution of the void fraction. When the CFL condition is not satisfied, the dynamic nature of this scheme can cause stability problems. Therefore, $\tilde{\phi}_f^*$ is corrected with respect to the local CFL number. This correction also forces continuous switching between schemes in the time domain:

$$\tilde{\phi}_f^{**} = \begin{cases} \tilde{\phi}_f^* & \text{if } CFL < 0.3 \\ \tilde{\phi}_D & \text{if } CFL > 0.7 \\ \tilde{\phi}_D + (\tilde{\phi}_f^* - \tilde{\phi}_D) \frac{0.7 - CFL}{0.7 - 0.3} & \text{if } 0.3 \leq CFL \leq 0.7 \end{cases} \quad (15.22)$$

When using this scheme in multiple dimensions, the local Courant number, C_f , is replaced by its cell definition.

Flux-Corrected Transport (FCT) Scheme

Flux-corrected transport (FCT) schemes limit both diffusive and dispersive numerical errors by constructing a net convective flux as a weighted average of the flux computed by a lower-order scheme and the flux computed by a higher-order scheme. In the context of a VOF simulation, CONVERGE uses FCT when solving the species density equations.

Chapter 15: Volume of Fluid Modeling

Front Capturing Schemes

Figure 15.6 shows a schematic of the FCT implementation in CONVERGE. At the n -th PISO/SIMPLE iteration, CONVERGE first calculates the lower-order fluxes ϕ^L and solves the species density equations according to

$$\frac{(\rho_m^{tr})^{TD}}{\Delta t} - \frac{(\rho_m^{tr})^{n-1}}{\Delta t} + \sum_f \phi_f^L (\rho_m^{tr})^{n-1} = s_m, \quad (15.23)$$

where the superscript TD denotes the transported/diffused solution and the sum is over the cell faces f . Then, CONVERGE calculates the higher-order fluxes ϕ^H and constructs the anti-diffusive fluxes, defined as

$$\phi^{AD} = \phi^H - \phi^L. \quad (15.24)$$

The anti-diffusive fluxes are corrected according to

$$\phi_C^{AD} = C\phi^{AD} = C(\phi^H - \phi^L), \quad (15.25)$$

where CONVERGE uses the method of [Zalesak \(1979\)](#) to compute the correction factor C . Finally, CONVERGE applies the corrected anti-diffusive fluxes to the transported/diffused solution to obtain an updated solution for iteration n , which satisfies

$$\frac{(\rho_m^{tr})^n}{\Delta t} - \frac{(\rho_m^{tr})^{TD}}{\Delta t} + \sum_f \phi_{C,f}^{AD} (\rho_m^{tr})^{TD} = s_m. \quad (15.26)$$

The FCT implementation in CONVERGE computes ϕ^L with a first-order upwind scheme and computes ϕ^H with a total variation diminishing (TVD) scheme.

Chapter 15: Volume of Fluid Modeling

Front Capturing Schemes
.....

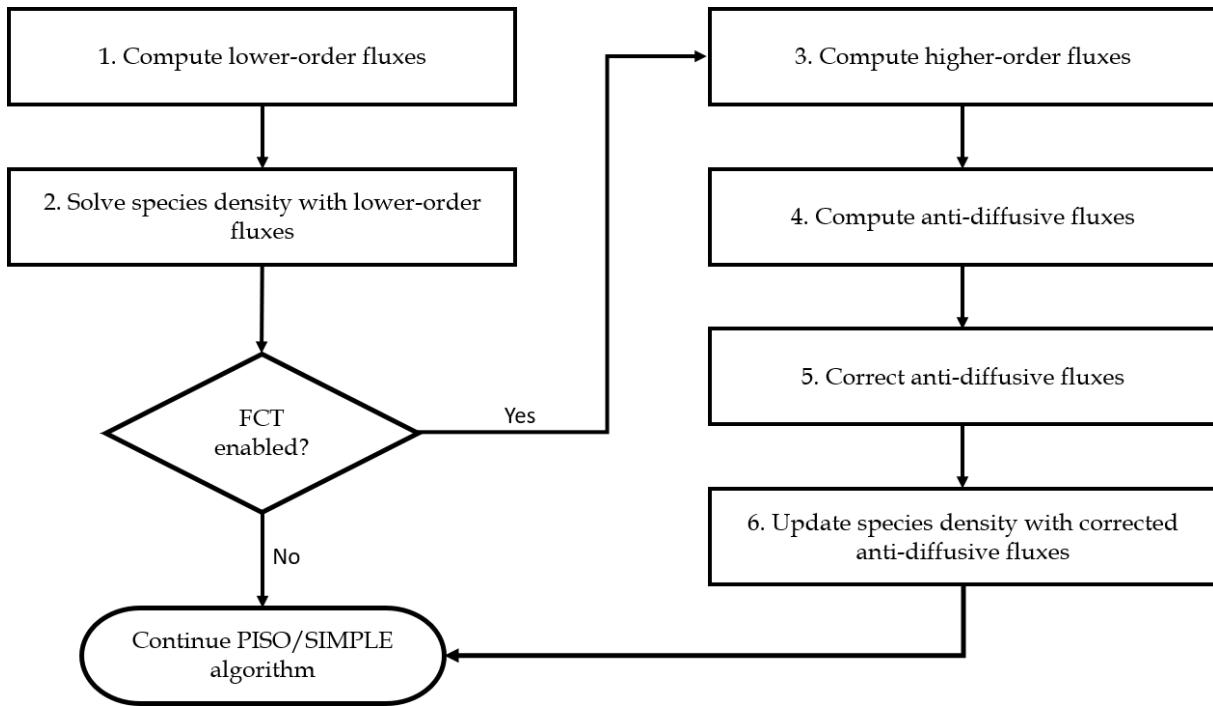


Figure 15.6: FCT implementation in CONVERGE.

15.4 Mixture Model

CONVERGE includes a multi-phase mixture model ([Manninen et al., 1996](#)) in which different species in a mixture can move at different velocities due to gravity or inertial effects. This model can be used to simulate phenomena such as bubbly, slurry, or droplet- or particle-laden flows, pneumatic transport, and mixture separation in a gravitational or rotational field. The mixture model is available for VOF simulations that use the [individual species solution method](#) with the species density solver ([inputs.in > solver_control > species_solver = 2](#)).

When the mixture model is active, CONVERGE solves for the species densities, mixture momentum, and mixture internal energy, incorporating the effects of differing species velocities via source terms in the transport equations. Each species is assumed to have a drift velocity relative to the mixture, defined as

$$u_{m,i}^{dr} = u_{m,i} - u_i, \quad (15.27)$$

where $u_{m,i}$ is the velocity of species m and u_i is the velocity of the mixture, which satisfies

$$u_i = \sum_m Y_m u_{m,i}. \quad (15.28)$$

The source term for the [species density equation](#) is given by

Chapter 15: Volume of Fluid Modeling

Mixture Model

$$-\frac{\partial(Y_m \rho u_{m,i}^{dr})}{\partial x_i}, \quad (15.29)$$

the source term for the [mixture momentum equation](#) is given by

$$-\frac{\partial \left(\sum_m Y_m \rho u_{m,i}^{dr} u_{m,j}^{dr} \right)}{\partial x_j}, \quad (15.30)$$

and the source terms for the [mixture internal energy equation](#) are given by

$$-\frac{\partial \left(\rho \sum_m Y_m e u_{m,i}^{dr} \right)}{\partial x_i} - P \frac{\partial \left(\sum_m \alpha_m u_{m,i}^{dr} \right)}{\partial x_i}. \quad (15.31)$$

The expressions in Equations 15.29-15.31 require an appropriate model for the drift velocity $u_{m,i}^{dr}$. Note that Equation 15.27 can be rewritten as

$$u_{m,i}^{dr} = u_{mq,i} - \sum_n Y_n u_{nq,i}, \quad (15.32)$$

where

$$u_{mq,i} = u_{m,i} - u_{q,i} \quad (15.33)$$

is the slip velocity of species m with respect to the reference species q . CONVERGE estimates the slip velocity according to

$$u_{mq,i} = \frac{\tau_m (\rho_m - \rho)}{f_{drag} \rho_m} a_i, \quad (15.34)$$

where ρ_m is the density of species m as defined in Equation 15.14. The expression in Equation 15.34 is derived from the force balance equation for particles (or bubbles or droplets) of species m that are dispersed in the mixture ([Manninen et al., 1996](#)). The relaxation time-scale τ_m is given by

Chapter 15: Volume of Fluid Modeling

Mixture Model

$$\tau_m = \frac{\rho_m d^2}{18\mu_q}, \quad (15.35)$$

where d is the particle diameter and μ_q is the viscosity of the reference species, and the drag coefficient is calculated from

$$f_{drag} = \begin{cases} 1 + 0.15 \text{Re}^{0.687} & \text{for } \text{Re} \leq 1000 \\ 0.0183 \text{Re} & \text{for } \text{Re} > 1000 \end{cases}. \quad (15.36)$$

The length scale used to compute the Reynolds number Re in Equation 15.36 is the particle diameter d . The acceleration a_i in Equation 15.34 is defined as

$$a_i = g_i - u_j \frac{\partial u_i}{\partial x_j} - \frac{\partial u_i}{\partial t}, \quad (15.37)$$

where g_i is the i -th component of the gravitational acceleration ([inputs.in > property_control > gravity](#)) and the last two terms represent the inertial acceleration.

To activate the mixture model, set [`vof.in > mixture_model > mixture_model_flag`](#) = 1 or 2 and configure related settings in the `mixture_model` settings block. The inertial acceleration terms are included in Equation 15.37 only if you set [`vof.in > mixture_model > inertial_effect_flag`](#) = 1. You can limit the magnitude of the inertial acceleration in [`vof.in > mixture_model > inertial_limit`](#).

When `mixture_model_flag` = 1, CONVERGE uses the mixture model for every cell in the simulation. When `mixture_model_flag` = 2, CONVERGE uses the long-scale interface model of [Márquez Damián \(2013\)](#) to determine which cells will be solved with the mixture model and which cells will be solved using the VOF method without the mixture model. This can reduce computational cost.

The long-scale interface model assumes that the flow contains both short-scale interfaces, characterized by the particle size d of the dispersed phase, and long-scale interfaces, associated with large local gradients of the species volume fraction α_m . The mixture model provides an accurate treatment of the short-scale interfaces, while the long-scale interfaces can be captured using the VOF method without the mixture model. For each cell, CONVERGE calculates a switching function on the cell faces f , given by

Chapter 15: Volume of Fluid Modeling

Mixture Model

$$\theta_f = \begin{cases} 1 & \text{if } \alpha_{m,f} \leq \varepsilon \text{ or } \alpha_{m,f} \geq 1 - \varepsilon, \\ 1 & \text{if } \gamma_f > \gamma_0, \\ 0 & \text{if } \gamma_f \leq \gamma_0, \end{cases} \quad (15.38)$$

where

$$\gamma_f = \left| \frac{\partial \alpha_m}{\partial x_i} \right|_f d_{PN,i} \quad (15.39)$$

and $d_{PN,i}$ is the center-to-center vector for the cells neighboring face f . When the volume fraction deviates from 1 or 0 by no more than ε , the cell can be considered to be in a single phase and the mixture model is unnecessary (*i.e.*, $\theta_f = 1$). Similarly, when $\gamma_f > \gamma_0$, the gradient is large enough to constitute a long-scale interface, and $\theta_f = 1$. If any species has $\theta_f = 0$ on all cell faces, CONVERGE uses the mixture model for that cell. Otherwise, CONVERGE uses the VOF method without the mixture model. CONVERGE sets the model constants to $\varepsilon = 0.005$ and $\gamma_0 = 0.33$.

15.5 Surface Tension and Wall Adhesion

Both VOF models available in CONVERGE account for the free-space effects of surface tension, *i.e.* along the interface between the liquid and gas phases. Surface tension effects between liquids and walls are further characterized by adhesion effects. You can describe surface adhesion with a contact angle, described later in this section.

Free-space surface tension is characterized by a source term, $F_{sf,i}$, in the momentum transport equation:

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial \sigma_{ij}}{\partial x_j} + F_{sf,i}. \quad (15.40)$$

To compute the surface tension, CONVERGE calculates the surface curvature from local gradients in the surface normal at the interface:

$$n_i = \frac{\partial \alpha}{\partial x_i} \left(\left| \frac{\partial \alpha}{\partial x_i} \right| \right)^{-1}. \quad (15.41)$$

The curvature, κ , is defined in terms of the divergence of the unit normal, n_i as

Chapter 15: Volume of Fluid Modeling

Surface Tension and Wall Adhesion

$$\kappa = -\frac{\partial n_i}{\partial x_i}. \quad (15.42)$$

In the VOF method, the interface between the liquid and gas phases is not tracked explicitly. Therefore, CONVERGE models the surface tension with a continuum surface force (CSF) model (Brackbill et al., 1992), which alleviates the interface topology constraints. The model interprets surface tension as a continuous, three-dimensional effect across an interface, rather than as a boundary value condition on the interface.

To formulate the volume force, we define a modified void fraction, $\tilde{\alpha}(x)$, that varies smoothly over a thickness h across the interface by convolving the characteristic function, $\alpha(x)$ with a Gaussian filter interpolation function, φ , as

$$\tilde{\alpha}(x) = \frac{1}{h^3} \int_v \alpha(x') \varphi(x' - x) d^3 x', \quad (15.43)$$

where φ has bounded support

$$\varphi(x) = 0 \text{ for } |x| \geq \frac{h}{2}. \quad (15.44)$$

The interpolation function is defined so that the modified $\tilde{\alpha}(x)$ approaches the characteristic function $\alpha(x)$ as the scale length $h \rightarrow 0$:

$$\lim_{h \rightarrow 0} \tilde{\alpha}(x) = \alpha(x). \quad (15.45)$$

CONVERGE computes the external force exerted in the cell by surface tension through a surface tension coefficient, σ .

$$F_{sfi}(x) = \sigma \kappa(x) \frac{\partial \tilde{\alpha}(x)}{\partial x_i}. \quad (15.46)$$

You can specify a wall adhesion angle in conjunction with the surface tension model. Rather than impose this boundary condition at the wall itself, the assumed fluid-wall contact angle adjusts the normal vectors of the cells near the wall. This dynamic boundary condition adjusts the curvature of the surface near the wall. If θ_w is the contact angle at the wall, then the surface normal at the live cell next to the wall is

Chapter 15: Volume of Fluid Modeling

Surface Tension and Wall Adhesion

$$n_i = n_{w\ i} \cos(\theta_w) + t_{w\ i} \sin(\theta_w), \quad (15.47)$$

where $n_{w\ i}$ and $t_{w\ i}$ are the unit vectors normal and tangential to the wall, respectively. The local curvature of the surface is determined by the combination of this contact angle and the normally calculated normal vector one cell removed from the wall. This curvature adjusts the body force term in the surface tension calculation.

15.6 Cavitation Modeling

CONVERGE's cavitation model is based on the flash-boiling hypothesis of [Shields et al., \(2011\)](#), with rapid heat transfer between the liquid and vapor phase.

The vaporization process in cavitation is very similar to that of flash-boiling, with a specific thermodynamic difference. While cavitation represents the vapor formed through a constant temperature system experiencing a drop in pressure, flash-boiling represents the same system, with a lower pressure drop and elevated temperatures. A homogenous relaxation model (HRM) predicts the mass exchange between the liquid and vapor. This model describes the rate at which the liquid-vapor mass interchange will approach its equilibrium value. This interchange is parametrized by the vapor mass fraction (also called the vapor quality or dryness fraction) x ,

$$x = \frac{m_g}{m_g + m_l}. \quad (15.48)$$

[Bilicki and Kestin \(1990\)](#) proposed a simple linearized form for the evolution of the vapor mass fraction,

$$\frac{Dx}{Dt} = \frac{\bar{x} - x}{\theta}. \quad (15.49)$$

In the above equation, x represents the instantaneous mass, \bar{x} represents the equilibrium mass and θ represents the time scale over which x relaxes to \bar{x} . For evaporation, the time scale θ_E can be represented as follows:

$$\theta_E = \theta_0 \alpha^{-0.54} \varphi^{-1.76}. \quad (15.50)$$

For condensation, the time scale θ_C can be represented as

$$\theta_C = F \theta_0 \alpha^{-0.54} \varphi^{-1.76}, \quad (15.51)$$

Chapter 15: Volume of Fluid Modeling

Cavitation Modeling

where F is the time scale factor ([*vof.in*](#) > *cavitation_model* > *condensation_time_factor*) by which the evaporation time scale is factored. A typical value for F is 5000, meaning condensation occurs 5000 times slower than evaporation under similar conditions. The value of the coefficient θ_0 is 3.84e-7 s and the non-dimensional pressure ratio (φ) is given by

$$\varphi = \frac{P_{sat} - P}{P_c - P_{sat}}, \quad (15.52)$$

where P_c is the critical pressure.

CONVERGE calculates the equilibrium vapor mass fraction \bar{x} from conservation of mass and energy. From state 0 to state 1, enthalpy conservation can be written as

$$m_{g,1}h_{g,1} + m_{l,1}h_{l,1} = m_{g,0}h_{g,0} + m_{l,0}h_{l,0}. \quad (15.53)$$

From state 0 to state 1, mass conservation can be written as

$$m_{g,1} + m_{l,1} = m_{g,0} + m_{l,0}, \quad (15.54)$$

written alternately as

$$m_{g,0} = m_{g,1} + m_{l,1} - m_{l,0}. \quad (15.55)$$

Substituting Equation 15.55 into Equation 15.53 and rearranging yields

$$m_{g,1}(h_{g,1} - h_{g,0}) + m_{l,1}(h_{l,1} - h_{g,0}) = m_{l,0}(h_{l,0} - h_{g,0}). \quad (15.56)$$

Because

$$h_{l,1} - h_{g,0} = h_{l,1} - h_{g,0} + (h_{l,0} - h_{l,0}), \quad (15.57)$$

Equation 15.56 can be rewritten as

$$m_{g,1}(h_{g,1} - h_{g,0}) + m_{l,1}(h_{l,1} - h_{l,0}) = (m_{l,1} - m_{l,0})(h_{g,0} - h_{l,0}). \quad (15.58)$$

CONVERGE approximates the enthalpy change using the trapezoidal rule,

$$\Delta h = \int c_p dT = \overline{c_p} \Delta T, \quad (15.59)$$

Chapter 15: Volume of Fluid Modeling

Cavitation Modeling

where

$$\overline{c_p} = \frac{c_p(T_1) + c_p(T_0)}{2} \quad (15.60)$$

and T_1 is the saturation temperature. Thus, CONVERGE approximates Equation 15.58 as

$$m_{g,1} \overline{c_{p,g}} (T_1 - T_0) + m_{l,1} \overline{c_{p,l}} (T_1 - T_0) = (m_{l,1} - m_{l,0}) h_{vap}, \quad (15.61)$$

where

$$h_{vap} = h_{g,0} - h_{l,0}, \quad (15.62)$$

which is evaluated as the latent heat for the saturation temperature T_0 .

Dividing both sides by the cell mass m , Equation 15.61 can be rewritten as

$$(1 - Y_{l,1}) \overline{c_{p,g}} (T_1 - T_0) + Y_{l,1} \overline{c_{p,l}} (T_1 - T_0) = (Y_{l,1} - Y_{l,0}) h_{vap}, \quad (15.63)$$

where

$$Y_{l,1} = \frac{m_{l,1}}{m} \quad (15.64)$$

and

$$Y_{l,0} = \frac{m_{l,0}}{m}. \quad (15.65)$$

Finally, Equation 15.63 is rearranged as

$$Y_{l,1} = \frac{\overline{c_{p,g}} (T_1 - T_0) + Y_{l,0} h_{vap}}{\overline{c_{p,g}} (T_1 - T_0) - \overline{c_{p,l}} (T_1 - T_0) + h_{vap}}. \quad (15.66)$$

To use the cavitation model in CONVERGE, set `inputs.in > vof_flag = 1` and set `vof.in > cavitation_model > cavitation_flag = 1`. Refer to the `vof.in` section of [Chapter 25 - Input and Data Files](#) for more information about the other cavitation-related parameters in `vof.in`.

15.7 VOF-Spray One-Way Coupling

CONVERGE features one-way coupling between VOF and Lagrangian spray modeling. You can initialize a spray simulation with parcel data from a VOF simulation. This option allows you to combine a high fidelity simulation of a fuel injector and nozzle system with a computationally inexpensive parcel-based fuel injector spray simulation.

First, run a VOF simulation. In this simulation, set `vof.in > vof_spray_flag = 1` and set `inputs.in > feature_control > parcel_mode > liquid_parcel = 0`. In `vof_spray.in`, identify injectors and nozzles based on the the region IDs of the regions on either side of each nozzle (see Figures 15.7 and 15.8 below). After this simulation, CONVERGE generates a `vof_spray.out` file for each stream listed in `vof_spray.in`. This file contains parcel position, velocity, turbulence, temperature, and cell size information from the VOF simulation. CONVERGE writes this information for the two layers of cells on either side of a region interface that defines a nozzle. We recommend that you place this region interface slightly within the nozzle (as seen below for Nozzles 3 and 7 in Figure 15.7) to better demarcate between the two regions. This information is written, at the interval specified by `vof_spray.in > vof_spray_injections > stream > twrite_vof_spray`, for all injectors specified in `vof_spray.in > vof_spray_injections > stream > injection_list`. Rename this file `vof_spray.dat` (CONVERGE requires this exact file name).

Chapter 15: Volume of Fluid Modeling

VOF-Spray One-Way Coupling

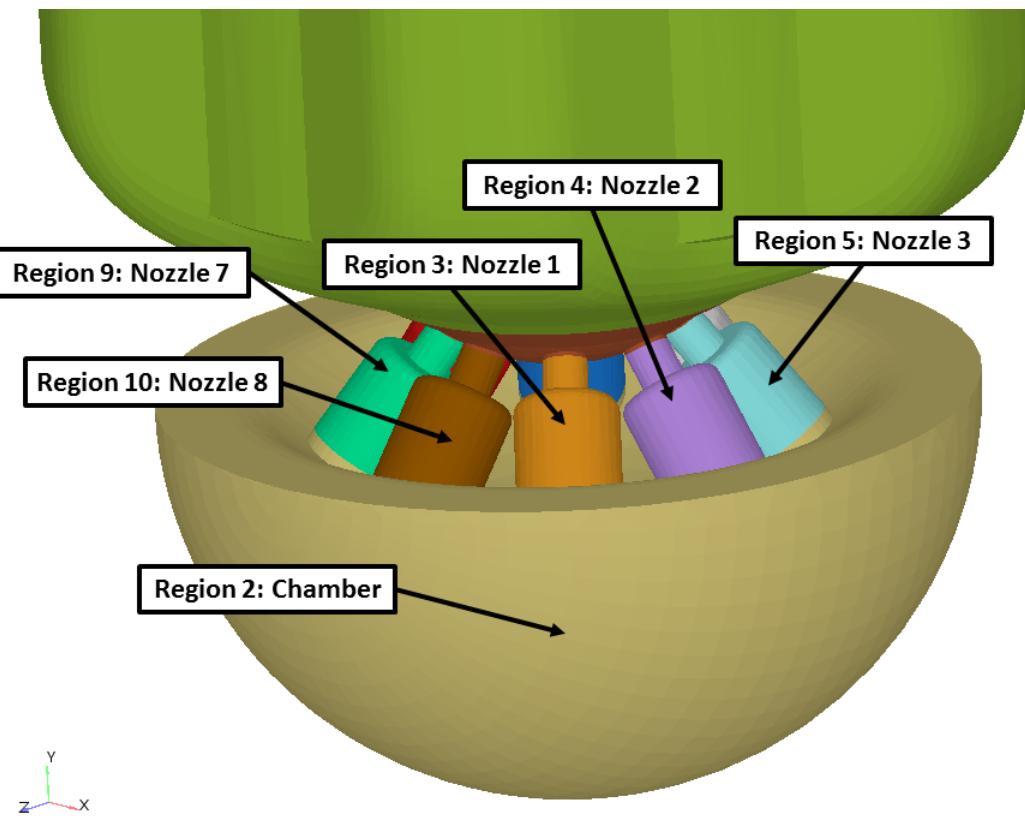


Figure 15.7: Example of nozzle and chamber regions for a simulation with VOF-spray one-way coupling. Note that not all nozzles are labeled.

```
version: 3.1
---
vof_spray_injections:
  - stream:
    stream_id: 0
    twrite_vof_spray: 1e-20
    injection_list:
      - injection:
        vof_spray_injector_name: 1
        nozzles:
          - nozzle:
            name: 1
            region_pairs: [3, 2]
          - nozzle:
            name: 2
            region_pairs: [4, 2]
          - nozzle:
            name: 3
            region_pairs: [5, 2]
          - nozzle:
            name: 4
            region_pairs: [6, 2]
          - nozzle:
            name: 5
            region_pairs: [7, 2]
          - nozzle:
            name: 6
            region_pairs: [8, 2]
```

Chapter 15: Volume of Fluid Modeling

VOF-Spray One-Way Coupling

```
- nozzle:  
  name: 7  
  region_pairs: [9, 2]  
- nozzle:  
  name: 8  
  region_pairs: [10, 2]
```

Figure 15.8: An excerpt of a [*vof_spray.in*](#) file. The region IDs correspond to those of the nozzles and the chamber in Figure 15.7 above.

Next, run a Lagrangian simulation with spray modeling enabled (set [*inputs.in*](#) > *feature_control* > *parcel_mode* > *liquid_parcel* = 1) and VOF modeling disabled (set [*inputs.in*](#) > *feature_control* > *vof_flag* = 0). For this simulation, set [*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *spray_bc_model* = ONEWAY and [*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *vof_oneway_mapping* > *filename* = *vof_spray.dat*. This simulation uses the data and injector settings (position, orientation, and diameter) in [*vof_spray.dat*](#) to create parcels and represent the spray. If you have multiple [*vof_spray.dat*](#) files for multiple streams, create [*stream-specific input directories*](#) and put each file in the appropriate directory.

To ensure that CONVERGE assigns each block of VOF data to the appropriate injector and nozzle in the Lagrangian simulation, the *Injector_NAME* and *Nozzle_NAME* from [*vof_spray.dat*](#) must match the injector name and nozzle name in [*parcel_introduction.in*](#). However, if [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzle_control* > *coord_sys* = POLAR-COPY, the names of individual nozzles are not specified in [*parcel_introduction.in*](#). In this situation, the values of *Nozzle_NAME* in [*vof_spray.dat*](#) must match the nozzle names generated by CONVERGE during the Lagrangian simulation. The latter names have the format *<base name>_<number>*, where *<base name>* is the name specified in [*parcel_introduction.in*](#) for the nozzle to be copied. For example, if *<base name>* = *noz*, the nozzle names are *noz_0*, *noz_1*, *noz_2*, etc. The one exception is if *<base name>* = 0, in which case the nozzle names are 0, 1, 2, etc.

CONVERGE does not verify that corresponding injectors are in the same location. If the position of the injector in the Lagrangian spray simulation does not match the position of the corresponding injector in the VOF simulation, CONVERGE will use the injector position and orientation that appears in [*vof_spray.dat*](#). If this position happens to be outside of the domain in the Lagrangian simulation (and you have not specified [*transformation options*](#)), the simulation will crash.

If the Lagrangian simulation is *crank angle-based* (*i.e.*, if [*inputs.in*](#) > *simulation_control* > *crank_flag* is non-zero), CONVERGE automatically converts the timing information in [*vof_spray.dat*](#), which is always in *seconds* for VOF simulations, to *crank angle degrees*. Also, if the start time of the VOF simulation differs from the start time of the Lagrangian spray simulation, CONVERGE shifts the timing in [*vof_spray.dat*](#) such that it matches the spray start time of the Lagrangian simulation ([*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *start_time*). The spray duration in the Lagrangian simulation ([*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *duration*) must be less than or equal to the duration given in [*vof_spray.dat*](#).

Chapter 15: Volume of Fluid Modeling

VOF-Spray One-Way Coupling

CONVERGE uses the position, velocity, turbulence, and temperature information from [*vof_spray.dat*](#) to initialize the parcels for the spray simulation. For each injector, specify the maximum mass (in kg) that CONVERGE assigns to each liquid parcel ([*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *vof_oneway_mapping* > *mass_per_parcel*) and the volume fraction threshold ([*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *vof_oneway_mapping* > *liq_frac_threshold*). If a cell has a liquid volume fraction above the threshold you enter, CONVERGE injects at least one liquid parcel using the data from [*vof_spray.dat*](#). If a cell has a liquid volume fraction below the threshold, CONVERGE does not inject liquid parcels or fuel vapor. CONVERGE adjusts the number of parcels injected to match the total injected mass from [*vof_spray.dat*](#), so the liquid mass injected at a given time-step in the Lagrangian simulation is equal to the liquid mass at the corresponding time in the VOF simulation. The injection scheme is as follows.

1. At a given time-step, determine the total mass to inject from the corresponding *Total Mass Liquid* in [*vof_spray.dat*](#).
2. For each data point in [*vof_spray.dat*](#), sum the liquid volume fraction (*Liquid VOF* in [*vof_spray.dat*](#)) of data points for which the liquid volume fraction exceeds the threshold ([*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *vof_oneway_mapping* > *liq_frac_threshold*).
3. Divide the total mass (determined in step 1) by the sum of the acceptable liquid volume fractions (determined in step 2). The result is the mass per unit volume fraction.
4. For each data point with a liquid volume fraction above the threshold, multiply the mass per unit volume fraction by the liquid volume fraction of the cell. The result is the mass to inject per cell.
5. Divide the mass to inject per cell by the maximum parcel mass ([*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *vof_oneway_mapping* > *mass_per_parcel*) and add 1:

$$\text{parcels per cell} = \frac{\text{mass to inject per cell}}{\text{vof_spray_mass_per_parcel}} + 1 \quad (15.67)$$

This process ensures that each cell has at least one parcel.

Currently, for cells in which CONVERGE injects multiple parcels, all parcels have the same radius, which is a function of the square root of *Ca* and the *Diameter* in [*vof_spray.dat*](#):

$$\text{parcel radius} = \sqrt{Ca} \cdot 0.5 \cdot \text{Diameter}. \quad (15.68)$$

CONVERGE calculates *Diameter* in [*vof_spray.dat*](#) from the area of the nozzle (assuming a circular nozzle).

If a data point in [*vof_spray.dat*](#) meets the criteria given by the injection scheme described above and CONVERGE does inject a parcel, the parcel location is that given by the corresponding *X*, *Y*, and *Z* data (unless you activate [*transformation options*](#)). To avoid creating a uniform stream of parcels in a given cell, CONVERGE perturbs the parcel position

Chapter 15: Volume of Fluid Modeling

VOF-Spray One-Way Coupling

via a random number within the bounds of a cube described by dx , dy , and dz in [*vof_spray.dat*](#).

Lagrangian Simulation: Transformation Options

If the injector in the Lagrangian simulation has physical properties that differ from the data in [*vof_spray.dat*](#), you can specify a transformation option in [*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *vof_oneway_mapping* > *transformation*.

To overwrite the nozzle diameter given in [*vof_spray.dat*](#), set *transformation* = NOZZLE-DIAMETER. In this case, CONVERGE will use the nozzle diameter given by [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzles* > *nozzle* > *geometry* > *diameter*. This feature is useful for nozzles that are not complete circles (e.g., for modeling half of a nozzle with a SYMMETRY boundary condition). Note that Equation 15.68 gives the parcel radius, but in this case the *Diameter* is given by [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzles* > *nozzle* > *geometry* > *diameter*.

If the injector in the Lagrangian simulation has a different location or orientation than that specified in [*vof_spray.dat*](#), use one of the options listed below.

- If *transformation* = TRANSLATION, CONVERGE translates the data in [*vof_spray.dat*](#). To calculate the translation amount, CONVERGE determines the distance and direction between the location of the injector (given by [*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *position*) and the injector location given as *Injector_Center* in [*vof_spray.dat*](#). Then, CONVERGE applies this translation to each data point in [*vof_spray.dat*](#) to translate the data to the location of the injector in the Lagrangian simulation.
- If *transformation* = ROTATION, CONVERGE rotates the data contained in [*vof_spray.dat*](#). To calculate the rotation amount, CONVERGE determines the angle between the normal vector of the injector exit (calculated based on the injector angle given by [*parcel_introduction.in*](#) > *injections* > *injection* > *injection_control* > *angle_xy_inj* and [*injections*](#) > *injection* > *injection_control* > *angle_xz_inj*) and the normal vector of the injector given as *Injector_Axi_Vec* in [*vof_spray.dat*](#). Then, CONVERGE applies this rotation to each data point in [*vof_spray.dat*](#) to rotate the data to the orientation of the injector in the Lagrangian simulation.
- If *transformation* = TRANSLATION-ROTATION, CONVERGE applies both translation and rotation as described above.
- If *transformation* = INJECTOR-COPY, CONVERGE copies VOF data from the base injector and then applies translation and rotation as described above. The base injector is another injector in [*parcel_introduction.in*](#) for which *transformation* ≠ INJECTOR-COPY. One base injector is required.
- If *transformation* = NONE, CONVERGE will not transform the data in [*vof_spray.dat*](#).
- If you want to specify only a single nozzle and have CONVERGE copy the attributes of this nozzle to the remaining nozzles, set *transformation* = NOZZLE-COPY. For this option, only one nozzle per injector is allowed, and you must also set [*parcel_introduction.in*](#) > *injections* > *injection* > *nozzle_control* > *coord_sys* = POLAR-COPY.

Chapter 15: Volume of Fluid Modeling

VOF-Spray One-Way Coupling

For an injector embedding in [*embedded.in*](#), CONVERGE automatically places the embedding at the injector location (from [*vof_spray.dat*](#), including any specified transformation options).

For parcel temperature, tke, and eps (all specified in [*parcel_introduction.in*](#)), you can supply constant values, use the values directly from [*vof_spray.dat*](#), or scale and/or offset the values from [*vof_spray.dat*](#). CONVERGE uses the values of these turbulence quantities to perturb the initial parcel velocity in the spray simulation. For the gas phase, use [*parcel_introduction.in* > injections > injection > injection_control > init_cell_turb](#) to control the turbulence initialization. If you plan to use the [KH-ACT breakup model](#) in the spray simulation, you must use a [k-epsilon turbulence model](#) in the VOF simulation.

Note that for calculation of vapor penetration length in the Lagrangian spray simulation, CONVERGE uses the cone angle given by [*parcel_introduction.in* > injections > injection > nozzles > nozzle > spray_cone > angle](#).

CONVERGE Studio contains example case setups for VOF and Lagrangian spray simulations (*File > Load example case > Fuel Injectors and Sprays > Spray studies*).

15.8 Dissolved Gas Modeling

Under certain conditions, a gas may dissolve into a liquid such that the two fluids are in solution. Likewise, as the pressure changes, the gas in the solution exits the solution and returns to a free gas phase. In CONVERGE, Henry's Law governs the mass transfer between the free gas phase and the dissolved gas/liquid solution. Equations 15.69 and 15.70 below use N2 as an example gas. Henry's law can be written as

$$\bar{c}_{N2,liq} = \frac{p}{K_H(T)}, \quad (15.69)$$

where \bar{c} is the mole or mass fraction of the gas in solution, p is the pressure, and K_H is the Henry's law constant (in Pu). Henry's law specifies the amount of a gas that should be in solution at the given pressure and temperature. By comparing this equilibrium value with the calculated value of dissolved gas in a computational cell, CONVERGE calculates the appropriate amount of dissolved gas in solution. Equation 15.70 presents the rate equation used to calculate mass transfer between the free gas phase and the dissolved gas phase:

$$\frac{dY_{N2,liq}}{dt} = \frac{\bar{Y}_{N2,liq} - Y_{N2,liq}}{\tau}, \quad (15.70)$$

where $Y_{N2,liq}$ is the mass fraction of dissolved gas in solution (N2 in this example), $\bar{Y}_{N2,liq}$ is the equilibrium mass fraction of dissolved gas in solution, and τ is the time-scale. Note that in the rate equation above, the time-scale τ is constant.

Chapter 15: Volume of Fluid Modeling

Dissolved Gas Modeling

To activate dissolved gas modeling, set `vof.in > dissolved_gas_model > dissolved_gas_flag = 1` and specify the dissolved gas information beneath that parameter, as described in [Chapter 25 - Input and Data Files](#). For each dissolved gas to be modeled, CONVERGE tracks a [passive](#) that represents the dissolved gas in solution with the liquid. You must include these passives in both `vof.in` and `species.in`. CONVERGE treats the dissolved gas as a liquid in solution. Supply the species name of this liquid in solution and the gas phase species that corresponds to the liquid in solution. CONVERGE uses the molecular weight of the corresponding gas for the molecular weight of the liquid.

Chapter



16

Combustion Modeling

16 Combustion Modeling

This chapter describes the combustion models available in CONVERGE.

CONVERGE includes combustion models for both [premixed](#) and [non-premixed](#) combustion. The following table summarizes the combustion models available in CONVERGE and their uses.

Table 16.1: Combustion models in CONVERGE.

Model	Uses
SAGE detailed chemical kinetics solver	Premixed and non-premixed combustion
CEQ chemical equilibrium solver	Premixed and non-premixed combustion
Flamelet Generated Manifold (FGM) model	Premixed and non-premixed combustion
G-Equation model	Premixed combustion
Extended Coherent Flamelet Model (ECFM)	Premixed combustion
3-Zone Extended Coherent Flamelet Model (ECFM3Z)	Non-premixed combustion
Shell+Characteristic Time Combustion (CTC) model	Non-premixed combustion
Representative Interactive Flamelet (RIF) model	Non-premixed combustion

This chapter describes [turbulent and laminar flamespeed calculations](#), the [surface chemistry model](#), the [adaptive zoning acceleration strategy](#), [combustion-related output](#), and [combustion time-step control](#).

CONVERGE includes several chemistry-related utilities, including [zero-dimensional chemistry tools](#), [one-dimensional chemistry tools](#), and the [mechanism reduction tool](#), which are discussed elsewhere in this manual.

16.1 Premixed Combustion Models

Premixed combustion requires that the fuel and oxidizer species be completely mixed before combustion is allowed to take place. This premixing is possible only at sufficiently low temperatures where the chain-breaking mechanism that drives the reaction chain in hydrogen and hydrocarbon oxidation is unable to compete with the effect of three-body chain-breaking reactions.

Under such low-temperature conditions, combustion reactions are considered frozen. The frozen state is metastable because a sufficiently strong heat source, a spark for example, can raise the temperature above the threshold and initiate combustion. Once the fuel and oxidizer species have been homogeneously mixed and a heat source is supplied, a flame front can propagate through the mixture.

Typically, the gas behind the flame front rapidly approaches the burned gas state close to chemical equilibrium. The mixture in front of the flame typically remains in the unburned state. Therefore, the combustion system contains two stable states: the unburned (index u) and the burned gas state (index b) as shown in the figure below. In premixed combustion, both states concurrently exist in the system. They are spatially separated by the flame front, where the transition from one to the other takes place.

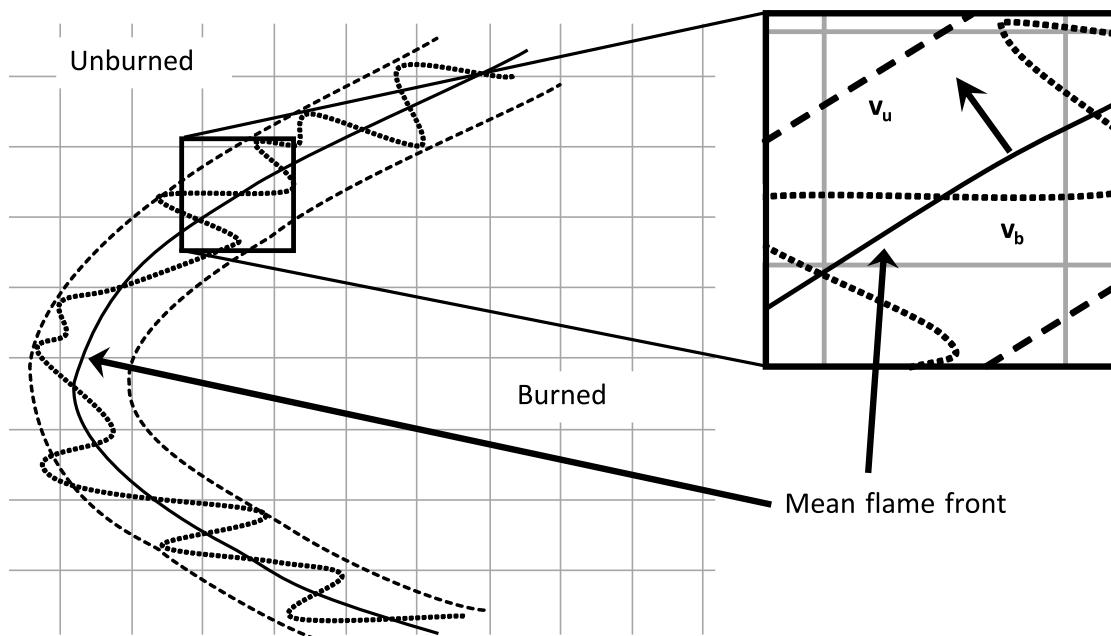


Figure 16.1: A schematic illustrating the mean flame front and the turbulent flame brush superimposed on the computational mesh. Figure based on [Tan and Reitz \(2006\)](#).

16.2 Non-Premixed Combustion Models

In diesel engines, the fuel-air mixture undergoes chemical reactions that lead to ignition as shown below in Figure 16.2. Ignition may not occur until several crank angles after the start of injection (SOI). Once ignition occurs, the fuel that has been injected and mixed with air undergoes the premixed combustion phase. When the premixed fuel-air mixture has been consumed, mixing-controlled combustion occurs. As the name suggests, burning in this combustion phase is controlled by the fuel-air mixing process.

Chapter 16: Combustion Modeling

Non-Premixed Combustion Models

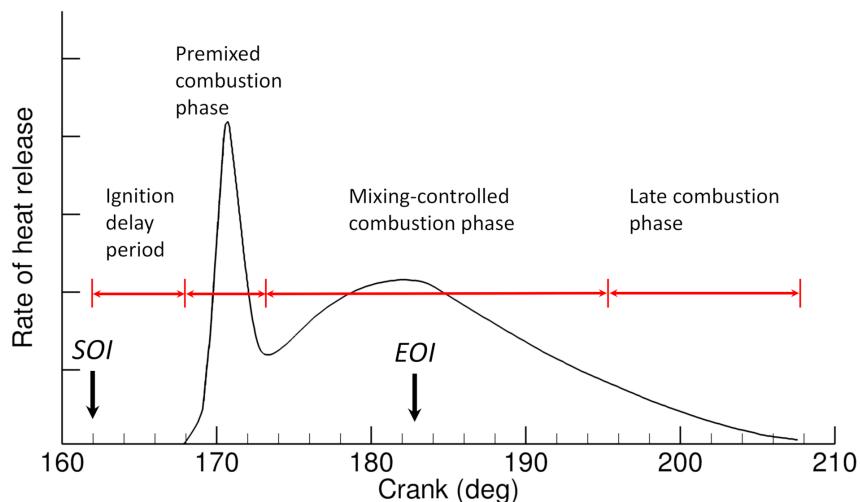


Figure 16.2: Schematic of a typical diesel engine heat release rate showing ignition delay, premixed combustion, and the mixing-controlled combustion phases ([Heywood, 1988](#)).

CONVERGE offers two different methods for modeling the ignition and combustion processes. The first method uses separate models for ignition and combustion. These models are based on the [Shell ignition model](#) and the [Characteristic Time Combustion model](#), and this method is relatively computationally inexpensive.

The second approach, described in the [SAGE detailed chemical kinetics solver](#) section, considers much of the chemistry taking place in combustion applications. Although the runtime using detailed chemistry can be significantly longer than with the first approach, the accuracy of the simulation may be greatly enhanced with the inclusion of detailed chemistry. Nonetheless, there is still often a need for the rapid turnaround time that can be achieved when the simpler models are used. The simpler ignition and combustion models are described in the following sections.

16.3 SAGE Detailed Chemical Kinetics Solver

CONVERGE offers the SAGE detailed chemical kinetics solver ([Senecal et al., 2003](#)), which solves detailed chemical kinetics via a set of CHEMKIN-formatted input files. To solve the systems of ordinary differential equations (ODEs), SAGE uses the CVODE solver, which is part of the SUNDIALS (SUite of Nonlinear and DIfferential/ALgebraic equation Solvers) package ([SUNDIALS, 2015](#)). The necessary routines have been incorporated into CONVERGE, so you do not need to install this package separately. In `combust.in > sage_model > ode_solver > type`, you can choose from several implementations of CVODE.

A chemical reaction mechanism is a set of elementary reactions that describe an overall chemical reaction. The combustion of different fuels can be modeled by changing the mechanism (e.g., there are mechanisms for iso-octane, gasoline, n-heptane, natural gas, etc.). SAGE calculates the reaction rates for each elementary reaction while the CFD solver solves the transport equations. Given an accurate mechanism, SAGE (in addition to AMR) can be used for modeling many combustion regimes (ignition, premixed, mixing-controlled). You

Chapter 16: Combustion Modeling

SAGE Detailed Chemical Kinetics Solver

can use SAGE to model either constant-volume or constant-pressure combustion. Note that SAGE consistently uses CGS units for all calculations.

As described by Turns (1996), a multi-step chemical reaction mechanism can be written in the following form:

$$\sum_{m=1}^M \nu'_{m,i} \chi_m \rightleftharpoons \sum_{m=1}^M \nu''_{m,i} \chi_m \quad \text{for } i=1,2,\dots,I \quad (16.1)$$

where $\nu'_{m,i}$ and $\nu''_{m,i}$ are the stoichiometric coefficients for the reactants and products, respectively, for species m and reaction i ; I is the total number of reactions; and χ_m is the chemical symbol for species m . The net production rate of species m is given by

$$\dot{\omega}_m = \sum_{i=1}^I \nu_{m,i} q_i \quad \text{for } m=1,2,\dots,M \quad (16.2)$$

where M is the total number of species and

$$\nu_{m,i} = \nu''_{m,i} - \nu'_{m,i}. \quad (16.3)$$

The rate-of-progress parameter q_i for the i -th reaction is

$$q_i = k_{i,f} \prod_{m=1}^M [X_m]^{\nu'_{m,i}} - k_{i,r} \prod_{m=1}^M [X_m]^{\nu''_{m,i}}. \quad (16.4)$$

In Equation 16.4, $[X_m]$ is the molar concentration of species m , and $k_{i,f}$ and $k_{i,r}$ are the forward and reverse rate coefficients for reaction i . In SAGE, the forward rate coefficient is expressed by the Arrhenius form as

$$k_{i,f} = A_i T^{\beta_i} \exp\left(\frac{-E_i}{RT}\right), \quad (16.5)$$

where A_i is the pre-exponential factor, β_i is the temperature exponent, E_i is the activation energy (in *cal/mol*), and R is the ideal gas constant. In addition, the reverse rate coefficient can either be specified in an analogous fashion as Equation 16.5, or calculated from the equilibrium coefficient $K_{i,c}$ as

Chapter 16: Combustion Modeling

SAGE Detailed Chemical Kinetics Solver

$$k_{i,r} = \frac{k_{i,f}}{K_{i,c}}. \quad (16.6)$$

The equilibrium coefficient $K_{i,c}$ is determined from the thermodynamic properties and is given by

$$K_{i,c} = K_{i,p} \left(\frac{P_{atm}}{RT} \right)^{\sum_{m=1}^M v_{mi}}, \quad (16.7)$$

where P_{atm} is the atmospheric pressure, R is the ideal gas constant, and T is the temperature. The equilibrium constant $K_{i,p}$ is obtained via

$$K_{i,p} = \exp \left(\frac{\Delta S_i^0}{R} - \frac{\Delta H_i^0}{RT} \right). \quad (16.8)$$

The Δ refers to the change that occurs in passing completely from reactants to products in the i -th reaction, specifically,

$$\frac{\Delta S_i^0}{R} = \sum_{m=1}^M v_{m_i} \frac{S_m^0}{R} \quad (16.9)$$

and

$$\frac{\Delta H_i^0}{RT} = \sum_{m=1}^M v_{m_i} \frac{H_m^0}{RT}, \quad (16.10)$$

where S and H denote entropy and enthalpy, respectively.

It should also be noted that SAGE allows for third-body reactions with the capability of specifying different third body efficiencies for different species. SAGE also allows for the solution of pressure dependent reactions in either the Lindemann, Troe, SRI, or PLOG form.

With the above information, the governing equations for mass and energy conservation can be solved for a given computational cell. The governing equation for mass is

$$\frac{d[X_m]}{dt} = \dot{\omega}_m. \quad (16.11)$$

Chapter 16: Combustion Modeling

SAGE Detailed Chemical Kinetics Solver

The governing equation for energy is

$$\frac{dT}{dt} = \frac{\frac{dP}{dt} - \sum_m (\bar{h}_m \dot{\omega}_m)}{\sum_m ([X_m] \bar{c}_{p,m})} \quad (16.12)$$

for constant-volume combustion and

$$\frac{dT}{dt} = \frac{\sum_m (\bar{h}_m \dot{\omega}_m)}{\sum_m ([X_m] \bar{c}_{p,m})} \quad (16.13)$$

for constant-pressure combustion. In the previous equations, T is temperature, P is pressure, $\dot{\omega}_m$ is determined by Equation 16.2, and \bar{h}_m and $\bar{c}_{p,m}$ are the molar specific enthalpy and molar constant-pressure specific heat of species m , respectively. The above equations are solved at each computational time-step and the species are updated appropriately. It is important to note that the temperature obtained from Equation 16.13 is used to update only the rate coefficients as SAGE is solving the system of rate equations and is not used to update the CONVERGE cell temperature. The cell temperature is updated after the detailed chemistry calculation has converged using the computed species concentrations.

In order to expedite the detailed chemistry calculations, kinetics are not solved in cells that fall below a minimum cell temperature (T_{cut}) and a minimum mole fraction (HC_{min}). The minimum mole fraction is the total mole fraction of CO, H₂, and the hydrocarbon species. The minimum mole fraction includes more than just the hydrocarbon species to allow carbon monoxide chemistry to take place in computational cells that do not include hydrocarbon species.

Acceleration of the SAGE Detailed Chemistry Solver

In addition to [adaptive zoning](#), CONVERGE contains features to accelerate the SAGE detailed chemistry solver. The first option is to not solve for the temperature given by Equation 16.12 or 16.13 unless the change in cell temperature from combustion from the previous time-step exceeds a specified value. This option allows CONVERGE to avoid recalculating the rate coefficients given in Equation 16.4, which can result in significant time savings. This option can be activated by setting [`combust.in > sage_model > solve_temp = 0`](#) (a value of 1 indicates that temperature will always be solved). The parameter [`combust.in > sage_model > delta_temp`](#) indicates the magnitude of the temperature change (above which temperature will be re-solved when `solve_temp = 0`). Set `delta_temp` to 2 K or lower (2 K is normally used). However, for certain mechanisms it may be possible to exceed a value of 2 K. You can run test cases for your particular application and kinetic mechanism if you want to exceed a value of 2 K.

Chapter 16: Combustion Modeling

SAGE Detailed Chemical Kinetics Solver

A second option, related to the Jacobian matrix calculation, may also speed up the SAGE chemistry solver. By default, CVODES calculates this matrix numerically. However, CONVERGE includes the option to pass an analytically calculated Jacobian to the solver which can potentially speed up the calculation. A case where using the analytically calculated Jacobian matrix does not speed up the solution has not been found. Therefore, we recommend that you set [`combust.in > sage_model > analyt_jac = 1`](#).

The tolerance in the detailed chemistry solver can also affect the runtime. The first tolerance, [`combust.in > sage_model > ode_solver > abs_tol`](#), is the iteration error for each species. The second, [`combust.in > sage_model > ode_solver > rel_tol`](#), is the iteration error for each species divided by the magnitude of the species.

Input Parameters for the SAGE Detailed Chemistry Solver

Mechanisms and thermodynamic data are defined by the [reaction mechanism](#) and [thermodynamic](#) data files, respectively. The SAGE parameters are located in the [`combust.in > sage_model`](#) settings block.

Stiffness-Based Load Balancing

For a SAGE simulation that does not include [adaptive zoning](#), you must include the [non-transport passive](#) `CHEM_STIFF` in [`species.in`](#). The inclusion of this non-transport [passive](#) invokes [stiffness-based load balancing](#) for the SAGE solver. For a SAGE simulation that includes adaptive zoning, `CHEM_STIFF` is optional. If you do not include `CHEM_STIFF`, CONVERGE simply distributes the combustion cells amongst all available processors.

Thickened Flame Model

In most large eddy simulations (LES) of premixed flames, the cells are not fine enough to resolve the laminar flame thickness. Thickened flame models (TFM) are designed to increase the flame thickness without changing the laminar flamespeed. In this way, the macroscopic combustion dynamics can be simulated without resolving the flame front explicitly.

The thickened flame model is available only when using the [SAGE detailed chemistry solver](#) and [LES turbulence modeling](#). This model is not appropriate for simulations with [RANS turbulence modeling](#) because RANS models automatically add a thickening effect.

CONVERGE incorporates the dynamic TFM of [Legier et al. \(2000\)](#). This model dynamically alters the nature of the flame front and the turbulence-chemistry interaction through E , an efficiency factor, and F , a local thickening factor. F is the linear scaling factor for the flame thickness. In the flame front, F has a value substantially greater than 1.0. Far from the flame front, F relaxes to 1.0. Because F is active only near the flame front, large-scale mixing is relatively unaffected. Table 16.2 below shows the TFM alterations to flame diffusivity, pre-exponential term, flamespeed, and flame thickness.

Table 16.2: TFM flame parameter alterations.

	Diffusivity	Pre-exponential term	Flamespeed	Flame thickness
Thin flame	D	A	s_l	δ_l
Thickened flame	$E \cdot F \cdot D$	$E \cdot A / F$	$E \cdot s_l$	$F \cdot \delta_l$

The species conservation equations are solved with these scaling factors applied:

$$\frac{\partial \rho Y_m}{\partial t} + \frac{\partial \rho Y_m u_j}{\partial x_j} = \frac{\partial}{\partial x_j} \left(\rho \cdot E \cdot F \cdot D \frac{\partial Y_m}{\partial x_j} \right) + \frac{E}{F} \dot{\omega}_m. \quad (16.14)$$

Thickening Methods

The general expression for the thickening factor F is given by

$$F = 1 + (F_{max} - 1)S, \quad (16.15)$$

where F_{max} is the maximum scaling factor and S is the [flame sensor](#) used to localize the effect to the area near the flame front. The thickening methods available in CONVERGE correspond to different ways of setting F_{max} .

When [`combust.in`](#) > `thickened_flame_model` > `thickening_method` > `option = 0`, you specify F_{max} directly. When `thickening_method` > `option = 1`, you specify the number of grid points across the flame (n_{res}), and CONVERGE calculates the maximum scaling factor as

$$F_{max} = \frac{n_{res} \Delta_x}{\delta_l}, \quad (16.16)$$

where Δ_x is the local grid spacing and δ_l is the laminar flame thickness. When `thickening_method` > `option = 2`, you specify the target flame thickness (δ_{target}), and CONVERGE calculates the maximum scaling factor as

$$F_{max} = \frac{\delta_{target}}{\delta_l}. \quad (16.17)$$

Flame Sensor Models

CONVERGE offers several models for the flame sensor S . The simplest is uniform thickening, in which $S = 1$ and $F = F_{max}$ throughout the domain. This option generally results in poorer mixing prediction and is not recommended for general use.

With the standard reaction rate model, CONVERGE calculates S from

$$S = \max \left[\min \left(\beta \frac{|\bar{\dot{\omega}}_{sens}|}{\dot{\Omega}_{sens,0}(\phi)} - 1, 1 \right), 0 \right], \quad (16.18)$$

where $\dot{\Omega}_{sens,0}$ is the maximum reaction rate of the sensor species from a [1D laminar premixed flame](#), $|\bar{\dot{\omega}}_{sens}|$ is the local reaction rate of the sensor species calculated by SAGE, and β is a model parameter. The species used for the flame sensor is user-specified ([combust.in > thickened_flame_model > flame_sensor_model > sensor_species](#)) and must match one of the sensor species available in the TLF table.

With Jaravel's sensor model, CONVERGE first calculates S from Equation 16.18 and then filters the sensor to capture the species gradients at the flame foot and tail (Jaravel 2016). This approach introduces a [passive](#) $\tilde{\psi}$, the indicator function, which is transported according to

$$\frac{\partial \bar{\rho} \tilde{\psi}}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_j \tilde{\psi}}{\partial x_i} = \frac{\partial}{\partial x_i} \left(F \Xi_\Delta \bar{\rho} \tilde{D}_\psi \frac{\partial \tilde{\psi}}{\partial x_i} \right) + \frac{\Xi_\Delta}{F} \bar{\dot{\omega}}_\psi. \quad (16.19)$$

In the indicator function transport equation, $\bar{\dot{\omega}}_\psi$ is a relaxation source term, which takes the values

$$\bar{\dot{\omega}}_\psi = \begin{cases} \frac{-\tilde{\psi}}{\tau_1} & \text{if } S < 0.05, \\ \frac{\psi_0 - \tilde{\psi}}{\tau_0} & \text{if } S > 0.8, \\ 0 & \text{if } 0.8 > S > 0.05, \end{cases} \quad (16.20)$$

and τ_0 and τ_1 are relaxation times. The first relaxation time τ_0 is correlated to the local time-step. The second relaxation time τ_1 is calculated based on the characteristic flame time τ_c according to

$$\tau_1 = \alpha_1 \tau_c, \quad \alpha_1 = \begin{cases} \alpha_{1,cold} & \text{if } T \leq T_s \\ \alpha_{1,hot} & \text{if } T > T_s \end{cases}, \quad (16.21)$$

where τ_c is calculated based on the laminar flame thickness, δ_l , and the [laminar flamespeed](#), s_l , as

$$\tau_c = \frac{\delta_l}{s_l}. \quad (16.22)$$

Finally, the initial value of S is adjusted according to

$$S = \max \left[\min \left(\tilde{\psi}, 1 \right), S \right]. \quad (16.23)$$

Efficiency Function

In Equation 16.19, Ξ_Δ is the efficiency function precursor. There are two models available for this precursor. The model of Charlette et al. (2002) calculates this term as

$$\Xi_\Delta = \left(1 + \min \left[\frac{\Delta}{\delta_l} - 1, \Gamma_\Delta \left(\frac{\Delta}{\delta_l}, \frac{u'_\Delta}{s_l}, Re_\Delta \right) \frac{u'_\Delta}{s_l} \right] \right)^\beta, \quad (16.24)$$

where Γ_Δ is an effective straining function, s_l is the laminar flamespeed, and u'_Δ is the sub-grid scale turbulent velocity. The model of [Colin and Ducros \(2000\)](#) calculates the efficiency function precursor as

$$\Xi_\Delta = 1 + \beta_{Colin} \frac{2 \ln(2)}{3 c_{ms} \left[Re_t^{1/2} - 1 \right]} \Gamma_{Colin} \left(\frac{\Delta}{\delta_l}, \frac{u'_\Delta}{s_l} \right) \frac{u'_\Delta}{s_l}, \quad (16.25)$$

where c_{ms} is a model constant with a value of 0.28 and β_{Colin} is a model parameter (unrelated to the parameter in Equation 16.18). The efficiency function E is calculated as

$$E = \frac{\Xi|_{\delta=\delta_l}}{\Xi|_{\delta=F\delta_l}}. \quad (16.26)$$

In addition to the models described above, you have the option to set E to a constant value, although this is not recommended for general use.

TFM Coupling with AMR

The thickened flame model can be coupled with [Adaptive Mesh Refinement](#) to reduce computational cost. This coupling is through a target thickening factor F_{target} and a target number of grid cells across the thickened flame. The target AMR level n_{AMR} is calculated from

$$F_{target} = \frac{n_{res}\Delta_x^{AMR}}{\delta_l}, \quad (16.27)$$

where

$$\Delta_x^{AMR} = \frac{\Delta_x^{Base}}{2^{n_{AMR}}} \quad (16.28)$$

and

$$n_{AMR} = \text{int} \left[\frac{1}{\log(2)} \log \left(\frac{n_{res}\Delta_x^{Base}}{\delta_l F_{target}} \right) \right]. \quad (16.29)$$

In TFM simulations with potential flame-wall interaction, if the flame reaches the wall, the model will calculate an inaccurate thermal flux at the wall. CONVERGE incorporates a wall treatment to address this. This approach limits the thickening factor to 1.0 near the wall, which has the effect of deactivating the model locally.

Input Parameters for the Thickened Flame Model

The thickened flame model parameters are located in the [`combust.in > thickened_flame_model`](#) settings block. When using the thickened flame model, you must activate the [SAGE detailed chemistry solver](#) (*i.e.*, set [`combust.in > sage_model > active = 1`](#)) and [LES turbulence modeling](#) (*i.e.*, set [`turbulence.in > turbulence_model = LES_*`](#)).

The thickened flame models defines several internal [passives](#), presented below in Figure 16.3. These [passives](#) are available in [`post.in`](#).

```
OMEGA_FUEL_GAS
TFM_THICKENING_FACTOR
TFM EFFICIENCY_FACTOR
TFM_SENSOR
TFM_UPRIME
LAM_FLAMESPEED
LAM_FLAMETHICKNESS
TFM_OMEGA
TFM_IND
TFM_SENSOR_INTERM
PROG_VAR_TFM
YC_UNBURNNT_TFM
YC_EQ_TFM
```

Figure 16.3: Internal passives for the thickened flame model.

Three-Point PDF Method

When running with SAGE, CONVERGE evaluates the chemical source term $\dot{\omega}$ in each cell at each time-step (or in each adaptive zone) using the resolved temperature and species concentration. In turbulent combusting flows, there are two primary turbulence-chemistry interaction (TCI) physical effects. The primary effect is enhanced mixing, which CONVERGE accounts for by modeling turbulent viscosity and turbulent diffusivity. The secondary effect is a commutation error $\dot{\omega}'$, which is produced by the RANS temporal averaging or LES spatial filtering operation,

$$\widehat{\dot{\omega}_m(Y_m, T)} = \widehat{\dot{\omega}_m}(\widetilde{Y}_m, \widetilde{T}) + \dot{\omega}'_m. \quad (16.30)$$

In LES, this commutation error limits to zero as the cell size is reduced to zero, but this is not true for RANS. In many combusting flows, this error is relatively small, and you can produce predictive results without accounting for it. In other flows (especially high Re turbulent combustion with local extinction or re-ignition), it must be modeled. To correct for the commutation error, CONVERGE can account for either equivalence ratio fluctuations (primarily for non-premixed flames) or temperature fluctuations (primarily for premixed flames) with a three-point delta PDF method.

In the three-point PDF method, CONVERGE treats each cell (or adaptive zone) as though it has three simultaneous states in the fluctuating variable (denoted ψ). Note that equivalence ratio and temperature cannot both be treated as fluctuating. The fluctuating variable states ψ_k are defined with one above and one below the mean value $\bar{\psi}$,

$$\begin{aligned} \psi_2 &= \bar{\psi}, \\ \psi_1 &= \bar{\psi} - f_R \psi''^2, \\ \psi_3 &= \bar{\psi} + f_R \psi'^2, \end{aligned} \quad (16.31)$$

where ψ''^2 is the variance in the fluctuating variable and f_R is a relaxation function. The relaxation function is designed so that the effect of the three-point PDF method limits to zero in the limit of small time-step, and is calculated as

$$f_R = \left(1 - \exp \left[\frac{-C_R dt}{\tau_{turb}} \right] \right), \quad (16.32)$$

where C_R is a user-specified relaxation constant and τ_{turb} is the turbulent time scale, calculated as

$$\tau_{turb} = \frac{k}{\varepsilon}. \quad (16.33)$$

CONVERGE calculates the chemical source term $\dot{\omega}$ as

$$\dot{\omega}_i = P_1 \dot{\omega}_{i,1} + P_2 \dot{\omega}_{i,2} + P_3 \dot{\omega}_{i,3}, \quad (16.34)$$

where $\dot{\omega}_{i,k}$ is the source term evaluated at the k -th fluctuating state and P_k is the probability of each state, calculated as

$$P_2 = \exp \left(\frac{-C_p \psi''^2}{\bar{\psi}} \right) \quad (16.35)$$

and

$$P_1 = P_3 = 0.5 - 0.5P_2. \quad (16.36)$$

Because the source term must effectively be solved three times for each cell (or adaptive zone), the three-point PDF method is about three times as computationally expensive as the SAGE solution without the three-point PDF.

For either fluctuating temperature or fluctuating equivalence ratio, you have two options for how to transport the variance. You can solve an additional variance transport equation,

$$\frac{\partial \bar{\rho} \psi''^2}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_i \psi''^2}{\partial x_i} = \frac{\partial}{\partial x_j} \left(\bar{\rho} D_i \frac{\partial \psi''^2}{\partial x_j} \right) + C_\psi \mu_i \left(\frac{\partial \bar{\psi}}{\partial x_i} \right)^2 - C_d \bar{\rho} \frac{\varepsilon}{k} \psi''^2. \quad (16.37)$$

Alternately, you can solve for the variance explicitly,

$$\psi''^2 = C \frac{\mu_t}{\rho} \frac{k}{\varepsilon} \left(\frac{\partial \bar{\psi}}{\partial x_i} \right)^2, \quad (16.38)$$

where C is [*combust.in*](#) > *three_points_pdf_model* > *variance_cmu*. We generally recommend the explicit method.

16.4 Chemical Equilibrium (CEQ) Model

If the chemical time scales are fast relative to the transport time scales, equilibrium may be used for the chemistry model. CONVERGE uses the CEQ ([Pope, 2003](#)) chemical equilibrium solver licensed from Ithaca Combustion Enterprise. Unlike many equilibrium solvers, the CEQ solver is guaranteed to converge for any combination of gas species.

[*combust.in*](#) > *ceq_model* > *active* = 1

Since CEQ uses data provided in the thermodynamic data and reaction mechanism files, these files must be accurately provided in order for the solver to calculate the correct equilibrium concentrations. To activate the CEQ equilibrium solver, set [*combust.in*](#) > *ceq_model* > *active* = 1. To reduce runtime, two additional inputs are provided: [*combust.in*](#) > *general_control* > *temp_cutoff* and *general_control* > *hc_minimum*. Use *temp_cutoff* to specify a temperature below which the CEQ equilibrium solver will not be used. Use *hc_minimum* to specify a minimum hydrocarbon mole fraction below which the CEQ equilibrium solver will not be used. The hydrocarbon mole fraction also includes the mole fraction of CO. The mole fraction of CO is included to allow carbon monoxide chemistry to take place in computational cells that do not include hydrocarbon species.

[*combust.in*](#) > *ceq_model* > *active* = 2

In many cases, the mixing zone between the fuel and oxidizer is not sufficiently resolved. In this case, you can slow the progress to equilibrium by using a mixing time scale. Similar to the [*Characteristic Time Combustion model*](#), the combustion progress may be slowed with a mixing time scale given by

$$\frac{\partial \rho_m}{\partial t} = - \frac{\rho_m - \rho_m^*}{\tau_{mix}} \quad (16.39)$$

where m is the species index, ρ_m is the species density, ρ_m^* is the instantaneous equilibrium species density, and the time scale is given by

$$\tau_{mix} = C_2 \frac{k}{\varepsilon} \quad (16.40)$$

where C_2 is a constant, k is the turbulent kinetic energy, and ε is the turbulent dissipation.

Chapter 16: Combustion Modeling

Chemical Equilibrium (CEQ) Model

Species Limitation

You can reduce the computational time of a CEQ simulation by limiting the number of species that are solved as part of the CEQ equilibrium solver. Activate this feature by specifying species via [combust.in](#) > `ceq_model` > `subset_species`. The species name must match the names in the reaction mechanism and thermodynamic data files.

Note that if you choose to use the [NASA 9 format](#) for the thermodynamic data file with CEQ solver, the data file must be specified according to the standard equations rather than the general form of the equations.

Input Parameters for the CEQ Chemical Equilibrium Solver

The CEQ model parameters are located in the [combust.in](#) > `ceq_model` settings block.

16.5 Flamelet Generated Manifold (FGM) Model

The Flamelet Generated Manifold (FGM) model ([van Oijen and de Goey, 2000](#)) reduces the reaction mechanism to two [scalars](#) in order to reduce computational time compared to models that use a full chemical mechanism. The FGM model is typically used for systems not dominated by kinetic phenomena such as local extinction or slow-forming pollutants (e.g., gas turbines at full power or coal-fired furnaces). This model captures kinetic phenomena such as ignition, flame extinction, and flame quenching and provides accurate flame dynamics, fuel effects, and emissions.

Prior to a 3D CONVERGE simulation with FGM, you must use CONVERGE to [populate an FGM lookup table](#) with a range of model solutions based on the user-specified flamelet type (zero-dimensional ignition, one-dimensional diffusion, or one-dimensional premixed) and other parameters. During the simulation, CONVERGE accesses the lookup table to quickly retrieve thermochemical information.

Species Parameters

The FGM model simplifies the chemistry to two scalars: the mixture fraction, Z , and the reaction progress variable, c . The lookup table contains these scalars in addition to other thermodynamic data. The normalized ratio of fuel to oxidizer, Z , has a unique value for each grid cell at each time-step.

Note that when simulating premixed systems, it is computationally optimal to define the system's fuel ($Z = 1$) as the premixed composition of fuel and oxidizer.

The reaction progress variable, c , is defined as the sum of the product mass fractions normalized by their equilibrium values, as

$$c = \frac{\sum \alpha_k Y_k}{\sum \alpha_k Y_k^{eq}}. \quad (16.41)$$

Chapter 16: Combustion Modeling

Flamelet Generated Manifold (FGM) Model

In Equation 16.0, Y_k denotes species mass fraction and α_k is equal to 0 for most reactants. We recommend that only α_{CO_2} and α_{CO} equal 1 and all other α values equal 0. In the FGM model, it is important to select α_k so that c increases monotonically from unburned ($c = 0$) to burned ($c = 1$) regions.

Flamelet Types

The FGM model in CONVERGE offers three flamelet types: 0D ignition, 1D diffusion, and 1D premixed.

CONVERGE simulates the [0D ignition](#) flamelet using solutions from an experimental 0D homogeneous reactor over a range of equivalence ratios, initial temperatures, and initial pressures. The lookup table contains the calculated solutions for Z , c , pressure (P), and enthalpy (H) in binary format. The 0D ignition flamelet type best captures the combustion flame dynamics in internal combustion engines in which pressure and total enthalpy can change substantially over time.

CONVERGE simulates the [1D diffusion flamelet](#) using solutions from an experimental 1D opposed-flow diffusion flame setup at constant pressure. CONVERGE calculates the pre-ignition regime from unsteady quenching flamelets at the extinction strain rate and the ignition regime from steady flamelets over a range of stoichiometric scalar dissipation rates, χ_{st} via

$$\chi(Z) = \frac{\chi_{st} f(Z)}{f(Z_{st})}. \quad (16.42)$$

In the previous equation, $\chi(Z)$ is the scalar dissipation rate profile, Z_{st} is the stoichiometric mixture fraction, and $f(Z)$ is defined in a [later section](#). The FGM model assumes that heat loss does not affect the species composition. CONVERGE calculates the non-adiabatic heat loss as the adiabatic temperature in Z space multiplied by user-specified upper and lower bounds. The lookup table contains the calculated solutions for Z , c , H , and $Z^{1/2}$, the variance of Z . The 1D diffusion flamelet type best represents the combustion flame in constant pressure combustors, such as a coal-fired furnace or a liquid fuel gas turbine combustor.

CONVERGE simulates the 1D premixed flamelet using solutions developed from an experimental 1D freely propagating premixed flame setup solved in physical space over the entire range of equivalence ratios. CONVERGE calculates non-adiabatic effects similar to the method described for the 1D diffusion flamelet. The lookup table contains the calculated solutions for Z , c , H , and $Z^{1/2}$. The 1D premixed flamelet type best represents turbulent premixed flames in constant pressure combustors, such as lean, premixed gas turbines.

Turbulence Closure

RANS and LES turbulence models used in conjunction with the FGM model require additional equations to solve for the turbulence/flame interaction. The additional closure

Chapter 16: Combustion Modeling

Flamelet Generated Manifold (FGM) Model

equations are presumed shape probability density functions (PDFs). The presumed shape PDFs are based on the theoretical distribution of each parameter in the lookup table.

The 0D ignition flamelet has a different set of presumed shape PDFs than both of the 1D flamelets. The 0D ignition flamelet uses delta functions for Z , c , enthalpy, and pressure PDFs. The 1D diffusion and premixed flamelets use a beta function for the Z PDF and delta functions for the c , E , and $Z^{''2}$ PDFs.

Since the species details are simplified in the FGM model, CONVERGE solves the Favre mean (mass-averaged) equations of $Z^{''2}$ and c . For RANS simulations, CONVERGE solves the transport equation (in a [later section](#)) for $Z^{''2}$, while for LES, CONVERGE calculates the variance by solving

$$Z^{''2} = C_{var} \Delta^2 |\nabla Z|^2 , \quad (16.43)$$

in which C_{var} is a constant and Δ is the LES sub-grid length scale. The transport equation of the Favre mean reaction progress variable is given by

$$\frac{\partial}{\partial t}(\bar{\rho}\tilde{c}) + \frac{\partial}{\partial x_i}(\bar{\rho}\tilde{u}_i\tilde{c}) = \frac{\partial}{\partial x_i}(\bar{\rho}D_t \frac{\partial \tilde{c}}{\partial x_i}) + \bar{\rho}\tilde{c} , \quad (16.44)$$

in which the first term is the time-dependent term, the second term is the convection-dependent term, the third term is the diffusion term, and the fourth term is the mean unclosed source term.

CONVERGE evaluates the mean source term, $\bar{\rho}\tilde{c}$, for the reaction progress with either a kinetic rate model or turbulent flamespeed model. For the 1D diffusion flamelet, the model calculates the mean source term from the kinetic rate model, in which the c PDF, $\delta(c - \tilde{c})$, is multiplied by the presumed shape PDF, $P(Z)$, as

$$\bar{\rho}\tilde{c} = \int_0^1 \left(\frac{\sum \alpha_k \dot{\omega}_k}{\sum \alpha_k Y_k^{eq}} \right) P(Z) \delta(c - \tilde{c}) dc dZ . \quad (16.45)$$

In the previous equation, $\dot{\omega}_k$ is the net reaction rate of the k -th species. For the 1D premixed flamelet type, CONVERGE calculates the mean source term from the turbulent flamespeed model as

$$\bar{\rho}\tilde{c} = \rho_u s_t |\nabla \tilde{c}| , \quad (16.46)$$

Chapter 16: Combustion Modeling

Flamelet Generated Manifold (FGM) Model

in which ρ_u is the unburned density and s_t is the turbulent flamespeed. When using the turbulent flamespeed model, you must specify how to solve for s_t ; set [combust.in > st_model > active = 1](#) to solve using the Peters model (use the [turbulent flamespeed equation](#)), set [st_model > active = 2](#) (solve using the Zimont model), or set [st_model > active = 4](#) (solve using the [viscosity ratio correlation](#)). The Zimont model ([Zimont et al., 1998](#)) defines the turbulent flamespeed as

$$s_t = A u' D a^{1/4}, \quad (16.47)$$

in which A is the user-specified Zimont constant ([combust.in > st_model > st_option > zimont_correlation > a](#)), u' is the root mean square of the turbulent fluctuating velocity, and Da is the [Damkohler number](#).

For the 0D ignition flamelet type, CONVERGE evaluates the mean source term with the kinetic rate model when $\text{grad } c$ is a small value; otherwise, CONVERGE solves the turbulent flamespeed model.

Input Parameters for the FGM Model

FGM model parameters are located in the [combust.in > fgm_model](#) settings block as well as in an [FGM table file](#) (e.g., *fgm_table.h5*).

For an FGM simulation, you must set [inputs.in > property_control > eos_flag = 1](#) and [parcels.in > liquid_parcels > parcel_list > parcel > evap_control > evap_source_flag = 0](#).

For the 0D ignition flamelet, CMEAN and ZMEAN are the internal [passives](#). For the 1D diffusion and 1D premixed, CMEAN, ZMEAN, and ZVAR are the internal [passives](#).

After running the 3D simulation with FGM, CONVERGE writes all final thermodynamic and passive information to the [thermo.out](#) and [passive.out](#) files, respectively.

16.6 G-Equation Model

CONVERGE includes the level set G-Equation model ([Peters, 2000](#)), which you can use to simulate premixed combustion. Because the derivation of this model includes several simplifications, it should be considered a less predictive combustion model.

In the formulation described below, the G-Equation model is written in terms of a flame front tracking variable G and a progress variable C . CONVERGE and CONVERGE Studio names the equivalent internal [passives](#) *G_EQN_TRANSPORT* and *G_EQN_PROGRESS*, respectively.

Ignition Approach for the G-Equation Model

Ignition for the G-Equation model is typically achieved through a source term. To source G , set [inputs.in > feature_control > source_flag = 1](#) and [define a passive source \(G_EQN_TRANSPORT\)](#) in [source.in](#). As shown below in Figure 16.4, the source begins at -10

Chapter 16: Combustion Modeling

G-Equation Model

CAD (the spark timing) and has a duration of 3.0 CAD. Specify the location of the source (for a spark) at the electrode. The maximum value (*source > max_value*) of the *G_EQN_TRANSPORT passive* allowed is set to 0.35e-3 and is proportional to the grid size (typically, twice the local grid size).

To use this method for sourcing the G-Equation directly, add embedding around the spark region such that the cells around the source (spark) are approximately 0.125-0.250 mm. Rather than sourcing G directly, you can alternatively add energy directly via an energy source to simulate the spark. In this case, when the temperature exceeds the temperature cutoff (*combust.in > g_eqn_model > temp_cutoff*) in a cell, positive G will be initialized in the cell and the G-Equation model will propagate the flame from the initialized cells.

```
version: 3.1
---

- source:
    description: Source 1
    equation: G_EQN
    type: PER_UNIT_VOLUME_TIME
    value: 8.0e11
    temporal_control:
        type: SEQUENTIAL
        cyclic_period: 720
        source_start_time: -10.0
        source_end_time: -7.0
    max_value: 0.35e-3
    shape:
        type: SPHERE
        x_center: [-0.003, 0, 0.0091]
        radius: 2.5e-4
    moving_control:
        moving_flag: STATIONARY
        velocity: [1, 2, 3]
        max_displace: 0.002
        reset_source_flag: DO_NOT_RESET
    mult_dt_source: 1
```

Figure 16.4: Sample *source.in* file that sources G directly to simulate a spark.

You can optionally model ignition with a modified version of the spark channel ignition monitoring model (SparkCIMM, [Dahms, 2009](#)), which treats the spark kernel as a set of parcels. In effect, it is a sub-grid spark ignition model. This model deposits spark energy into a set of parcels arrayed across the spark gap, 100 mJ/s multiplied by the specified spark efficiency factor. The spark kernel parcels are advected. At each parcel location, CONVERGE evaluates the local Karlovitz number,

$$Ka = \left(\frac{\eta}{s_l} \right)^2, \quad (16.48)$$

where η is the Kolmogorov length scale and s_l is the [laminar flamespeed](#) plus a plasma expansion speed contribution. When the local Karlovitz number is below a specified critical Karlovitz number, SparkCIMM flags the parcel as combusting, specifies a 0.5 mm

combustion parcel diameter, and allows the parcel to grow and merge with other combusting parcels. Once the parcel diameter is as large as the local grid spacing, CONVERGE adds a positive *G_EQN_TRANSPORT passive* source according to the above direct-source method.

Because we recommend using embedding in the spark region down to a cell spacing of 0.125-0.250 mm, SparkCIMM has identical behavior to sourcing the G-Equation directly, with additional computational cost. Combusting parcels are already larger than the local cell spacing, so the G-Equation is sourced immediately. Because the behavior is identical, this kernel-parcel ignition method is not recommended for general use.

To model ignition with spark kernel parcels, set *combust.in* > *g_eqn_model* > *spark* > *active* = 1 and include the *g_eqn_spark.in* file in your case setup.

Initialization of G

The basic G-Equation model tracks the location of the flame front via the transport of the *G_EQN_TRANSPORT passive*. The parameter *G* indicates the distance to the flame front. A value of zero for *G* indicates that the flame is at that location. A negative value indicates the region is unburned and a positive value indicates that the region is burned (see Figure 16.5 below). Initialize *G* via *combust.in* > *g_eqn_model* > *init_value*. You can specify a single numerical value or a file name to set up [region-based initialization](#).

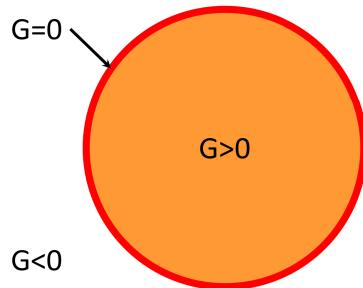


Figure 16.5: Unburned ($G<0$), burned ($G>0$), and flame ($G=0$) zones.

Transport of G

The G-Equation model is based on the assumption that the premixed turbulent combustion occurs in either the corrugated flamelet or the thin reaction zone regime ([Peters, 2000](#)). With this assumption, the turbulent flame front can be tracked by solving for the mean (and possibly the variance) of a non-reacting scalar, *G* ([Ewald and Peters, 2005](#)), as

$$\frac{\partial \rho \tilde{G}}{\partial t} + \frac{\partial \rho \tilde{u}_i \tilde{G}}{\partial x_i} = -\rho D_t \tilde{\kappa} \left| \frac{\partial \tilde{G}}{\partial x_i} \right| + \rho_u s_t \left| \frac{\partial \tilde{G}}{\partial x_i} \right| \quad (16.49)$$

and

Chapter 16: Combustion Modeling

G-Equation Model

$$\frac{\partial \rho \widetilde{G}^{''2}}{\partial t} + \frac{\partial \rho \widetilde{u}_i \widetilde{G}^{''2}}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D_t \frac{\partial \widetilde{G}^{''2}}{\partial x_i} \right) + 2\rho D_t \frac{\partial \widetilde{G}}{\partial x_i} \frac{\partial \widetilde{G}}{\partial x_i} - c_s \rho \widetilde{G}^{''2} \frac{\varepsilon}{k}, \quad (16.50)$$

where s_t is the turbulent flamespeed, ρ_u is the unburned density, k is the turbulent kinetic energy, ε is the turbulent dissipation, and c_s is a user-supplied constant ([combust.in > st_model > st_option > peters_correlation > g_prime_cs](#)). It should be noted that there are two versions of the G-Equation model. In one version, the variance of G is solved (we call this version the Peters flamespeed model with G -prime) (see Equation 16.50 above). In the other version, the variance of G is not solved (we call this version the Peters flamespeed model without G -prime). Use [combust.in > st_model > active](#) to specify whether CONVERGE solves for the variance of G .

For large eddy simulations, Equation 16.50 incorporates an additional term, as follows:

$$\begin{aligned} \frac{\partial \rho \widetilde{G}^{''2}}{\partial t} + \frac{\partial \rho \widetilde{u}_i \widetilde{G}^{''2}}{\partial x_i} &= \frac{\partial}{\partial x_i} \left(\rho D_t \frac{\partial \widetilde{G}^{''2}}{\partial x_i} \right) + 2\rho D_t \frac{\partial \widetilde{G}}{\partial x_i} \frac{\partial \widetilde{G}}{\partial x_i} - c_s \rho \widetilde{G}^{''2} \frac{\varepsilon}{k} - \rho \widetilde{\omega}, \\ \widetilde{\omega} &= c_2 \left(s_l \overline{\sigma} \right)^2 \frac{C_s \Delta}{v_{\Delta}'}, \end{aligned} \quad (16.51)$$

where the additional term compensates for kinematic restoration ([Pitsch, 2002](#)). In this additional term, s_l is the laminar flamespeed, C_s is the appropriate Smagorinsky constant, Δ is the filter width, v'_{Δ}' is the sub-grid velocity fluctuation, and c_2 is a proportionality factor. The latter is evaluated according to Equation 16.51 as

$$c_2 = \frac{2}{Sc_{\Delta} b_1^2}, \quad (16.52)$$

where the coefficient b_1 is taken to be 2.0 ([Peters, 2000](#)).

The last term on the right-hand side of Equation 16.49 is attributed to the averaged turbulent mass burn rate. The first term of Equation 16.49 accounts for the influence of curvature on the flame front. For this term, the mean flame front curvature can be found from

$$\tilde{\kappa} = -\frac{\partial}{\partial x_i} \left(\frac{\partial \widetilde{G}}{\partial x_i} \Bigg/ \left| \frac{\partial \widetilde{G}}{\partial x_i} \right| \right). \quad (16.53)$$

The turbulent diffusion terms are given by

Chapter 16: Combustion Modeling

G-Equation Model

$$D_t = \frac{c_\mu}{Sc} \frac{k^2}{\varepsilon}. \quad (16.54)$$

When the variance of G is solved (*i.e.*, when [*combust.in*](#) > *st_model* > *active* = 11) ([Ewald and Peters, 2005](#)),

$$D'_t = \sqrt{c_s \frac{k}{2} \frac{c_\mu}{Sc} \widetilde{G'^2}}. \quad (16.55)$$

When the variance of G is not solved (*i.e.*, when [*combust.in*](#) > *st_model* > *active* = 1),

$$D'_t = \frac{c_\mu}{Sc} \frac{k^2}{\varepsilon}. \quad (16.56)$$

The level set approach is used to solve Equation 16.49. The level set method is a numerical method designed to track and maintain a sharp interface. The level set method is well-suited for the G-Equation model (in which the flame front must be maintained and tracked). With this implementation, the mean flame front position is defined as the location where $G(x,t) = 0.0$ in the solution of Equation 16.49. The interface divides the flow field into two regions: (1) an unburned region ($G < 0.0$) and a burned region ($G > 0.0$) ([Tan and Reitz, 2003](#)). Outside the flame surface, the scalar is required to satisfy

$$\left| \frac{\partial G}{\partial x_i} \right| = 1. \quad (16.57)$$

Transport of Progress Variable

The CONVERGE implementation of the G-Equation model also transports the *G_EQN_PROGRESS* [passive](#). In short, the progress variable C retards the completion of combustion in some cases, to prevent nonphysical behavior. The flame thickness divided by the flame speed gives a characteristic flame time τ_{flame} . The progress variable approach allows the flame to progress by a fraction of (dt/τ_{flame}) at each time-step. As such, the progress variable is a local adjustment to G , and the rate of combustion process is governed according to C , not to G . C varies between zero and one. $C = 0$ corresponds to no combustion, and $C = 1$ corresponds to full combustion calculated from equilibrium chemistry.

The progress variable C is transported by solving

$$\frac{\partial \rho \tilde{C}}{\partial t} + \frac{\partial \rho \tilde{u}_i \tilde{C}}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D_t \frac{\partial \tilde{C}}{\partial x_i} \right) + \rho S, \quad (16.58)$$

Chapter 16: Combustion Modeling

G-Equation Model

where S is a source term of the progress variable. The source term S is evaluated according to the turbulent flamespeed s_t and the turbulent flame length scale l_t :

$$S = \frac{s_t}{\max\left[\frac{\Delta x}{2}, l_t\right]}, \quad l_t = \frac{3}{\Pr} \frac{\mu_t}{\rho s_t}. \quad (16.59)$$

The multiplicative value of 3 is a scaling factor derived from numerical experiment. S is further restricted in value according to

$$0 < \tilde{C}^{n+1} + dt \cdot S < 1. \quad (16.60)$$

Combustion Module

In principle, the G-Equation model does not need a combustion solver to track the flame front. Hence the G-Equation model typically runs faster than detailed chemistry. However, we recommend the use of [SAGE detailed chemistry](#) (in conjunction with a RANS turbulence model and AMR) to obtain combustion products and to predict emissions.

You can direct CONVERGE to use the [CEQ equilibrium solver](#) or the [SAGE detailed chemistry solver](#) in the burned region (inside the flame) via `combust.in > g_eqn_model > burned_region`. You can direct CONVERGE to use the [CEQ equilibrium solver](#) or the [SAGE detailed chemistry solver](#) at the flame front via `combust.in > g_eqn_model > on_flame`. Finally, you can direct CONVERGE to use the [SAGE detailed chemistry solver](#) in the unburned region (outside the flame) via `combust.in > g_eqn_model > unburned_region`. Five combinations of these options are supported. These are listed below.

Figure 16.6 below depicts `combust.in > g_eqn_model > burned_region = CEQ`, `g_eqn_model > on_flame = CEQ`, and `g_eqn_model > unburned_region = NONE`, which is the typical setting for the G-Equation model. This setting runs faster than the subsequent options. The species for the CEQ equilibrium solver can be defined either in the [reaction mechanism file](#) or in [species.in](#).

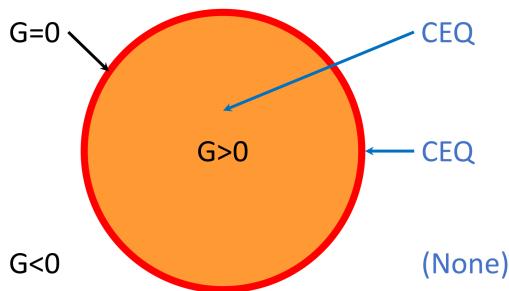


Figure 16.6: G-Equation model when `combust.in > burned_region = CEQ`, `on_flame = CEQ`, and `unburned_region = NONE`.

Chapter 16: Combustion Modeling

G-Equation Model

Figure 16.7 below depicts `combust.in > g_eqn_model > burned_region = CEQ`, `g_eqn_model > on_flame = CEQ`, and `g_eqn_model > unburned_region = SAGE`. This option requires a [reaction mechanism file](#). This option helps to predict knock.

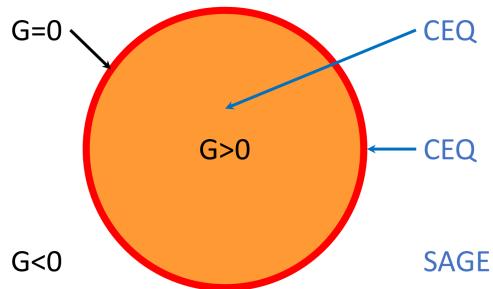


Figure 16.7: G-Equation model when `combust.in > burned_region = CEQ`, `on_flame = CEQ`, and `unburned_region = SAGE`.

Figure 16.8 below depicts `combust.in > g_eqn_model > burned_region = SAGE`, `g_eqn_model > on_flame = CEQ`, and `g_eqn_model > unburned_region = SAGE`. This option requires a [reaction mechanism file](#). This option helps to accurately predict both knock and NOx emissions.

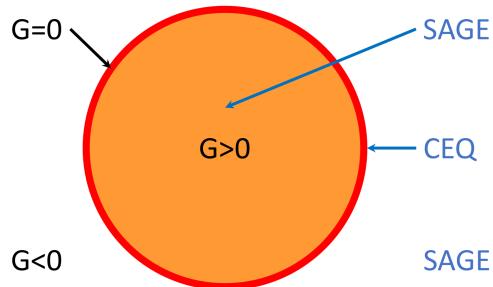


Figure 16.8: G-Equation model when `combust.in > burned_region = SAGE`, `on_flame = CEQ`, and `unburned_region = SAGE`.

Figure 16.9 below depicts `combust.in > g_eqn_model > burned_region = SAGE`, `g_eqn_model > on_flame = SAGE`, and `g_eqn_model > unburned_region = SAGE`. This option requires a [reaction mechanism file](#). Use this option to couple the G-Equation model with the [PM](#) and [PSM](#) detailed soot models.

Chapter 16: Combustion Modeling

G-Equation Model

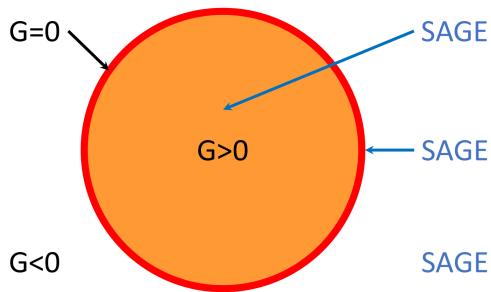


Figure 16.9: G-Equation model when `combust.in > burned_region = SAGE`, `on_flame = SAGE`, and `unburned_region = SAGE`.

Figure 16.10 below depicts `combust.in > g_eqn_model > burned_region = SAGE`, `g_eqn_model > on_flame = SAGE`, and `g_eqn_model > unburned_region = NONE`. This option requires a [reaction mechanism file](#). Use this option to couple the G-Equation model with the [PM](#) and [PSM](#) detailed soot models.

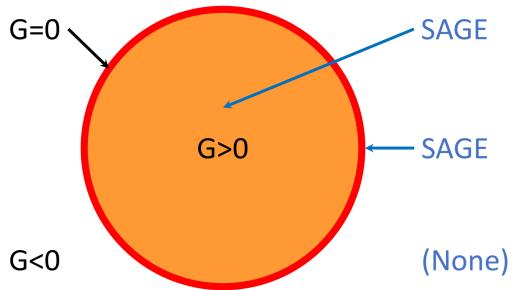


Figure 16.10: G-Equation model when `combust.in > burned_region = SAGE`, `on_flame = SAGE`, and `unburned_region = NONE`.

If `combust.in > g_eqn_model > active = 1`, set `combust.in > sage_model > active = 11`.

Acceleration Options

If you have activated the G-Equation model, you can invoke [adaptive zoning](#) to accelerate the combustion calculations.

Input Parameters for the G-Equation Model

The G-equation parameters are included in the `combust.in > g_eqn_model` settings block.

16.7 Extended Coherent Flame Models

The Extended Coherent Flame Model (ECFM) and 3-Zone Extended Coherent Flame Model (ECFM3Z) are extensions of the Coherent Flame Model proposed by [Marble and Broadwell \(1977\)](#). The ECFM is a premixed combustion model, while the ECFM3Z is a partially- or non-premixed combustion model. The following sections describe the theory for each model and the ECFM/ECFM3Z setup in CONVERGE.

Extended Coherent Flame Model (ECFM)

The ECFM ([Colin et al., 2003](#)) can be used to accurately model the combustion process, especially in spark-ignited engines. ECFM was re-implemented in CONVERGE 2.4.15, and for CONVERGE 2.4.15+, ECFM is only species-based (ECFM-SB). Tracer species are no longer used to close the atomic balances of C, O, N, and H. This change resolves the mass and atom conservation issues. Tracer species are still required to define locally the input parameters of TKI and laminar flamespeed tables. This section describes the theory of the ECFM and optional spark model.

To determine the flame surface density, the ECFM uses the fuel/air equivalence ratio in fresh gases, the composition (including residual gases), and the temperature near the flame. The resulting flame surface density is used to describe large scale burned/unburned stratification.

CONVERGE uses different transport equations for the flame surface density depending on whether a [RANS](#) or [LES](#) turbulence model is activated. This section describes the transport equation used for RANS simulations. The [ECFM for LES](#) section describes the transport equation used for LES.

Tracking Flame Propagation

The flame surface density (Σ) is determined by the following transport equation:

$$\frac{\partial \Sigma}{\partial t} + \frac{\partial u_i \Sigma}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\frac{\mu}{Sc} \frac{\partial (\Sigma / \bar{\rho})}{\partial x_i} \right) + (At_{sgs} + At_{res} + Curv_{sgs}) \Sigma - D + P_k, \quad (16.61)$$

where μ is the molecular viscosity, Sc is the Schmidt number, $At_{sgs} = \alpha K_t$ is the flame surface production by [turbulent stretch](#), $At_{res} = \frac{2}{3} \frac{\partial \tilde{u}_i}{\partial x_i}$ is the production by the mean flow dilatation, $Curv_{sgs} = \frac{2}{3} \frac{\rho_u}{\rho_b} s_l \frac{1 - \tilde{c}_\Sigma}{\tilde{c}_\Sigma} \Sigma$ models the effects of the flame thermal expansion and curvature, $D = \beta s_l \frac{\Sigma^2}{1 - \bar{c}}$ is destruction due to consumption, P_k is the source term (such as a spark plug), ρ_u is the density of unburned gases, ρ_b is the density of burned gases, s_l is the [laminar flamespeed](#), \tilde{c} is the mass progress variable, and \bar{c} is the volume progress variable. Specify the values of α in At_{sgs} and β in D via [combust.in > ecfm_model > stretch_alpha](#) and [ecfm_model > destruct_beta](#), respectively.

Figure 16.11 shows a schematic of the flame surface density (FSD) progress variable \tilde{c}_Σ .

Chapter 16: Combustion Modeling

Extended Coherent Flame Models

Extended Coherent Flame Model (ECFM)

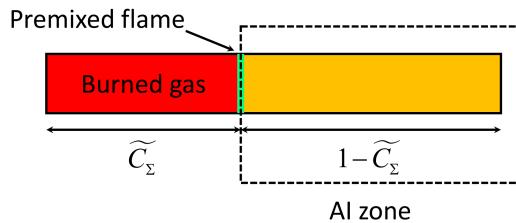


Figure 16.11: Real bimodal decomposition of ECFM-SB.

With the species-based ECFM, the FSD and autoignition progress variables are linked based on two assumptions: that the premixed flame separates fresh autoignition gases and burned gases and that the cell mass fraction of species is given by

$$\tilde{Y}_i = (1 - \tilde{c}_\Sigma) Y_i|^{ai} + \tilde{c}_\Sigma Y_i|^b. \quad (16.62)$$

The fresh autoignition gas $Y_i|^{ai}$ is treated in a bimodal way (in reality, a homogeneous zone), as follows:

$$Y_i|^{ai} = (1 - \tilde{c}_{ai}) Y_i|^{u} + \tilde{c}_{ai} Y_i|^b, \quad (16.63)$$

where the superscripts u and b represent the unburned and burned zones, respectively. Figure 16.12 below shows the model bimodal decomposition.

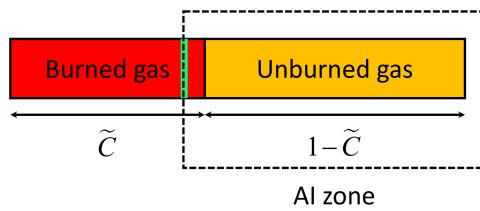


Figure 16.12: Model bimodal decomposition of ECFM-SB.

The mass progress variable is defined as

$$\tilde{c} = \frac{\sum_i \tilde{Y}_i^b}{\sum_i \tilde{Y}_i^u + \sum_i \tilde{Y}_i^b} \quad (16.64)$$

and the burned gas fraction from autoignition \tilde{c}_{ai} is calculated as ([Colin et al., 2018](#))

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$\tilde{c}_{ai} = \frac{\tilde{Y}_{N2}^{b,ai}}{\tilde{Y}_{N2}^{b,ai} + \tilde{Y}_{N2}^{b,\Sigma}} \tilde{c}, \quad (16.65)$$

where $\tilde{Y}_{N2}^{b,ai}$ and $\tilde{Y}_{N2}^{b,\Sigma}$ represent the N2 mass fractions from autoignition and premixed flame oxidation, respectively. The autoignition progress variable c_{ai} is defined as

$$c_{ai} = \frac{\tilde{c}_{ai}}{1 - \tilde{c} + \tilde{c}_{ai}}. \quad (16.66)$$

Finally, the FSD progress variable \tilde{c}_Σ and volume progress variable \bar{c} are calculated as

$$\tilde{c}_\Sigma = \tilde{c} - \tilde{c}_{ai} \quad (16.67)$$

and

$$\bar{c} = \frac{\rho}{\rho_b} \tilde{c}. \quad (16.68)$$

Species source terms are defined in terms of these progress variables. For unburned species, the species source term is evaluated as

$$\tilde{\omega}_{\tilde{Y}_i^u} = -Y_i^u \left[(1 - \tilde{c}_\Sigma) \tilde{\omega}_c^{TKI} + (1 - c_{ai}) (\tilde{\omega}_c^\Sigma + \tilde{\omega}_c^{ISSIM}) \right], \quad (16.69)$$

where Y_i^u , the conditional species mass fraction, is calculated as

$$Y_i^u = \frac{\tilde{Y}_i^u}{1 - \tilde{c}}. \quad (16.70)$$

The premixed flame propagation source term is calculated as

$$\tilde{\omega}_c^\Sigma = \frac{\rho_u}{\bar{\rho}} S_l \bar{\Sigma}. \quad (16.71)$$

The calculations of the [TKI](#) source term and the [ISSIM](#) source terms are described below.

Burned gas species source terms are evaluated as

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$\tilde{\omega}_{\tilde{Y}_i^b} = Y_i^{b^*} \left[(1 - \tilde{c}_\Sigma) \tilde{\omega}_c^{TKI} + (1 - c_{ai}) (\tilde{\omega}_c^\Sigma + \tilde{\omega}_c^{ISSIM}) \right] + (1 - \tilde{c}) \tilde{\omega}_i^{bg}, \quad (16.72)$$

where $Y_i^{b^*}$ is the burned gas composition given by the two-step ECFM chemistry and $\tilde{\omega}_i^{bg}$ is the burned gas post-oxidation reaction rate. N2 is evaluated with a slightly different expression to account for the difference between autoignition and flame propagation reaction rates,

$$\tilde{\omega}_{Y_{N2}^{b,ai}} = Y_{N2}^{b^*} (1 - \tilde{c}_\Sigma) \tilde{\omega}_c^{TKI} + (1 - \tilde{c}) \frac{\tilde{Y}_{N2}^{b,ai}}{\tilde{Y}_{N2}^{b,ai} + \tilde{Y}_{N2}^{b,\Sigma}} \tilde{\omega}_{N2}^{bg} \quad (16.73)$$

and

$$\tilde{\omega}_{Y_{N2}^{b,\Sigma}} = Y_{N2}^{b^*} (1 - c_{ai}) (\tilde{\omega}_c^\Sigma + \tilde{\omega}_c^{ISSIM}) + (1 - \tilde{c}) \frac{\tilde{Y}_{N2}^{b,\Sigma}}{\tilde{Y}_{N2}^{b,ai} + \tilde{Y}_{N2}^{b,\Sigma}} \tilde{\omega}_{N2}^{bg}, \quad (16.74)$$

where $Y_{N2}^{b^*}$ is the conditional N2 mass fraction in the burned gases.

Turbulent Stretch Model

The Intermittent Turbulent Net Flame Stretch (ITNFS) model shows how intermittent turbulence affects the distribution of stretch along the flame front. The ITNFS model is a library of the net flame stretch constructed by [Meneveau and Poinsot \(1991\)](#). In the model, turbulent stretch (K_t) is given as

$$\frac{K_t}{\varepsilon / k} = ITNFS_{factor} \cdot \Gamma_k \left(\frac{u'}{s_l}, \frac{L}{\delta_l} \right), \quad (16.75)$$

where ε is turbulent dissipation, k is turbulent kinetic energy, $u' = \sqrt{2k/3}$ is turbulence

RMS velocity, s_l is the [laminar flamespeed](#), L is integral length scale, δ_l is the laminar flame thickness calculated as described in [Blint \(1986\)](#), and Γ_k is the net stretch ratio function, which is typically evaluated as an integration of the efficiency function, C , as shown in [Meneveau and Poinsot \(1991\)](#). In CONVERGE, there are seven choices for modeling the turbulence stretch ratio functions, Γ_k .

When `combust.in > ecfm_model > itnfs_model = 0`, the stretch ratio is evaluated using a fitted equation instead of integrating the efficiency function, as shown in Meneveau and Poinsot (1991). The net stretch ratio is given by $\Gamma_{k1} - \Gamma_{k2}$ where

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$\Gamma_{k1} = 10^{r\left(s, \frac{u'}{s_l}\right)}, \quad (16.76)$$

where

$$r\left(s, \frac{u'}{s_l}\right) = -\frac{1}{s+0.4} \exp(-[s+0.4]) + (1-\exp(-[s+0.4])) \left(\sigma_1\left(\frac{u'}{s_l}\right) s - 0.11 \right), \quad (16.77)$$

$$s = \log_{10}\left(\frac{L}{\delta_l}\right), \quad (16.78)$$

and

$$\sigma_1\left(\frac{u'}{s_l}\right) = \frac{2}{3} \left(1 - \frac{1}{2} \exp\left(\frac{u'}{s_l}\right)^{\frac{1}{3}} \right), \quad (16.79)$$

and where

$$\Gamma_{k2} = \frac{3}{2} \left(\frac{L}{\delta_l} \right) \left(\frac{u'}{s_l} \right)^{-1} \ln\left(\frac{1}{1-P_q}\right), \quad (16.80)$$

where

$$P_q = \frac{1}{2} [1 + \tanh(\operatorname{sgn}[x] x^2)], \quad (16.81)$$

$$x = \frac{\log_{10}\left(\frac{u'}{s_l}\right) - g\left(\frac{L}{\delta_l}\right)}{\sigma\left(\frac{L}{\delta_l}\right)}, \quad (16.82)$$

$$g\left(\frac{L}{\delta_l}\right) = \left(0.7 + \frac{1}{s}\right) \exp(-s) + (1 - \exp(-s))(1 + 0.36s), \quad (16.83)$$

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

and

$$\sigma \left(\frac{L}{\delta_l} \right) = 0.04 \log_{10} \left(\frac{L}{\delta_l} \right). \quad (16.84)$$

When $\text{combust.in} > \text{ecfm_model} > \text{itnfs_model} = 1$ through 5, the net stretch ratio, Γ_k , is evaluated as an integration of the efficiency function, C . For $\text{combust.in} > \text{ecfm_model} > \text{itnfs_model} = 1$, the integration formula is given by [Meneveau and Poinsot \(1991\)](#) as

$$\Gamma_k = \frac{c_{ms}}{\ln 2} \int_{scales} C_{MP} \left(\frac{r}{\delta_l} \right) \cdot \left(\frac{L}{r} \right)^{\left(\frac{2}{3} - \frac{\mu}{9} \right)} d \left[\ln \left(\frac{L}{r} \right) \right], \quad (16.85)$$

where $c_{ms} = 0.28$, $\mu = 0.26$. The integration is performed in the r/L space. The efficiency function is

$$C_{MP} \left(\frac{r}{\delta_l} \right) = 10^{-c(s)}, \quad (16.86)$$

where

$$c(s) = \frac{0.545}{s + 0.364} \quad (16.87)$$

and

$$s = \log_{10} \left(\frac{r}{\delta_l} \right). \quad (16.88)$$

The subscript MP indicates the efficiency function formula is from the research by [Meneveau and Poinsot \(1991\)](#).

While $\text{combust.in} > \text{ecfm_model} > \text{itnfs_model} = 2$, the integration is based on the $p = \ln(r/L)$ space, and the integrated form becomes

$$\Gamma_k = \frac{c_{ms}}{\ln 2} \int_0^{p_{\max}} C_{MP} \left(\frac{r}{\delta_l} \right) \exp \left(p \left[\frac{2}{3} - \frac{\mu}{9} \right] \right) dp, \quad (16.89)$$

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

where the upper limit of integration, $p_{\max} = \frac{3}{4} \ln(Re)$, corresponds to the smallest scale that the eddies can stretch the flame.

When $\text{combust.in} > \text{ecfm_model} > \text{itnfs_model} = 3, 4$, or 5 , Equation 16.89 is used with a different efficiency function, C , proposed by different groups. When $\text{combust.in} > \text{ecfm_model} > \text{itnfs_model} = 3$, the efficiency function ([Charlette et al., 2002](#)) is

$$C_{CMV} = C_{CDVP} \cdot C_V, \quad (16.90)$$

where

$$C_{CDVP}\left(\frac{r}{\delta_l}, \frac{u'}{s_l}\right) = \frac{1}{2} \left[1 + \operatorname{erf}\left(0.6 \ln\left(\frac{r}{\delta_l}\right) - \left(\frac{u'}{s_l}\right)^{-\frac{1}{2}} \right) \right] \quad (16.91)$$

and

$$C_V = \frac{1}{2} \left[1 + \operatorname{erf}\left(3 \log\left(\frac{2u'}{s_l}\right) \right) \right]. \quad (16.92)$$

In this model, the correct factor C_V ensures that the very slow eddies do not stretch the flames.

When $\text{combust.in} > \text{ecfm_model} > \text{itnfs_model} = 4$, the efficiency function ([Bougrine et al., 2014](#)) is

$$C_{Bougrine} = \frac{1 + \operatorname{erf}\left(0.9 \ln\left(\frac{r}{\delta_l}\right) - 2 \right)}{1 + 0.3 \left(\frac{u'}{s_l} \right) \cdot \left(1 + \operatorname{erf}\left(0.9 \ln\left(\frac{r}{\delta_l}\right) - 2 \right) \right)} \cdot \left[\frac{1}{Le} (1.76 + \tanh(Le - 2)) \right], \quad (16.93)$$

where Le is the Lewis number.

When $\text{combust.in} > \text{ecfm_model} > \text{itnfs_model} = 5$, the Suillaud efficiency function is used, which is defined as

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$C_{Suillard} = \frac{1 + \operatorname{erf} \left(0.8 \ln \left(\frac{r}{\delta_l} \right) - \frac{0.83}{\sqrt{u'/s_l}} \right)}{1 + 0.3 \left(\frac{u'}{s_l} \right) \cdot \left(1 + \operatorname{erf} \left(0.8 \ln \left(\frac{r}{\delta_l} \right) - \frac{0.83}{\sqrt{u'/s_l}} \right) \right)} \cdot \left[\frac{1}{Le} (1.76 + \tanh(Le - 2)) \right]. \quad (16.94)$$

Finally, when `combust.in > ecfm_model > itnfs_model = 6`, a simple flame stretch model proposed by [Cant and Bray \(1989\)](#) is used:

$$K_t = \sqrt{\frac{\varepsilon}{\nu}}, \quad (16.95)$$

where ε is the dissipation of turbulent kinetic energy and ν is the kinematic viscosity.

Correction Factors

To account for the decrease in turbulent strain near a WALL boundary, the At_{sgs} and $Curv_{sgs}$ terms in the [FSD equation](#) are multiplied by the wall stretch correction factor C_w , given by

$$C_w = \begin{cases} 0 & \text{for } l_{wall} < l_{buffer} \\ \frac{l_{wall} - l_{buffer}}{l_{wall}} & \text{for } l_{wall} \geq l_{buffer}, \end{cases} \quad (16.96)$$

where l_{wall} is the distance to the wall and the length scale l_{buffer} is given by

$$l_{buffer} = \frac{57.5\mu}{(0.09)^{1/4} \rho \sqrt{k}}. \quad (16.97)$$

The At_{sgs} and $Curv_{sgs}$ terms are also multiplied by the time stretch correction factor of [Bruneaux et al. \(1997\)](#), which accounts for the reduction in turbulent strain that occurs when the strain time scale becomes close to the flame time scale. This correction factor is given by

$$C_t = \frac{1}{1 + \tau_f (At_{sgs} + Curv_{sgs})}, \quad (16.98)$$

where the strain time scale is $1/(At_{sgs} + Curv_{sgs})$ and the flame time scale is defined as

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$\tau_f = \frac{\delta_l^0}{Z_e s_l^0}, \quad (16.99)$$

where δ_l^0 is the unstrained flame thickness and s_l^0 is the unstrained flamespeed. The factor Z_e is given by

$$Z_e = (7500) \frac{T_b - T_u}{T_b^2}, \quad (16.100)$$

where T_b and T_u are the temperatures of the burned and unburned gases, respectively. For most applications, the flame time scale is small compared to the strain time scale, such that $C_t \approx 1$. The correction C_t is significantly smaller than unity only for very turbulent flows or for very slow flames.

CONVERGE also applies the flamespeed correction factor described in [Bruneaux et al. \(1997\)](#). To account for heat losses near WALL boundaries, the laminar flamespeed s_l is multiplied by

$$flspeed_cor = \exp[-2Z_e(1-F)], \quad (16.101)$$

where

$$F = (1 - \tilde{c}) + \tilde{c} \frac{\tilde{H} - \tilde{H}_u}{\Delta h}, \quad (16.102)$$

\tilde{H} is the total enthalpy, \tilde{H}_u is the enthalpy of the unburned gases, and Δh is the heat formation from burned species. When there is no heat loss, $F = 1$ and $flspeed_cor = 1$.

Imposed Stretch Spark Ignition Model

The Imposed Stretch Spark Ignition Model (ISSIM) ([Colin and Truffin, 2011](#)) can be coupled with the ECFM. The ISSIM uses the same electrical circuit description as the AKTIM spark ignition model from [Duclos and Colin \(2001\)](#). The ISSIM was first developed for LES and later adapted to RANS.

The main purpose of the ISSIM is to simulate the reaction rate due to the flame surface density (FSD) starting at the moment of ignition. This model simultaneously represents both the electrical circuit energy deposition and the flame surface and mass deposition.

Electric circuit model: Figure 16.13 below shows a simplified electrical diagram of the inductive system for a spark plug.

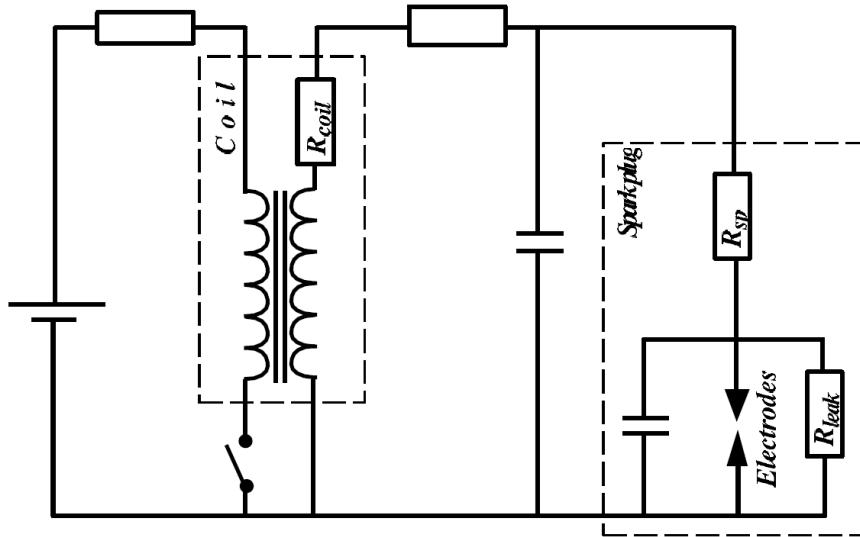


Figure 16.13: The electrical scheme of the inductive system.

The primary circuit includes the battery and the primary inductance. When the switch is open, the energy is stored in the primary inductance. Approximately 60% of the energy is transferred to the spark plug ([Verhoeven, 1997](#)), while the remaining energy is dissipated by the secondary inductance. The ISSIM considers only the secondary part of the inductive ignition system. The amount of energy transferred to the secondary circuit at the beginning of the simulation is specified by `issim.in > ignitions > ignition > initial_sec_energy`.

The life of the spark is generally divided into three phases. The breakdown and arc phases, which last typically less than a microsecond, are taken into account by considering an instantaneous energy deposit in the gas, E_{bd} , which is given as ([Duclos and Colin, 2001](#))

$$E_{bd} = \frac{V_{bd}^2}{C_{bd}^2 d_{ie}}, \quad (16.103)$$

where V_{bd} is the breakdown voltage (approximated by Paschen's law [[Reinmann, 1998](#)]), C_{bd} is a constant (1.5e6), and d_{ie} is the inter-electrode distance specified via `issim.in > spark_plugs > spark_plug > electrode_distance`.

The glow phase, which can last a few milliseconds, is modeled by solving the electrical circuit ODEs. During this phase, the spark voltage V_{ie} is equal to

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$V_{ie}(t) = V_{cf} + V_{af} + V_{gc}, \quad (16.104)$$

where V_{cf} is the cathode fall voltage, V_{af} is the anode fall voltage, and V_{gc} is the gas column voltage. The gas column voltage ([Kim and Anderson, 1995](#)) is given by

$$V_{gc} = 40.46 l_{spk} i_s^{-0.32} p^{0.51}, \quad (16.105)$$

where l_{spk} is the spark length, i_s is the current, and p is the pressure. The power transferred to the gas phase is given by

$$\text{power_transferred_to_gas} = V_{gc} i_s \exp\left(-\frac{0.5 \times \text{electrode_diameter}}{l_{spk}}\right) \quad (16.106)$$

where *electrode_diameter* is the electrode diameter given in [*issim.in*](#). Equation 16.106 is obtained by fitting experimental data from IFP Energies nouvelles.

Kim and Anderson suggest that the cathode fall voltage is 7.6 V during the arc phase and the anode fall voltage is 252 V during the glow phase. The current is given by

$$i_s = \sqrt{\frac{2E_s}{L_s}}, \quad (16.107)$$

where the energy E_s and the inductance L_s are user-specified parameters. The electrical energy available on the secondary circuit is given by

$$\frac{\partial E_s(t)}{\partial t} = -R_s i_s^2(t) - V_{ie} i_s(t). \quad (16.108)$$

The glow phase lasts as long as $E_s(t)$ is positive.

When the spark voltage reaches the breakdown voltage (*i.e.*, when $V_{ie} > V_{bd}$), the gas breaks down and becomes conductive. At breakdown, the spark length is initially equal to the user-defined spark gap ([*issim.in*](#) > *spark_plugs* > *spark_plug* > *electrode_distance*), as follows:

$$l_{spk}(t=0) = d_{ie}. \quad (16.109)$$

Then the spark is stretched by convection and turbulent motion of the flow. The total spark length is given by

Chapter 16: Combustion Modeling

Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$l_{spk} = l_{spk,mean} \Xi_{spk}, \quad (16.110)$$

where the mean length is given by

$$\frac{dl_{spk,mean}}{dt} = 2u_{conv}, \quad (16.111)$$

in which u_{conv} is the mean convection, and the spark wrinkling is given by

$$\frac{d\Xi_{spk}}{dt} = K_t, \quad (16.112)$$

where K_t considers the turbulence wrinkling effect, which is similar to the ITNFS function in the flame surface density equation.

The total energy received by the gas, E_{ign} , is the sum of the energy received at breakdown (E_{bd}) and the energy received during the glow phase and can be expressed as

$$E_{ign} = 0.6E_{bd} + \int_{t_{spk}}^t V_{gc} i_s dt. \quad (16.113)$$

If E_{ign} is larger than the critical energy E_{crit} , ignition will occur. The critical energy ([Adelman, 1981](#)) is given by

$$E_{crit} = 4 \frac{\gamma}{\gamma-1} l_{spk} p \pi \delta_L^2, \quad (16.114)$$

where δ_L is the local flame thickness and γ is the gas heat capacity ratio.

Initial burned gas kernel phase: When ignition begins, a mass of burning gases, $m_{bg,ign}$, is deposited near the spark. In the ISSIM implementation in CONVERGE, based on experimental observation that the initial kernel size does not depend on the mixture composition but rather scales with the deposited electrical energy, the following formulation is used:

$$m_{bg,ign} = C_{bg} \left\langle \frac{E_{ign}}{C_p \Delta T_{ign}} \right\rangle, \quad (16.115)$$

where ΔT_{ign} is a characteristic temperature of the plasma in the early instants. As this temperature rapidly decreases after breakdown, it is difficult to define it with accuracy. A value of 20,000 K is chosen arbitrarily. The user-specified correction factor C_{bg} ([issim.in > general_control > c_ignition_mass](#)) allows to correct this initially deposited mass. The brackets denote an average in the cells within a sphere of radius of twice the spark gap.

The target burned gas volume fraction is given by

$$\bar{c}_{ign}(x, t_{ign}) = c_0 \exp\left(-\left(\frac{|x - x_{spk}|}{0.5d_{ie}}\right)^2\right), \quad (16.116)$$

where x gives the cell coordinates, x_{spk} gives the spark plug location, d_{ie} is the spark gap length, and c_0 satisfies

$$\int \rho_b \bar{c}_{ign} dV = m_{bg}^{ign}. \quad (16.117)$$

In order to impose the volume fraction on the 3D CFD domain, CONVERGE calculates the reaction rate of the progress variable from

$$\dot{\bar{c}}_c = \max\left(\rho_u s_l \bar{\Sigma}_{\bar{c}}, \rho_b (\bar{c}_{ign} - \bar{c}_{\Sigma}) dt^{-1}\right). \quad (16.118)$$

The ignition flame surface density (FSD) source is introduced at the start of ignition. As suggested by experiments, the initial flame kernel is assumed to be a sphere and its radius $r_{b,ign}$ can be defined as

$$r_{b,ign} = \left(\frac{3}{4\pi} \int \bar{c}_{ign} dV \right)^{1/3}. \quad (16.119)$$

The FSD can be defined as

$$\Sigma_{ign} = C_{surf} \frac{3\bar{c}}{r_b^{ign}}, \quad (16.120)$$

where C_{surf} is a user-specified value ([issim.in > general_control > c_flame_wrinkling](#)) that corresponds to an initial wrinkling value, allowing to account for the non-perfect sphericity of the flame kernel.

The ignition FSD source term is defined as

Chapter 16: Combustion Modeling

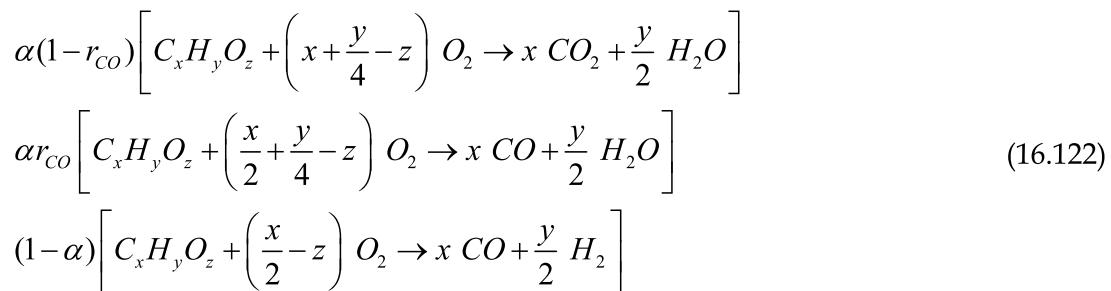
Extended Coherent Flame Models Extended Coherent Flame Model (ECFM)

$$\bar{\omega}_{\Sigma,ign} = \max((\Sigma_{ign} - \bar{\Sigma}_{\tilde{c}})dt^{-1}, 0). \quad (16.121)$$

The source term in Equation 16.121 applies to both [RANS](#) and [LES](#) simulations. For LES simulations, an additional source term is required to model FSD creation at the sub-grid scale. This additional source term is described in the [ECFM for LES](#) section.

Unburned Fuel Oxidation

For a generic hydrocarbon fuel $C_xH_yO_z$, unburned fuel oxidation is given by a global two-stage reaction as ([Colin and Benkenida, 2004](#))



where $r_{CO} = 0.01$ and the value of α is determined based on the average equivalence ratio $\bar{\phi}$:

$$\begin{aligned} \bar{\phi} < 1: & \quad \alpha = 1 \\ 1 \leq \bar{\phi} < \phi_{crit}: & \quad \alpha = \frac{1}{2x+y} \left(\frac{4x+y-2z}{\bar{\phi}} - 2(x-z) \right) \\ \phi_{crit} \leq \bar{\phi}: & \quad \alpha = 0, \end{aligned} \quad (16.123)$$

where

$$\phi_{crit} = \frac{2}{x-z} \left(x + \frac{y}{4} - \frac{z}{2} \right). \quad (16.124)$$

Post-Flame Stage

CONVERGE includes several methods for calculating the post-flame kinetics and post-fuel oxidation. All of these parameters are in [combust.in](#).

The first method follows equilibrium reactions to solve the chemistry as described in [Colin et al. \(2003\)](#). To use this method, set [combust.in](#) > `ecfm_model > post_flame_model = 0` or [combust.in](#) > `ecfm3z_model > post_flame_model = 0`.

Chapter 16: Combustion Modeling

Extended Coherent Flame Models

Extended Coherent Flame Model (ECFM)

The post-flame equilibrium reactions are as follows:



and the equation for CO to CO₂ oxidation is



The second method is to use the [CEQ solver](#) for the post-flame stage. The CEQ solver solves the equilibrium species after the flame front. To use this method, set `combust.in > ecfm_model > post_flame_model = 1` or `combust.in > ecfm3z_model > post_flame_model = 1`. This method is available for comparison purposes but is not recommended for general use.

The third method uses SAGE detailed chemistry for the post-flame region. To use this method, set `combust.in > ecfm_model > post_flame_model = 2` or `combust.in > ecfm3z_model > post_flame_model = 2`. When using this model, set `combust.in > sage_model > active = 0` and configure SAGE settings in the `combust.in > sage_model` settings block. You can activate the adaptive zone model by setting `combust.in > adaptive_zone_model > active = 1`.

3-Zone Extended Coherent Flame Model (ECFM3Z)

For partially- or non-premixed combustion, the [ECFM](#) is coupled with a mixing model and is known as the ECFM3Z (3-Zone Extended Coherent Flame Model) ([Colin and Benkenida, 2004](#)). This method creates a mixed zone between the air (+ exhaust gas recirculation) and fuel zones.

Mixing Model

The mixing of air and fuel (as shown below in Figure 16.14) forms the mixed zone.

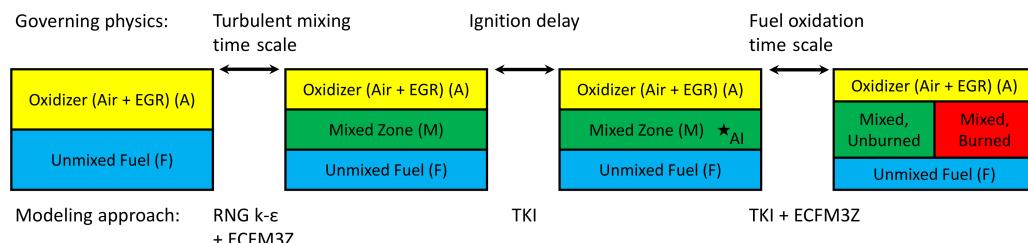


Figure 16.14: Mixing model.

Unmixed fuel and O₂ are solved as

Chapter 16: Combustion Modeling

Extended Coherent Flame Models 3-Zone Extended Coherent Flame Model (ECFM3Z)

$$\frac{\partial \bar{\rho} \tilde{Y}_{Fu}^F}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_i \tilde{Y}_{Fu}^F}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\frac{\mu}{Sc} \frac{\partial \tilde{Y}_{Fu}^F}{\partial x_i} \right) + \bar{\rho} \tilde{S}_{Fu} + \bar{\rho} \bar{E}_{Fu}^{F \rightarrow M} \quad (16.127)$$

and

$$\frac{\partial \bar{\rho} \tilde{Y}_{O2}^A}{\partial t} + \frac{\partial \bar{\rho} \tilde{u}_i \tilde{Y}_{O2}^A}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\frac{\mu}{Sc} \frac{\partial \tilde{Y}_{O2}^A}{\partial x_i} \right) + \bar{\rho} \bar{E}_{O2}^{A \rightarrow M}. \quad (16.128)$$

Mixing terms $\bar{E}_{Fu}^{F \rightarrow M}$ and $\bar{E}_{O2}^{A \rightarrow M}$ are a function of the turbulence time scale, which is given as follows:

$$\frac{1}{\tau_m} = \beta_m \frac{\varepsilon}{k}, \quad (16.129)$$

where β_m is the mixing rate constant. This value is specified by [*combust.in > ecfm3z_model > mix_betam*](#). The evaporation source term \tilde{S}_{Fu} in the unmixed fuel of Equation 16.127 is the gaseous fuel mass production rate already presented for the unburned and burned fuel equations.

In the species-based ECFM3Z model, multiple fuels are allowed in the simulation. Each fuel species must be set up as described in the [ECFM/ECFM3Z Setup](#) section.

Autoignition model

Tabulated Kinetics of Ignition ([*combust.in > ecfm_model > auto_ignition > active = 1*](#) or [*combust.in > ecfm3z_model > auto_ignition > active = 1*](#))

The tabulated kinetics of ignition (TKI) model was first proposed by [Colin et al. \(2005\)](#). In this first version autoignition was described in two steps. In the first step, an autoignition delay was used in the intermediate species transport equation $Y_{F,j}$ to reach the autoignition time. Once the delay was reached, the progress of reaction was advanced in time using the fuel transport equation and a tabulated progress variable reaction rate.

The second version of the TKI model ([Robert et al., 2015](#)) did not use intermediate species. Instead, the entire autoignition process was described by the fuel mass fraction burned by autoignition $Y_{F,ai}$.

The third version of the TKI model ([Colin et al., 2018](#)) is implemented in CONVERGE. This version describes autoignition based on the fraction \tilde{c}_{ai} of burned products that are formed by autoignition. This fraction is deduced from the fraction of N2 formed by autoignition in the burned zone (Equation 16.65).

Chapter 16: Combustion Modeling

Extended Coherent Flame Models 3-Zone Extended Coherent Flame Model (ECFM3Z)

In order to tabulate the autoignition reaction rate, the TKI model is designed to solve the 0D constant pressure reactors at different initial conditions. These initial conditions reproduce the in-cylinder conditions that occur in the engine. This method uses four parameters: pressure, fresh gas temperature, equivalence ratio, and dilution rate. For each parameter, you can select a range and discretization fidelity to match the local sensitivity of chemistry to the parameter. Table 16.3 shows an example of the TKI parameter variation and discretization.

Table 16.3: Example of the TKI parameter variation and discretization.

Parameter	Range	Number of points
Fresh gas temperature	500 - 1500 K	50
Pressure	1 - 20 MPa	10
Fuel/air equivalence ratio	0.1 - 3.0	10
EGR fraction to air	0 - 80%	4

During tabulation, CONVERGE extracts the time t_i at which the 0D calculation reaches the progress of reaction c_{ai} :

The progress variable in the TKI model is an image of the amount of heat released during fuel oxidation, as follows:

$$c_{ai} = \frac{T - T_{ini}}{T_{eq} - T_{ini}}, \quad (16.130)$$

where T_{ini} is the initial temperature and T_{eq} is the final temperature of the 0D simulation. The progress variable reaction rate is defined as

$$\dot{\omega}_{c,TKI} = \frac{dc_{ai}}{dt}. \quad (16.131)$$

This ordinary differential equation is approximated as follows:

$$0 < c < c_1 : \dot{\omega}_{c,TKI} = \frac{c_1}{t_1} \quad (16.132)$$
$$c_1 < c < c_{i+1} (\text{for } i \geq 1) : \dot{\omega}_{c,TKI} = \frac{c_{i+1} - c_i}{t_{i+1} - t_i}$$

Note that t_1 can be viewed as the so-called ignition delay of TKI, although the progress of the reaction at this time, c_1 , needs to be set as a very small number (typically 1e-6) to ensure a proper description of the heat release rate during the entire autoignition process, and thus t_1 cannot be compared to a standard autoignition delay, which is typically chosen at c close to 0.5. In the second approximation in Equation 16.132, the progress variable reaction rate is constant between t_{i+1} and t_i . This choice was made in [Colin et al. \(2005\)](#) because it proved to be more accurate than a linear approximation.

The autoignition (TKI) table (*tki_table.h5*) can be generated with the [zero-dimensional chemistry utility](#). This utility tabulates the ignition delay and reaction rates in HDF5 format and can be viewed using utilities provided by The HDF Group, such as HDFView. To generate a TKI table, set `zero_d_solver.in > zero_d_output_control > generate_tki_table_flag = 1` and use `zero_d_template.in` to configure the TKI table values and ranges. To use a TKI table, set `combust.in > ecfm_model > auto_ignition > active = 1` (if using the ECFM model) or `combust.in > ecfm3z_model > auto_ignition > active = 1` (if using the ECFM3Z model), and then specify the TKI file name in `ecfm_model > auto_ignition > tki_table_filename` or `ecfm3z_model > auto_ignition > tki_table_filename`.

ECFM for LES

Some details of the ECFM implementation are different for [LES](#) and [RANS](#) simulations. For LES, CONVERGE solves the FSD transport equation proposed in [Richard et al., 2007](#) and calculates ISSIM source terms as described below. Progress variables, species reaction rates, and the [TKI source term](#) are calculated in the same way as in RANS simulations.

The FSD transport equation for LES is given by

$$\frac{\partial \bar{\Sigma}}{\partial t} + \frac{\partial}{\partial x_i} (\tilde{u}_i \bar{\Sigma} + P_{res}) = T_{sgs} + S_{sgs} + C_{sgs} + C_{res} + S_{res}, \quad (16.133)$$

where $\bar{\Sigma}$ is the LES-filtered flame surface density and P , T , S , and C represent contributions from propagation, transport, strain, and curvature, respectively. The resolved propagation and curvature terms are based on the normal to the iso-surface of the filtered progress variable, defined as

$$N_i = -\frac{\partial \tilde{c}/\partial x_i}{\sqrt{(\partial \tilde{c}/\partial x_j)(\partial \tilde{c}/\partial x_j)}}. \quad (16.134)$$

The resolved propagation term is given by

$$P_{res} = S_d N_i \bar{\Sigma}, \quad (16.135)$$

Chapter 16: Combustion Modeling

Extended Coherent Flame Models ECFM for LES

where

$$S_d = (1 + \tau \tilde{c}) s_l, \quad (16.136)$$

s_l is the [laminar flamespeed](#), and

$$\tau = \frac{\rho_u}{\rho_b} - 1 \quad (16.137)$$

is the thermal expansion ratio across the flame front. The resolved curvature term is given by

$$C_{res} = S_d \left(\frac{\partial N_i}{\partial x_i} \right) \bar{\Sigma}, \quad (16.138)$$

while the unresolved curvature term is given by

$$C_{sgs} = \left[\frac{2(1 + \tau)}{3\bar{c}} - \frac{1}{1 - \bar{c}} \right] s_l \bar{\Sigma} (\bar{\Sigma} - \Sigma_{lam}), \quad (16.139)$$

where

$$\Sigma_{lam} = \sqrt{\left(\frac{\partial \tilde{c}}{\partial x_i} \right) \left(\frac{\partial \tilde{c}}{\partial x_i} \right) + (\bar{c} - \tilde{c}) \frac{\partial N_i}{\partial x_i}}. \quad (16.140)$$

The unresolved transport term is defined as

$$T_{sgs} = \frac{\partial}{\partial x_i} \left(\mu_t \frac{\partial \bar{\Sigma}}{\partial x_i} \right), \quad (16.141)$$

where μ_t is the turbulent viscosity.

For the resolved strain,

$$S_{res} = A_T \bar{\Sigma}, \quad (16.142)$$

CONVERGE uses the Hawkes model by default, given by

Chapter 16: Combustion Modeling

Extended Coherent Flame Models ECFM for LES

$$A_T = \frac{\partial \tilde{u}_i}{\partial x_i} - K_{ij} \frac{\partial \tilde{u}_i}{\partial x_j}, \quad (16.143)$$

where

$$K_{ij} = \delta_{ij} - \left[\frac{N_i N_j}{\Xi_{sgs}^2} + \delta_{ij} \left(1 - \frac{1}{\Xi_{sgs}^2} \right) \right] \quad (16.144)$$

and the sub-grid-scale wrinkling factor Ξ_{sgs} is defined as

$$\Xi_{sgs} = \frac{\bar{\Sigma}}{\Sigma_{lam}}. \quad (16.145)$$

The unresolved strain is modeled as

$$S_{sgs} = a_T \bar{\Sigma}, \quad (16.146)$$

where

$$a_T = \Gamma_k \left(\frac{\hat{u}'}{s_l}, \frac{\hat{\Delta}}{\delta_l} \right) \frac{\hat{u}'}{\hat{\Delta}}. \quad (16.147)$$

The efficiency function Γ_k is based on the choice of [ITNFS model](#) specified in [combust.in > ecfm_model > itnfs_model](#). The length scale $\hat{\Delta}$ represents the combustion filter size, which is related to the grid filter Δ by ([Colin et al., 2000](#))

$$\hat{\Delta} = n_{res} \Delta \quad (16.148)$$

and \hat{u}' is the turbulent velocity fluctuation at scale $\hat{\Delta}$. CONVERGE sets $n_{res} = 8$ and calculates \hat{u}' using the same method that is specified for the thickened flame model ([combust.in > thickened_flame_model > efficiency_factor_model > uprime_model](#)).

When the [spark ignition model](#) is activated ([combust.in > ecfm_model > spark > active = 1](#)), CONVERGE uses a modified version of the FSD equation to account for FSD creation by the flame kernel during spark ignition. As proposed by [Colin and Truffin, 2011](#), the modified equation is

Chapter 16: Combustion Modeling

Extended Coherent Flame Models ECFM for LES

$$\frac{\partial \bar{\Sigma}}{\partial t} + \frac{\partial}{\partial x_i} \left[\tilde{u}_i \bar{\Sigma} + (1-\alpha) P_{res} \right] = T_{sgs} + S_{sgs} + C_{sgs} + (1-\alpha)(C_{res} + S_{res}) \\ + \alpha S_{ign} + \bar{\omega}_{\Sigma, ign}. \quad (16.149)$$

The source term $\bar{\omega}_{\Sigma, ign}$, as described in an [earlier section](#), is introduced at the time of ignition. An additional source term, S_{ign} , accounts for FSD creation at the sub-grid scale during the few milliseconds after ignition when the flame kernel radius r_b is smaller than $\hat{\Delta}$. For this term, CONVERGE uses the expression derived in [Robert et al., 2015](#), given by

$$S_{ign} = \frac{2}{3} \left(\frac{1+\tau}{\bar{c}_\Sigma} \right) S_l \bar{\Sigma}. \quad (16.150)$$

When the transition factor α is close to 1, the contributions from resolved strain and curvature in Equation 16.149 are suppressed and replaced with S_{ign} . CONVERGE computes α from

$$\alpha = 0.5 \left[1 - \tanh \left(\frac{r_b^+ - 1}{0.15} \right) \right], \quad (16.151)$$

where

$$r_b^+ = \frac{r_b}{\hat{\Delta}}. \quad (16.152)$$

The model for r_b proposed in [Colin and Truffin, 2011](#), requires the solution of an additional transport equation for the passive Ψ , which is related to r_b via

$$\bar{\rho} \Psi = \bar{\rho} \tilde{c}_\Sigma (2 r_b^{-1}) = \frac{2}{3} \rho_b \Sigma_{ign}. \quad (16.153)$$

CONVERGE uses the transport equation for Ψ proposed in [Robert et al., 2015](#), which can be written as

$$\bar{\rho} \frac{\partial \Psi}{\partial t} = - \frac{\partial}{\partial x_i} \left(\bar{\rho} \tilde{u}_i \Psi \right) + \frac{\partial}{\partial x_i} \left(\mu_t \frac{\partial \Psi}{\partial x_i} \right) - \frac{1}{2} S_{ign} \Psi + 2 r_b^{-1} \dot{\omega}_c^\Sigma + S_{\Psi, ign}, \quad (16.154)$$

Chapter 16: Combustion Modeling

Extended Coherent Flame Models ECMF for LES

where

$$\dot{\omega}_c^\Sigma = \max \left[\rho_u s_l \bar{\Sigma}_{\tilde{c}}, \rho_b (\bar{c}_{ign} - \bar{c}_\Sigma) dt^{-1} \right] \quad (16.155)$$

and

$$S_{\Psi,ign} = \max \left[0, \rho_b (\bar{c}_{target} - \bar{c}_\Sigma) \frac{2}{r_b^{ign} dt} \right]. \quad (16.156)$$

ECFM/ECFM3Z Setup

Required input files for ECFM/ECFM3Z are listed below. You can use ECFM/ECFM3Z for single- or dual-fuel simulations. When running a simulation with ECFM/ECFM3Z, the reaction mechanism file must be removed from the Case Directory unless you are running ECFM/ECFM3Z with SAGE.

combust.in

The [combust.in](#) file is required for all ECFM/ECFM3Z simulations. Relevant settings are in the *ecfm_model* or *ecfm3z_model* settings block.

issim.in

The [issim.in](#) file is required if *combust.in* > *ecfm_model* > *spark* > *active* = 1.

ecfm3z_reinit.in

The [ecfm3z_reinit.in](#) file is required if *combust.in* > *ecfm_model* > *reinit_flag* = 1 or *ecfm3z_model* > *reinit_flag* = 1. You can use this file to reinitialize the combustion domain in specific regions at specific times. For example, for a multi-cycle or multi-cylinder engine case, the combustion regions must be reinitialized for the next spark or autoignition event to occur.

parcel*.in and liquid.dat

The [parcels.in](#), [parcel_introduction.in](#), [parcel_parcel_interaction.in](#), [parcel_wall_interaction.in](#), and [liquid.dat](#) files are required only for ECFM3Z.

species.in and post.in

Table 16.4 below lists the gases that must be included in [species.in](#). In CONVERGE Studio, click Create required species (*Case Setup* > *Physical Models* > *Combustion modeling* > *Models* > ECFM/ECFM3Z) to add these species automatically.

Table 16.5 lists the internal passives for ECFM/ECFM3Z. CONVERGE declares internal passives automatically, so you do not need to list them in [species.in](#). If desired, you can specify them in [post.in](#) for visualization purposes.

Table 16.6 lists optional variables that may be useful for visualization. These variables are defined, stored, and transported only if you specify them in [post.in](#).

Thermodynamic data file, *initialize.in*, and *boundary.in*

In *initialize.in* and *boundary.in*, you must specify the initial gas species or the inflow and outflow gas species. For species tracers, you must initialize passives ($TF_{<\text{fuel name}>}$, T_{CO} , T_{O2} , T_{CO2} , T_{N2} , T_{H2O} , and T_{H2}) with the same names and values as their corresponding species (mixed fuel, CO, O₂, CO₂, N₂, H₂O, and H₂, respectively). Do not list the intermediate species (O, H, N, and OH) in *initialize.in* and *boundary.in*.

For gasoline internal combustion engine simulations, follow these general setup rules.

- Intake: Set the $U_{<\text{species name}>}$ and $UF_{<\text{fuel name}>}$ to the unburned gas composition of the intake. Set all of the $<\text{species name}>$ to 0. Set all $T_{<\text{species name}>} = U_{<\text{species name}>}$ and $TF_{<\text{fuel name}>} = UF_{<\text{fuel name}>}$.
- Exhaust: Set the $<\text{species name}>$ and the $<\text{fuel name}>$ to the burned gas composition of the mean engine fuel/air ratio. Set all $U_{<\text{species name}>} = 0$. Set all $T_{<\text{species name}>} = 0$ and $TF_{<\text{fuel name}>} = 0$.

You need to make two copies of the thermodynamic properties of fuel species in the [thermodynamic data file](#) and rename the fuel species as $UF_{<\text{fuel name}>}$ and $UMF_{<\text{fuel name}>}$. Also, you need to copy the species properties (O₂, N₂, CO₂, H₂, and H₂O) and rename the species as $U_{<\text{species name}>}$ and $UNMIX_{<\text{species name}>}$. Finally, copy the N₂ properties and rename as AI_{N2} . Note that CONVERGE Studio completes this task automatically when you click [Create required species/passives](#) (*Case Setup > Physical Models > Combustion modeling > Models > ECFM/ECFM3Z*).

Table 16.4: Species required in *species.in* for ECFM/ECFM3Z (example with C3H8 as fuel species).

O2	Species in the mixed zone.
N2	
CO2	
H2O	
H	
H2	
O	
N	
OH	
CO	
C3H8	Fuel in the burned zone.
U_{O2}	Species in the unburned zone.
U_{N2}	
U_{CO2}	

Chapter 16: Combustion Modeling

Extended Coherent Flame Models ECFM/ECFM3Z Setup

<i>U_CO</i>	
<i>U_H2</i>	
<i>U_H2O</i>	
<i>UF_C3H8</i>	Fuel in the unburned zone.
<i>UNMIX_O2</i>	Species in the unmixed zone.
<i>UNMIX_N2</i>	
<i>UNMIX_CO2</i>	
<i>UNMIX_CO</i>	
<i>UNMIX_H2</i>	
<i>UNMIX_H2O</i>	
<i>UMF_C3H8</i>	Fuel in the unmixed zone.
<i>AI_N2</i>	N2 in burned gas species (used for autoignition progress variable).

Table 16.5: Available internal passives for ECFM/ECFM3Z. (Do not include in *species.in*. Add to *post.in* if desired.)

<i>AI_PROG_VAR</i>	Progress variable for autoignition (c_{ai}).
<i>AI_T_EGR_MASS_RATIO</i>	EGR mass ratio in autoignition zone.
<i>AI_T_EGR_VOL_RATIO</i>	EGR volume ratio in autoignition zone (used as input for TKI table).
<i>AI_T_EQUIV_RATIO</i>	Fuel/air equivalence ratio in autoignition zone (used as input for TKI table).
<i>AI_T_FUEL_MW</i>	Molecular weight of mean fuel in autoignition zone.
<i>AI_T_FUEL_NUM_C</i>	Number of carbon atoms of mean fuel in autoignition zone.
<i>AI_T_FUEL_NUM_H</i>	Number of hydrogen atoms of mean fuel in autoignition zone.
<i>AI_T_FUEL_NUM_O</i>	Number of oxygen atoms of mean fuel in autoignition zone.
<i>AT_SGS</i>	Unresolved turbulent strain ($At_{sgs}\Sigma$).
<i>AT_RES</i>	Resolved turbulent strain ($At_{res}\Sigma$).
<i>CURV_SGS</i>	Unresolved stretch by curvature ($Curv_{sgs}\Sigma$).
<i>CURV_RES</i>	Resolved stretch by curvature (available only for LES).
<i>BURNED_DEN</i>	Density of burned gases.
<i>BURNED_EQUIV_RATIO</i>	Equivalence ratio of burned gases.
<i>BURNED_SIE</i>	Sensible enthalpy of burned gases.

Chapter 16: Combustion Modeling

Extended Coherent Flame Models

ECFM/ECFM3Z Setup

BURNED_TEMP	Temperature of burned gases.
CHEM_SRC	Total heat release rate.
DENSITY_MIX_CD_M	Mean density in the mixed zone.
DIVN	$n = -\frac{\nabla \bar{c}}{ \nabla \bar{c} }$ (available only for LES). Divergence of flame normal vector defined by
E3Z_DENSITY_UNMIX	Mean density of unmixed zones (including fuel zone and air+EGR zone).
E3Z_UNBURNED_DEN	Density of unburned gases.
E3Z_UNBURNED_SIE	Sensible enthalpy of unburned gases.
ECFM3Z_SRC_UNBURN ED_ENTH_EVAP_EX	Mean enthalpy source from spray and film evaporation. Does not correspond to final unburned enthalpy source term.
ECFM_TEMP_EVAP_EX	Not used.
ECFM_UPRIME	Turbulent velocity fluctuation (available only for LES).
EQUIV_RATIO	Global equivalence ratio.
FLAME_SPEED	Laminar flamespeed.
FLMTDEN	Flame surface density divided by ρ . To recover standard flame surface density, multiply <i>FLMTDEN</i> by global density.
FUEL_HTFORM	Formation enthalpy of mean ECFM3Z fuel (average over all fuel components present in the cell) in unburned mixed zone.
FUEL_MW	Molecular weight of mean ECFM3Z fuel in unburned mixed zone.
FUEL_NUM_C	Number of carbon atoms of mean ECFM3Z fuel in unburned mixed zone.
FUEL_NUM_H	Number of hydrogen atoms of mean ECFM3Z fuel in unburned mixed zone.
FUEL_NUM_O	Number of oxygen atoms of mean ECFM3Z fuel in unburned mixed zone.
GRAD1, GRAD2, GRAD3	Components of filtered progress variable gradient.
HEATLOSS	Flamespeed correction factor due to wall heat losses. Equal to 1 for an adiabatic flame. Less than 1 for an under-adiabatic flame with reduced flamespeed.
ISSIM_ALPHA	ISSIM transition factor. 1 = Ignition source term is added to the FSD equation, 0 = No ignition source term in the FSD equation.

Chapter 16: Combustion Modeling

Extended Coherent Flame Models ECFM/ECFM3Z Setup

<i>ISSIM_RBIGN</i>	Radius of burned gases given by the transport equation of the passive Ψ in the ISSIM.
<i>KNOCK_INDEX</i>	Ratio of $OMEGA_TKI / (OMEGA_TKI + OMEGA_CFM)$.
<i>LAM_FLAMESPEED</i>	Equal to <i>FLAME_SPEED</i> .
<i>LAM_FLAME_THICKNESS</i> <i>S</i>	Laminar flame thickness.
<i>LAM_SIGMA</i>	Resolved flame surface density (available only for LES). Defined as $\Sigma_{lam} = \nabla \tilde{c} + (\bar{c} - \tilde{c}) \nabla \cdot n$
<i>MEAN_ENTH</i>	Mean sensible enthalpy defined as transported passive.
<i>NORM1, NORM2, NORM3</i>	Working arrays for ECFM in LES.
<i>OMEGA_BG</i>	Fuel reaction rate (s^{-1}) due to the burned gases post-oxidation (from simplified kinetics or SAGE).
<i>OMEGA_CFM</i>	Fuel reaction rate (s^{-1}) due to flame propagation from the flame surface density model.
<i>OMEGA_TKI</i>	Fuel reaction rate (s^{-1}) due to autoignition from the TKI model .
<i>PROG_VAR</i>	Global mass progress variable \tilde{c} .
<i>PROG_VAR_CLIP</i>	Indicates zones where <i>DIVN</i> cannot be calculated (<i>PROG_VAR_CLIP</i> = 0 in these zones).
<i>PROG_VAR_SIGMA</i>	Mass progress variable due to premixed flame propagation \tilde{c}_Σ .
<i>SIE_MIX_CD_M</i>	Mean sensible enthalpy in mixed zone.
<i>SRC_RIF_FUEL_EVAP_E</i> <i>X</i>	Total fuel spray and film source mass ($kg/m^3/s$).
<i>TEMPERATURE</i>	Mean cell temperature.
<i>TF_<fuel species></i> (e.g., <i>TF_C8H18</i>)	Fuel tracer mass fraction (used to define autoignition zone).
<i>T_<species></i> (e.g., <i>T_CO</i>)	Tracers for CO, CO ₂ , H ₂ , H ₂ O, and N ₂ (used to define autoignition zone).
<i>UNBURNED_ENTH</i>	Sensible enthalpy of unburned gases.
<i>UNBURNED_TEMP</i>	Temperature of unburned gases.
<i>VOL_PROG_VAR</i>	Volumetric global progress variable \bar{c} .
<i>VOL_PROG_VAR_SIGMA</i>	Volumetric progress variable due to flame propagation \bar{c}_Σ .

Chapter 16: Combustion Modeling

Extended Coherent Flame Models ECFM/ECFM3Z Setup

<i>BF_<fuel species></i> (e.g., <i>BF_C8H18</i>)	Fuel mass fraction in burned gases zone.
<i><fuel species></i> (e.g., <i>C8H18</i>)	Fuel mass fraction in unburned gases zone.
<i><species></i> (e.g., <i>CO</i>)	All species names (except for fuel species) without a prefix correspond to the species mass fractions in the burned gases zone.
<i>UMF_<fuel species></i> (e.g., <i>UMF_C8H18</i>)	Fuel mass fraction in unmixed fuel zone.
<i>UNMIX_<species></i> (e.g., <i>UNMIX_CO</i>)	Species mass fractions of CO, CO ₂ , H ₂ , H ₂ O, O ₂ , and N ₂ in unmixed air+EGR zone.
<i>U_<species></i> (e.g., <i>U_CO</i>)	Species mass fractions of CO, CO ₂ , H ₂ , H ₂ O, O ₂ , and N ₂ in mixed unburned gases zone.

Chapter 16: Combustion Modeling

Extended Coherent Flame Models

ECFM/ECFM3Z Setup

Table 16.6: Optional variables (add to *post.in* if desired).

<i>DAMKOHLER</i>	
	$Da = \frac{\sqrt{Re}}{Ka}$ Damkohler number, defined as
<i>ECFM_ACTIVATE</i>	Level of cell activation in ECFM. 0 = No ECFM computation, 1 = Burned gases reactions considered, but unburned gases state cannot be defined (e.g., because progress variable is too close to 1), 2 = Unburned and burned gases zones are defined and the TKI reaction rate (autoignition) can be computed, 4 = Unburned and burned gases zones are defined, the TKI reaction rate (autoignition) can be computed, and laminar flamespeed is greater than the threshold value of 0.01 m/s at which the FSD reaction rate (premixed flame propagation) can be computed.
<i>ECFM_ADIAB_BURNED_TEMP</i>	Estimate of adiabatic burned gases temperature. Can be compared to <i>BURNED_TEMP</i> , which accounts for enthalpy losses.
<i>ECFM_MIXED_MASS_F_RAC</i>	Mass fraction of mixed gases. Equal to 1 for ECFM. Less than or equal to 1 for ECFM3Z.
<i>ECFM_SIGMA_WALL_C</i> <i>ORR</i>	Wall stretch correction factor.
<i>ECFM_SIGMA_TIME_C</i> <i>ORR</i>	Time stretch correction factor.
<i>KARLOVITZ_NUMBER</i>	$Ka = \frac{\delta_t u_\eta}{s_t \eta}$ Karlovitz number, defined as $s_t \eta$, where u_η is the Kolmogorov velocity scale and η is the Kolmogorov length scale.
<i>REYNOLDS_NUMBER</i>	$Re = \frac{\rho_u u' l_t}{\mu(T_u)}$ Reynolds number for unburned gases, given by $\frac{\rho_u u' l_t}{\mu(T_u)}$, where u' and l_t are the velocity fluctuation and integral turbulent length scale, respectively, from the RANS or LES turbulence model, $\mu(T_u)$ is the laminar viscosity at the temperature of the unburned gases, and ρ_u is the density of unburned gases.

16.8 Shell+CTC Model

Original Shell Ignition Model

To model diesel ignition delay, a multi-step kinetics model based on the Shell model has been implemented in CONVERGE. The original Shell model ([Halstead et al., 1977](#)), which was developed to predict knock in gasoline engines, uses a simplified reaction mechanism to simulate autoignition in diesel engines. The eight reactions in this model are given by

Chapter 16: Combustion Modeling

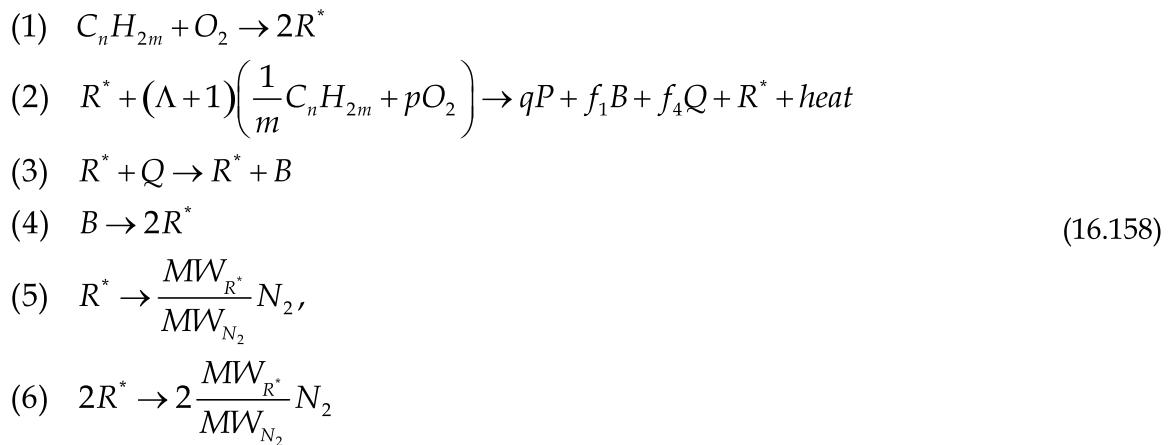
Shell+CTC Model



where C_nH_{2m} is the hydrocarbon fuel, R^* is a radical formed from the fuel, B is the branching agent, Q is a labile intermediate species, and P represents products (CO, H₂, CO₂, and H₂O).

Modified Shell Ignition Model

[Schapertons and Lee \(1985\)](#) noted that the use of the above reaction scheme can violate mass conservation if used after the local onset of ignition due to the propagation steps $R^* \rightarrow R^* + B$ and $R^* \rightarrow R^* + Q$ (reactions (3) and (4) in Equation 16.157 above). They modified the reactions to account for mass conservation. The resulting scheme is given by



where Λ is given by

$$\Lambda = \frac{(f_1 MW_B + f_4 MW_Q)}{(MW_{C_nH_{2m}}/m + p MW_{O_2})} \tag{16.159}$$

and MW_{R^*} , for example, refers to the molecular weight of R^* .

Furthermore,

Chapter 16: Combustion Modeling

Shell+CTC Model

$$p = \frac{n(2 - \gamma) + m}{2m} \quad (16.160)$$

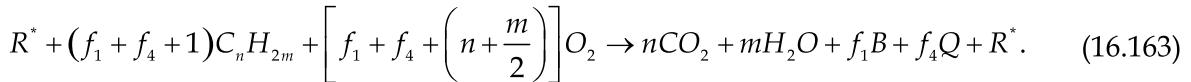
and

$$q = 1 + n/m, \quad (16.161)$$

where γ is a stoichiometric factor set equal to 0.67 ([Halstead et al., 1977](#)). The reaction scheme given in Equation 16.158 is commonly used to model the diesel engine ignition process (e.g., [Kong and Reitz, 1993](#)). In Equation 16.158, the molecular weights associated with the species R^* , B , and Q are given by ([Schapertons and Lee, 1985](#))

$$\begin{aligned} MW_{R^*} &= \frac{(MW_{C_nH_{2m}} + MW_{O_2})}{2} \\ MW_B &= MW_{C_nH_{2m}} + MW_{O_2} \\ MW_Q &= MW_{C_nH_{2m}} + MW_{O_2}. \end{aligned} \quad (16.162)$$

In order to accurately interface the ignition model with a combustion model based on equilibrium (e.g., the [CTC model](#) described below), it is critical that the C, H, O, and N atoms are conserved in the ignition process. In order to modify the reaction scheme in Equation 16.158 to conserve atoms, R^* , B , and Q are assigned atoms based on their molecular weights given above in Equation 16.162. For example, R^* has $n/2$ carbon atoms, m hydrogen atoms, and one oxygen atom. While reaction (2) in Equation 16.158 has been formulated to conserve mass, it must be modified to also conserve atoms. The resulting reaction is given by



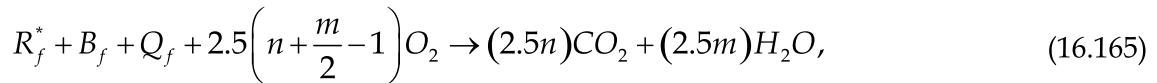
In addition to reaction (2), reactions (5) and (6) in Equation 16.158 also violate atom conservation and lead to a net production of N_2 from the ignition event, which can lead to inaccuracies because this process removes potentially combustible atoms and converts them to inert nitrogen. One possible solution is to have R^* go to products in the termination reactions. However, this possible solution would also require additional oxygen for the atoms to conserve. Another possibility is to keep reactions (5) and (6) as-is in Equation 16.158 and then put the nitrogen back into R^* when the ignition process is complete. In other words,

$$R_f^* = R_{1-6}^* + \frac{MW_{R^*}}{MW_{N_2}} \Delta N_2, \quad (16.164)$$

Chapter 16: Combustion Modeling

Shell+CTC Model

where R_{1-6}^* is the final value of R^* from the reaction scheme and ΔN_2 is the change (increase) in nitrogen from the termination reactions. This final amount of R^* , R_f^* is then patched (along with $B_f=B_{1-6}$ and $Q_f=Q_{1-6}$) to products with the reaction



which was formulated to conserve atoms. The conversion of the radical species to products requires some additional oxygen from the gas mixture. Also note that Equations 16.164 and 16.165 are applied only once when the ignition process is complete. The final reaction scheme with the above improvements is shown in Table 16.7. This mechanism conserves atoms (and mass) between the pre- and post-ignition gas mixture.

Chapter 16: Combustion Modeling

Shell+CTC Model

Table 16.7: Modified Shell ignition reaction mechanism as implemented in CONVERGE.

Number	Modified Shell ignition reaction	Process	Rate
1.	$C_nH_{2m} + O_2 \rightarrow 2R^*$	Initiation	k_q
2.	$R^* + (f_1 + f_4 + 1)C_nH_{2m} + \left[f_1 + f_4 + \left(n + \frac{m}{2} \right) \right] O_2 \rightarrow nCO_2 + mH_2O + f_1B + f_4Q + R^*$	Propagation	k_p
3.	$R^* + Q \rightarrow R^* + B$	Propagation	f_2k_p
4.	$B \rightarrow 2R^*$	Branching	k_b
5.	$R^* \rightarrow \frac{MW_{R^*}}{MW_{N_2}} N_2$	Linear termination	f_3k_p
6.	$2R^* \rightarrow 2 \frac{MW_{R^*}}{MW_{N_2}} N_2$	Quadratic termination	k_t
7.	$R_f^* = R_{1-6}^* + \frac{MW_{R^*}}{MW_{N_2}} \Delta N_2$ $B_f = B_{1-6}$ $Q_f = Q_{1-6}$ $R_f^* + B_f + Q_f + 2.5 \left(n + \frac{m}{2} - 1 \right) O_2 \rightarrow (2.5n)CO_2 + (2.5m)H_2O$	This step performed only once (at the end of the ignition process).	

The rate constants in the reaction scheme shown below in Table 16.7 are similar to those used by Halstead [et al.](#) (1977) and are given by

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\begin{aligned} f_1 &= A_{f01} \exp\left(\frac{-E_{f1}}{RT}\right) [O_2]^{x_1} [C_n H_{2m}]^{y_1} \\ f_2 &= A_{f01} \exp\left(\frac{-E_{f1}}{RT}\right) \\ f_3 &= A_{f03} \exp\left(\frac{-E_{f3}}{RT}\right) [O_2]^{x_3} [C_n H_{2m}]^{y_3} \\ f_4 &= A_{f04} \exp\left(\frac{-E_{f4}}{RT}\right) [O_2]^{x_4} [C_n H_{2m}]^{y_4}, \end{aligned} \quad (16.166)$$

$$\begin{aligned} k_q &= A_q \exp\left(\frac{-E_q}{RT}\right) \\ k_b &= A_b \exp\left(\frac{-E_b}{RT}\right) \\ k_t &= A_t \exp\left(\frac{-E_t}{RT}\right), \end{aligned} \quad (16.167)$$

$$\begin{aligned} k_{p1} &= A_{p1} \exp\left(\frac{-E_{p1}}{RT}\right) \\ k_{p2} &= A_{p2} \exp\left(\frac{-E_{p2}}{RT}\right) \\ k_{p3} &= A_{p3} \exp\left(\frac{-E_{p3}}{RT}\right), \end{aligned} \quad (16.168)$$

and

$$k_p = \frac{1}{\frac{1}{k_{p1}} [O_2] + \frac{1}{k_{p2}} + \frac{1}{k_{p3}} [C_n H_{2m}]}, \quad (16.169)$$

where R is the ideal gas constant and $[O_2]$, for example, denotes the species concentration of O2 in mol/cm³.

Equations 16.166 through 16.169 include ten pre-exponential factors (A_i), ten activation energies (E_i), and six concentration dependence parameters (x_i , y_i) for a total of 26 model

Chapter 16: Combustion Modeling

Shell+CTC Model

constants. [Kong and Reitz \(1993\)](#) found that the formation rate of the labile intermediate species Q is the rate-limiting step in the kinetic path. In addition, [Theobald and Cheng \(1987\)](#) found that the total ignition delay is sensitive to the pre-exponential factor A_{f04} . As a result, `combust.in > shell_model > af04` is a user-specified parameter, while the other 25 parameters have fixed values in CONVERGE as shown below in Table 16.8. Note that the activation energy is given in units of *cal/mole* to be consistent with standard chemical reaction conventions.

Chapter 16: Combustion Modeling

Shell+CTC Model

Table 16.8: Shell model parameters from Halstead [et al.](#) (1977). Units: A_i (cm, mol, s), E_i (cal/mole), R (cal/mol-K).

Shell model parameter	Value
A_{f01}	7.3e-4
A_{f02}	180.0
A_{f03}	1.47
A_{f04}	Set via combust.in > shell_model > af04.
A_q	1.2e12
A_b	4.4e17
A_t	3.0e12
A_{p1}	1.0e12
A_{p2}	1.0e11
A_{p3}	1.0e13
E_{f1}/R	-7.55e03
E_{f2}/R	-3.53e03
E_{f3}/R	5.04e03
E_{f4}/R	1.51e04
E_q/R	1.76e04
E_b/R	2.27e04
E_t/R	0.0
E_{p1}/R	0.0
E_{p2}/R	7.56e03
E_{p3}/R	428.0
x_1	1.0
x_3	0.0
x_4	-1.0
y_1	0.0
y_3	0.0
y_4	0.35

Chapter 16: Combustion Modeling

Shell+CTC Model

In the present implementation, reaction rates are frozen to their value at 950 K for temperatures above 950 K as in the model from [Schapertons and Lee \(1985\)](#).

As described by [Theobald and Cheng \(1987\)](#), subcycling is needed in order to accurately solve the modified Shell model reaction mechanism. Subcycling is based on the net production rate of B , assuming that the reaction orders of $CnH2m$ and $O2$ are zero. The time rate of change of $[B]$ is then given by

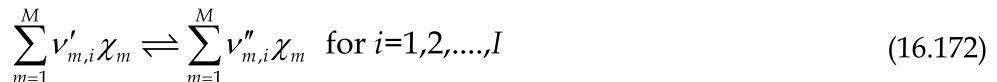
$$\frac{d[B]}{dt} = -k_b[B] + f_1 k_p[R^*] + f_2 k_p[R^*][Q]. \quad (16.170)$$

A subcycling time-step dt_{shell} can be written from Equation 16.170 as

$$dt_{shell} = \left| \frac{[B]}{-k_b[B] + f_1 k_p[R^*] + f_2 k_p[R^*][Q]} \right|, \quad (16.171)$$

where the quantity $[X_m]$ represents a molar concentration. The computational time-step dt is discretized into sub-time-steps (given by Equation 16.171) to solve the Shell model equations.

The above Shell model reaction scheme is solved using a reference species approach as described in [Amsden et al. \(1989\)](#). This approach is valid for the following reaction scheme:



where $v'_{m,i}$ and $v''_{m,i}$ are the stoichiometric coefficients for the reactants and products, respectively, for species m and reaction i and χ_m represents the chemical symbol for species m .

As described by Amsden et al. [\(1989\)](#), the reaction rate, R_i , for this approach is given by

$$R_i = \frac{\rho_{ref,i} (\Psi_1 / \Psi_2 - 1)}{dt MW_{ref,i} (v''_{ref,i} - v'_{ref,i})'} \quad (16.173)$$

where

$$\Psi_1 = \rho_{ref,i} + dt MW_{ref,i} (v''_{ref,i} \Omega_{i,f} + v'_{ref,i} \Omega_{i,r}), \quad (16.174)$$

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\Psi_2 = \rho_{ref,i} + dt MW_{ref,i} (\nu'_{ref,i} \Omega_{i,f} + \nu''_{ref,i} \Omega_{i,r}), \quad (16.175)$$

and $\rho_{ref,i}$ is the reference species of reaction i with a molecular weight of $MW_{ref,i}$. The reference species is the species whose density has the highest likelihood of going negative. In the above equations, the quantities $\Omega_{i,f}$ and $\Omega_{i,r}$ are given by

$$\Omega_{i,f} = k_{i,f} \prod_{m=1}^M (\rho_m / MW_m)^{\tilde{\nu}'_{m,i}} \quad (16.176)$$

and

$$\Omega_{i,r} = k_{i,r} \prod_{m=1}^M (\rho_m / MW_m)^{\tilde{\nu}''_{m,i}}, \quad (16.177)$$

where $k_{i,f}$ and $k_{i,r}$ are the forward and reverse rate constants, respectively. In addition, $\tilde{\nu}'_{m,i}$ and $\tilde{\nu}''_{m,i}$ in Equations 16.176 and 16.177 are not necessarily equal to the stoichiometric coefficients in Equation 16.172 so that empirical reaction orders can be used ([Amsden et al., 1989](#)).

Each of the reactions in the current set progress only in the forward direction. As a result, $k_{i,r}$ (and hence $\Omega_{i,r}$) is zero for each of the reactions. With this simplification, it can be shown that Equation 16.173 reduces to

$$R_i = \frac{(\rho_{ref,i} / MW_{ref,i}) \Omega_{i,f}}{(\rho_{ref,i} / MW_{ref,i}) + \nu'_{ref,i} \Omega_{i,f} dt}, \quad (16.178)$$

which is used in CONVERGE to calculate the progress rates for the Shell model.

The reference species is selected for each reaction by comparing the values of $MW_m \Omega_{i,f} (\nu''_{m,i} - \nu'_{m,i}) / \rho_m$ for each of the species involved in that reaction. The species with the lowest value of this quantity is called the reference species ([Amsden et al., 1989](#)). The rate coefficients are given in Table 16.7 above for each of the reactions. Once the reference species ref,i has been determined, the reaction rate R_i for reaction i is calculated via Equation 16.178 above. The change in density in species m is then calculated from

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\rho_m = \rho_{m,0} + MW_m \sum_{i=1}^I (\nu''_{m,i} - \nu'_{m,i}) R_i dt, \quad (16.179)$$

where $\rho_{m,0}$ is the species density value before the start of the Shell model calculations for the current subcycle.

Input Parameters for the Shell Model

The Shell model parameters are located in [combust.in](#). In addition, *rshell*, *bshell*, *qshell*, and *shell-n2* must be present in the [reaction mechanism](#) and [thermodynamic](#) data files in order to model the ignition process.

Characteristic Time Combustion Model

CONVERGE includes the Characteristic Time Combustion (CTC) model ([Abraham et al., 1985](#) and [Xin et al., 1997](#)). The CTC model simulates the rate of change of the density of species m , ρ_m , as

$$\frac{\partial \rho_m}{\partial t} = -\frac{\rho_m - \rho_m^*}{\tau_c}, \quad (16.180)$$

where ρ_m^* is the local and instantaneous thermodynamic equilibrium value of the species density and τ_c is the characteristic time to achieve equilibrium. Following [Kong et al. \(1995\)](#), the characteristic time is calculated via

$$\tau_c = \tau_{chem} + f \tau_{turb}, \quad (16.181)$$

where τ_{chem} is the chemical-kinetics time, τ_{turb} is the turbulent mixing time, and f is a delay coefficient which simulates the increasing influence of turbulence on combustion ([see previous equation](#)). The chemical time scale is modeled as ([Kong et al. \(1995\)](#))

$$\tau_{chem} = \frac{[C_n H_{2m}]^{0.75} \exp\left(\frac{E_{chem}}{RT_g}\right)}{2 A_{chem} [O_2]^{1.5}}, \quad (16.182)$$

where A_{chem} is a user-specified constant ([combust.in](#) > *ctc_model* > *denomc*), E_{chem} is the activation energy given by 18,475 cal/mol ([Kong et al. \(1995\)](#)), R is the ideal gas constant, and T_g is the gas temperature. In addition, the turbulent time scale is given as

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\tau_{turb} = C_2 \frac{k}{\varepsilon}, \quad (16.183)$$

where C_2 is a user-specified constant ([combust.in > ctc_model > cm2](#)). The turbulent time scale acts as a sub-grid model that accounts for the non-uniformity of species in a cell. The sub-grid non-uniformity of species, which cannot be accounted for directly, may act to slow the combustion process. For this reason, the turbulent time scale is added to slow the combustion.

If an [LES](#) model is used, the characteristic time for the CTC model is calculated as

$$\tau_c = \tau_{chem} + f \tau_{sub}, \quad (16.184)$$

where τ_{sub} is based on the sub-grid velocity field given by

$$u_i' \cong C_{les} \frac{dx^2}{24} \frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j}. \quad (16.185)$$

In Equation 16.185, dx is a characteristic cell size (given by the cube root of the cell volume), \bar{u}_i is the resolved velocity field in dimension i , C_{les} is a scaling constant, and j is a dimension that may or may not be the same as dimension i . The sub-grid time scale is then given by

$$\tau_{sub} = C_2 \frac{k_{sub}}{\varepsilon_{sub}}, \quad (16.186)$$

where

$$k_{sub} = \frac{1}{2} u_i' u_i' \quad (16.187)$$

and

$$\varepsilon_{sub} = \frac{k_{sub}^{3/2}}{dx}. \quad (16.188)$$

In the case of the single-scale CTC model, solving Equation 16.180 for the updated species density due to combustion yields

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\rho_m = \rho_m^* + (\rho_{m,0} - \rho_m^*) \exp\left(\frac{-dt}{\tau_c}\right), \quad (16.189)$$

where $\rho_{m,0}$ is the species density value at the beginning of the time-step and dt is the time-step.

In addition to the single-scale CTC model, the multi-scale model proposed by Xin [et al.](#) (1997) is included in CONVERGE. In this model, the following expressions are used for the destruction of fuel, CO, and H2:

$$\frac{\partial \rho_{fuel}}{\partial t} = -\frac{\rho_{fuel} - \rho_{fuel}^*}{\tau_c}, \quad (16.190)$$

$$\frac{\partial \rho_{CO}}{\partial t} = -\frac{\rho_{CO} - \rho_{CO}^*}{x_{CTC} \tau_c}, \quad (16.191)$$

and

$$\frac{\partial \rho_{H_2}}{\partial t} = -\frac{\rho_{H_2} - \rho_{H_2}^*}{x_{CTC} \tau_c}, \quad (16.192)$$

where x_{CTC} is a user-specified fraction between 0 and 1 ([combust.in > ctc_model > tau_fraction](#)). The above model assumes that the time scale for CO and H2 are a fraction of the fuel conversion time scale. Note, however, that this assumption represents only the fastest possible time scales for CO and H2. If there is not enough oxygen in a computational cell to perform this conversion, the time scales are increased accordingly.

Solving Equations 16.190 through 16.192 yield the following updated species densities:

$$\rho_{fuel} = \rho_{fuel,0} + \Delta[fuel] MW_{fuel} = \rho_{fuel}^* + (\rho_{fuel,0} - \rho_{fuel}^*) \exp\left(\frac{-dt}{\tau_c}\right), \quad (16.193)$$

$$\rho_{CO} = \rho_{CO,0} + \Delta[CO] MW_{CO} = \rho_{CO}^* + (\rho_{CO,0} - \rho_{CO}^*) \exp\left(\frac{-dt}{x_{CTC} \tau_c}\right), \quad (16.194)$$

and

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\rho_{H_2} = \rho_{H_2,0} + \Delta[H_2]MW_{H_2} = \rho_{H_2}^* + (\rho_{H_2,0} - \rho_{H_2}^*)\exp\left(\frac{-dt}{x_{CTC}\tau_c}\right). \quad (16.195)$$

The remaining species densities can be determined from the atom conservation equations as

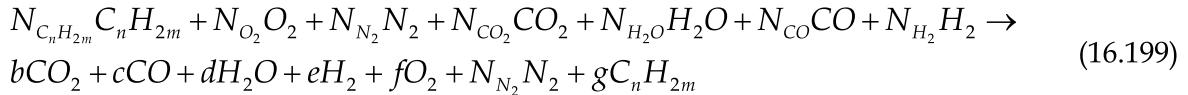
$$\rho_{CO_2} = \rho_{CO_2,0} + (-\Delta[CO] - n\Delta[fuel])MW_{CO_2}, \quad (16.196)$$

$$\rho_{H_2O} = \rho_{H_2O,0} + (-\Delta[H_2] - m\Delta[fuel])MW_{H_2O}, \quad (16.197)$$

and

$$\rho_{O_2} = \rho_{O_2,0} + (0.5\Delta[CO] + 0.5\Delta[H_2] + (n + 0.5m)\Delta[fuel])MW_{O_2}. \quad (16.198)$$

The equilibrium species density values must be determined in order to solve Equation 16.189 or Equations 16.193 through 16.198. As in Kong *et al.* (1995), it is assumed that seven species are involved in the combustion process: fuel (C_nH_{2m}), O₂, N₂, CO₂, H₂O, CO, and H₂. With these species, the combustion of an arbitrary hydrocarbon can be written as



where N_m represents the initial number of moles of species m per gram of mixture and the coefficients b , c , d , e , f , and g are the unknown quantities. For non-rich conditions, it is assumed that the equilibrium value of fuel is zero, and hence $g = 0$. Atom balances for carbon, hydrogen, and oxygen yield the following expressions:

$$\begin{aligned} car_{tot} &= nN_{C_nH_{2m}} + N_{CO} + N_{CO_2} = b + c \\ hyd_{tot} &= 2(mN_{C_nH_{2m}} + N_{H_2} + N_{H_2O}) = 2d + 2e \\ oxy_{tot} &= 2(N_{O_2} + N_{CO_2}) + N_{H_2O} + N_{CO} = 2b + c + d + 2f. \end{aligned} \quad (16.200)$$

In order to solve for the five unknowns, two additional equations are needed. Two equilibrium reactions are used for this purpose. The first, the so-called water-gas shift reaction, is given by



Chapter 16: Combustion Modeling

Shell+CTC Model

and accounts for the simultaneous presence of CO and H₂. The second reaction is



It can be shown that the equilibrium constants for Equations 16.201 and 16.202 are given by

$$K_{p,1} = \exp\left(\frac{\Delta S^o}{R} - \frac{\Delta H^o}{RT}\right) = \frac{b \cdot e}{c \cdot d} \quad (16.203)$$

and

$$K_{p,2} = \exp\left(\frac{\Delta S^o}{R} - \frac{\Delta H^o}{RT}\right) \left(\frac{RT}{p_{atm}}\right) \rho_{tot} = \frac{b^2}{c^2 f}, \quad (16.204)$$

where K_p is the equilibrium constant in pressure units and the subscript 1 and 2 refer to Equations 16.201 and 16.202, respectively. In Equations 16.203 and 16.204 the change in standard state molar entropy and enthalpy are given by

$$\begin{aligned} \frac{\Delta S^o}{R} &= \frac{(S_{CO_2}^o + S_{H_2}^o - S_{CO}^o - S_{H_2O}^o)}{R} \\ \frac{\Delta H^o}{RT} &= \frac{(H_{CO_2}^o + H_{H_2}^o - H_{CO}^o - H_{H_2O}^o)}{RT} \end{aligned} \quad (16.205)$$

for the water-gas shift equilibrium reaction and by

$$\begin{aligned} \frac{\Delta S^o}{R} &= \frac{(2S_{CO_2}^o - 2S_{CO}^o - S_{O_2}^o)}{R} \\ \frac{\Delta H^o}{RT} &= \frac{(2H_{CO_2}^o - 2H_{CO}^o - H_{O_2}^o)}{RT}, \end{aligned} \quad (16.206)$$

for Equation 16.202. Furthermore, in Equation 16.204 ρ_{tot} is the sum of the species densities for the seven main combustion species (C_nH_{2m}, O₂, N₂, CO₂, H₂O, CO, and H₂).

Equations 16.200, 16.203, and 16.204 include five equations and five unknowns. In order to solve for the equilibrium coefficients, the above equations were combined to write the following polynomial for the CO coefficient c :

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\begin{aligned} & \left(1 - K_{p,1}\right)c^4 + \left[\left(1 - K_{p,1}\right)oxy_{tot} - \left(3 - 2K_{p,1}\right)car_{tot} - \frac{hyd_{tot}}{2} - 2\frac{\left(1 - K_{p,1}\right)}{K_{p,2}} \right]c^3 + \\ & \left[\left(\frac{hyd_{tot}}{2} - oxy_{tot} + 2car_{tot} + \frac{6 - 4K_{p,1}}{K_{p,2}} \right)car_{tot} \right]c^2 + \left[\left(2\frac{\left(K_{p,1} - 3\right)}{K_{p,2}} \right)car_{tot}^2 \right]c + \frac{2car_{tot}^3}{K_{p,2}} = 0, \end{aligned} \quad (16.207)$$

which is solved in CONVERGE using a Newton-Raphson solver. Once the CO coefficient c is determined, the other four coefficients can be found through Equations 16.200 and 16.204. These coefficients are then used to determine the equilibrium species densities ρ_m^* .

For rich conditions ($car_{tot} > oxy_{tot}$) it is assumed that only CO, H₂, and C_nH_{2m} are products of equilibrium. As a result, $b = d = f = 0$, and the atom balances yield

$$\begin{aligned} c &= oxy_{tot} \\ g &= (car_{tot} - c)/n \\ e &= (hyd_{tot} - 2mg)/2 \end{aligned} \quad (16.208)$$

for the remaining coefficient values.

Input Parameters for the CTC Model

The CTC model parameters are located in [combust.in](#). When using the CTC model, you must activate turbulence modeling (*i.e.*, set [inputs.in](#) > *solver_control* > *turbulence_solver* = 1).

Shell+CTC Model

A common application of the Shell and CTC models is to implement them concurrently. The Shell model acts on computational cells that are in the ignition phase and the CTC model acts on cells that are in the combustion phase. As in [\(Xin et al. \(1997\)\)](#), the two phases are distinguished by the local gas temperature and a delay coefficient f given by

$$f = \frac{1 - e^{-\gamma}}{0.632}, \quad (16.209)$$

where γ is a production fraction given by

$$\gamma = \frac{\rho_{CO_2,react} + \rho_{H_2O,react} + \rho_{CO,react} + \rho_{H_2,react}}{\rho_{tot} - \rho_{N_2} - \rho_{CO_2,resid} - \rho_{H_2O,resid}}, \quad (16.210)$$

Chapter 16: Combustion Modeling

Shell+CTC Model

where the subscript *react* refers to a reactive species, not including any residual or exhaust gas recirculation (EGR) amounts, and the subscript *resid* refers to a residual amount. ρ_{tot} is the sum of the species densities for the seven active Shell+CTC model species ($CnH2m$, O_2 , N_2 , CO_2 , H_2O , CO , and H_2). Equation 16.210 above represents a ratio of the amount of products to the amount of reactive species. Note that Equation 16.210 is based on the assumption that only residual amounts of CO_2 and H_2O are included in the simulation.

Equation 16.209 represents a measure of combustion completeness and is used in the CTC model to delay the effect of turbulence on combustion until combustion has proceeded to a certain point ([see previous equation](#)). The f parameter is used in conjunction with a critical temperature T_{chop} (`combust.in > ctc_model > temp_cutoff`) to determine whether a computational cell should undergo Shell or CTC calculations. Table 16.9 below presents a definition of both ignition and combustion phases when the Shell+CTC model is used.

Table 16.9: Definition of ignition and combustion phases for the Shell+CTC model.

Model	Ignition and combustion phase criteria
Shell	$T_g < T_{chop}$ and $f \leq 0.1$
CTC	$T_g \geq T_{chop}$ or $f > 0.1$

When a cell is in the combustion phase, the radicals R^* , B , and Q must be patched to the products (*i.e.*, CO , CO_2 , and H_2O). This patching happens only once per cell as the radical species values are set to zero after the patching is complete.

For the non-modified Shell model, the ignition radicals are patched to products in the proportion proposed by [Schapertons and Lee \(1985\)](#), which yields the following updated species densities (for CO , CO_2 , and H_2O , respectively):

$$\Delta\rho_{CO} = \frac{\gamma(n/m)MW_{CO}}{\gamma(n/m)MW_{CO} + (1-\gamma)(n/m)MW_{CO_2} + MW_{H_2O}} \rho_{rad}, \quad (16.211)$$

$$\Delta\rho_{CO_2} = \frac{(1-\gamma)(n/m)MW_{CO_2}}{\gamma(n/m)MW_{CO} + (1-\gamma)(n/m)MW_{CO_2} + MW_{H_2O}} \rho_{rad}, \quad (16.212)$$

and

$$\Delta\rho_{H_2O} = \frac{MW_{H_2O}}{\gamma(n/m)MW_{CO} + (1-\gamma)(n/m)MW_{CO_2} + MW_{H_2O}} \rho_{rad}, \quad (16.213)$$

where

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\rho_{rad} = \rho_{R_f^*} + \rho_{B_f} + \rho_{Q_f} \quad (16.214)$$

is the sum of the radical species densities.

For the modified Shell model, the change in product species as a result of radical patching can be found from Reaction 7 [in a previous table](#). The amount of oxygen $\Delta\rho_{O_2}$ needed to convert the radicals to products is given by

$$\Delta\rho_{O_2} = \left[\frac{2.5\left(n + \frac{m}{2} - 1\right)MW_{O_2}}{MW_{K^*} + MW_B + MW_Q} \right] \rho_{rad}, \quad (16.215)$$

where

$$\rho_{rad} = \rho_{R_f^*} + \rho_{B_f} + \rho_{Q_f} \quad (16.216)$$

is the sum of the final radical species densities (R_f^* includes the additional N_2 formed during the ignition process $\Delta\rho_{N_2}$ as explained previously). The change in products from the radical conversion process is given by

$$\Delta\rho_{CO_2} = \frac{(2.5n)MW_{CO_2}(\rho_{rad} + \Delta\rho_{O_2})}{(2.5n)MW_{CO_2} + (2.5m)MW_{H_2O}} \quad (16.217)$$

and

$$\Delta\rho_{H_2O} = \frac{(2.5m)MW_{H_2O}(\rho_{rad} + \Delta\rho_{O_2})}{(2.5n)MW_{CO_2} + (2.5m)MW_{H_2O}}. \quad (16.218)$$

The seven species involved in the patching process are modified as follows:

Chapter 16: Combustion Modeling

Shell+CTC Model

$$\begin{aligned}\rho_{O_2} &= \rho_{O_2} - \Delta\rho_{O_2} \\ \rho_{N_2} &= \rho_{N_2} - \Delta\rho_{N_2} \\ \rho_{CO_2} &= \rho_{CO_2} + \Delta\rho_{CO_2} \\ \rho_{H_2O} &= \rho_{H_2O} + \Delta\rho_{H_2O} \\ \rho_R^* &= 0 \\ \rho_B &= 0 \\ \rho_Q &= 0\end{aligned}\tag{16.219}$$

Input Parameters for the Shell+CTC Model

The Shell+CTC model parameters are located in the [combust.in](#) > *shell_model* and [combust.in](#) > *ctc_model* settings blocks. When using the Shell+CTC model, you must activate turbulence modeling (*i.e.*, you must set [inputs.in](#) > *solver_control* > *turbulence_solver* = 1 and include a [turbulence.in](#) file in your case setup).

16.9 Representative Interactive Flamelet (RIF) Model

Combustion in non-premixed turbulent flow is restricted to relatively thin, molecularly mixed regions that lie within flammability limits. The interaction of the highly turbulent flow field with the combustion process is governed by nonlinear chemistry as the chemistry time scale is much less than the turbulence time scale. The Representative Interactive Flamelet (RIF) model effectively decouples the solution of the turbulent flow field and diffusion flamelets.

The RIF concept for non-premixed combustion in CONVERGE is based on a coordinate transformation that uses the mixture fraction as an independent coordinate. Therefore, it is possible to transform the balance equations for enthalpy and species into mixture fraction space, which leads to resolution of the inner structure of the reaction zone in a one-dimensional calculation.

Flamelet Equations

The first-order flamelet equations have been derived by [Peters \(1984\)](#), where a local coordinate transformation and boundary layer arguments are used. The resulting equations for the species mass fraction and temperature, respectively, are

$$\rho \frac{\partial Y_i}{\partial t} - \rho \frac{\chi}{2\widehat{Le}_i} \frac{\partial^2 Y_i}{\partial Z^2} = \dot{\omega}_i, \quad \widehat{Le}_i = \begin{cases} Le_i & (\text{Laminar}) \\ \hat{D}/(\hat{D}+D_i) & (\text{Turbulent}) \approx 1.0 \end{cases} \tag{16.220}$$

and

Chapter 16: Combustion Modeling

Representative Interactive Flamelet (RIF) Model

$$\rho \frac{\partial T}{\partial t} - \rho \frac{\chi}{2} \frac{\partial^2 T}{\partial Z^2} - \rho \frac{\chi}{2c_p} \left[\sum_{i=1}^{n_s} \frac{c_{p_i}}{\widehat{Le}_i} \frac{\partial Y_i}{\partial Z} + \frac{\partial c_p}{\partial Z} \right] \frac{\partial T}{\partial Z} = \frac{1}{c_p} \left(\frac{\partial p}{\partial t} - \sum_{i=1}^{n_s} \dot{\omega}_i h_i \right). \quad (16.221)$$

In Equations 16.220 and 16.221, n_s denotes the number of chemical species, c_p is the heat capacity at constant pressure, Y_i is the mass fraction of the chemical species i , h_i is the enthalpy of the chemical species i , $\dot{\omega}_i$ is the net chemical production rate, D_i is the diffusivity, \widehat{D} is the turbulent diffusivity, and \widehat{Le}_i is the Lewis number of species i . For simulations that include RIF, CONVERGE sets the Lewis number to 1 for all species.

A characteristic of Equations 16.220 and 16.221 is that after the transformation into mixture fraction (Z) space, the convective terms disappear. Since all the scalars are convected with the same velocity in physical space, no relative convective velocities exist between the mixture fraction and the other scalars, such as species mass fractions or temperature. Coupling the equations in phase space to the flow field in physical space occurs through the pressure and scalar dissipation rate.

$$\tilde{Y}_i(x_i, t) = \int_0^1 \tilde{P}(Z; x_i, t) Y_i(Z, t) dZ \quad (16.222)$$

Governing Equations of Mixture Fraction

The RIF model is based on the presumed probability density function (PDF) approach described later in this section. The RIF model requires knowledge of the Favre mean mixture fraction \tilde{Z} and its variance $\widetilde{Z''^2}$ in space and time. The conservation equations for the mean and the variance of the mixture fraction are given by

$$\frac{\partial(\tilde{\rho}\tilde{Z})}{\partial t} + \frac{\partial}{\partial x_i} (\tilde{\rho}\tilde{u}_i \tilde{Z}) = -\frac{\partial}{\partial x_i} (\tilde{\rho}\tilde{u}_i'^2 \widetilde{Z''^2}) + S_{RIF} \quad (16.223)$$

and

$$\frac{\partial(\tilde{\rho}\widetilde{Z''^2})}{\partial t} + \frac{\partial}{\partial x_i} (\tilde{\rho}\widetilde{u}_i'^2 \widetilde{Z''^2}) = -\frac{\partial}{\partial x_i} (\tilde{\rho}\widetilde{u}_i'^2 \widetilde{Z''^2}) - 2(\tilde{\rho}\widetilde{u}_i'^2 \widetilde{Z''}) \frac{\partial \tilde{Z}}{\partial x_i} - \tilde{\rho}\tilde{\chi}, \quad (16.224)$$

where

Chapter 16: Combustion Modeling

Representative Interactive Flamelet (RIF) Model

$$\widetilde{u_i'^2 Z''^2} = -D_t \frac{\partial \tilde{Z}}{\partial x_i}. \quad (16.225)$$

In Equations 16.223 to 16.225, $\tilde{\rho}$ is the density, x_i is the space coordinate, t is time, S_{RIF} is the source term, D_t is the turbulent diffusivity, and $\tilde{\chi}$ is the scalar dissipation rate, which can be modeled as

$$\tilde{\chi} \cong c_\chi \frac{\tilde{\varepsilon}}{\tilde{k}} \widetilde{Z''^2}, \quad (16.226)$$

where $\tilde{\varepsilon}$ is the turbulent dissipation, \tilde{k} is the turbulent kinetic energy and c_χ is a time scale ratio. In CONVERGE we assume c_χ to be constant, and the default value is 2.0. The scalar dissipation rate $\tilde{\chi}$ is calculated from the flow solver of CONVERGE and then is integrated along flamelet surface.

The scalar dissipation rate conditioned on stoichiometric mixture fraction is obtained from

$$\widetilde{\chi}_{st}(\mathbf{x}, t) = \frac{\tilde{\chi} f(Z_{st})}{\int_0^1 f(Z) \tilde{P}(Z; \mathbf{x}, t) dZ}, \quad (16.227)$$

where $\tilde{P}(Z^{1/2}; \mathbf{x}, t)$ is the probability density function and

$$f(Z) = \exp\left(-2\left[\operatorname{erfc}^{-1}(2Z)\right]^2\right) \quad (16.228)$$

The surface averaged value for scalar dissipation rate at the stoichiometric mixture fraction can be derived as

$$\hat{\chi}_{st} = \frac{\int_V \tilde{\rho}(\mathbf{x}) \tilde{\chi}_{st}^{3/2}(\mathbf{x}) \tilde{P}(Z_{st}) dV'}{\int_V \tilde{\rho}(\mathbf{x}) \tilde{\chi}_{st}^{1/2}(\mathbf{x}) \tilde{P}(Z_{st}) dV'}, \quad (16.229)$$

where Z_{st} is fuel mass fraction at an equivalence ratio of 1.0 and V is the volume of the entire computational domain.

Chapter 16: Combustion Modeling

Representative Interactive Flamelet (RIF) Model

Multiple Flamelets Theory

In unsteady flamelet modeling, the history of the scalar dissipation rate and the boundary conditions determine the solution of a flamelet, with the consequence that different flamelet histories must be calculated if these parameters vary too much in the physical domain. The scalar dissipation rate ($\tilde{\chi}$) is not uniform during spray injection and droplet vaporization process. Therefore, multiple flamelets are used for the spatial inhomogeneity of χ (Barths et al., 1998), as

$$\hat{\chi}_{st}(l) = \frac{\int_V \tilde{I}_l(\mathbf{x}) \tilde{\rho}(\mathbf{x}) \chi_{st}^{3/2}(\mathbf{x}) \tilde{P}(Z_{st}) dV'}{\int_V \tilde{I}_l(\mathbf{x}) \tilde{\rho}(\mathbf{x}) \chi_{st}^{1/2}(\mathbf{x}) \tilde{P}(Z_{st}) dV'}, \quad (16.230)$$

where

$$\tilde{I}_l(\mathbf{x}) = \frac{Z_l}{\sum Z_l}. \quad (16.231)$$

Here Z_l is the mixture fraction corresponding to each flamelet and l can vary from 0 to the number of flamelets.

For multiple flamelet simulations, the first flamelet is initialized at the start of the spray ([parcel_introduction.in > injections > injection > injection_control > start_time](#)). In engine simulations, additional flamelets are initialized at intervals of 0.5 crank angle degrees.

RIF Model and CFD Solver Interaction

The energy conservation equation is solved to get temperature of CFD cells. Figure 16.15 below shows how the flow solver and the flamelet code interact. From the turbulent flow and mixing field the domain averaged flamelet parameters, scalar dissipation rate conditioned on stoichiometric mixture χ_{st} and the pressure, p , are extracted. They are transferred to the flamelet code (RIF model) at each CFD time-step (n). The flamelet code computes the solution of the flamelet equations for the species mass fractions and the temperature as given in Equations 16.220 and 16.221 for the next time-step ($n+1$), resolving the chemical time scales by subcycling the time-step applied by the CFD code. Weighting this solution with a presumed probability density function as a function of the mixture fraction and the mixture fraction variance leads to turbulent mean values for the species mass fractions in physical space. Species transport and the energy equation are then solved to get the updated mass fractions of all the species and the cell temperature.

Chapter 16: Combustion Modeling

Representative Interactive Flamelet (RIF) Model

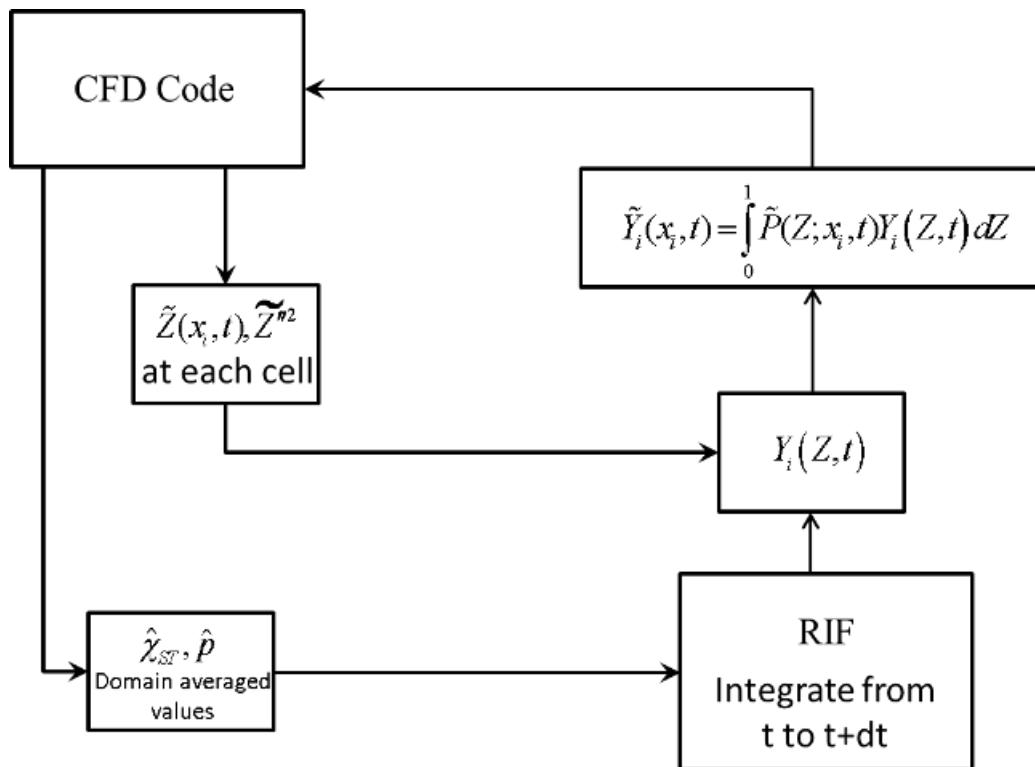


Figure 16.15: A schematic of the Representative Interactive Flamelet concept by [Peters \(2000\)](#).

Figure 16.16 shows the coupling of the RIF model with the [order of solution of transport equations](#) in CONVERGE. The species mass fractions from the flamelet code (RIF model) act as sources for the species equations. The transport equations for mean mixture fraction and its variance (\tilde{Z} and \tilde{Z}''^2 , respectively) are solved as [passives](#). These are used to calculate the scalar dissipation rate and the pressure which is fed back to the flamelet code (RIF model).

To provide the RIF model with values for the Favre mean mixture fraction (\tilde{Z}) and its variance (\tilde{Z}''^2), you must define [passives](#) for both of these values. You must also define a [passive](#) for each flamelet to be included in the RIF model. To define these [passives](#), include the following names, followed by a Schmidt number (0.78 is recommended), under the [passive](#) section in the [species.in](#) file, as shown in the following figure.

```

.
.
passive:
  - RIF_ZMEAN:          0.78
  - RIF_ZVAR:           0.78
  - RIF_FLMT_ZMEAN1:   0.78
  - RIF_FLMT_ZMEAN2:   0.78
  
```

Figure 16.16: An excerpt of *species.in* showing passive inputs for RIF parameters.

Chapter 16: Combustion Modeling

Representative Interactive Flamelet (RIF) Model

Include one instance of *RIF_FLMT_ZMEAN#* (e.g., *RIF_FLMT_ZMEAN1*) for each flamelet that you are including in the RIF model. Specify the number of flamelets via [combust.in > rif_model > num_flamelets](#).

Figure 16.17 below describes the coupling process between the RIF model and the [iterative algorithm](#) (e.g., PISO).

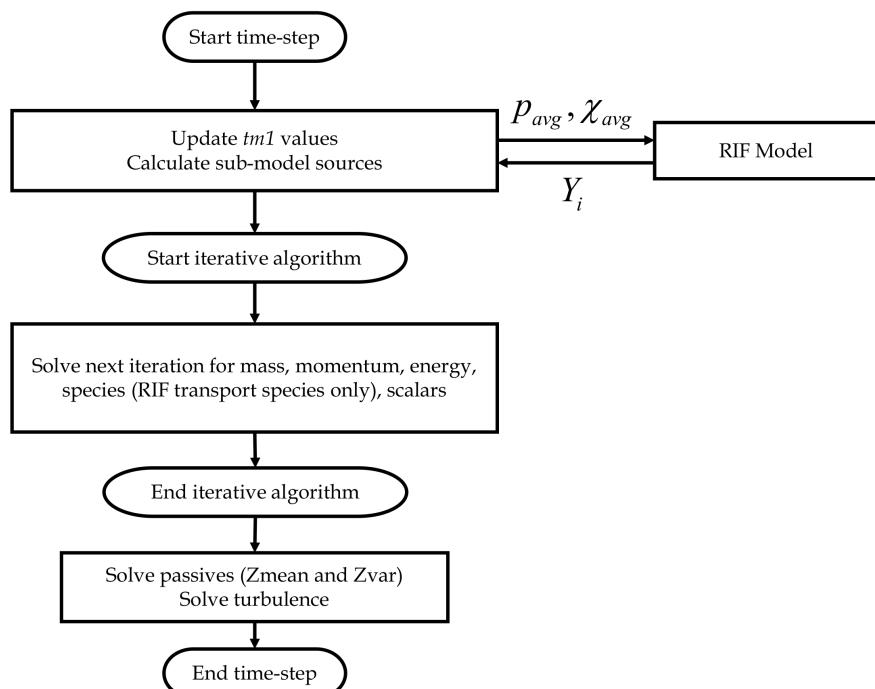


Figure 16.17: A schematic of the CONVERGE iterative algorithm coupling with RIF model.

Presumed Shape Probability Density Function Approach

Equations 16.226 through 16.231 are used to calculate the mean mixture fraction and the mixture fraction variance at each point of the turbulent flow field. CONVERGE solves equations for the turbulent flow field, the Reynolds stress equations (or the equation for the turbulent kinetic energy, \tilde{k}), and the equation for the dissipation, $\tilde{\varepsilon}$.

In this approach, a suitable two-parameter probability density function (PDF) is presumed in advance, thereby fixing the functional form of the PDF by relating the two parameters to known values of \tilde{Z} and $\widetilde{Z^{1/2}}$ at each point of the flow field.

Beta Function Probability Density Function Approach

In a two-feed system, the mixture fraction Z varies between $Z=0$ and $Z=1$, so the beta function PDF is commonly used for the Favre PDF in non-premixed turbulent combustion. The beta function has the form

Chapter 16: Combustion Modeling

Representative Interactive Flamelet (RIF) Model

$$\tilde{P}(Z; x, t) = \frac{Z^{\alpha-1} (1-Z)^{\beta-1}}{\Gamma(\alpha)\Gamma(\beta)} \Gamma(\alpha + \beta), \quad (16.232)$$

where Γ is a gamma function. The two parameters α and β are related to the Favre mean and variance by \tilde{Z} and $\widetilde{Z''^2}$ by

$$\begin{aligned}\alpha &= \tilde{Z}\gamma \\ \beta &= (1-\tilde{Z})\gamma,\end{aligned} \quad (16.233)$$

where

$$\gamma = \frac{\tilde{Z}(1-\tilde{Z})}{\widetilde{Z''^2}} - 1 \geq 0. \quad (16.234)$$

Using the presumed PDF approach, mean values of any quantity that depends only on the mixture fraction can be calculated. For instance, the mean value of ψ_i can be obtained from

$$\tilde{\psi}_i(x, t) = \int_0^1 \psi_i(Z, t) \tilde{P}(Z; x, t) dZ. \quad (16.235)$$

Clipped Gaussian PDF Approach

The β -PDF is not well-posed when the mean mixture fraction is exactly zero or exactly one. The clipped Gaussian PDF approach avoids this singular behavior by adopting the Gaussian distribution for $0 < Z < 1$ and a Dirac delta function at $Z = 0$ and $Z = 1$. In the Favre form, the clipped Gaussian PDF is given by

$$P(Z) = a_1 \delta(Z) + (1-a_1-a_2) \frac{G(Z)}{I_G} + a_2 \delta(1-Z), \quad (16.236)$$

where $\delta(Z-Z_0)$ is the Dirac delta function centered at $Z = Z_0$, whose integral is unity if $Z = Z_0$ and is zero otherwise. The parameter I_G is defined as

$$I_G = \int_0^1 G(Z) dZ. \quad (16.237)$$

Chapter 16: Combustion Modeling

Representative Interactive Flamelet (RIF) Model

The unknown free parameters a_1 and a_2 and the variance of the Gaussian distribution, g , are a function of the mean value of \tilde{Z} and the variance of the mixture fraction $\widetilde{Z}^{1/2}$. These parameters and variance can be evaluated using the computationally inexpensive method suggested by [Cleary \(2004\)](#).

Input Parameters for the RIF Model

The RIF model parameters are located in the [*combust.in*](#) > *rif_model* settings block.

16.10 Eddy Dissipation Model (EDM)

In the Eddy Dissipation Model (EDM), the rate of combustion is determined by the rate of eddy dissipation. The original formulation of this model ([Magnussen and Hjertager, 1977](#)) assumes a single-step chemical reaction of the form



where F is the fuel species, ν_{O_2} is the stoichiometric coefficient for O_2 , and P is the product. CONVERGE uses a generalized formulation that allows for a reaction mechanism with multiple reactions, multiple products, and/or multiple species with non-unity stoichiometric coefficients. Although there is no firm limit on the number of reactions, we recommend using one reaction when [*combust.in* > *eddy_dissipation_model* > *hybrid_flag* = 0](#) and up to two reactions when [*hybrid_flag* = 1](#).

In the original formulation of the EDM, the rate of combustion is calculated as the minimum of three mixing rates. The source term for fuel species transport can be written as

$$S_{F, EDM} = -A\tau_{turb}^{-1} \min\left(\rho_F, \frac{\rho_{O_2}}{r_F}, \frac{B\rho_P}{1+r_F}\right), \quad (16.239)$$

where A and B are user-defined constants, ρ is density (kg/m^3), and

$$r_F = \nu_{O_2} \frac{MW_{O_2}}{MW_F}, \quad (16.240)$$

where MW represents molecular weight. The turbulent time scale τ_{turb} is defined as

$$\tau_{turb} = \frac{k}{\varepsilon}, \quad (16.241)$$

where k is the turbulent kinetic energy (tke) and ε is the rate of dissipation of tke.

Chapter 16: Combustion Modeling

Eddy Dissipation Model (EDM)

The source terms for oxygen and the product species are given by

$$S_{O_2, EDM} = r_F S_{F, EDM} \quad (16.242)$$

and

$$S_{P, EDM} = -(1 + r_F) S_{F, EDM}, \quad (16.243)$$

respectively.

In the CONVERGE implementation of the EDM, the source term for each species m is calculated as

$$S_m = MW_m \dot{\omega}_m, \quad (16.244)$$

where the net production rate $\dot{\omega}_m$ is given by

$$\dot{\omega}_m = \sum_{i=1}^I (\nu''_{m,i} - \nu'_{m,i}) q_i. \quad (16.245)$$

In Equation 16.245, $\nu'_{m,i}$ and $\nu''_{m,i}$ are the stoichiometric coefficients for the reactants and products, respectively, for species m and reaction i , and I is the total number of reactions.

When `combust.in > eddy_dissipation_model > hybrid_flag = 0`, the rate parameter q_i is based on a generalization of Equation 16.239, given by

$$q_i = q_{i, EDM} = A \tau_{turb}^{-1} \min \left[\min_{\nu'_{m,i} \neq 0} \left(\frac{\rho_m}{\nu'_{m,i} MW_m} \right), \frac{B \sum_{\nu''_{m,i} \neq 0} \rho_m}{\sum_{\nu''_{m,i} \neq 0} \nu''_{m,i} MW_m} \right]. \quad (16.246)$$

When `combust.in > eddy_dissipation_model > hybrid_flag = 1`, CONVERGE compares the EDM mixing rate to the reaction rate computed from chemical kinetics. The smaller rate determines the rate of combustion. Thus,

$$q_i = \min(q_{i, EDM}, q_{i, kinetic}), \quad (16.247)$$

Chapter 16: Combustion Modeling

Eddy Dissipation Model (EDM)

where $q_{i,kinetic}$ is calculated from the Arrhenius form of the reaction rate coefficients, as shown in an [earlier section](#).

To use the EDM, activate turbulence modeling ([inputs.in](#) > *solver_control* > *turbulence_solver* = 1) and configure the model parameters in [combust.in](#) > *eddy_dissipation_model*. You can specify a maximum value for τ_{turb}^{-1} in [combust.in](#) > *eddy_dissipation_model* > *max_mixing_rate*.

Since the EDM is based on RANS turbulence modeling, we recommend choosing a [RANS turbulence model](#). Additionally, because the EDM is intended for continuous flow combustion, you should set the initial product species mass fractions to non-negligible values.

When [combust.in](#) > *eddy_dissipation_model* > *hybrid_flag* = 0, we recommend using a *mech.dat* file with only one reaction.

When [combust.in](#) > *eddy_dissipation_model* > *hybrid_flag* = 1, we recommend using a *mech.dat* file with no more than two reactions. To visualize which cells are using the rate from chemical kinetics, specify [post.in](#) > *cells* > *passives* > *edm_kinetic_cell*. CONVERGE sets *edm_kinetic_cell* = 1 for the cells in which $q_i = q_{i,kinetic}$ for at least one reaction.

16.11 Flamespeed Calculation

Turbulent Flamespeed Calculations

[Tracking the turbulent flame front](#) in the G-Equation model requires the turbulent flamespeed, s_t . For RANS turbulence models, a turbulent burning velocity relationship ([Peters, 2000](#)) is used to calculate the turbulent flamespeed as

$$s_t = s_l + u' \left\{ -\frac{a_4 b_3^2}{2b_1} Da + \left[\left(\frac{a_4 b_3^2}{2b_1} Da \right)^2 + a_4 b_3^2 Da \right]^{1/2} \right\}, \quad (16.248)$$

where u' is the root mean square of the turbulent fluctuating velocity; s_l is the [laminar flamespeed](#); a_4 , b_1 , and b_3 are modeling constants; and Da is the Damkohler number. The Damkohler number is calculated according to [Tan and Reitz \(2003\)](#) as

$$Da = \frac{s_l l_t}{u' \delta_l}, \quad (16.249)$$

where

Chapter 16: Combustion Modeling

Flamespeed Calculation

$$\delta_l = \frac{(\lambda / c_p)_o}{(\rho s_l)_u} \quad (16.250)$$

and

$$l_t = c_{\mu}^{3/4} \frac{k^{3/2}}{\varepsilon}. \quad (16.251)$$

In Equation 16.250, λ is the molecular conductivity, the subscript o indicates a cell value, and the subscript u indicates the unburned region value. If the [variance of G is solved](#) (i.e., if [combust.in > st_model > active = 11](#)), then the turbulent flamespeed is calculated from

$$s_t = s_l + u' \left\{ -\frac{a_4 b_3^2}{2b_1} l^* Da + \left[\left(\frac{a_4 b_3^2}{2b_1} l^* Da \right)^2 + a_4 b_3^2 l^* Da \right]^{1/2} \right\}, \quad (16.252)$$

where

$$l^* = \frac{l_{ft}}{\frac{k^{3/2}}{\varepsilon} \sqrt{\frac{2c_{\mu}}{c_s Sc}}} \quad (16.253)$$

and

$$l_{ft} = \frac{\sqrt{\widetilde{G}^{1/2}}}{\left| \partial \widetilde{G} / \partial x_i \right|}. \quad (16.254)$$

For LES turbulence models, the turbulent flamespeed is calculated ([Pitsch, 2002](#)) as

$$s_t = s_l \left(1 - \frac{b_3^2 s_l \mu_t}{2b_1 \mu u'} + \sqrt{\left(\frac{b_3^2 \mu_t s_l}{2b_1 \mu u'} \right)^2 + \frac{b_3^2 \mu_t}{\mu}} \right), \quad (16.255)$$

where s_l is the laminar flamespeed, u' is the sub-grid scale velocity, and b_1 and b_3 are modeling constants.

Chapter 16: Combustion Modeling

Flamespeed Calculation

Alternately, the turbulent flamespeed may be linked to the laminar flamespeed through the turbulent viscosity. This approach calculates the turbulent flamespeed as

$$s_t = s_l \sqrt{\frac{\mu_t}{\mu}}. \quad (16.256)$$

This approach is available for both [G-Equation](#) and [FGM](#).

Laminar Flamespeed Calculations

For a general fuel, the laminar flamespeed can be calculated in several different ways.

The first method is to use the [Metghalchi and Keck \(1982\)](#) correlation, given by

$$s_{l,ref} = B_m + B_2(\phi - \phi_m)^2, \quad (16.257)$$

where ϕ is the equivalence ratio and B_m , B_2 , and ϕ_m are user-supplied constants appropriate for the fuel and oxidizer used in the simulation.

The second method is to use the [Gulder \(1984\)](#) correlation, given by

$$s_{l,ref} = \omega \phi^\eta \exp[-\xi(\phi - 1.075)^2], \quad (16.258)$$

where ω , η , and ξ are user-supplied constants appropriate for the fuel and oxidizer used in the simulation.

For both the Metghalchi and Gulder methods, once the reference laminar flamespeed $s_{l,ref}$ is calculated at the reference pressure and temperature, the laminar flamespeed is adjusted for the actual pressure and temperature using the following equation:

$$s_l = s_{l,ref} \left(\frac{T_u}{T_{u,ref}} \right)^\gamma \left(\frac{P}{P_{ref}} \right)^\beta (1 - 2.1Y_{dil}), \quad (16.259)$$

where T_u is the unburned temperature, $T_{u,ref}$ is the reference unburned temperature, P is the pressure, P_{ref} is the reference pressure, Y_{dil} is the mass fraction of dilution species.

The temperature exponent in Equation 16.259 is defined as

$$\gamma = a_T + m_T(\phi - 1), \quad (16.260)$$

Chapter 16: Combustion Modeling

Flamespeed Calculation

where a_T and m_T are specified as [combust.in > sl_model > general_control > temp_a](#) and [sl_model > general_control > temp_m](#), respectively.

The pressure exponent in Equation 16.259 is defined as

$$\beta = a_p + m_p(\phi - 1), \quad (16.261)$$

where a_p and m_p are specified as [combust.in > sl_model > general_control > pres_a](#) and [sl_model > general_control > pres_m](#), respectively.

CONVERGE contains an additional option for laminar flamespeed correlation for gasoline (a reference gasoline with average molecular weight of 107 and H/C ratio of 1.69) for the Metghalchi and Gulder methods. If you invoke this option, the laminar flamespeed is correlated by the following equations:

$$\gamma_g = a_T + m_T \phi^{3.51} \quad (16.262)$$

and

$$\beta_g = a_p + m_p \phi^{2.77} - 0.008 P_{atm} / 6, \quad (16.263)$$

where $a_T = 2.18$, $m_T = -0.8$, $a_p = -0.16$, $m_p = 0.22$ and P_{atm} is the unburned pressure in *atm*. To activate this option for gasoline, set [combust.in > sl_model > active = 11](#) (Metghalchi) or 21 (Gulder).

Another method for calculating the laminar flamespeed of a general fuel makes use of user-supplied data tables. These data tables must have laminar flamespeed tabulated as a function of mixture fraction, temperature, pressure, and dilution fraction. You can use the one-dimensional chemistry utility to [generate a tabulated laminar flamespeed \(TLF\) table \(tlf_table.h5\)](#) prior to performing a combustion simulation. This table is an HDF5-formatted file and can be viewed using utilities provided by The HDF Group, such as HDFView.

To calculate the laminar flamespeed for methane combustion, the correlation of [Rahim et al. \(2002\)](#) is available. The laminar flamespeed is calculated as

$$s_l = s_{l,0} \left[1 + a_1(\phi - 1) + a_2(\phi - 1)^2 \right] \left(\frac{T_u}{T_{u,0}} \right)^\alpha \left(\frac{P}{P_0} \right)^\beta, \quad (16.264)$$

Chapter 16: Combustion Modeling

Flamespeed Calculation

where T_u is the unburned temperature, $T_{u,0} = 298 \text{ K}$ is the reference temperature, $P_0 = 101325 \text{ Pa}$ is the reference pressure, and $s_{l,0}$ is the reference laminar flamespeed of 0.7077 m/s . The model constants are given as $\alpha = 1.39$, $\beta = -0.016$, $a_1 = 0.237$, and $a_2 = -3.411$.

IFPEN Metghalchi and Keck Laminar Flamespeed Model

Set [combust.in > sl_model > active = 4](#) to enable the IFPEN Metghalchi and Keck laminar flamespeed model. This model is used primarily when running a simulation with the ECFM or ECFM3Z model. The laminar flamespeed is calculated by

$$s_l = s_{l,0} \left(\frac{T^u}{T_0} \right)^\alpha \left(\frac{P^u}{P_0} \right)^\beta \left(1 - 2.1X_{EGR} \right), \quad (16.265)$$

where

$$\begin{aligned} s_{l,0} &= B_m + B_\phi (\bar{\phi} - \phi_m)^2 \\ \alpha &= 2.18 - 0.8(\bar{\phi} - 1) \\ \beta &= -0.16 + 0.22(\bar{\phi} - 1) - \frac{0.008P_{atm}}{6}, \end{aligned} \quad (16.266)$$

$T_0 = 298 \text{ K}$, and $P_0 = 1.0 \text{ MPa}$.

The coefficients B_m , B_ϕ , and ϕ_m are known only for propane and iso-octane. These values are summarized below in the following table.

Table 16.10: Coefficients for propane and iso-octane.

Fuel	B_m	B_ϕ	ϕ_m
Propane	0.342	-1.387	1.08
Isooctane	0.263	-0.847	1.13

For a fuel with the formula C_xH_y , the coefficients are calculated via a linear interpolation:

$$K^{C_xH_y} = 0.2 \cdot ((8-x)K^{C_3H_8} + (x-3)K^{C_8H_{18}}), \quad (16.267)$$

where K is B_m , B_ϕ , or ϕ_m . Table 16.11 below summarizes the laminar flamespeed equation for various values of ϕ .

Chapter 16: Combustion Modeling

Flamespeed Calculation

Table 16.11: Laminar flamespeed equations.

Range of ϕ	Laminar flamespeed equation
$\phi < 0.5$	$s_l = s_{l,\phi=0.7} \cdot \phi \cdot \frac{6}{7}$
$0.5 \leq \phi < 0.7$	$s_l = s_{l,\phi=0.7} \cdot (\phi - 0.35) \cdot \frac{20}{7}$
$0.7 \leq \phi \leq 1.4$	$s_l = s_{l,0} \left(\frac{T^u}{T_0} \right)^\alpha \left(\frac{P^u}{P_0} \right)^\beta (1 - 2.1 X_{EGR})$
$1.4 < \phi \leq 2$	$s_l = s_{l,\phi=1.4} \cdot (2.2 - \phi) \cdot 1.25$
$\phi > 2.0$	$s_l = s_{l,\phi=1.4} \cdot (2.7 - \phi) \cdot \frac{2.5}{7}$

16.12 Surface Chemistry Model

CONVERGE has a surface chemistry model that can simulate heterogeneous reactions between a solid surface and an adjacent gas (*e.g.*, adsorption or desorption). You can simulate surface chemistry on two types of surfaces: a non-moving WALL boundary that is specified in [boundary.in](#) or a porous medium that is specified as a region in [initialize.in](#).

The surface chemistry model calculates the gas-phase chemistry only with the [SAGE detailed chemistry solver](#) (with or without [adaptive zoning](#) to accelerate the SAGE solver). You cannot use most of the other combustion models or other physical models (*e.g.*, radiation modeling or film modeling) in conjunction with surface chemistry modeling, although you can use [conjugate heat transfer modeling](#) with surface chemistry.

CONVERGE uses an empty site convention for surface chemistry in which the empty site is considered a species. With adsorption, the gas species replaces the empty surface site. With desorption, the empty site replaces the surface species as it becomes a gas.

To model surface chemistry, CONVERGE can use either a [segregated two-step](#) calculation or [coupled gas-phase chemistry to surface chemistry](#) calculations.

Surface Chemistry Theory

Given a gas/solid reaction



CONVERGE calculates the production rate for the solid species $B(s)$ and $C(s)$ as

Chapter 16: Combustion Modeling

Surface Chemistry Model

$$\dot{\omega}_B = \frac{d\theta_B}{dt} \quad (16.269)$$

and

$$\dot{\omega}_C = \frac{d\theta_C}{dt}, \quad (16.270)$$

respectively, in which θ is a unitless coverage fraction. Because θ is a conserved quantity,

$$\sum_{j=1}^J \theta_j = 1 \quad (16.271)$$

for each surface species j . We define the surface coverage of species j as

$$\theta_j = \frac{[X_j] \sigma_j}{\Gamma_s}, \quad (16.272)$$

in which $[X_j]$ is the molar concentration of species j , σ_j is the number of surface sites for species j , and Γ_s is the surface site density. For boundary cases, $[X_j]$ and Γ_s have units of mol/cm^2 . For porous region cases, $[X_j]$ and Γ_s have units of mol/cm^3 .

For solid-phase chemistry, CONVERGE calculates the production rate for the gas in the cell as

$$\dot{\omega}_A = \frac{d[X_A]}{dt} = R_A \left(\frac{A}{V} \right) \eta F, \quad (16.273)$$

in which R_A is the reaction rate defined in Equation 16.274, F is the user-specified washcoat factor, η is the user-specified [effectiveness factor](#), and A/V is the area of the catalytic surface divided by the volume of the gas-phase in the cell. For boundary cases, CONVERGE computes A/V with the cell properties, and for porous region cases, you specify A/V as the effective surface-to-volume ratio ([surface_chemistry.in > boundary_control > boundary > surface_area_to_volume_ratio](#)). The reaction rate for the gas species is

$$R_A = -k_f [X_A] \theta_B, \quad (16.274)$$

in which k_f is the forward rate coefficient. There are two types of forward reaction rate calculations for surface chemistry, which are shown below in the [Reaction Options](#) section.

Chapter 16: Combustion Modeling

Surface Chemistry Model

The CVODES solver ([SUNDIALS, 2015](#)) uses Equations 16.269, 16.270, and 16.273 to calculate the final Y_g'' , θ'' , and T'' values.

CONVERGE considers the production and consumption of gas-phase species at the surface to be a source term, S_A , given as

$$S_A = \frac{1}{\Delta t} \int_t^{t+\Delta t} \dot{\omega}_A M W_A dt \quad (16.275)$$

in which $M W_A$ is the molecular weight of species $A(g)$. By sub-cycling the time-step, Δt , CONVERGE approximates the source term as

$$S_A \approx \rho_g \frac{Y_g'' - Y_g}{\Delta t}, \quad (16.276)$$

CONVERGE solves the [species transport equation](#) with the resulting S_A value.

The surface chemistry reaction rate is a strong function of surface area or cell volume. For most applications, the chemistry will reach a steady state and thus changing the mesh resolution inside a porous region or near the surface chemistry surface during the simulation will introduce fluctuations to both species and temperature. You can avoid this problem by using a fixed grid.

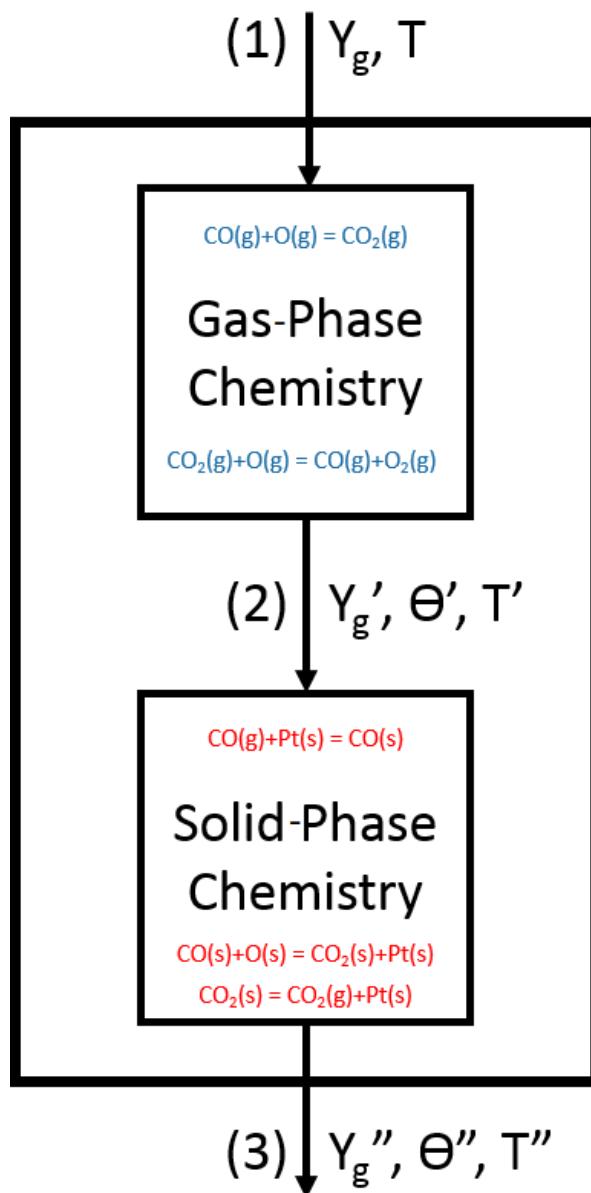
Two-Step Surface Chemistry

Figure 16.18: The two-step surface chemistry process in which Y_g is the gas-phase mass fraction and θ is the solid-phase species coverage fraction.

Figure 16.17 depicts the two-step calculation. First, CONVERGE calculates the gas-phase chemistry, which yields new gas-phase mass fraction and temperature data. Next, CONVERGE uses new values, along with the solid-phase coverage fraction data, to calculate solid-phase chemistry. Finally, CONVERGE updates all of the mass fraction, coverage fraction, and temperature data.

Chapter 16: Combustion Modeling

Surface Chemistry Model Two-Step Surface Chemistry

CONVERGE calculates the production rate for the gas species using Equations 16.268 to 16.276 during the gas-phase chemistry step.

In the solid-phase chemistry step, CONVERGE uses the updated gas mass fractions and temperature values along with the solid species coverage fractions as input.

Note that surface (solid) species are considered to be negligible in the calculation of enthalpy since surface species temperatures are unknown. Thus, [the governing equations for mass and energy](#) remain the same as for the gas-phase. It is important to note that the temperature obtained from Equation 16.80 is used only to update the rate coefficients rather than to update the CONVERGE cell temperature. The cell temperature is updated after the detailed chemistry calculation has converged using the computed species concentrations. In order to expedite the detailed chemistry calculations, kinetics are not solved in cells that fall below a minimum cell temperature (T_{cut} as specified as `surfchem_temp_cutoff` in `surface_chemistry.in`).

To activate the decoupled (two-step) calculation option, set `surface_chemistry_control > coupled_solver_flag = 0` in `surface_chemistry.in`.

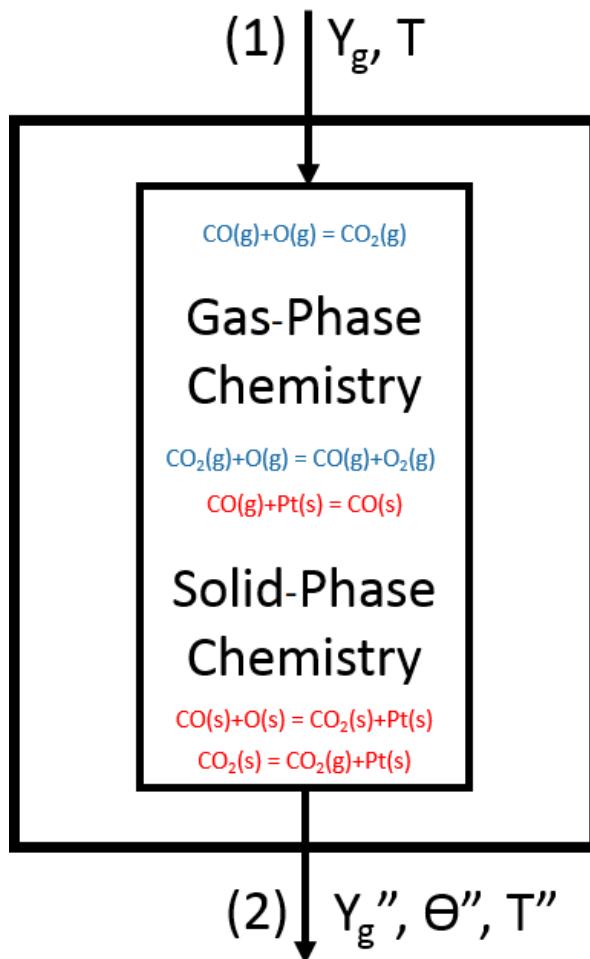
Coupled Gas-Surface Chemistry

Figure 16.19: The coupled gas-surface chemistry calculation in which Y_g is the gas-phase mass fraction and θ is the solid-phase species coverage fraction.

Figure 16.19 depicts the coupled gas-surface chemistry calculation in which CONVERGE calculates the gas-phase and solid-phase chemistry simultaneously using Equations 16.268 to 16.276 and the gas-phase equivalent Equations 16.68 to 16.78. CONVERGE updates the mass fraction, coverage fraction, and temperature data for the cell at the end of the calculation.

Surface (solid) species contribute to the calculation of enthalpy, and thus CONVERGE applies [the governing equations for mass and energy](#) to both the solid- and gas-phase species. In order to expedite the detailed chemistry calculations, kinetics are not solved in cells that fall below a minimum cell temperature ([`surface_chemistry.in > surface_chemistry_control > temp_cutoff`](#)).

To activate the coupled calculation option, set [`surface_chemistry.in > surface_chemistry_control > coupled_solver_flag = 1`](#).

Reaction Options

When calculating the forward reaction rate in the solid phase, there are two reaction options available: coverage-dependent or sticking probability. For the coverage-dependent reaction option, CONVERGE calculates the activation energy to determine the reaction rate. For the sticking reaction option, CONVERGE calculates the sticking coefficient to determine the reaction rate. The [surface mech.dat](#) file contains these options. Note that surface chemistry supports the [FORD](#) and [USER](#) options.

Coverage-Dependent (COV) Reaction Option

Gas-phase species that adsorb onto the surface can change the behavior of a surface reaction. CONVERGE computes the forward rate coefficient in a solid-phase reaction as the coverage-dependent Arrhenius expression:

$$k_{i,f} = A_i T^{\beta_i} \exp\left(-\frac{E_{a,i}}{RT}\right) \prod_{j=1}^J 10^{\chi_{ji}[\theta_j(n)]} [\theta_j(n)]^{\mu_{ji}} \exp\left[-\frac{\varepsilon_{ji}[\theta_j(n)]}{RT}\right], \quad (16.277)$$

in which χ_{ji} , μ_{ji} , and ε_{ji} are user-specified parameters. For a surface reaction i to be coverage dependent on multiple surface species j , you must list the three coverage parameters for each species.

Sticking (STICK) Reaction Option

For the sticking reaction option, CONVERGE computes the forward rate coefficients in solid-phase reactions in terms of sticking coefficients γ_i . Sticking coefficients quantify the probability that a gas molecule adsorbs when hitting the surface. We define the sticking coefficients, which are between 0 and 1, as

$$\gamma_i = a_i T^{\beta_i} \exp\left(-\frac{c_i}{RT}\right), \quad (16.278)$$

in which a_i , β_i , and c_i are user-specified parameters. The sticking coefficients are temperature dependent and increase with surface area.

We convert from a unitless sticking coefficient to a forward kinetic rate coefficient:

$$k_{i,f} = \gamma_i \frac{\prod_{j=1}^J \sigma_j^{\nu'_{ji}}}{(\Gamma_{tot})^y} \sqrt{\frac{RT}{2\pi MW_j}} \quad (16.279)$$

Chapter 16: Combustion Modeling

Surface Chemistry Model Reaction Options

in which ν'_{ji} is the reaction order for that surface species in reaction i , y is the sum of all the stoichiometric coefficients of surface species reactants, and MW_j is the molecular weight of species j .

[Motz and Wise \(1960\)](#) showed that Equation 16.279 becomes less accurate when the sticking coefficient is close to unity. They proposed the following correction to the forward rate coefficient:

$$k_{i,f} = \left(\frac{\gamma_i}{1 - \frac{\gamma_i}{2}} \right) \prod_{j=1}^J \sigma_j^{\nu'_{ji}} \frac{\sqrt{RT}}{(\Gamma_{tot})^y} \sqrt{\frac{RT}{2\pi MW_j}} \quad (16.280)$$

You may wish to activate the Motz-Wise correction factor if your simulation has a sticking coefficient near one. Refer to [surface_mech.in](#) for details.

Reverse Reaction Rate

You can specify the Arrhenius coefficients for the reverse reaction with the keyword *REV*. The forward and reverse rate coefficients are proportional to one another, and their proportionality constant is equal to the equilibrium coefficient, K_{cr} . The equilibrium coefficient for surface reactions accounts for both solid and gas-phase species with the following relationship:

$$K_{cr} = \left(\frac{P_{atm}}{RT} \right)^{\sum_{m=1}^M \nu_{mr}} \prod_{n=1}^N (\Gamma_s^o)^{\sum_{j=1}^J \nu_{ji}} \prod_{j=1}^J \sigma_j^{-\nu_{ji}} K_{pr} \quad (16.281)$$

in which P_{atm} is the atmospheric pressure, R is the ideal gas constant, T is the temperature, and $\sigma_j^{-\nu_{ji}}$ is the user-specified surface site occupancy of surface species j . Equation 16.75 defines the gas equilibrium constant, K_{pr} . In Equation 16.281, the first summation is over the gas-phase species and the second summation over the surface species j of surface site type n . CONVERGE updates the surface species concentrations, gas species concentrations, and temperature.

Note that surface chemistry supports the [FORD](#) and [USER](#) options.

Effectiveness Factor

The effectiveness factor η in Equation 16.273 accounts for diffusion through the washcoat to the catalyst and can range from 0 to 1. In a thin (*i.e.*, a few mm) washcoat, the gas species diffusion into the catalyst sites may be fast, and thus you can consider the effectiveness factor to be one. In a thick washcoat, the gas species diffusion is slower and η is less than

Chapter 16: Combustion Modeling

Surface Chemistry Model Effectiveness Factor

one. The [*surface_chemistry.in > effectiveness_factor_control*](#) settings block contains the relevant parameters.

CONVERGE solves for the effectiveness factor

$$\eta = \frac{\tanh \phi}{\phi} \quad (16.282)$$

using the Thiele modulus ([Deutschmann et al., 2014](#))

$$\phi = L \sqrt{\frac{\dot{\omega}_j F}{D_{j,eff} c_j L}} \quad (16.283)$$

In Equation 16.283, F is the user-specified washcoat factor, L is the user-specified washcoat thickness and $D_{j,eff}$ is the effective diffusion coefficient for the user-specified surface species j . CONVERGE solves for the effective diffusion coefficient via

$$D_{eff} = \frac{\varepsilon_p}{\tau} \bar{D}_j \quad (16.284)$$

in which ε_p is the user-specified washcoat porosity and τ is the user-specified washcoat tortuosity. The mixed diffusion coefficient, \bar{D}_j , is defined as

$$\frac{1}{\bar{D}_j} = \frac{1}{D_{mol}} + \frac{1}{D_{knud,j}} \quad (16.285)$$

To solve for the mixed diffusion coefficient, CONVERGE calculates the Knudsen diffusion coefficient as

$$D_{knud,j} = c \frac{\varepsilon_p}{\tau} \frac{d_p}{3} \sqrt{\frac{8RT}{\pi M W_j}} \quad (16.286)$$

where both c is a constant, D is the molecular diffusion coefficient (specified via [*inputs.in > property_control > species_diffusion_model*](#)), and MW_j is the molecular weight of species j .

In Equation 16.286, d_p is the user-specified pore diameter. CONVERGE calculates the molecular diffusion coefficient using the CONVERGE-calculated viscosity ν of the cell and the Schmidt number Sc .

16.13 Adaptive Zoning

CONVERGE includes adaptive zoning ([Babajimopoulos et al., 2005](#)), which accelerates the chemistry calculations by grouping together similar computational cells and then invoking the chemistry solver once per group rather than once per cell. Adaptive zoning was implemented jointly by Convergent Science and Lawrence Livermore National Laboratory.

General Procedure

At any given time t , each cell is at some thermodynamic state. Based on the thermodynamic states of the cells, CONVERGE groups the cells into [zones](#). The thermodynamic state of each zone is based on the average temperature and composition of all of the cells in that zone. CONVERGE invokes the chemistry solver once on each zone.

The chemical kinetic equations for a closed-volume homogeneous reactor of each zone can be written as

$$\frac{\partial Y_k}{\partial t} \Big|_{\text{zone}} = \frac{\dot{\omega}_k M W_k}{\rho} \Big|_{\text{zone}} \quad (k=1,\dots,K) \quad (16.287)$$

and

$$\frac{\partial T}{\partial t} \Big|_{\text{zone}} = -\frac{1}{\rho \bar{c}_v} \sum_{k=1}^K \dot{\omega}_k M W_k e_k \Big|_{\text{zone}}, \quad (16.288)$$

where $\dot{\omega}_k$ is the production/consumption rate of species k ; $M W_k$ and e_k are the molecular weight and the specific internal energy of the species k , respectively; and \bar{c}_v is the constant volume specific heat of the gas mixture. All of these quantities are zone-based values. Each zone is allowed to react from time t to $t+\Delta t$. Once the new zone composition is obtained, CONVERGE maps the zonal temperatures and mass fractions to the computational cells such that temperature and composition non-uniformities are preserved. The [mapping strategy](#), which is described below, is critical to ensure the quality of the solution.

Zoning Strategy

When adaptive zoning is active, CONVERGE groups cells into zones (also known as bins). Figure 16.20 below shows an example of the zoning process. It is important to note that the zoning strategy in CONVERGE is similar but not identical to the strategy in [Babajimopoulos et al. \(2005\)](#).

Chapter 16: Combustion Modeling

Adaptive Zoning

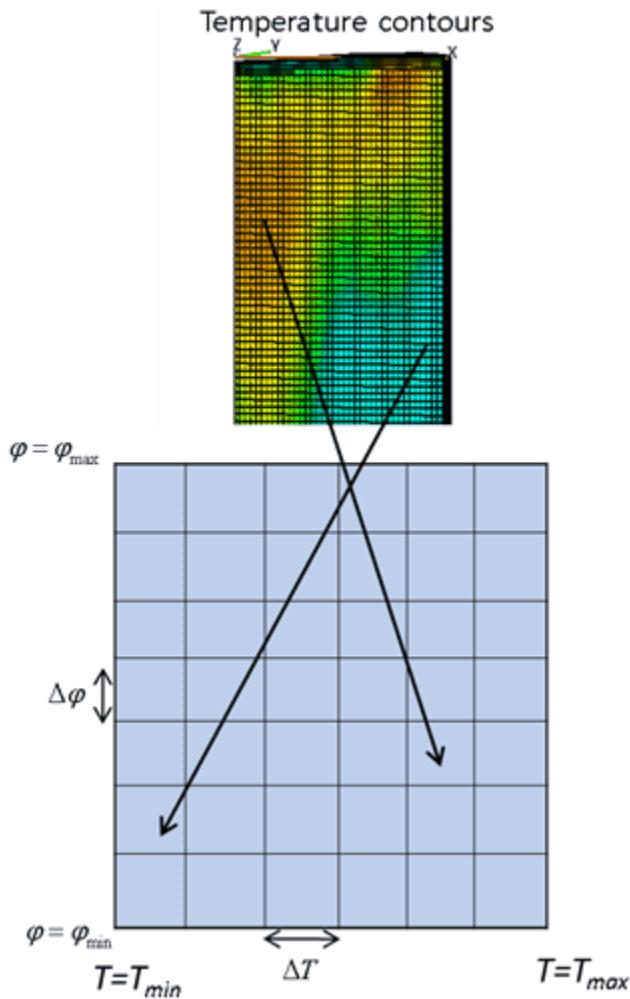


Figure 16.20: An example zoning process based on temperature and progress equivalence ratio.

CONVERGE has a multi-dimensional zoning strategy, which means different variables can be specified for the zoning process. Temperature and progress equivalence ratio (defined below) are required. You can select additional variables (pressure, total equivalence ratio, the cube root of the mass fraction of a specified species) if desired. The number of variables is the number of dimensions of the zoning strategy. In general a two-dimensional (temperature and progress equivalence ratio) zoning strategy works well for single fuel simulations. A higher-dimension strategy may be required for multi-fuel applications. Specify the zoning strategy variables in the [combust.in > adaptive_zone_model > bin_options](#) settings block.

Note that cells that belong to different regions are automatically placed in different zones.

The progress equivalence ratio, φ , and the total equivalence ratio, φ^t , are given as

Chapter 16: Combustion Modeling

Adaptive Zoning

$$\varphi = \frac{2C_{-CO_2}^{\#} + \frac{H_{-H_2O}^{\#}}{2}}{O_{-CO_2-H_2O}^{\#}} \quad (16.289)$$
$$\varphi^t = \frac{2C^{\#} + \frac{H^{\#}}{2}}{O^{\#}},$$

where $C^{\#}$ is the number of carbon atoms, $H^{\#}$ is the number of hydrogen atoms, $C_{-CO_2}^{\#}$ is the number of carbon atoms excluding CO₂, and $H_{-H_2O}^{\#}$ is the number of hydrogen atoms excluding H₂O. Progress equivalence ratio is preferable to total equivalence ratio as the progress ratio tracks the progress of the reactions.

In CONVERGE, the number of zones is not predetermined. Instead, the number of zones varies dynamically with the stratification of the flow field parameters. The mass of each zone is non-uniform as the cells are distributed non-uniformly.

The accuracy of SAGE with adaptive zoning depends on the user-supplied zone sizes. Accuracy increases as the zone size is reduced, but the computational effort increases. There are two methods to specify bin sizes. You can have some variables with one bin size method and other variables with the other bin size method.

The first option is a fixed bin size. For example, you could have a fixed temperature bin size of 10 K or a fixed progress equivalence ratio bin size of 0.1. The second option is a variable bin size, where the bin size can vary as a function of the bin variable. You can use the second option, for example, to refine the zones in the temperature range in which combustion is most important. When appropriately configured, a variable bin size saves computational time compared to a fixed (fine) bin size for the entire temperature range. The bin size options are controlled by [combust.in > adaptive_zone_model > bin_options](#).

Mapping Strategy

For adaptive zoning, CONVERGE uses the mapping technique outlined by [Babajimopoulos et al. \(2005\)](#). This mapping is based on the ch value of the individual cells in the zone. The ch value of an individual cell is defined as

$$ch_{cell} = 2C_{-CO_2}^{\#} + \frac{H_{-H_2O}^{\#}}{2}, \quad (16.290)$$

where $C_{-CO_2}^{\#}$ is the number of carbon atoms excluding CO₂ and $H_{-H_2O}^{\#}$ is the number of hydrogen atoms excluding H₂O. CONVERGE calculates the ch value for each cell and each zone before starting the zonal chemical calculations. The sum of all the ch values of the individual cells (ch_{cell}) in a zone will be equal to the ch value of the zone (ch_{zone}).

Chapter 16: Combustion Modeling

Adaptive Zoning

CONVERGE uses the following equation to calculate all the individual species except CO₂, H₂O, O₂, and N₂L:

$$m_{k,cell} = \frac{ch_{cell}}{ch_{zone}} m_{k,zone}. \quad (16.291)$$

CONVERGE attempts to conserve the carbon, hydrogen, and oxygen atoms in each cell by readjusting the mass of CO₂, H₂O, and O₂ as follows:

$$\sum_k \frac{m_{k,cell}}{MW_k} c_k + \frac{m_{CO_2,cell}}{MW_{CO_2}} = C_{cell}^{\#}, \quad (16.292)$$

$$\sum_k \frac{m_{k,cell}}{MW_k} h_k + 2 \frac{m_{H_2O,cell}}{MW_{H_2O}} = H_{cell}^{\#}, \quad (16.293)$$

and

$$\sum_k \frac{m_{k,cell}}{MW_k} o_k + 2 \frac{m_{O_2,cell}}{MW_{O_2}} = O_{cell}^{\#}, \quad (16.294)$$

where $m_{k,cell}$ is the mass of species k in a cell; MW_k is the molecular weight of species k ; c_k , h_k and o_k are the number of carbon, hydrogen, and oxygen atoms in species k ; and $C_{cell}^{\#}$, $H_{cell}^{\#}$, and $O_{cell}^{\#}$ are the number of moles of carbon atoms, hydrogen atoms and oxygen atoms in a given cell.

The total mass in an individual cell is conserved by readjusting the mass of the remaining N₂. If there is no N₂ in the system, or if any of the CO₂, H₂O, or O₂ masses are negative, CONVERGE applies the mapping of [Liang et al. \(2009\)](#). This calculates the density of each species k in cell i at the next time-step $t+dt$, $\rho_{k,i}^{t+dt}$, as

$$\begin{aligned} \rho_{k,i}^{t+dt} &= \rho_{k,i}^t + \Delta m_{k,zone} \frac{\rho_i^t}{\sum_{l=1}^K \rho_l^t V_l} && \text{if } \Delta m_{k,zone} > 0, \\ \rho_{k,i}^{t+dt} &= \rho_{k,i}^t + \Delta m_{k,zone} \frac{\rho_{k,i}^t}{\sum_{l=1}^K \rho_l^t V_l} && \text{if } \Delta m_{k,zone} < 0, \end{aligned} \quad (16.295)$$

where $\Delta m_{k,zone}$ is the change of the total mass of species k in the adaptive zone. The summation in the denominator is for all the cells in the adaptive zone where l indicates the cell index in the adaptive zone and K is the total number of cells in the adaptive zone. ρ_i indicates the density of cell i and $\rho_{k,i}$ indicates the density of species k in cell i .

For improved NOx predictions, CONVERGE allows mapping based on nitrogen atoms ([*combust.in*](#) > *adaptive_zone_model* > *nox_flag*).

Heat release mapping, which allows mapping based on heat release rate and the species mass fractions, is available in CONVERGE. The [*combust.in*](#) > *adaptive_zone_model* > *hr_map_flag* parameter controls this option. Heat release mapping maps the species mass fraction based on the *ch* value described above. The ch_{cell} value calculated in Equation 16.291 is then used to map the zone heat release rate ($\Delta H_{r,zone}$) to the individual cell heat release rate ($\Delta H_{r,cell}$) according to

$$\Delta H_{r,cell} = \frac{ch_{cell}}{ch_{zone}} \Delta H_{r,zone}. \quad (16.296)$$

When performing heat release mapping, the internal energy before and after combustion are not assumed to be identical. This is in contrast to mapping based on only species mass fractions, when the internal energy is assumed to be the same both before and after combustion. The source for sensible internal energy is calculated based on both the mapped species mass fractions and the heat release rate.

Input Parameters for Adaptive Zoning

The adaptive zoning parameters are located in the [*combust.in*](#) > *adaptive_zone_model* settings block.

16.14 Skip Species

The species transport calculations, which CONVERGE performs when [*inputs.in*](#) > *solver_control* > *species_solver* = 1, may be computationally expensive. The skip species feature allows CONVERGE to avoid transporting species that have insignificant mass. By skipping species that are present only in very small amounts (and by invoking this feature only at certain times in the simulation), CONVERGE reduces the time required for the species transport calculations without compromising the accuracy of the simulation results. Skip species is available only in simulations in which [*combust.in*](#) > *sage_model* > *active* is non-zero. CONVERGE does not allow the time periods during which skip species is active to overlap with the time periods during which the combustion model is active.

At the skip species start time, CONVERGE sorts the species in descending order by mass. Beginning at the top of the list, CONVERGE retains species until the cumulative mass equals or exceeds the user-specified percentage of the total mass. You can force CONVERGE to keep specific species (e.g., the fuel and oxidizer in an engine simulation) regardless of mass. In this case, CONVERGE sums the mass of the "keep" species and then adds species from the mass-ordered list until the sum equals or exceeds the user-specified percentage of total mass.

CONVERGE converts the discarded hydrocarbons into retained hydrocarbons. There are two options for the conversion process. You can specify the species and mass fraction(s) to which the discarded hydrocarbons will be converted. Alternatively, if you do not specify species and mass fractions, CONVERGE will convert the discarded hydrocarbons into the retained hydrocarbons according to the mass fractions of the retained hydrocarbons. Analogous options are available for converting the discarded non-hydrocarbons. It is important to note that the conversions do not conserve the number of atoms in the simulation. Also, to avoid changes to total enthalpy due to the conversions, CONVERGE adjusts the cell temperatures and thus the post-conversion mass may be slightly different.

To activate skip species, set [*inputs.in*](#) > *feature_control* > *skip_species_flag* = 1 and include the [*skip_species.in*](#) file in your case setup.

16.15 Combustion Time-Step Control

CONVERGE allows you to specify a variable time-step based on combustion. If the ignition or combustion model is active, the maximum time-step for combustion is given by

$$dt_chem = dt * \min\left(\frac{T}{\Delta T}\right)^* mult_dt_chem, \quad (16.297)$$

where *dt* is the current time-step, [*inputs.in*](#) > *temporal_control* > *mult_dt_chem* is a user-specified multiplier, *T* is the initial cell temperature for the current time-step, and ΔT is the change in cell temperature due to combustion.

16.16 Combustion-Related Output

Mixture Fraction

CONVERGE will calculate the mixture fraction if [*combust.in*](#) > *mix_frac_output* > *mix_frac* > *active* = 1. The mixture fraction is calculated from

$$Z = \sum_{i=1}^{n_s} \frac{N_{C_i} MW_C Y_i}{MW_i} + \sum_{i=1}^{n_s} \frac{N_{H_i} MW_H Y_i}{MW_i}, \quad (16.298)$$

where *i* is the species; *n_s* is the number of species; *N_{C,i}* and *N_{H,i}* are the number of carbon and hydrogen atoms, respectively, in species *i*; *MW_C* and *MW_H* are the molecular weights of

Chapter 16: Combustion Modeling

Combustion-Related Output

carbon and hydrogen, respectively; Y_i is the species mass fraction; and MW_i is the molecular weight of species i .

CONVERGE will calculate the mixture fraction variance if [combust.in > mix_frac_output > mix_frac_var](#) > active = 1. The mixture fraction variance is calculated as

$$\frac{\partial}{\partial t} (\rho Z^{1/2}) + \frac{\partial}{\partial x_i} (\rho u_i Z^{1/2}) = \frac{\partial}{\partial x_i} \left(\frac{\mu_t}{Sc_t} \frac{\partial Z^{1/2}}{\partial x_i} \right) + \frac{2\mu_t}{Sc_t} \left(\frac{\partial Z}{\partial x_i} \right)^2 - \rho \chi. \quad (16.299)$$

The scalar dissipation, χ , is modeled by

$$\chi = c_\chi \frac{\varepsilon}{k} Z^{1/2}, \quad (16.300)$$

where c_χ is a model constant ([combust.in > mix_frac_output > c_chi](#)). Note that, if you set [combust.in > mix_frac_output > mix_frac_var > active = 1](#), you must include a [passive](#) called [mix_frac_var](#) in [species.in](#).

Post-Processing Parameters

CONVERGE will keep track of the hydrocarbon species and CO, H, and H₂ if you include the keyword *hc* in [species.in](#) as a non-transport [passive](#). Figure 16.21 below shows an example excerpt of [species.in](#).

```
:
:
passive:
    hiroy_soot:      0.78
    nox:             0.78
passive_nt:
    - hc
```

Figure 16.21: Excerpt of [species.in](#) that includes a non-transport passive for hydrocarbons (*hc*).

CONVERGE will write all relevant output quantities (mass, mean and standard deviation) for the hydrocarbons to [passive.out](#). To generate cell-by-cell information related to hydrocarbons, define *hc* as [post.in > passive > hc](#). In G-Equation simulations, you can generate cell-by-cell information related to the value of *G* by including [post.in > passive > G_EQN](#).

Chapter



17

Emissions Modeling

17 Emissions Modeling

CONVERGE features many models to simulate [soot](#) or [NOx](#) production. These models are described in this chapter.

Species of interest such as CO, CO₂, and unburned hydrocarbons are always calculated or interpolated in CONVERGE, provided they are included in the [reaction mechanism file](#) and are a part of the combustion model used by the simulation.

To model emissions, activate [combustion modeling](#) (set [inputs.in > feature_control > combustion_flag = 1](#) and include a [combust.in](#) file in your case setup), set [combust.in > emissions_flag = 1](#), and include an [emissions.in](#) file in your case setup.

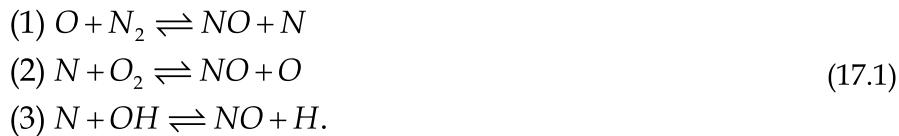
17.1 NOx Modeling

CONVERGE offers three options for predicting NOx formation: the [thermal NOx model](#), the [prompt NOx model](#), and [NOx calculation via the SAGE detailed chemical kinetics solver](#). The thermal NOx and prompt NOx models use internal passives to calculate NOx and do not require the NOx kinetics mechanism in the [reaction mechanism file](#) (e.g., *mech.dat*). You must specify NOx reactions in the [reaction mechanism file](#) when calculating NOx with SAGE.

The [emissions.in > nox_model](#) settings block contains parameters for each NOx model. Additionally, the Schmidt number ([inputs.in > property_control > schmidt_turb](#)) is sometimes adjusted to tune NOx with typically only a small effect on the global combustion characteristics. In general, for diffusion flames, as the Schmidt number increases, the predicted fuel-air mixing decreases, which yields a lower NOx concentration. For premixed flames, as the Schmidt number increases, the turbulent diffusivity decreases, which yields a higher NOx concentration.

Thermal NOx Model

The extended Zel'dovich mechanism as presented by [Heywood \(1988\)](#) is used to calculate NO formation. This mechanism is given by the following set of reactions:



By default, the rate constants for the reactions in Equation 17.1 are given by

Chapter 17: Emissions Modeling

NOx Modeling Thermal NOx Model

$$\begin{aligned}
 k_{1,f} &= 7.6 \times 10^{13} \exp\left(\frac{-38,000}{T}\right) \\
 k_{1,r} &= 1.6 \times 10^{13} \\
 k_{2,f} &= 6.4 \times 10^9 T \exp\left(\frac{-3,150}{T}\right) \\
 k_{2,r} &= 1.5 \times 10^9 T \exp\left(\frac{-19,500}{T}\right) \\
 k_{3,f} &= 4.1 \times 10^{13} \\
 k_{3,r} &= 2.0 \times 10^{14} \exp\left(\frac{-23,650}{T}\right),
 \end{aligned} \tag{17.2}$$

where the subscript *f* denotes a forward reaction and the subscript *r* denotes a reverse reaction. The units of the rate constants in Equation 17.2 are $\text{cm}^3/\text{mol}\cdot\text{s}$. You can specify alternate rate constants in [passive_nox_rate.dat](#).

From the reactions in Equation 17.1, the rate of formation of NO can be written as

$$\begin{aligned}
 \frac{d[\text{NO}]}{dt} &= k_{1,f} [\text{O}][\text{N}_2] - k_{1,r} [\text{NO}][\text{N}] + k_{2,f} [\text{N}][\text{O}_2] - k_{2,r} [\text{NO}][\text{O}] \\
 &\quad + k_{3,f} [\text{N}][\text{OH}] - k_{3,r} [\text{NO}][\text{H}],
 \end{aligned} \tag{17.3}$$

where $[\text{NO}]$, for example, denotes the species concentration of NO in mol/cm^3 . Equation 17.3 can be rewritten by assuming that the steady-state approximation is valid for $[\text{N}]$. The rate of formation of $[\text{N}]$ can be written as

$$\begin{aligned}
 \frac{d[\text{N}]}{dt} &= k_{1,f} [\text{O}][\text{N}_2] - k_{1,r} [\text{NO}][\text{N}] - k_{2,f} [\text{N}][\text{O}_2] + k_{2,r} [\text{NO}][\text{O}] \\
 &\quad - k_{3,f} [\text{N}][\text{OH}] + k_{3,r} [\text{NO}][\text{H}].
 \end{aligned} \tag{17.4}$$

If $d[\text{N}]/dt$ is set equal to 0 in Equation 17.4, $[\text{N}]$ can be eliminated in Equation 17.3. It can be shown that the formation rate of $[\text{NO}]$ then becomes

$$\frac{d[\text{NO}]}{dt} = 2k_{1,f} [\text{O}][\text{N}_2] \frac{1 - [\text{NO}]^2 / (K[\text{O}_2][\text{N}_2])}{1 + k_{1,r} [\text{NO}] / (k_{2,f} [\text{O}_2] + k_{3,f} [\text{OH}])}, \tag{17.5}$$

Chapter 17: Emissions Modeling

NOx Modeling Thermal NOx Model

where

$$K = \left(k_{1,f} / k_{1,r} \right) \left(k_{2,f} / k_{2,r} \right) \quad (17.6)$$

If equilibrium values of $[O]$, $[OH]$, and $[H]$ are assumed (set `emissions.in > nox_model > nox_radical_model = 0`), Equation 17.5 can be rewritten as

$$\frac{d[NO]}{dt} = \frac{2R_1 \left[1 - \left(\frac{[NO]}{[NO]_{eq}} \right)^2 \right]}{1 + \left(\frac{[NO]}{[NO]_{eq}} \right) R'}, \quad (17.7)$$

where

$$R' = \frac{R_1}{R_2 + R_3} \quad (17.8)$$

and

$$\begin{aligned} R_1 &= k_{1,r} [NO]_{eq} [N]_{eq} \\ R_2 &= k_{2,f} [N]_{eq} [O_2]_{eq} \\ R_3 &= k_{3,f} [N]_{eq} [OH]_{eq}, \end{aligned} \quad (17.9)$$

where the subscript eq is used to denote the equilibrium species concentration.

Equations 17.7 through 17.9 require that equilibrium species concentrations are known. It is assumed that $[H_2]$ is in equilibrium at the local conditions. $[NO]_{eq}$ can be calculated from the equilibrium constants K_1 and K_2 for reactions (1) and (2) in Equation 17.1, as follows:

$$K_1 = \frac{k_{1,f}}{k_{1,r}} = \frac{[NO]_{eq} [N]_{eq}}{[O]_{eq} [N_2]_{eq}} \quad (17.10)$$

and

Chapter 17: Emissions Modeling

NOx Modeling Thermal NOx Model

$$K_2 = \frac{k_{2,f}}{k_{2,r}} = \frac{[NO]_{eq} [O]_{eq}}{[N]_{eq} [O_2]_{eq}}. \quad (17.11)$$

By eliminating $[N]_{eq}/[O]_{eq}$ in Equations 17.10 and 17.11, it can be shown that

$$[NO]_{eq} = \sqrt{K_1 K_2 [O_2]_{eq} [N_2]_{eq}}. \quad (17.12)$$

Similarly, $[N]_{eq}$ can be calculated from reaction (2) via

$$[N]_{eq} = \frac{k_{2,r}}{k_{2,f}} \frac{[NO]_{eq} [O]_{eq}}{[O_2]_{eq}}, \quad (17.13)$$

which requires $[O]_{eq}$ to be known. As described by [Heywood \(1988\)](#), $[O]_{eq}$ can be determined from

$$[O]_{eq} = \frac{K_O}{(RT)^{1/2}} [O_2]_{eq}^{1/2}, \quad (17.14)$$

which is derived from the reaction



In Equation 17.14, K_O is the equilibrium constant given by [Heywood \(1988\)](#) as

$$K_O = 3.6 \times 10^3 \exp\left(\frac{-31,090}{T}\right) \quad (atm^{1/2}) \quad (17.16)$$

and R is the ideal gas constant in $atm\cdot cm^3/K\cdot mol$. Finally, the $[OH]_{eq}$ can be calculated from

$$[OH]_{eq} = \left(K_{OH} [O_2]_{eq} [H_2]_{eq}\right)^{1/2}, \quad (17.17)$$

which is derived from the reaction



Chapter 17: Emissions Modeling

NOx Modeling Thermal NOx Model

The equilibrium constant, $K_{OH'}$, in Equation 17.17 has been obtained from a curve fit and is given by

$$K_{OH} = 23.4936 \exp\left(\frac{-8567.17}{T}\right). \quad (17.19)$$

With the equilibrium constants calculated, Equation 17.7 can now be solved by rewriting it as

$$\frac{d\alpha}{dt} = \frac{2R_1[1-\alpha^2]}{[NO]_{eq}(1+\alpha R')}, \quad (17.20)$$

where

$$\alpha = \frac{[NO]}{[NO]_{eq}}. \quad (17.21)$$

Integration of Equation 17.20 yields

$$\alpha_2 - \alpha_1 = \beta(1 - \alpha_2^2), \quad (17.22)$$

where

$$\beta = \frac{2R_1 dt}{[NO]_{eq}(1 + \alpha R')}, \quad (17.23)$$

and it is assumed that α in the term $(1 - \alpha_2^2)$ on the right side of Equation 17.22 is evaluated at the end of the computational time-step dt . Recasting Equation 17.22 in the form of a quadratic equation leads to the following solution for the updated value of α :

$$\alpha_2 = \frac{-1 + \sqrt{1 + 4\beta(\alpha_1 + \beta)}}{2\beta}, \quad (17.24)$$

where β , given by Equation 17.23, is evaluated using α_1 , which is the value of α at the start of the computational time-step. Equation 17.24 is used to calculate the updated value of α and Equation 17.21 is then used to calculate $[NO]$ at the end of the time-step. It can be shown that

$\alpha_2 \rightarrow \alpha_1$ in the limit of $\beta \rightarrow 0$. This limit is recovered in the CONVERGE implementation of Equation 17.24.

Concentration of O and OH Radicals

For modeling NOx, CONVERGE employs several methods for computing the concentration of O and OH radicals.

In CONVERGE, the equilibrium approach ([*emissions.in*](#) > *nox_model* > *nox_radical_model* = 0) assumes equilibration of combustion reactions. Such an assumption is possible because the kinetics of the thermal NOx formation rate are slower than the main hydrocarbon oxidation rate and thus most of the thermal NOx forms after combustion is complete. Additionally, this assumption is valid for high temperature (> 2200 K) combustion. For this approach, Equations 17.14 and 17.17 give the formulations for the concentration of O and OH radicals, respectively.

The partial equilibrium approach ([*emissions.in*](#) > *nox_model* > *nox_radical_model* = 1) for computing the concentration of O radicals accounts for third-body reactions in the O₂ dissociation-recombination process to improve the equilibrium approach described previously. Equation 17.25 describes the third-body reactions and Equations 17.26 to 17.27 describe the equations for radical concentration ([Warnatz, 2001](#)).



With this approach, the equilibrium O atom concentration is

$$[O] = 36.64 T^{1/2} [O_2]^{1/2} \exp\left(\frac{-27123}{T}\right). \quad (17.26)$$

To compute the concentration of the OH radical, CONVERGE uses

$$[OH] = 2.129e2 T^{-0.57} \exp\left(\frac{-4595}{T}\right) [O]^{1/2} [H_2O]^{1/2}. \quad (17.27)$$

The third approach ([*emissions.in*](#) > *nox_model* > *nox_radical_model* = 2) requires a combustion model that invokes the [SAGE detailed chemical kinetics solver](#) (e.g., [SAGE](#) by itself, or a [G-Equation](#) or [FGM](#) option that invokes SAGE). Based on reactions in the [reaction mechanism file](#), SAGE predicts the O atom concentration and CONVERGE simply takes the concentration of O radicals as the local O species mass fraction. Likewise, CONVERGE takes the concentration of OH radicals from the local OH species mass fraction.

Thermal NOx Model Setup

To activate the [thermal NOx](#) (based on the Extended Zel'dovich mechanism) model, set [*emissions.in*](#) > *nox_model* > *nox_thermal* > *active* = 1. You can adjust the model through

parameters in [*emissions.in*](#), including *nox_model > nox_thermal > rate_flag* (which requires *passive_nox_rate.dat*) and *nox_model > nox_thermal > nox_radical_model*, which specifies whether the equilibrium assumption is made for the O/OH radical model. When [*emissions.in > nox_model > nox_thermal > rate_flag = 0*](#), CONVERGE applies the default rate constants described in [Heywood \(1988\)](#). These values are generally recommended for ICE cases. For gas turbine cases, we recommend the rate constants described in [Hanson and Salimian \(1984\)](#), which are available in CONVERGE Studio during case setup ([*emissions.in > nox_model > nox_thermal > rate_flag = 1*](#)).

By default, a factor of 1.533 (the ratio of molecular weights of NO₂ to NO) is used to convert the predicted NO to NOx. You must verify that this factor reflects the experimental measurements for the conversion of NO to NO₂ (by mass). If measurements report only NO or if both NO and NO₂ were measured, then set [*emissions.in > nox_model > nox_scaling_factor*](#) to the appropriate value.

Prompt NOx Model

[Fenimore \(1971\)](#) identifies a rapid transient formation of NO (so-called "prompt NO") in fuel-rich, low-temperature conditions. This mechanism is most useful for gas turbine, surface combustion, and staged combustion applications. The prompt NO mechanism is complex and CONVERGE employs a simplified model to ensure computational feasibility.

To solve the reaction rates, CONVERGE uses the [De Soete \(1975\)](#) global kinetic parameter, which is given as

$$\frac{d[NO]}{dt} = R_{NO_x} - R_{N_2}, \quad (17.28)$$

where R_{NO_x} is the overall prompt NOx formation rate and R_{N_2} is the overall molecular nitrogen formation rate.

When prompt NOx forms in a fuel-rich environment where the concentration of O is high, the N radical primarily forms NOx instead of nitrogen. Thus, the rate of prompt NOx formation is approximately equal to the rate of overall prompt NOx formation (shown in Equation 17.29). This scenario occurs during initial stages of the flame.

$$\frac{d[NO]}{dt} = k_{pr} [O_2]^r [N_2] [fuel\ species] \exp\left(\frac{-E_a}{RT}\right) \quad (17.29)$$

where [NO], for example, denotes the species concentration of NO, a is defined below in Equation 17.33, E_a is the activation energy, R is the ideal gas constant, and T is temperature. By applying a correction factor to the De Soete model, the performance of the model given by Equation 17.29 improves for fuels with higher hydrocarbon content and in fuel-rich conditions. This factor accounts for the effect of fuel type and air-to-fuel ratio. Applying the correction factor f yields

Chapter 17: Emissions Modeling

NOx Modeling

Prompt NOx Model

$$\frac{d[NO]}{dt} = f k'_{pr} [O_2]^a [N_2] [\text{fuel species}] \exp\left(\frac{-E'_a}{RT}\right). \quad (17.30)$$

The correction factor f is

$$f = 4.75 + 0.0819n - 23.2\phi + 32\phi^2 - 12.2\phi^3, \quad (17.31)$$

where ϕ is the equivalence ratio ([emissions.in > nox_model > nox_prompt > equiv_ratio](#)) and n is the number of carbon atoms per molecule for hydrocarbon fuel (calculated from the fuel listed for [combust.in > fuel_name](#)). Note that the correction factor f is valid for equivalence ratios between 0.6 and 1.6 and f is a curve fit to experimental data and thus valid for aliphatic alkane hydrocarbon fuels ([Dupont et al., 1993](#)). Values for k'_{pr} and E'_a are hard-coded in CONVERGE based on values from [Dupont et al., 1993](#). Currently, the prompt NOx model supports only single-component fuels. If you use this model in a simulation that contains a multi-component fuel or in a simulation that contains multiple fuels, CONVERGE will calculate n only based on the [combust.in > fuel_name](#) (the single-component fuel) and the results will be incorrect.

After applying the correction factor to Equation 17.29, the source term due to the prompt NOx formulation is

$$S_{prompt,NO} = MW_{NO} \frac{d[NO]}{dt}. \quad (17.32)$$

[De Soete \(1975\)](#) asserts that the oxygen reaction order depends uniquely on the oxygen mole fraction (X_{O_2}) in each cell. Equation 17.33 describes the reaction order.

$$a = \begin{cases} 1.0 & X_{O_2} \leq 4.1e-3 \\ -3.95 - 0.9 \ln X_{O_2} & 4.1e-3 \leq X_{O_2} \leq 1.11e-2 \\ -0.35 - 0.1 \ln X_{O_2} & 1.11e-2 \leq X_{O_2} < 0.03 \\ 0 & X_{O_2} \geq 0.03 \end{cases} \quad (17.33)$$

Prompt NOx Model Setup

To activate the prompt NOx model, set [emissions.in > nox_model > nox_prompt > active = 1](#) and include the [passive](#) NOx in [species.in](#). You must ensure that you include both O and OH reactions in the [reaction mechanism file](#). You can also set the the global equivalence ratio used for the flame during prompt NOx calculations via [emissions.in > nox_model > nox_prompt > equiv_ratio](#).

Detailed Chemistry NOx

As an alternative to the reduced-order NOx formation models described above, CONVERGE can directly calculate NOx production via the [SAGE detailed chemical kinetics solver](#). If SAGE is activated (*i.e.*, if `sage_model > active = 1` in `combust.in`), you can specify reactions in the [mechanism data file](#) and the [thermodynamic data file](#) to model NO or both NO and NO₂. It is important to carefully compare the results from the detailed chemistry solver to measured values.

Additionally, if your simulation includes [adaptive zoning](#) (*i.e.*, if `adaptive_zone_model > active = 1` in `combust.in`), you can set `adaptive_zone_model > nox_flag = 1` in `combust.in` to improve the prediction of NOx emissions while simultaneously reducing the accuracy of the prediction of other species.

17.2 Soot Modeling

The soot models outlined in the sections below describe the complex soot formation and oxidation process using several global steps, including soot inception, surface growth, coagulation, and oxidation.

- Soot inception is the formation of the smallest solid soot particles from the gas-phase hydrocarbon molecules (*e.g.*, the PAH [polycyclic aromatic hydrocarbons] species). Inception serves as the link between gas-phase chemistry and soot particle dynamics and can be described by the collision of two PAH molecules.
- Soot surface growth is necessary to accomplish two-way coupling with the gas phase. Soot mass growth and heterogeneous reactions on surfaces as well as loss of soot particles due to reactions with gas phase species must be included in a detailed presentation of soot formation.
- Soot coagulation is a physical process of collisions between small soot particles leading to the formation of larger soot particles.
- Soot condensation indicates the gas phase species (such as PAHs) coagulate together and form large soot particles.

The following figure shows a summary of the models used in CONVERGE to describe the various steps of soot formation. The reaction rates of these global steps are determined by either empirical expression (you can adjust some parameters to match the experimental results) or by implementing simplified physical models.

Chapter 17: Emissions Modeling

Soot Modeling

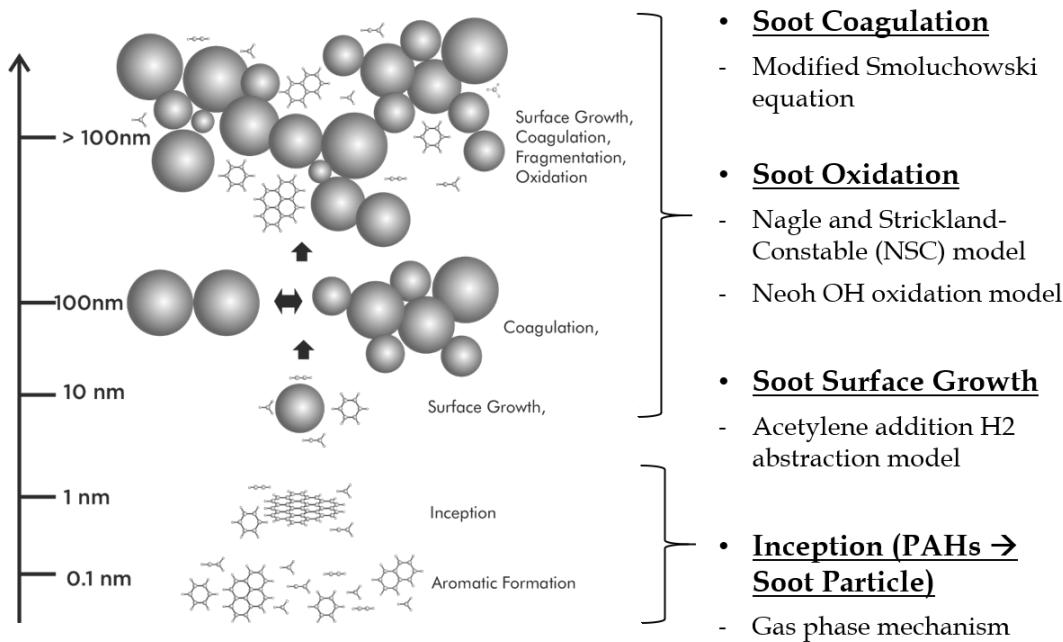


Figure 17.1: A descriptive overview of soot formation, with the various models used to simulate soot formation in CONVERGE.

The soot models implemented in CONVERGE typically are solved with detailed PAH chemistry.

CONVERGE offers three categories of soot models. In order of increasing complexity, these models are categorized as empirical, phenomenological, and detailed. Phenomenological approaches model inception, surface growth, coagulation and oxidation steps. Detailed soot models are more computationally expensive, but they perform better over a wider range of operating conditions. Detailed models are a better choice when more comprehensive soot formation analysis is needed. They describe the stages of soot formation directly. The following table lists the soot models available in CONVERGE.

Table 17.1: CONVERGE soot models.

Category	Model name
Empirical	Hiroyasu-NSC
Phenomenological	Gokul Dalian Waseda
Detailed	Particulate Mimic (PM) Particulate Size Mimic (PSM)

Empirical Soot Model: Hiroyasu-NSC

CONVERGE offers one empirical soot model: the Hiroyasu model, which is coupled with the [Nagle and Strickland-Constable model \(1962\)](#) to simulate soot oxidation.

The production of soot mass M_s (g) within a computational cell can be determined from a single-step competition between the soot mass formation rate \dot{M}_{sf} (g / s) and the soot mass oxidation rate \dot{M}_{so} according to [Hiroyasu and Kadota \(1976\)](#):

$$\frac{dM_s}{dt} = \dot{M}_{sf} - \dot{M}_{so}, \quad (17.34)$$

where the formation rate is given by

$$\dot{M}_{sf} = SF M_{form} \quad (17.35)$$

and

$$SF = A_{sf} P^{0.5} \exp\left(\frac{-E_{sf}}{RT}\right). \quad (17.36)$$

In Equations 17.35 and 17.36, M_{form} is the mass of the soot formation species in *grams*, P is the cell pressure in *bar*, R is the ideal gas constant in *cal/mol-K*, T is the cell temperature in *K*, E_{sf} is the activation energy in *cal/mol*, and A_{sf} is the Arrhenius pre-exponential factor with units of $s^{-1}\text{-bar}^{-0.5}$.

There are two options for the soot formation species. The first option treats all hydrocarbon species as soot formation species, and is available for all combustion models. To use all HC species for soot formation, set [*emissions.in* > *hiroy_soot_model* > *hiroy_form_flag* = 0](#). The second option (*hiroy_form_flag* = 1) uses C2H2 as the soot formation species, and requires that a detailed [reaction mechanism](#) is available. This option is compatible with the [SAGE detailed chemical kinetics solver](#) and the [RIF combustion model](#), and with the [G-Equation combustion model](#) as long as at least one of [*combust.in* > *g_eqn_model* > *burned_region*](#), [*g_eqn_model* > *on_flame*](#), or [*g_eqn_model* > *unburned_region* = SAGE](#).

To model soot oxidation, CONVERGE uses the [Nagle and Strickland-Constable model \(1962\)](#) (NSC). This model considers carbon oxidation by two mechanisms. The rates of these mechanisms depend on surface chemistry involving more reactive A sites and less reactive B sites. In this model, the total soot oxidation reaction rate R_{total} is given by

Chapter 17: Emissions Modeling

Soot Modeling Empirical Soot Model: Hiroyasu-NSC

$$R_{total} = \left(\frac{k_A P_{O_2}}{1 + k_Z P_{O_2}} \right) X + k_B P_{O_2} (1 - X) \quad (\text{mol/cm}^2\text{-s}), \quad (17.37)$$

where X is the proportion of A sites given by

$$X = \frac{P_{O_2}}{P_{O_2} + (k_T/k_B)}. \quad (17.38)$$

In the above equations, P_{O_2} is the oxygen partial pressure in *atmospheres* and the k values are rate constants for carbon given by

$$\begin{aligned} k_A &= 20 \exp(-30,000 / RT) \quad (\text{mol/cm}^2\text{-s-atm}) \\ k_B &= 4.46 \times 10^{-3} \exp(-15,200 / RT) \quad (\text{mol/cm}^2\text{-s-atm}) \\ k_T &= 1.51 \times 10^5 \exp(-97,000 / RT) \quad (\text{mol/cm}^2\text{-s}) \\ k_Z &= 21.3 \exp(4,100 / RT) \quad (\text{atm}^{-1}) \end{aligned} \quad (17.39)$$

where R is the ideal gas constant and T is the temperature. If it is assumed that the soot particles are spherical and uniform in size, the surface area S (cm^2) for oxidation can be written as

$$S = N_{p,soot} \pi D_s^2 = \frac{6M_s}{\rho_s D_s}, \quad (17.40)$$

where $N_{p,soot}$ is the total number of soot particles, D_s is the nominal soot particle diameter in cm , M_s is the total soot particle mass (*grams*), and ρ_s is the soot density in g/cm^3 . The oxidation rate \dot{M}_{so} is thus given by

$$\dot{M}_{so} = S R_{total} M W_c = \frac{6M_s}{\rho_s D_s} R_{total} M W_c \quad (\text{g/s}), \quad (17.41)$$

where $M W_c$ (g/mol) is the molecular weight of carbon. If a scaling factor A_{so} is included, the following expression for the soot oxidation rate is obtained:

$$\dot{M}_{so} = A_{so} \frac{6M_s}{\rho_s D_s} R_{total} M W_c. \quad (17.42)$$

Chapter 17: Emissions Modeling

Soot Modeling Empirical Soot Model: Hiroyasu-NSC

Equation 17.42 can be rewritten more concisely as

$$\dot{M}_{so} = SO \dot{M}_s, \quad (17.43)$$

with

$$SO = A_{so} \frac{6}{\rho_s D_s} R_{total} MW_c. \quad (17.44)$$

Using Equations 17.35 and 17.43, Equation 17.34 can be rewritten as a first-order linear equation given by

$$\frac{d\rho_s}{dt} + SO \rho_s = SF \rho_{form}, \quad (17.45)$$

which has been recast in terms of species densities, where ρ_{form} is the density of the soot formation species. Equation 17.45 can be integrated analytically with the method of constant coefficients. The result is the following expression for the soot species density ρ_s^{n+1} at the end of a computational time-step dt :

$$\rho_s^{n+1} = \frac{SF \rho_{form}}{SO} + C \exp(-SO dt). \quad (17.46)$$

The constant of integration C can be determined because the soot species density is equal to ρ_s^n at the start of a computational time-step. Substituting the result for C yields

$$\rho_s^{n+1} = \frac{SF \rho_{form}}{SO} + \left(\rho_s^n - \frac{SF \rho_{form}}{SO} \right) \exp(-SO dt), \quad (17.47)$$

which also can be written in terms of the difference in soot density at the start and end of the computational time-step:

$$\rho_s^{n+1} - \rho_s^n = \left(\frac{SF \rho_{form}}{SO} - \rho_s^n \right) \cdot [1 - \exp(-SO dt)]. \quad (17.48)$$

Equation 17.48 is used in CONVERGE to update the soot density in each computational cell for each time-step.

Chapter 17: Emissions Modeling

Soot Modeling Empirical Soot Model: Hiroyasu-NSC

Hiroyasu-NSC Model Setup

The Hiroyasu-NSC soot model parameters are located in the [*emissions.in*](#) > *hiroy_soot_model* settings block. The Hiroyasu-NSC predicted soot mass is written to the [*emissions.out*](#) file.

You can use the Hiroyasu-NSC model and other soot models simultaneously.

Phenomenological Soot Models

You can tune many of the parameters of the phenomenological soot models to adjust for varying operating conditions. CONVERGE solves these phenomenological models using the [SAGE detailed chemical kinetics solver](#) and requires a detailed mechanism to calculate averaged soot mass and number density. Phenomenological soot models are also supported when running a simulation with [G-Equation](#), as long as at least one of [*combust.in*](#) > *g_eqn_model* > *burned_region*, *g_eqn_model* > *on_flame*, or *g_eqn_model* > *unburned_region* = *SAGE*. The soot formation is assumed to be one-way coupled with the gas phase chemistry; that is, soot formation will not affect gas phase chemistry and system parameters such as temperature and pressure. These models generally result in a less than 10% increase in computational time compared to an engine simulation with detailed chemistry only.

In engine simulations, the soot number density and soot mass density are solved as global transport [passives](#) according to Equation 17.49 below:

$$\frac{DM}{Dt} = \nabla \left(\frac{\mu}{Sc} \nabla \left(\frac{M}{\rho} \right) \right) + \dot{S}_M, \quad (17.49)$$

where M represents either the soot species density, $\rho Y_{C(s)}$, or the soot number density, N , in SI units. μ is the cell viscosity, Sc is the Schmidt number, and \dot{S}_M represents the source term one of the models.

Table 17.2 below summarizes the model parameters employed by each of the three phenomenological soot models. A more detailed formulation of the source terms for each phenomenological model is included in the subsections that follow.

Chapter 17: Emissions Modeling

Soot Modeling

Phenomenological Soot Models

Table 17.2: Model parameters and typical application of each phenomenological soot model.

	Gokul model	Dalian model	Waseda model
Soot precursor	Pyrene (A4)	C50	Acenaphthylene radical (A2R5)
Soot inception (SI)	Arrhenius	Arrhenius	Arrhenius
Soot surface growth (SSG)	Simplified HACA (Hydrogen-Abstraction/Carbon-Addition)	Arrhenius	Simplified HACA
Soot oxidation (SO)	Revised NSC (Nagle and Strickland-Constable) model/OH radical	NSC model/OH radical	NSC/OH/NO
Soot coagulation (SC)	Revised Smoluchowski model	Kazakov-Foster model	Revised Smoluchowski model
Typical application	Low temperature diesel	Diesel PCCI (Premixed Charge Compression Ignition)	Medium-duty diesel

The [Gokul](#), [Dalian](#), and [Waseda](#) phenomenological soot models can be used with the dense or preconditioned solver options ([combust.in](#) > `sage_model > ode_solver > type = DENSE, ITERATIVE, ITERATIVE_SUPERLU, or ITERATIVE_CONVERGE`). However, you can set SAGE with [adaptive zoning](#) for these soot models. Soot-related output—averaged soot number density, soot mass, and the mass for different soot sub-processes (nucleation, surface growth, oxidation, and coagulation)—will be written to [phenom_soot_model.out](#).

Gokul Model

The Gokul phenomenological soot model ([Vishwanathan and Reitz, 2010](#)) is best suited for low temperature diesel engine simulations. Set [emissions.in](#) > `phenom_soot_model > active = GOKUL` to activate this model.

You must include the soot precursor for this model, pyrene (A4), in the [reaction mechanism](#). A brief description of each sub-process is given below.

Soot Inception



where R_{SI} is the rate of the soot inception reaction and k_{SI} is the rate constant for the soot inception reaction.

Chapter 17: Emissions Modeling

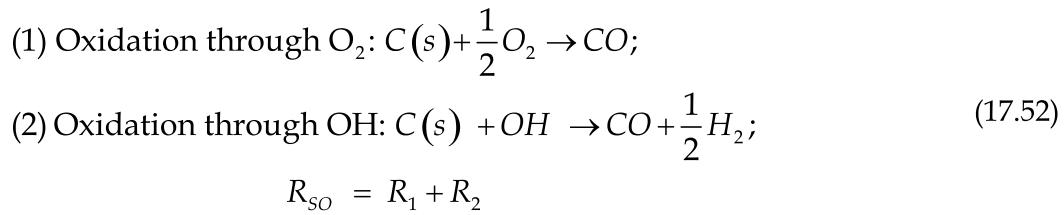
Soot Modeling Phenomenological Soot Models

Soot Surface Growth

$$\begin{aligned} C(s) + C_2H_2 &\rightarrow 3C(s) + H_2 \\ R_{SSG} &= k_{SSG} [C_2H_2] \\ k_{SSG} &= 9.0 \cdot 10^4 \exp\left(-\frac{12100}{T}\right) \sqrt{S} \\ S &= \pi d_p^2 N \\ d_p &= \left(\frac{6Y_{C(s)}\rho}{\pi\rho_{C(s)}N} \right)^{1/3}, \end{aligned} \quad (17.51)$$

where R_{SSG} is the rate of the soot surface growth reaction, k_{SSG} is the rate constant for the soot surface growth reaction, T is temperature, d_p is the particle size, $Y_{C(s)}$ is the soot mass fraction, N is the soot number density, ρ is the ambient density, and $\rho_{C(s)}$ is the density of graphite (2.0 g/cm³).

Soot Oxidation



where R_{SO} is the rate of the soot oxidation reaction, R_1 is the rate of reaction (1), and R_2 is the rate of reaction (2). Oxidation through O₂ uses the NSC model (consistent with the Hiroyasu empirical soot model), while oxidation through OH uses the Neoh model ([Neoh et al., 1984](#)).

Soot Coagulation

$$nC(s) \rightarrow C_n(s)$$
$$R_{SC} = 2Ca \left(\frac{6MW_{C(s)}}{\pi\rho_{C(s)}} \right)^{1/6} \left(\frac{6k_B T}{\rho_{C(s)}} \right)^{1/2} \left(\frac{\rho Y_{C(s)}}{MW_{C(s)}} \right)^{1/6} N^{11/6} \quad (17.53)$$

where R_{SC} is the rate of the soot coagulation reaction, Ca is the coagulation coefficient, $MW_{C(s)}$ is the carbon molecular weight (12 g/mol), and k_B is the Boltzmann constant.

The source terms for the soot species density and number density are calculated as follows:

Chapter 17: Emissions Modeling

Soot Modeling

Phenomenological Soot Models

$$\begin{aligned} \text{For species density: } & \dot{S}_M = (16R_{SI} + 2R_{SSG} - R_{SO})M_{C(s)} \\ \text{For number density: } & \dot{S}_M = 16R_{SI} \frac{M_{C(s)}}{M_{nuci}} - R_{SC} \\ & M_{nuci} = \frac{\pi}{6} d_{nuci}^3 \rho_{C(s)}, \end{aligned} \quad (17.54)$$

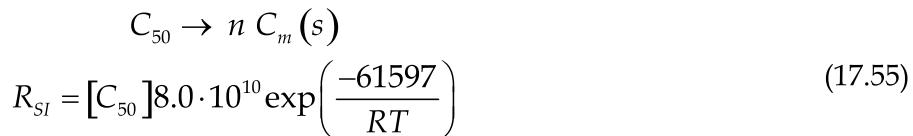
where M_{nuci} is the mass of incipient soot, assumed to have a diameter, $d_{nuci} = 1.28 \text{ nm}$ (approximately equivalent to 100 carbon atoms).

Dalian Model

The Dalian phenomenological soot model ([Jia et al., 2009](#)) is best suited for diesel premixed charge compression ignition (PCCI) engine simulations. Set `emissions.in > phenom_soot_model > active = DALIAN` to activate this model.

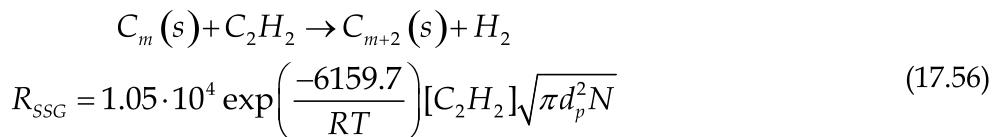
A brief description of each sub-process is given below.

Soot Inception



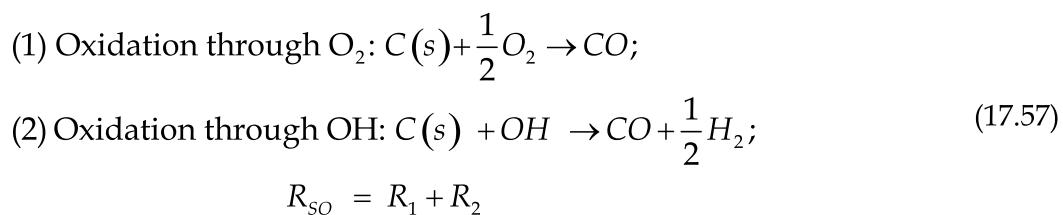
where R_{SI} is the rate of the soot inception reaction, $[C_{50}]$ is the concentration of C50, R is the ideal gas constant, and T is the temperature.

Soot Surface Growth



where R_{SSG} is the rate of the soot surface growth reaction, $[C_2H_2]$ is the concentration of C2H2, d_p is the particle size, and N is the soot number density.

Soot Oxidation



Chapter 17: Emissions Modeling

Soot Modeling Phenomenological Soot Models

where R_{SO} is the rate of the soot oxidation reaction, R_1 is the rate of reaction (1), and R_2 is the rate of reaction (2). Oxidation through O₂ uses the NSC model (consistent with the Hiroyasu empirical soot model), while oxidation through OH uses the Neoh model ([Neoh et al., 1984](#)).

Soot Coagulation

$$nC_m(s) \rightarrow C_{n^*m}(s)$$
$$R_{SC} = \frac{1}{2}\beta N^2, \quad (17.58)$$

where R_{SC} is the rate of the soot coagulation reaction, and β is the collision coefficient, which is estimated by the Kazakov-Foster model ([Kazakov and Foster, 1998](#)) by considering the particle collision in both the free-molecular and near-continuum regime.

The source terms for the soot species density and number density are calculated as follows:

$$\text{For species density: } \dot{S}_M = (50R_{SI} + R_{SSG} - R_{SO})MW_{C(s)} \quad (17.59)$$
$$\text{For number density: } \dot{S}_M = R_{SI} \frac{N_A}{2} - R_{SC},$$

where N_A is Avogadro's number and $MW_{C(s)}$ is the molecular weight of a carbon atom.

Waseda Model

The Waseda phenomenological soot model ([Kaminaga et al., 2008](#)) is best suited for medium duty diesel engine simulations. Set `emissions.in > phenom_soot_model > active = WASEDA` to activate this model.

You must include the soot precursor for this model, acenaphthylene radical (A2R5), in the [reaction mechanism](#). A brief description of each sub-process is given below.

Soot Inception

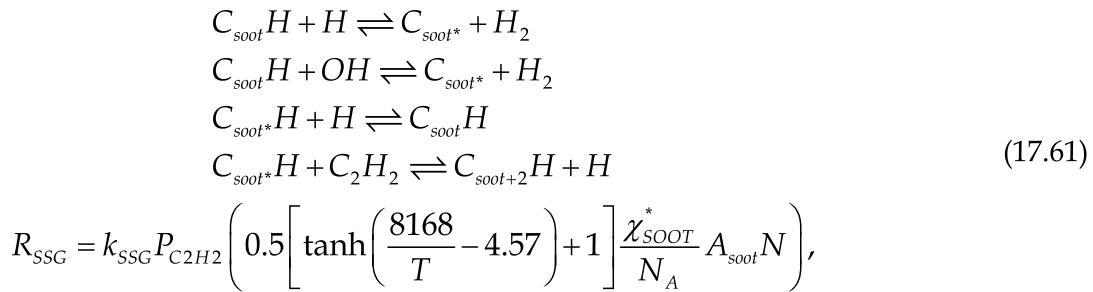
$$A2R5 \rightarrow 12C(s) + 4H_2$$
$$R_{SI} = [A2R5]1000 \exp\left(\frac{-5000}{RT}\right) \quad (17.60)$$

where R_{SI} is the rate of the soot inception reaction, $[A2R5]$ is the concentration of A2R5, R is the ideal gas constant, and T is the temperature.

Soot Surface Growth

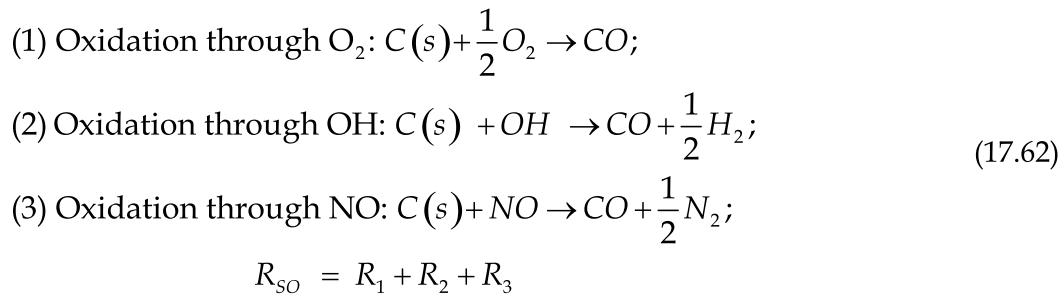
Chapter 17: Emissions Modeling

Soot Modeling Phenomenological Soot Models



where R_{SSG} is the rate of the soot surface growth reaction, k_{SSG} is the rate constant for the soot surface growth reaction, P_{C2H2} is the partial pressure of C_2H_2 , χ_{SOOT}^* is the active surface site density of soot particles, N_A is Avogadro's number, A_{soot} is the surface area fraction of soot particles per unit volume, and N is the soot number density. Refer to [Kaminaga et al. \(2008\)](#) for more details of these parameters.

Soot Oxidation



where R_{SO} is the rate of the soot oxidation reaction, R_1 is the rate of reaction (1), R_2 is the rate of reaction (2), and R_3 is the rate of reaction (3). Oxidation through O_2 uses the NSC model (consistent with the Hiroyasu empirical soot model). Oxidation through OH uses the Neoh model ([Neoh et al., 1984](#)). Oxidation through NO uses the reaction probability concept proposed by [Gersum and Roth \(1992\)](#).

Soot Coagulation



where R_{SC} is the rate of the soot coagulation reaction, k_{SC} is the rate constant for soot coagulation, f_v is the soot volume fraction, and N is the soot number density.

The source terms for the soot species density and number density are calculated as follows:

$$\begin{aligned} \text{For species density: } & \dot{S}_M = (R_{SI} + R_{SSG} - R_{SO}) MW_{C(s)} \\ \text{For number density: } & \dot{S}_M = R_{SI} 0.12 N_A - R_{SC}, \end{aligned} \tag{17.64}$$

where $MW_{C(s)}$ is the molecular weight of a carbon atom.

Detailed Soot Models

There are three detailed soot models available in CONVERGE: the Particulate Mimic soot model, the Particulate Size Mimic soot model, and the Sectional Soot model. The following sub-sections describe these models in more detail.

Particulate Mimic Soot Model

The detailed soot models available in CONVERGE solve the complex soot formation and oxidation with detailed chemistry. The detailed soot models feature good capability over wide ranges of operating conditions. The properties of a soot ensemble can be described by the particle size distribution function (PSDF). The PSDF of soot can be obtained by solving equations for the number density of all size classes if the physical and chemical processes changing the PSDF are known. In principle, this produces an infinitely large set of partial differential equations.

In practice, the integral features of the PSDF are of primary engineering interest. These provide the most important information about the soot particle ensemble, such as mean particle number density, mean mass or volume, and mean diameter of the particles.

The method of moments ([Frenklach and Wang, 1991](#); [Kazakov et al., 1995](#); and [Kazakov and Frenklach, 1998](#)) is based on the fact that solving an infinite set of equations for the statistical moments of the PSDF is equivalent to the direct simulations of the PSDF. This method can be shown to have sufficient accuracy using only a few moments for global observables, such as mean number density and soot mass. Usually a set of equations for the first two to six moments is applied. The accuracy of the approach increases with the number of moments used.

The main advantages of the method of moments is its computational efficiency and that the major features of the PSDF, such as mean number density and soot volume fraction, can be extracted from the moments. The Particulate Mimic (PM) model in CONVERGE is based on the method of moments introduced above. The different processes leading to the formation and oxidation of soot particles included in the model are briefly described below. Detailed discussion of the PM model can be found in [Mauss \(1998\)](#).

The basic physical and chemical processes assumed to be important for the formation of soot are: particle [inception](#), [coagulation](#), [condensation](#), and heterogeneous surface reactions (where the heterogeneous surface reactions are [surface growth](#) and oxidation by OH and O₂). The dynamics of the soot particle characteristics can be described by a set of equations for the moments of the soot particle size distribution function:

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

$$\frac{dM_z}{dt} = \dot{M}_{z,pi} + \dot{M}_{z,con} + \dot{M}_{z,coag} + \dot{M}_{z,sr}, \quad (17.65)$$

where $\dot{M}_{z,pi}$, $\dot{M}_{z,con}$, $\dot{M}_{z,coag}$, and $\dot{M}_{z,sr}$ are the rates of particle inception, condensation, coagulation, and surface reactions for the z -th moment of the PSDF, respectively. The moments are defined as

$$M_z = \sum_{i=1}^{\infty} i^z N_i, \quad (17.66)$$

where N_i is the number density of soot particles of size class i . It can be seen from the definition that the zeroth moment is related to the mean number density, whereas the first moment is related to the mean mass or mean volume of the soot particles.

Each moment in the particulate mimic model is solved as global transport [passive](#):

$$\begin{aligned} \frac{D(\dot{M}_z / \rho)}{Dt} &= \nabla \left(\frac{\mu}{Sc} \nabla \left(\frac{\dot{M}_z}{\rho} \right) \right) + \dot{S}_{Mr} \\ \dot{S}_{Mr} &= \dot{M}_{z,pi} + \dot{M}_{z,con} + \dot{M}_{z,sg} + \dot{M}_{z,ox} + \dot{M}_{z,coag}, \end{aligned} \quad (17.67)$$

where Sc is the Schmidt number, \dot{S}_{Mr} represents the source term, and $\dot{M}_{z,sg}$ and $\dot{M}_{z,ox}$ are the rates of particle surface growth and oxidation for the z -th moment of the PSDF, respectively. The moment source term is coupled with species source term and solved using the [SAGE detailed chemical kinetics solver](#). It is a two-way coupling, which means that the soot formation will affect the gas phase and system heat release.

Soot-related output for a PM simulation is written to [*soot_pm_model.out*](#). To account for the mass transfer between gas and soot in the detailed soot models, and therefore strictly conserve mass, add *SOOT* as a species in [*species.in*](#) > *gas* and [*therm.dat*](#). To add the *SOOT* species to [*therm.dat*](#), copy the *therm.dat* entry for either A4R5, A4 or A3R5m and rename the species *SOOT*.

Stages of the PM Model

Soot Inception

In the PM model, the rate of formation of particle of size i is thus given by

$$\dot{N}_{i,pi} = \frac{1}{2} \sum_{j=1}^{i-1} \beta_{j,i-j} {}^p N_j {}^p N_{i-j}, \quad (17.68)$$

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

where ${}^p N_j$ denotes the number density of a PAH molecule of size j and $\beta_{i,j}$ the frequency of free-molecular coalescence given by

$$\beta_{i,j} = \varepsilon \sqrt{\frac{8\pi k_B T}{\mu_{i,j}}} (r_i + r_j)^2, \quad (17.69)$$

where k_B is the Boltzmann constant, $\mu_{i,j}$ is the reduced mass and r_i and r_j are the radius of particles i and j , respectively, and ε is the collision enhancement due to inter-particle forces ([Balthasar et al., 2002](#)).

The rate of the moments of the soot size distribution can be obtained by multiplying Equation 17.68 by i^z and summing over all the size classes, leading to

$$\dot{M}_{z,pi} = \frac{1}{2} \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} (i+j)^z \beta_{i,j} {}^p N_i {}^p N_j. \quad (17.70)$$

By assuming constant density of soot particles and the size of two collided PAH molecules are of similar order, the formulation of soot inception can be simplified. Furthermore, the sizes i and j are approximated by the mean size $\langle i \rangle$, in terms of moments ${}^p M_1 / {}^p M_0$. It can be shown that the mean size of PAHs does not vary significantly in test simulations. It was found that a mean size of 24($M_1/M_0 = 12$) carbon atoms is a good approximation ([Mauss, 1998](#)). By setting the mean size of PAHs to a constant value, the rate of particle inception is in the subroutine calculated as

$$\dot{M}_{z,pi} = C_{pi} (T, {}^p M_0) (2\langle i \rangle)^z, \quad (17.71)$$

where C_{pi} is a pre-factor depending on the temperature and PAH formation rate.

Soot Surface Reactions

For both the PM and the PSM models, soot mass growth and loss of soot particles due to reactions with gas phase species on their surface is described by a detailed growth and oxidation mechanism, the Hydrogen Abstraction Acetylene Addition Ring Closure (HACARC) mechanism ([Mauss, 1998](#)):

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

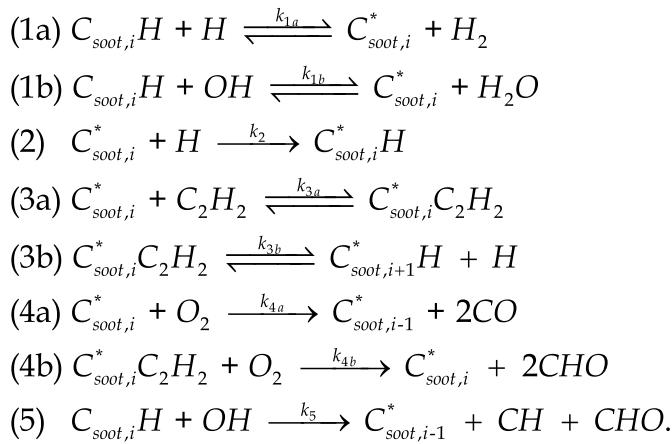


Figure 17.2: The reaction pathways of the HACARC mechanism.

$C_{soot,i}H$ is an active site on the surface of a soot particle with terminal H-C bound and $C_{soot,i}^*$ is an active radical site on the surface with i and $i+1$ denoting the size classes. Reactions (1) through (3) in the HACARC mechanism describe the surface growth mechanism while reactions (4) through (5) are for the oxidation. The detailed discussion on HACARC mechanism can be found in [Mauss \(1998\)](#). The rates of surface reactions can be formulated as follows:

$$\begin{aligned}
 \dot{M}_{z,sg} &= \alpha k_{3a,f} [C_2H_2] f_{3a} A \sum_{d=0}^{z-1} \binom{z}{d}^s M_{d+2/3} \Delta m^{z-d}, \quad z=1,2,\dots \\
 \dot{M}_{z,ox} &= \alpha (k_{4a}[O_2] + k_5[OH]) A \sum_{d=0}^{z-1} \binom{z}{d}^s M_{d+2/3} \Delta m^{z-d}, \quad z=1,2,\dots
 \end{aligned} \tag{17.72}$$

where α is the fraction of surface site ranging from 0 to 1 and Δm is the change of mass in one reaction step (*i.e.*, $\Delta m=1$ for surface growth and -1 for oxidation). A is the rate coefficient factor given by

$$\begin{aligned}
 A &= \frac{k_{1a,f}[H] + k_{1b,f}[H] + k_5[OH]}{k_{1a,r}[H_2] + k_{1b,r}[H_2O] + k_2[H] + k_{3a,f}[C_2H_2]f_{3a} + k_{4a}[O_2]} \\
 f_{3a} &= \frac{k_{3b,f}}{k_{3b,f} + k_{3a,r} + k_{4,r}[O_2]}
 \end{aligned} \tag{17.73}$$

where k represents the rate constant for the reaction number in Figure 17.2 and the subscript f and r differentiate between the forward and reverse reaction, respectively.

Soot Coagulation

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

In general (and also in the PM model), coagulation is described by the [Smoluchowski \(1917\)](#) equation as

$$\dot{N}_{i,coag} = \frac{1}{2} \sum_{j=1}^{i-1} \beta_{j,i-j} N_j N_{i-j} - \sum_{j=1}^{\infty} \beta_{i,j} N_i N_j. \quad (17.74)$$

The frequency factor β depends on the Knudsen number regime. For large Knudsen numbers, the coagulation rate of moments takes the following form:

$$\begin{aligned} \dot{M}_{0,coag}^{fm} &= -0.5 \langle \phi_{0.0} \rangle M_0^2 \\ \dot{M}_{0,coag}^{fm} &= 0 \\ \dot{M}_{z,coag}^{fm} &= \sum_{d=1}^{z-1} \binom{z}{d} \langle \phi_{d,z-d} \rangle M_0^2. \end{aligned} \quad (17.75)$$

The function $\langle \phi_{d,z-d} \rangle$ is evaluated based on a Logarithmic Lagrange interpolation for the reduced moment, M_z/M_0 .

The rate of coagulation in the continuum regime at small Knudsen numbers takes the following form:

$$\begin{aligned} \dot{M}_{0,coag}^c &= K \left[M_0^2 + M_{1/3} M_{-1/3} + 2.154 \lambda C_s (M_{-1/3} M_0 + M_{1/3} M_{-2/3}) \right] \\ \dot{M}_{z,coag}^c &= K \left[2 M_d M_{z-d} + M_{d+1/3} M_{z-d-1/3} + \right. \\ &\quad \left. + 2.154 \lambda C_s (M_{d-1/3} M_{z-d} + M_d M_{z-d-1/3} + M_{d+1/3} M_{z-d-2/3} + M_{d-2/3} M_{z-d+1/3}) \right], \end{aligned} \quad (17.76)$$

with $C_s = (\pi \rho_s / 6 m_1)^{1/3}$ and $K = 2k_B T / 3\eta$, where k_B is the Boltzmann constant, η is the gas viscosity, and λ is the mean free path.

The rate of coagulation valid for the entire regime of Knudsen numbers is then obtained by forming the harmonic mean of the continuum and free molecular rate as

$$\dot{M}_{z,coag} = \frac{\dot{M}_{z,coag}^{fm} \dot{M}_{z,coag}^c}{\dot{M}_{z,coag}^{fm} + \dot{M}_{z,coag}^c}, \quad z = 0, 2, 3, \dots \quad (17.77)$$

Soot Condensation

In the PM model, soot condensation is modeled as the coagulation of PAH molecules with soot particles described by the [Smoluchowski \(1917\)](#) equation in the following form:

$$\dot{M}_{z,con} = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} (i+j)^z \beta_{j,i-j} {}^p N_j {}^s N_i - \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} i^z \beta_{i,j} {}^p N_j {}^s N_i. \quad (17.78)$$

The first term describes the formation of particle of size i , while the second term indicates the loss of particles of size i due to coagulation with PAHs of all size classes.

By assuming the size of soot particles is much larger than that of PAHs, constant size of PAHs, the formulation of soot condensation can be simplified to

$$\dot{M}_{z,con} = C_{Lib,con} \sum_{d=0}^{z-1} j^{z-d-1/2} M_{d+2/3}, \quad (17.79)$$

where $C_{Lib,con}$ is a prefactor depending on the temperature and PAH formation rate and $M_{d+2/3}$ is the broken moment obtained by interpolation of all the moments ([Mauss, 1998](#)).

Particulate Size Mimic Soot Model

The PM model is efficient for obtaining detailed soot information such as cell averaged soot number density and mass, but it does not reveal the particle size distribution function (PSDF) for each cell. The Particulate Size Mimic model (PSM), based on the discrete sectional method ([Wen et al., 2005](#); [Kumar and Ramkrishna, 1996](#)), can obtain the PSDF information in addition to the detailed soot information that can also be obtained from the PM model.

The PSM model is based on the definition of sections containing the particles with the same volume. The boundary of each section is defined as:

$$\begin{aligned} v_{1,min} &= v_{MIN} \\ v_{i,min} &= v_{i-1,max}, \text{ for } i > 1 \\ v_{i,mean} &= \frac{v_{i,min} + v_{i,max}}{2}, \end{aligned} \quad (17.80)$$

where v_{MIN} is the volume of the smallest soot particle considered. In CONVERGE, v_{MIN} is defined as the volume of the soot precursor ($0.4e-27 m^3$ for pyrene), while the v_{MAX} is defined as the volume of the biggest soot particles (approximately $100 nm$ in diameter). Note that you can limit the size of the biggest soot particle by changing the value of [`emissions.in > detailed_soot_model > psm_biggestsoot_diameter`](#). We do not recommend setting a value above $100 nm$. $v_{i,min}$ and $v_{i,max}$ are the boundary soot volume for each section.

$q_i(v)$ is the distribution function of v for each section. The total volume fraction Q_i for each section will be

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

$$Q_i = \int_{v_{i,\min}}^{v_{i,\max}} q_i(v) dv. \quad (17.81)$$

In CONVERGE the distribution function $q_i(v)$ for v is presented in the following first-order polynomial:

$$\begin{aligned} q_i(v) &= q'_i v + q''_i \\ q'_i &= \frac{k_i^+ + k_i^-}{2}, \end{aligned} \quad (17.82)$$

where k_i^+ and k_i^- are the slopes of the left and right boundaries of $q_i(v)$ respectively. These values are calculated so that the soot streams transported through different sections are conserved. q''_i is then calculated as

$$q''_i = \frac{Q_i}{v_{i,\min} - v_{i,\max}} - q'_i v_{i,mean}. \quad (17.83)$$

The maximum boundary is increased using the following nonlinear formulation to obtain high computation efficiency ([Netzell, 2007](#)):

$$\begin{aligned} v_{1,max} &= v_{MIN} + v_{C2} \\ v_{i,max} &= (v_{MIN} + v_{C2}) \left(\frac{v_{MAX}}{v_{MIN} + v_{C2}} \right)^{\frac{i-1}{i_{max}-1}}, \end{aligned} \quad (17.84)$$

where v_{C2} is the volume of two carbon atoms in soot. This is based on soot molecular weight and density and is calculated to be 7.176e4 nm³.

Each section in the particulate size mimic model is solved as a global transport [passive](#), as follows:

$$\begin{aligned} \frac{D(\dot{Q}_i / \rho)}{Dt} &= \nabla \left(\frac{\mu}{Sc} \nabla \left(\frac{\dot{Q}_i}{\rho} \right) \right) + \dot{S}_{Qi} \\ \dot{S}_{Qi} &= \Delta \dot{Q}_{i,pi} + \Delta \dot{Q}_{i,con} + \Delta \dot{Q}_{i,sg} + \Delta \dot{Q}_{i,ox} + \Delta \dot{Q}_{i,coag}, \end{aligned} \quad (17.85)$$

where Sc is the Schmidt number, and \dot{S}_{Qi} represents the source term for each section. Similar to the PM model, the section source term is coupled with the species source term and

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

solved using the [SAGE detailed chemical kinetics solver](#). It is a two-way coupling, which means that the soot formation will affect the gas phase and system heat release. As with the PM model, in the PSM model, soot inception, coagulation, condensation and surface reactions are taken into account.

Soot-related output for the PSM model is written to [soot_psm_model.out](#). To account for the mass transfer between gas and soot in the detailed soot models, and therefore strictly conserve mass, add SOOT as a species in [species.in](#) > *gas* and [therm.dat](#). To add the SOOT species to [therm.dat](#), copy the *therm.dat* entry for either A4R5, A4 or A3R5m and rename the species SOOT.

Stages of the PSM Model

Soot Inception

For PSM, the soot inception model is based on the [Smoluchowski \(1917\)](#) equation with PAH species. By applying the same assumptions as were applied in the PM model, a simplified formulation was obtained:

$$\dot{Q}_{i,pi} = 2v_{PAH}\beta_{fm,pi}(v_{PAH})N_{PAH}^2, \quad (17.86)$$

where v_{PAH} is the volume of the PAH species, $\beta_{fm,pi}$ is the collision coefficient for the PAH species, and N_{PAH} is the number density of the PAH species. Note that nucleation is regarded as the first section, while the source term for other sections is zero.

Soot Surface Reactions

CONVERGE applies the [HACARC surface reaction model](#), as described in the PM model section, in the PSM model. For each section the surface growth rate $\Delta Q_{i,sg}$ and oxidation rate $\Delta Q_{i,ox}$ are calculated as ([Marchal, 2008](#)):

$$\begin{aligned} \Delta Q_{i,sg} &= \alpha v_{c2}^{\frac{3-\theta}{3}} (k_d - k_{rev}) \left(\frac{3}{3+\theta} q_i \left(v_{i,max}^{\frac{3+\theta}{3}} - v_{i,min}^{\frac{3+\theta}{3}} \right) + \frac{3}{\theta} q_i^\mu \left(v_{i,max}^{\frac{\theta}{3}} - v_{i,min}^{\frac{\theta}{3}} \right) \right) \\ \Delta Q_{i,ox} &= \alpha v_{c2}^{\frac{3-\theta}{3}} (k_{O2} - k_{OH}) \left(\frac{3}{3+\theta} q_i \left(v_{i,max}^{\frac{3+\theta}{3}} - v_{i,min}^{\frac{3+\theta}{3}} \right) + \frac{3}{\theta} q_i^\mu \left(v_{i,max}^{\frac{\theta}{3}} - v_{i,min}^{\frac{\theta}{3}} \right) \right), \end{aligned} \quad (17.87)$$

where θ is the fractional dimension of the soot ([Marchal, 2008](#)) and k is the reaction rate coefficient specified in [Marchal, 2008](#).

Soot Coagulation

In the PSM model, soot coagulation forms in a manner similar to soot condensation. The formulations for coagulation for each section are given as follows ([Marchal, 2008](#)):

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

$$\Delta Q_{i,coag} = \sum_{v_{i,min} < (v_d + v_j) < v_{i,max}} (v_d + v_j) N_d N_j \beta_{coag}(v_d, v_j) + \sum_{(v_d + v_i) < v_{i,max}} v_d N_d N_i \beta_{coag}(v_d, v_i) - v_i N_i \sum_{(v_i + v_j) > v_{i,max}} N_i \beta_{coag}(v_i, v_j) - 2v_i N_i N_j \beta_{coag}(v_i, v_j). \quad (17.88)$$

Soot Condensation

In the PSM model, soot condensation will cause the transportation of particles from one section to the other. Formulations of conserving the concentration and soot mass are briefly explained as follows. The condensation source term is calculated as

$$\Delta Q_{i,cond} = v_{PAH} N_{PAH} \int_{v_{i,min}}^{v_{i,max}} \beta_{fm,cond}(v_{PAH}, v) n(v) dv. \quad (17.89)$$

The concentration of PAHs must be evaluated in order to calculate the above equation. The PAH volume fraction for the condensation and inception models is calculated as

$$R_{PAH} = 2\beta_{fm,pi}(v_{PAH}, v_{PAH}) N_{PAH}^2 + \sum_{i=1}^{i_{MAX}} N_{PAH} \int_{v_{i,min}}^{v_{i,max}} \beta_{fm,cond}(v_{PAH}, v) n(v) dv. \quad (17.90)$$

The soot evolution by condensation of the section i results from the difference between the exit particle before condensation of section i and the entry particle after condensation with PAH from section $i+1$:

$$\Delta Q_{i,cond} = \Delta q_{i,cond}^\rightarrow - \Delta q_{i,cond}^\uparrow, \quad (17.91)$$

where $\Delta q_{i,cond}^\uparrow$ is the flow of soot particles leaving section i and $\Delta q_{i,cond}^\rightarrow$ is the flow of soot particles entering the section i . The distribution between the sections is then written as

$$\begin{aligned} \Delta q_{i,cond}^\uparrow &= \left(\frac{(v_{i+1,max} - v_{i+1,min}) \ln(v_{i,max} / v_{i,min})}{(v_{i,max} - v_{i,min}) \ln(v_{i+1,max} / v_{i+1,min})} - 1 \right)^{-1} \Delta Q_{i,cond} \\ \Delta q_{i,cond}^\rightarrow &= \left(1 - \frac{(v_{i,max} - v_{i,min}) \ln(v_{i+1,max} / v_{i+1,min})}{(v_{i+1,max} - v_{i+1,min}) \ln(v_{i,max} / v_{i,min})} \right)^{-1} \Delta Q_{i,cond}. \end{aligned} \quad (17.92)$$

The source terms of condensation then become

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

$$\begin{aligned}\Delta Q_{1,cond} &= -\Delta q_{i,cond}^{\uparrow} \\ \Delta Q_{i,cond} &= \Delta q_{i,cond}^{\rightarrow} - \Delta q_{i,cond}^{\uparrow}, i = 2, 3, \dots, i-1 \\ \Delta Q_{imax,cond} &= \Delta q_{imax-1,cond}^{\rightarrow}.\end{aligned}\quad (17.93)$$

Note that these expressions are correct when the PAHs jump only from one section to the next. If the number of sections increases to the point that the section sizes fall below the size of PAHs, condensation will then be assimilated to the coagulation.

Sectional Soot Model

The Sectional Soot Model (SSM) ([Aubagnac-Karkar, 2014](#)) is based on the [PSM model](#). SSM gives access to the instantaneous soot number density function (SNDF) at each time-step and location. Unlike in the PSM model, which divides particles based on volume, the soot particles in SSM are separated with respect to their mass. The mass interval $[m_{MIN}, m_{MAX}]$ is divided into i_{max} number of sections. The lower bound of the first section, m_{MIN} , is given as the mass of $N - 2$ carbon atoms, where N is the number of carbon atoms in the smallest user-specified precursor.

For each subsequent section, i , the boundary soot masses are given as

$$m_{i,min} = m_{i-1,max}$$

and

$$m_{i,max} = (m_{MIN}) \left(\frac{m_{MAX}}{m_{MIN}} \right)^{\frac{i-1}{i_{max}-1}} \quad \text{for } i > 1, \quad (17.94)$$

where m_{MAX} is determined from the user-defined largest soot diameter ([ssm_soot.in > ssm_biggestsoot_diameter](#)) and soot density, ρ_{soot} ([ssm_soot.in > ssm_soot_density](#)).

Each section of soot particles in the given mass range is governed by a standard transport equation for the soot mass fraction in this section, $\tilde{Y}_{i,soot}$, according to

$$\frac{\partial \bar{\rho} \tilde{Y}_{i,soot}}{\partial t} + \nabla \cdot (\bar{\rho} \tilde{u} \tilde{Y}_{i,soot}) = \nabla \cdot (\bar{\rho} D_{t,soot} \nabla \tilde{Y}_{i,soot}) + \bar{\rho} \tilde{\omega}_{i,soot} \quad (17.95)$$

where $\bar{\rho}$ is the gas-phase density, \tilde{u} is the gas velocity, $D_{t,soot}$ is the turbulent diffusion coefficient of soot, and $\bar{\rho} \tilde{\omega}_{i,soot}$ is the soot source term for section i . The soot volume fraction, Q_i , can be written as

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

$$Q_i = \frac{\bar{\rho}}{\rho_{soot}} \tilde{Y}_{i,soot} \quad (17.96)$$

assuming that the volume of all soot particles is negligible relative to the volume of the gas. Therefore, the soot mass fraction source term in Equation 17.95 is given as

$$\bar{\rho} \tilde{\omega}_{i,soot} = \rho_{soot} \dot{Q}_i = \rho_{soot} (\dot{Q}_{i,pi} + \dot{Q}_{i,con} + \dot{Q}_{i,sg} + \dot{Q}_{i,ox} + \dot{Q}_{i,coag}), \quad (17.97)$$

where $\dot{Q}_{i,pi}$, $\dot{Q}_{i,con}$, $\dot{Q}_{i,coag}$, $\dot{Q}_{i,sg}$, and $\dot{Q}_{i,ox}$ are the volume fraction source terms for the i -th section due to particle inception, condensation, coagulation, surface growth, and oxidation, respectively.

The soot volume fraction density in a section i can be calculated from the soot volume fraction as

$$q_i = \frac{Q_i}{v_{i,max} - v_{i,min}}. \quad (17.98)$$

Further, the particle number density $n_i(m)$ can be calculated as

$$n_i(m) = \frac{q_i \cdot \rho_{soot}}{m}, \quad (17.99)$$

where m is the mass of the particles in the section. Finally, the volume number of particles in section i is calculated as

$$N_i = \int_{m_{i,min}}^{m_{i,max}} n_i(m) dm = \frac{Q_i \cdot \rho_{soot}}{m_{i,max} - m_{i,min}} \ln\left(\frac{m_{i,max}}{m_{i,min}}\right). \quad (17.100)$$

Equation 17.100 then allows you to calculate the soot number density function (SNDF). If you know the soot mass fraction, $\tilde{Y}_{i,soot}$, you can then calculate the instantaneous SNDF or any soot variable distribution at each time-step and location.

In the SSM, particle inception, coagulation, condensation and surface reactions can be activated independently in [ssm_soot.in](#).

Stages of SSM

Soot Particle Inception (Nucleation)

Two precursor species can collide and create the smallest soot particle through particle inception (or nucleation). The source term for inception is derived from the Smoluchowski equation to give

$$\rho \dot{\omega}_{pi,a+b} = \eta_{a+b} \gamma_{a+b} \beta_{a+b}^{fm} N_a N_b (m_a + m_b), \quad (17.101)$$

where γ_{a+b} is the collision efficiency, η_{a+b} is the dampening factor described in [Aubagnack-Karkar et al. \(2018\)](#), β_{a+b}^{fm} is the collision frequency, N_x is the density of the presence of the nucleation species, and m_x the mass of the nucleation species.

To activate the soot nucleation model, set `ssm_soot.in > ssm_soot_nucleation_model > active` greater than 0. You can use either reversible or non-reversible particle inception as described in [Aubagnack-Karkar et al. \(2018\)](#). If `ssm_soot.in > ssm_soot_collision_model > efficiency_model = 0`, specify the precursor species and correlating collision efficiencies in `ssm_soot.in > ssm_soot_nucleation_model > precursor_species`.

Soot Condensation

Collision between precursors and soot particles increases the size of the soot particle through condensation. The source term for condensation is given as

$$\rho \dot{\omega}_{i,con} = \gamma_{PAH+i} \beta_{PAH+i}^{fm} N_{PAH} \int_{m_{i,min}}^{m_{i,max}} (m_{PAH} + m) n_i(m) dm, \quad (17.102)$$

where γ is the collision efficiency, $m_{i,min}$ and $m_{i,max}$ are the boundaries of the current section, β_{PAH+i}^{fm} is the collision frequency between precursor and soot particles of section i , $n_i(m)$ is the particle number density, and N_{PAH+i} is the number density of the precursor species.

Condensation can either be non-reversible or reversible. To use a non-reversible soot condensation model, set `ssm_soot.in > ssm_soot_condensation_model > active = 1`. To use a reversible condensation model, set `ssm_soot.in > ssm_soot_condensation_model > active = 2` and specify a *nu_factor*, which is the intermolecular vibrational wavenumber used to calculate reversibility as described in [Aubagnack-Karkar et al. \(2018\)](#).

If `ssm_soot.in > ssm_soot_collision_model > efficiency_model = 0`, specify the soot condensation species and collision effectiveness in `ssm_soot.in > ssm_soot_condensation_model > condensation_species`.

Soot Coagulation

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

Coagulation is the process by which soot particles collide and form larger particles. It is modeled similarly to condensation. The coagulation source term for section i is given as

$$\rho \dot{\omega}_{i,j}^{coag} = \iint_{\Delta_i \Delta_j} \gamma_{i+j} \beta_{i+j} (m_i + m_j) n_i(m_i) n_j(m_j) dm_i dm_j, \quad (17.103)$$

where γ is the collision efficiency (if `ssm_soot.in > ssm_soot_collision_model > efficiency_model = 0`), β_{i+j} is the collision frequency, and m_x the mass of the coagulation species.

To activate the soot coagulation model, set `ssm_soot_coagulation_model > active = 1`.

Collision

The SSM requires collision modeling (*i.e.*, `ssm_soot.in > ssm_soot_collision_model > active = 1`). The collision efficiency γ is modeled according to

$$\gamma = 1 - \left(1 + \frac{\Phi_0}{k_B T} \right) \exp \left(- \frac{\Phi_0}{k_B T} \right), \quad (17.104)$$

where Φ_0 is the interaction potential well depth between the two colliding PAH particles and k_B is Boltzmann's constant.

When using the collision efficiency as described in [Aubagnack-Karkar et al. \(2018\)](#), the interaction potential well depth is calculated as

$$\Phi_0 = E_d D_1 D_2, \quad (17.105)$$

where D_1 and D_2 are the particle diameters and E_d is the collision efficiency factor (J/m^2). To use this collision efficiency model, set `ssm_soot.in > ssm_soot_collision_model > efficiency_model = 0`.

Soot Surface Reactions

Surface growth is the addition of gaseous phase carbon atoms to soot particles. The growth and oxidation of PAH on the surface is represented with a variation of the Hydrogen Abstraction Carbon Addition (HACA) cycle. The SSM adopts this surface chemistry cycle because it was designed for engine conditions. The six reactions in this cycle are:

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

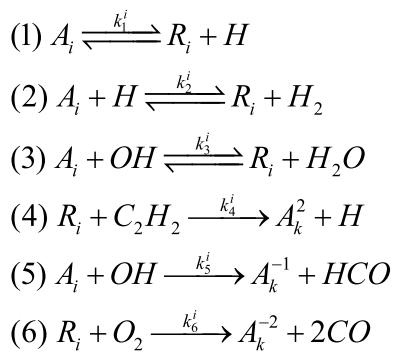


Figure 17.3: The reaction pathways of the HACA cycle.

In these reactions, A_i represents stable aromatic soot particles in section i (i.e., $C_{i,soot}$), R_i represents radical soot particles in section i (i.e., $C_{i,soot}^*$), A_k^p represents stable aromatic soot particles that are a product of a reaction that exchanges p carbon atoms with the gaseous phase, and k_n^i is the rate constant of the n -th reaction on particles of section i .

When `ssm_soot.in` > `ssm_soot_surface_growth_model` > `active = 1`, C_{soot}^* is a quasi-steady-state species and the total concentration of soot particles on a surface is given as

$$[C_{i,soot}] + [C_{i,soot}^*] = \int_{m_{i,\min}}^{m_{i,\max}} \alpha_{HACA} \frac{n_i(m)}{N_A} \left(\frac{m}{m_{C2}} \right)^{\theta/3} dm, \quad (17.106)$$

where α_{HACA} is the proportion of active sites, N_A is Avogadro's number, $n_i(m)$ is the particle number density, m is the mass, and θ is the steric factor, the ratio between the actual rate constant and the rate constant predicted by collision theory.

When `ssm_soot.in` > `ssm_soot_surface_growth_model` > `active = 2`, the total concentration of soot particles on a surface is given as

$$[C_{i,soot}] = \int_{m_{i,\min}}^{m_{i,\max}} \alpha_{HACA} \frac{n(m)}{N_A} \left(\frac{m}{m_{C2}} \right)^{\theta/3} dm. \quad (17.107)$$

The steric factor θ changes as a function of the size of the particles as depicted by the following graph.

Chapter 17: Emissions Modeling

Soot Modeling Detailed Soot Models

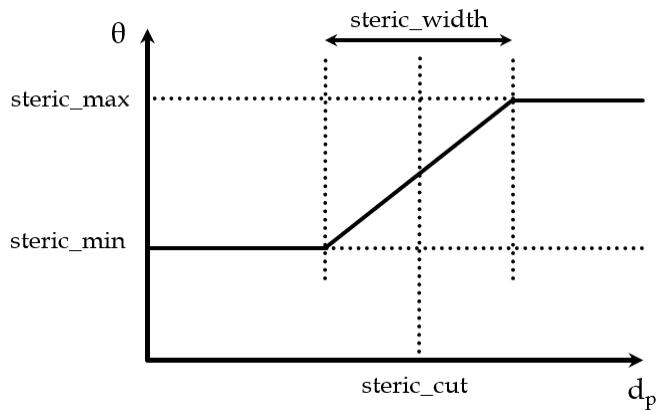


Figure 17.4: Steric factor as a function of particle size.

Surface growth

The volume fraction rate of soot particles in section i which are growing enough to switch to section $i + 1$ is modeled as

$$\dot{Q}_{sg}^{out} = N_A K_{i,sg} \int_{m_{C2}-m_{i,\max}}^{i,\max} (m+m_{C2}) \frac{\rho}{m} \left(\frac{m}{m_{C2}} \right)^{\theta/3} dm. \quad (17.108)$$

Oxidation

The volume fraction rate of soot transferred from section i to section $i-1$ is modeled as

$$\dot{Q}_{ox,i}^{out} = N_A K_{i,ox} \int_{m_{i,\min}}^{m_{i,\min}+m_{C2}} (m+m_{C2}) \frac{\rho}{m} \left(\frac{m}{m_{C2}} \right)^{\theta/3} dm. \quad (17.109)$$

To activate the soot surface reaction model, set `ssm_soot.in > ssm_soot_surface_growth_model > active = 1` and configure the parameters in the `ssm_soot_surface_growth_model` settings block.

Soot Modeling Case Setup

To activate soot modeling, set `combust.in > emissions_flag = 1` and include both `combust.in` and `emissions.in`.

You can use the Hiroyasu-NSC model along with the more detailed soot models. However, only one detailed or phenomenological soot model may be activated at one time when coupled with the [SAGE detailed chemical kinetics solver](#).

We recommend that you couple the [SAGE detailed chemical kinetics solver](#) with all the non-empirical soot models for the most accurate results. You can simulate detailed soot models with other combustion models, such as [G-Equation](#) (as long as at least one of `combust.in >`

`g_eqn_model > burned_region`, `g_eqn_model > on_flame`, or `g_eqn_model > unburned_region = SAGE`) or [RIF](#). With the detailed soot models, you can speed up the calculation via several settings: activate the iterative solver ([`combust.in > sage_model > ode_solver > type = ITERATIVE`](#) or [`ITERATIVE_SUPERLU`](#)), the analytical Jacobian ([`combust.in > sage_model > ode_solver > analyt_jac = 1`](#)), and [`adaptive zoning`](#) options ([`combust.in > adaptive_zone > bin_options`](#)). Detailed soot modeling is also compatible with dynamic mechanism reduction (set [`combust.in > sage_model > dmr_flag = 1`](#)).

While the detailed soot models are typically tightly two-way coupled with the [SAGE detailed chemical kinetics solver](#), you can force detailed soot modeling calculations to run with a different chemistry solver by setting [`emissions.in > detailed_soot_model > coupled_solver_flag = 0`](#) and [`detailed_soot_model > active = PM`](#) or [`PSM`](#) for the PM or PSM model, respectively.

Phenomenological models provide the soot number density, the soot mass and the soot mass due to nucleation, surface growth, oxidation and coagulation. You can obtain averaged soot number density, soot mass, soot size, and the mass for different soot sub-processes (nucleation, surface growth, oxidation, and coagulation) as output in both the PM and PSM detailed soot models. When using the PSM model, you can also obtain the PSDF.

Soot Precursors

When using the PM or PSM model, at least one of the following compounds must be included in the [reaction mechanism file](#): A4R5, A4 (pyrene), or A3R5- (acephenanphthryl). CONVERGE searches `mech.dat` for these compounds (in the order in which they are listed in the previous sentence) and assigns as the soot precursor the first one of these compounds that it finds in the [reaction mechanism file](#). For example, A4 will be the soot precursor only if A4R5 is not included in the [reaction mechanism file](#).

The soot precursor for each phenomenological soot model is hard-coded in CONVERGE.

For the PM, PSM, or phenomenological models, you can add other soot precursors via the [`emissions.in > custom_soot_precursor`](#) settings block. Set [`emissions.in > custom_soot_precursor > active = 1`](#) and then specify additional soot precursors via [`custom_soot_precursor > precursor_species`](#).

17.3 Emissions Post-Processing

Accurate prediction of emissions typically requires using a detailed chemical mechanism, which inevitably increases the computational cost. An alternative method involves using the emissions post-processing feature. This feature enables you to run a 3D simulation based on a previous run that was generated with a simple combustion model or reduced mechanism. It assumes that the major fluid quantities (e.g., velocity, temperature, pressure, density) do not vary greatly provided the geometry and inflow conditions remain constant. This assumption means that those quantities can be fixed and only the species and emissions quantities need to be solved, which can be very computationally efficient.

Chapter 17: Emissions Modeling

Emissions Post-Processing

To use this method, you must have a restart file that was generated from a simulation with combustion modeling active (*i.e.*, from a simulation with [*inputs.in*](#) > *feature_control* > *combustion_modeling* = 1). You can use the [FGM](#) combustion model or the [SAGE detailed chemical kinetics solver](#) to create this restart file. When using SAGE, we suggest using a reduced mechanism to save computational resources.

After you have obtained your restart file, set [*inputs.in*](#) > *simulation_control* > *restart_flag* = 1 (to direct CONVERGE to use this restart file) and then set [*inputs.in*](#) > *feature_control* > *post_emission_flag* = 1. In order to use emissions post-processing, you must invoke the [SAGE detailed chemical kinetics solver](#) (*i.e.*, [*combust.in*](#) > *sage_model* > *active* = 1) and activate at least one emissions model. We recommend that you use this method with the transient solver (*i.e.*, [*inputs.in*](#) > *solver_control* > *steady_solver* = 0). You must use the [steady-state monitor](#) (set [*inputs.in*](#) > *solver_control* > *monitor_steady_state* = 1 and include [*monitor_steady_state.in*](#) in your case setup) to determine when to stop the simulation. A general suggestion is to monitor the region-averaged emission quantities to determine the steady status.

The emissions post-processing feature can also help you optimize emissions parameters. Since it only solves species and emission quantities, it is much faster compared with re-running the detailed simulations.

Chapter



18

Source Modeling

18 Source Modeling

This chapter describes the options in CONVERGE for sources and sinks of energy, momentum, turbulent kinetic energy, turbulent dissipation rate, specific dissipation rate, species, [passives](#), [scalars](#), or porous media. To activate source/sink modeling, set [inputs.in](#) > *feature_control* > *source_flag* = 1 and specify the relevant parameters in [source.in](#). The source/sink volumes are defined by specifying the shape of the source (*i.e.*, box, sphere, cylinder, region, line, circle, boundary, or proximity) and the stream(s) to which it applies. The [source.in](#) file also contains information regarding the strength of the source, the start and end times of the source, and the motion of the source (if any).

If [source.in](#) > *source* > *shape* > *type* = LINE or CIRCLE, it represents a collection of point sources along the line or around the circumference of the circle, respectively. For LINE and CIRCLE sources, *source* > *shape* > *type* can be specified only by *source* > *value* (not by *source* > *unit_volume*). The volume of the cell encompassing the point source determines the local strength of the source. You do not have to consider the local grid refinements (due to embedding and AMR) while specifying the magnitude for *source* > *value* because the total strength of the source is equally divided by the number of points in a LINE or CIRCLE source, which is useful when specifying multiple sources of the same type with the same values.

For a LINE source, the first and the last point sources are located at [source.in](#) > *source* > *shape* > *x1_center* and *source* > *shape* > *x2_center*, respectively, while the remaining point sources are equidistantly spaced between the first and last points. For a CIRCLE source, the point sources are equidistantly spaced around the circumference of the circle, with a randomly selected starting point. The coordinates for each point source on the CIRCLE are not definite.

Note that in the case of LINE and CIRCLE sources with [source.in](#) > *source* > *moving_control* > *moving_flag* = MOVE_WITH_FLOW, each point can attain a different velocity due to the spatial variation of fluid velocity. A point source moving with the flow is useful for simulating the expanding arc of a spark energy source during engine simulations.

You can reset the source to its original coordinates in one of two ways, depending on the value of [source.in](#) > *source* > *moving_control* > *reset_source_flag*. When [source.in](#) > *reset_source_flag* = RESET_AT_MAX_DISPLACE, if all the points on the line or circle exceed [source.in](#) > *source* > *moving_control* > *max_displace*, CONVERGE resets the whole line or circle to its original location. When [source.in](#) > *source* > *moving_control* > *reset_source_flag* = RESET_LINE_CIRCLE, if any point on the line or circle exceeds the value of [source.in](#) > *source* > *moving_control* > *max_displace*, CONVERGE resets the whole line or the circle to its original location.

Use [source.in](#) > *source* > *mult_dt_source* to control the time-step size. This parameter limits the increase in magnitude of a source during a single time-step. For example, if *mult_dt_source* = 0.5, CONVERGE will adjust the time-step so that the source magnitude cannot increase more than 50% in a single time-step.

All source types (except for porous media) are limited by a user-specified maximum value of a monitored flow variable. If the monitored variable is approaching the maximum allowed value in a cell, CONVERGE will reduce the value of the source term to prevent the flow variable from substantially exceeding the specified limit.

CONVERGE limits the maximum value of the solution variable by neglecting the convection and diffusion terms of the appropriate transport equation and evaluating the local rate of change each time-step,

$$\frac{\partial \rho \phi}{\partial t} = S. \quad (18.1)$$

The local value of the source, S , is limited so that the monitored solution variable will reach the specified maximum after the next time-step. If the monitored solution variable is already greater than specified maximum, S will take on a negative value, sufficient to reduce the monitored variable to its specified maximum at the next time-step. The full transport equation is then solved with the modified source term. Because the source term is modified before convection and diffusion take place, the monitored flow variable will have a value of approximately the maximum limit, but not precisely.

18.1 Energy Source Modeling

CONVERGE solves for energy sources when `inputs.in > solver_control > energy_solver = INTERNAL` or `TOTAL` and `source.in > source > equation = ENERGY`. If `inputs.in > solver_control > energy_solver = INTERNAL`, CONVERGE solves the compressible form of the energy equation is given by

$$\frac{\partial \rho e}{\partial t} + \frac{\partial u_j \rho e}{\partial x_j} = -P \frac{\partial u_j}{\partial x_j} + \sigma_{ij} \frac{\partial u_i}{\partial x_j} + \frac{\partial}{\partial x_j} \left(K \frac{\partial T}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left(\rho D \sum_m h_m \frac{\partial Y_m}{\partial x_j} \right) + S, \quad (18.2)$$

where ρ is density, Y_m is the mass fraction of species m , D is the mass diffusion coefficient, P is the pressure, e is the specific internal energy, K is the conductivity, h_m is the species enthalpy, σ_{ij} is the stress tensor, T is the temperature, and S is the energy source term (`source.in > source > source_value` or `source.in > source > source_unit_volume`, depending on the value of `source.in > source > source_type`).

If `inputs.in > solver_control > energy_solver = TOTAL`, CONVERGE solves a modified form of the equation, given by

Chapter 18: Source Modeling

Energy Source Modeling

$$\begin{aligned} \frac{\partial}{\partial t} \left(\rho \left(e + \frac{1}{2} u_k^2 \right) \right) + \frac{\partial}{\partial x_j} \left(\rho u_j \left(e + \frac{1}{2} u_k^2 \right) \right) = \\ - \frac{\partial}{\partial x_j} (u_j P) + \frac{\partial}{\partial x_j} (u_i \sigma_{ij}) + \frac{\partial}{\partial x_j} \left(K \frac{\partial T}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left(\rho D \sum_m h_m \frac{\partial Y_m}{\partial x_j} \right) + S. \end{aligned} \quad (18.3)$$

Thermal Runaway Source Modeling

Thermal runaway is the process by which an increase in temperature accelerates the release of more energy, leading to an even greater increase in temperature. This positive feedback mechanism is an important safety concern for designing lithium-ion batteries. You can specify thermal runaway species in passive-based [AMR](#) to improve thermal runaway modeling.

CONVERGE offers models for thermal runaway of energy sources. To model thermal runaway in CONVERGE, set [source.in](#) > *source* > *equation* = ENERGY. Then set [source.in](#) > *source* > *type* = HATCHARD_MODEL or REN_MODEL for the [Hatchard-Kim model](#) or the [Ren model](#), respectively, and include [thermal_runaway_model.in](#) in your case setup.

Hatchard-Kim Model

The Hatchard-Kim model ([Hatchard et al., 2001](#) and [Kim et al., 2007](#)) calculates heat generation from four exothermic reactions: the decomposition of the solid electrolyte interface (SEI), the reaction between the anode and electrolyte, the reaction between the cathode and electrolyte, and the electrolyte decomposition.

Decomposition of SEI

The anode is initially protected from direct reaction with the electrolyte because of the solid electrolyte interface (SEI) film. The SEI begins to decompose in an exothermic reaction around 90 to 120 °C. The rate of the SEI decomposition reaction, R_{SEI} , is given by

$$\frac{dc_{SEI}}{dt} = -R_{SEI}(T, c_{SEI}) = -A_{SEI} \exp \left[-\frac{E_{a,SEI}}{RT} \right] c_{SEI}^{m_{SEI}}, \quad (18.4)$$

where R_{SEI} is the rate of SEI decomposition (s^{-1}), T is the temperature (K), c_{SEI} is the normalized concentration of SEI film (dimensionless), A_{SEI} is the pre-exponential factor (s^{-1}), $E_{a,SEI}$ is the activation energy for the SEI decomposition reaction (J/mole), R is the gas constant, and m_{SEI} is the reaction order of c_{SEI} . The heat produced from the SEI decomposition reaction, Q_{SEI} , is then given by

$$Q_{SEI} = H_{SEI} W_C R_{SEI}, \quad (18.5)$$

Chapter 18: Source Modeling

Energy Source Modeling Thermal Runaway Source Modeling

where H_{SEI} is the specific heat release of the SEI decomposition reaction (J/kg) and W_C is the specific carbon content in the jelly roll (kg/m^3).

Anode and electrolyte reaction

The anode and electrolyte react exothermically according to

$$\frac{dc_{An}}{dt} = -R_{An-E}(T, c_E, c_{An}, t_{SEI}) = -A_{An-E} \exp\left[-\frac{t_{SEI}}{t_{SEI,Ref}}\right] c_{An}^{m_{An}} \exp\left[-\frac{E_{a,An-E}}{RT}\right] \quad (18.6)$$

where R_{An-E} is the rate of the anode and electrolyte reaction (s^{-1}), T is the temperature (K), c_E is the amount of electrolyte (dimensionless), c_{An} is the amount of anode (dimensionless), A_{An-E} is the pre-exponential factor (s^{-1}), t_{SEI} is a measure of the SEI layer thickness (dimensionless) that reflects the amount of lithium in the SEI, $t_{SEI,Ref}$ is set as $t_{SEI,init}$ according to [Kim et al. \(2007\)](#), $E_{a,An-E}$ is the activation energy for the anode and electrolyte reaction ($J/mole$), R is the gas constant, and m_{An} is the reaction order of c_{An} . The heat produced from the anode and electrolyte reaction, Q_{An-E} , is then given by

$$Q_{An-E} = H_{An-E} W_C R_{An-E}, \quad (18.7)$$

where H_{An-E} is the specific heat release of the anode and electrolyte reaction (J/kg) and W_C is the specific carbon content in the jelly roll (kg/m^3).

Cathode and electrolyte reaction

The cathode active material reacts highly exothermically with the electrolyte according to

$$\frac{d\alpha}{dt} = R_{Cat-E}(T, \alpha, c_E) = A_{Cat-E} \alpha^{m_{Cat,1}} (1-\alpha)^{m_{Cat,2}} \exp\left[-\frac{E_{a,Cat-E}}{RT}\right] \quad (18.8)$$

where R_{Cat-E} is the rate of the cathode and electrolyte reaction (s^{-1}), T is the temperature (K), α is the fractional degree of conversion for the cathode, c_E is the amount of electrolyte (dimensionless), A_{Cat-E} is the pre-exponential factor (s^{-1}), $E_{a,Cat-E}$ is the activation energy for the cathode and electrolyte reaction ($J/mole$), R is the gas constant, $m_{Cat,1}$ is the reaction order of α , and $m_{Cat,2}$ is the reaction order for $(1-\alpha)$. The heat produced from the cathode and electrolyte reaction, Q_{Cat-E} , is then given by

$$Q_{Cat-E} = H_{Cat-E} W_{Cat} R_{Cat-E}, \quad (18.9)$$

Chapter 18: Source Modeling

Energy Source Modeling Thermal Runaway Source Modeling

where H_{Cat-E} is the specific heat release of the cathode and electrolyte reaction (J/kg) and W_{Cat} is the specific cathode active content in the jelly roll (kg/m^3).

Electrolyte decomposition reaction

The electrolyte can also decompose exothermically at high temperatures >200 °C. This decomposition reaction is expressed as

$$\frac{dc_E}{dt} = -R_E(T, c_E) = -A_E \exp\left[-\frac{E_{a,E}}{RT}\right] c_E^{m_E} \quad (18.10)$$

where R_E is the rate of electrolyte decomposition (s^{-1}), T is the temperature (K), c_E is the amount of electrolyte (dimensionless), A_E is the pre-exponential factor (s^{-1}), $E_{a,E}$ is the activation energy for the electrolyte decomposition ($J/mole$), R is the gas constant, and m_E is the reaction order of c_E . The heat produced from the electrolyte decomposition, Q_E , is then given by

$$Q_E = H_E W_E R_E, \quad (18.11)$$

where H_E is the specific heat release of the electrolyte decomposition reaction (J/kg) and W_E is the specific electrolyte content in the jelly roll (kg/m^3).

The total heat generated from the battery using this model is thus given as

$$Q_{generated} = Q_{SEI} + Q_{An-E} + Q_{Cat-E} + Q_E \quad (18.12)$$

Ren Model

The model proposed by [Ren et al. \(2018\)](#) calculates heat generation from six exothermic reactions: the decomposition of the solid electrolyte interface (SEI), the reaction between the anode and electrolyte, the reaction between the anode and binder, the decomposition of the cathode, the reaction between the cathode and anode, and the reaction between the cathode and binder.

Decomposition of SEI

The SEI begins to decompose in an exothermic reaction. The rate of the SEI decomposition reaction is given by

$$\frac{dc_{SEI}}{dt} = R_{SEI} = -A_{SEI} \exp\left[-\frac{E_{a,SEI}}{RT}\right] c_{SEI}^{m_{SEI}} \quad (18.13)$$

Chapter 18: Source Modeling

Energy Source Modeling Thermal Runaway Source Modeling

where R_{SEI} is the rate of SEI decomposition (s^{-1}), T is the temperature (K), c_{SEI} is the normalized concentration of SEI (dimensionless), A_{SEI} is the pre-exponential factor (s^{-1}), $E_{a,SEI}$ is the activation energy for the SEI decomposition reaction (J/mole), R is the gas constant, and m_{SEI} is the reaction order of c_{SEI} . The heat produced from the SEI decomposition reaction, $Q_{SEI'}$, is then given by

$$Q_{SEI} = m_{reactant} H_{SEI} R_{SEI}, \quad (18.14)$$

where H_{SEI} is the specific heat release of the SEI decomposition reaction (J/kg) and $m_{reactant}$ is the mass density of the reactants (kg/m^3).

Reaction between anode and electrolyte

The anode and electrolyte react exothermically according to

$$R_{An-E} = A_{An-E} \exp\left[-\frac{E_{a,An-E}}{RT}\right] c_{An-E}^{m_{An-E}} \quad (18.15)$$

where R_{An-E} is the rate of the anode and electrolyte reaction (s^{-1}), T is the temperature (K), c_{An-E} is the normalized concentration of anode active material (dimensionless), A_{An-E} is the pre-exponential factor (s^{-1}), $E_{a,An-E}$ is the activation energy for the anode and electrolyte reaction (J/mole), R is the gas constant, and m_{An-E} is the reaction order of c_{An-E} . The heat produced from the anode and electrolyte reaction, $Q_{An-E'}$, is then given by

$$Q_{An-E} = m_{reactant} H_{An-E} R_{An-E}, \quad (18.16)$$

where H_{An-E} is the specific heat release of the anode and electrolyte reaction (J/kg) and $m_{reactant}$ is the mass density of the reactants (kg/m^3).

Reaction between anode and binder

The reaction between the anode active material and the binder material proceeds according to

$$R_{An-B} = A_{An-B} \exp\left[-\frac{E_{a,An-B}}{RT}\right] c_{An-B}^{m_{An-B}} \quad (18.17)$$

where R_{An-B} is the rate of the anode and binder reaction (s^{-1}), T is the temperature (K), c_{An-B} is the normalized concentration of binder material (dimensionless), A_{An-B} is the pre-exponential factor (s^{-1}), $E_{a,An-B}$ is the activation energy for the anode and binder reaction (J/mole), R is the

Chapter 18: Source Modeling

Energy Source Modeling Thermal Runaway Source Modeling

gas constant, and m_{An-B} is the reaction order of c_{An-B} . The heat produced from the anode and electrolyte reaction, Q_{An-B} , is then given by

$$Q_{An-B} = m_{reactant} H_{An-B} R_{An-B}, \quad (18.18)$$

where H_{An-B} is the specific heat release of the anode and electrolyte reaction (J/kg) and $m_{reactant}$ is the mass density of the reactants (kg/m^3).

Decomposition of cathode

The cathode decomposes according to

$$\frac{dc_{Cat}}{dt} = R_{Cat} = -A_{Cat} \exp\left[-\frac{E_{a,Cat}}{RT}\right] c_{Cat}^{m_{Cat}} \quad (18.19)$$

where R_{Cat} is the rate of cathode decomposition (s^{-1}), T is the temperature (K), c_{Cat} is the normalized concentration of cathode active material (dimensionless), A_{Cat} is the pre-exponential factor (s^{-1}), $E_{a,Cat}$ is the activation energy for the cathode decomposition reaction ($J/mole$), R is the gas constant, and m_{Cat} is the reaction order of c_{Cat} . The heat produced from the anode and electrolyte reaction, Q_{Cat} , is then given by

$$Q_{Cat} = m_{reactant} H_{Cat} R_{Cat}, \quad (18.20)$$

where H_{Cat} is the specific heat release of the anode and electrolyte reaction (J/kg) and $m_{reactant}$ is the mass density of the reactants (kg/m^3).

Reaction between cathode and anode

The reaction between the cathode and anode proceeds according to

$$R_{Cat-An} = A_{Cat-An} \exp\left[-\frac{E_{a,Cat-An}}{RT}\right] c_{Cat-An}^{m_{Cat-An}} \quad (18.21)$$

where R_{Cat-An} is the rate of the cathode and anode reaction (s^{-1}), T is the temperature (K), c_{Cat-An} is the normalized concentration of anode active material (dimensionless), A_{Cat-An} is the pre-exponential factor (s^{-1}), $E_{a,Cat-An}$ is the activation energy for the cathode and anode reaction ($J/mole$), R is the gas constant, and m_{Cat-An} is the reaction order of c_{Cat-An} . The heat produced from the anode and electrolyte reaction, Q_{Cat-An} , is then given by

$$Q_{Cat-An} = m_{reactant} H_{Cat-An} R_{Cat-An}, \quad (18.22)$$

Chapter 18: Source Modeling

Energy Source Modeling Thermal Runaway Source Modeling

where H_{Cat-An} is the specific heat release of the anode and electrolyte reaction (J/kg) and $m_{reactant}$ is the mass density of the reactants (kg/m^3).

Reaction between cathode and binder

The reaction between the cathode active material and the binder material proceeds according to

$$R_{Cat-B} = A_{Cat-B} \exp\left[-\frac{E_{a,Cat-B}}{RT}\right] c_{Cat-B}^{m_{Cat-B}} \quad (18.23)$$

where R_{Cat-B} is the rate of the cathode and binder reaction (s^{-1}), T is the temperature (K), c_{Cat-B} is the normalized concentration of binder material (dimensionless), A_{Cat-B} is the pre-exponential factor (s^{-1}), $E_{a,Cat-B}$ is the activation energy for the cathode and binder reaction ($J/mole$), R is the gas constant, and m_{Cat-B} is the reaction order of c_{Cat-B} . The heat produced from the anode and electrolyte reaction, Q_{Cat-B} , is then given by

$$Q_{Cat-B} = m_{reactant} H_{Cat-B} R_{Cat-B}, \quad (18.24)$$

where H_{Cat-B} is the specific heat release of the anode and electrolyte reaction (J/kg) and $m_{reactant}$ is the mass density of the reactants (kg/m^3).

Mass balance equations for the reactions between anode, cathode, electrolyte, and binder
For Equations 18.18, 18.22, and 18.24, the mass balance equations are given as

$$\frac{dc_{An-E}}{dt} = \frac{dc_{Cat-An}}{dt} = -(R_{An-E} + R_{Cat-An}) \quad (18.25)$$

$$\frac{dc_{An-B}}{dt} = \frac{dc_{Cat-B}}{dt} = -\left(\frac{\gamma}{1+\gamma} R_{An-B} + R_{Cat-B}\right) \quad (18.26)$$

where γ is [thermal_runaway_model.in > bind_gamma_factor](#).

Given the above equations, the total heat generated from this model is then given as

$$Q_{generated} = Q_{SEL} + Q_{An-E} + Q_{An-B} + Q_{Cat} + Q_{Cat-An} + Q_{Cat-B}. \quad (18.27)$$

Energy Sources in Engine Models

The purpose of using energy sources in engine applications is to represent fundamental physical processes at the appropriate point in the engine cycle. There are three common instances where you specify energy sources in engine applications:

- Applying specified measurements or simulation data for [heat release rate](#),
- Adding [spark energy sources](#) directly to represent the energy discharge from a spark plug, and
- Using a measured or simulated [cylinder pressure trace](#) to represent the mean pressure of the cylinder during an engine cycle.

Heat Release Data

Use the heat release rate option ([*source.in*](#) > *source* > *type* = HEAT_RELEASE_DATA) to apply a known heat release rate over an entire region (or shape). The heat release rate option can help you avoid some of the computationally expensive chemistry calculations for engine simulations in CONVERGE. Note that this approach should only be used when the details of the combustion are not of interest. The specified heat release rate can be obtained from CONVERGE (*i.e.*, single cylinder results), from experimental data, or from 1-dimensional simulations.

To approximate combustion using heat release rate, specify a file name (*e.g.*, *heat_release.in*) that contains the heat release data for [*source.in*](#) > *source* > *value*. The heat release file must have the keyword HEAT_RELEASE as shown in the figure below. The units for heat release data are in J/s if [*inputs.in*](#) > *simulation_control* > *crank_flag* = 0 and in J/CAD if *crank_flag* is non-zero.

```
TEMPORAL
CYCLIC
crank           HEAT_RELEASE
-204.98        2.150E-03
-204.82        4.760E-03
-204.33        1.100E-02
.
.
.
119.72         1.040E+04
120.21         0.000E+00
515.02         2.150E-03
```

Figure 18.1: An excerpt of a heat release data file.

Spark Energy

In a typical spark discharge, voltage rises between the two electrodes until there is an electrical breakdown in the spark gap. In this first stage of the electrical discharge (called the breakdown phase), the mixture between the electrodes is ionized into plasma, which propagates from one electrode to another.

The breakdown phase is followed by the arc phase, in which the thin cylindrical plasma expands largely due to heat conduction and diffusion. The arc phase is followed by a glow discharge phase, in which, depending on the details of the ignition system, the energy

storage device will dump its energy into the discharge circuit. With inflammable mixtures, the exothermic reactions in the arc phase lead to a self-sustaining propagating flame during the glow discharge phase.

Time scales in the breakdown phase are significantly smaller than the arc/glow phase. Since the discharge times of the arc and glow discharge phases are similar, this energy is often combined together as one energy source. In a CONVERGE simulation, you will therefore specify two energy sources that briefly overlap. You must ensure that the breakdown phase occurs earlier and is of shorter duration than arc/glow phase by correctly specifying [*source.in* > source > temporal_control > source_start_time](#) and [*source_end_time*](#) for the two sources, with consideration for the engine speed. Note that the two overlapping spark sources cause a step change with respect to time in the energy release. For a spark plug, specify the *source > type* as SPHERE, BOX, CYLINDER, or LINE centered at the midpoint between the two electrodes. Typically, *source > type* = SPHERE is used to simulate the spark. For an example of a SPHERE source to simulation the spark, consult the SI8 engine SAGE PFI or SAGE premixed example cases.

Modeling Spark Energy with a LINE Source

Fluid motion in the spark region is quite significant. You can expect both mean and fluctuating velocities to be in the range of 1 to 10 m/s in this region, as described by Heywood (1988). On the time scale of the breakdown phase, this fluid motion is not important. But in the arc/glow phase, the arc between the electrodes will be advected with the flow and stretches out in length.

To more accurately simulate the arc/glow phase advection, specify the source as moving with the flow by setting [*source.in* > source > moving_control > moving_flag](#) = MOVE_WITH_FLOW. This option will displace the source with the velocity of the flow in the arc/glow phase. A LINE source may more accurately represent an advected arc/glow phase because it is a collection of point sources, and each point in the LINE can attain different velocities due to the spatial variation of fluid velocities. Points close to the electrode will be affected by the boundary layer of the fluid motion while points in between will be advected by the bulk motion of the flow. Hence, a LINE source can closely resemble the stretched-out arc.

Figure 18.2 below shows a typical [*source.in*](#) file that includes a *source > type* = LINE to represent spark energy. In this example, *source > value* = 25 mJ for the breakdown phase for 0.5 crank angle degrees (700.0 to 700.5) and *source > value* = 25 mJ for the arc/glow phase for 10 crank angle degrees (700.0 to 710.0). For multiple-cycle simulations of four-stroke engines, the spark discharge repeats after 720 crank angle degrees, which is specified by the CYCLIC 720 row. Note that the *source > type* can be either 0 or 1 for specifying spark energy.

```
version: 3.0
---
- source:
    description: breakdown
    equation: ENERGY
    type: TOTAL_VALUE
```

Chapter 18: Source Modeling

Energy Source Modeling Energy Sources in Engine Models

```
value: 0.025
temporal_control:
    type: CYCLIC
    cyclic_period: 720
    source_start_time: 700.0
    source_end_time: 700.5
max_value: 50000
shape:
    type: LINE
    x_center: [-0.000355, 0.00555, 0.0522]
    radius: [-0.000355, 0.00555, 0.0516]
moving_control:
    moving_flag: STATIONARY
    velocity: [1, 2, 3]
    max_displace: 0.005
    reset_source_flag: DO_NOT_RESET
    mult_dt_source: 1
-
source:
    description: arc_glow
    equation: ENERGY
    type: TOTAL_VALUE
    value: 0.025
    temporal_control:
        type: CYCLIC
        cyclic_period: 720
        source_start_time: 700.0
        source_end_time: 710.0
    max_value: 50000
    shape:
        type: LINE
        x_center: [-0.000355, 0.00555, 0.0522]
        radius: [-0.000355, 0.00555, 0.0516]
    moving_control:
        moving_flag: MOVE_WITH_FLOW
        velocity: [1, 2, 3]
        max_displace: 0.005
        reset_source_flag: RESET_LINE_CIRCLE
    mult_dt_source: 1
```

Figure 18.2: An example *source.in* for spark energy input.

Pressure Trace Data

Similar to the specification of the [heat release rate](#), you can provide cylinder pressure trace data. CONVERGE will use this pressure trace data to provide an energy source that represents the heat release due to combustion. Note that [*source.in*](#) > *source* > *type* must be set to *PRESSURE_TRACE*. You can obtain the pressure trace data by running one cycle of combustion calculations in CONVERGE, from experimental data, or from one-dimensional simulations.

To approximate combustion using pressure trace data, specify a file (e.g., *pressure.in*) that contains the pressure trace data as the [*source.in*](#) > *source* > *value*. This data file must contain the keyword *PRESSURE_CURVE* as shown below in Figure 18.3. The pressure trace data has units of *Pa*.

```
TEMPORAL
CYCLIC
crank      PRESSURE_CURVE
-204.98 96823.79
-204.82 96866.36
-204.33 97002.27
:
:
```

Chapter 18: Source Modeling

Energy Source Modeling Energy Sources in Engine Models

118.76	92645.65
120.21	91368.29
515.2	96823.79

Figure 18.3: An example of a pressure trace data file.

18.2 Momentum Source Modeling

CONVERGE solves for momentum sources when [*inputs.in*](#) > *solver_control* > *momentum_solver* = 1 and [*source.in*](#) > *source* > *equation* = *U-EQ*, *V-EQ* or *W-EQ*. The momentum equation is given by

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_j} = - \frac{\partial P}{\partial x_i} + \frac{\partial \sigma_{ij}}{\partial x_j} + S_i. \quad (18.28)$$

In Equation 18.28, u_i is velocity, ρ is density, P is pressure, σ_{ij} is the stress tensor, and S_i is the *momentum source* term. Depending on the direction of the momentum source, (*i.e.*, x , y , and z vector components), the momentum source will be *U-EQ*, *V-EQ*, or *W-EQ*, respectively. Note that the *max_value* for a momentum source is specified in terms of the absolute value of velocity (m/s).

18.3 Turbulent Kinetic Energy Source Modeling

You can specify turbulent kinetic energy sources for simulations that use a [RANS turbulence model \(k-epsilon or k-omega\)](#) or an [LES one-equation model](#). CONVERGE solves for the turbulent kinetic energy source when [*inputs.in*](#) > *solver_control* > *turbulence_solver* = 1 and [*source.in*](#) > *source* > *equation* = *TKE*. The general form of the turbulent kinetic energy transport equation is given by

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_i k}{\partial x_i} = P + D - E + S, \quad (18.29)$$

where P is the production term, D is the diffusion term, E is the dissipation term, and S is the source term. See [Chapter 13 - Turbulence Modeling](#) for details about these terms. Note that the diffusion and dissipation terms vary depending on the type of turbulence model (RANS k-epsilon/LES one-equation versus RANS k-omega).

18.4 Turbulent Dissipation Source Modeling

You can specify turbulent dissipation sources for simulations that use a [RANS k-epsilon turbulence model](#). CONVERGE solves for the turbulent dissipation source when [*inputs.in*](#) > *solver_control* > *turbulence_solver* = 1 and [*source.in*](#) > *source* > *equation* = *EPS*. The turbulent dissipation transport equation is given by

Chapter 18: Source Modeling

Turbulent Dissipation Source Modeling

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial (\rho u_i \varepsilon)}{\partial x_i} = \frac{\partial}{\partial x_j} \left(\frac{\mu}{Pr_\varepsilon} \frac{\partial \varepsilon}{\partial x_j} \right) + c_{\varepsilon 3} \rho \varepsilon \frac{\partial u_i}{\partial x_i} + \left(c_{\varepsilon 1} \frac{\partial u_i}{\partial x_j} \tau_{ij} - c_{\varepsilon 2} \rho \varepsilon + c_s S_s \right) \frac{\varepsilon}{k} + S - \rho R_\varepsilon, \quad (18.30)$$

where ε is the turbulent dissipation; $c_{\varepsilon 1}$, $c_{\varepsilon 2}$, and $c_{\varepsilon 3}$ are model constants; ρ is the density; S is the user-supplied source term; and S_s is the source term that represents interactions with discrete phase (spray). Note that S and S_s are distinct from one another. In Equation 18.30, R_ε depends on which RANS k - ε model you are using. If you are using the Standard k - ε model, $R_\varepsilon = 0$. If you are using the RNG k - ε model, then R_ε is defined by the [linked equation](#).

18.5 Specific Dissipation Source Modeling

You can specify specific dissipation sources for simulations that use a [RANS k-omega turbulence model](#). CONVERGE solves for the specific dissipation source when `inputs.in > solver_control > turbulence_solver = 1` and `source.in > source > equation = OMEGA`. The specific dissipation transport equation is given by

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_j \omega}{\partial x_j} = \frac{\alpha \omega}{k} P - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + S, \quad (18.31)$$

where α , β , and μ_t are model constants; ρ is the density; ω is the specific dissipation, and S is the user-specified source term.

18.6 Species Source Modeling

CONVERGE solves for species sources when `inputs.in > solver_control > species_solver = 1` and `source.in > source > equation` is set to the name of the species (e.g., H2O2). The species transport equation is given by

$$\frac{\partial \rho_m}{\partial t} + \frac{\partial \rho_m u_j}{\partial x_j} = \frac{\partial}{\partial x_j} \left(\rho D \frac{\partial Y_m}{\partial x_j} \right) + S_m, \quad (18.32)$$

where

$$\rho_m = Y_m \rho, \quad (18.33)$$

Chapter 18: Source Modeling

Species Source Modeling

and where u is velocity, ρ is density, ρ_m is the density of the species m , Y_m is the mass fraction of species m , D is the mass diffusion coefficient, and S_m is the *species source* term of species m .

When you specify a species source, CONVERGE automatically calculates additional source terms for the energy and momentum equations. These additional sources account for the energy and momentum of the species source and are calculated based on [*source.in > source > species_control > temp*](#) and [*source > species_control > velocity*](#).

18.7 Passive Source Modeling

CONVERGE solves for passive sources when [*passives*](#) are specified in [*species.in*](#) and when [*source.in > source > equation*](#) is set to the name of the passive. The passive transport equation is given by

$$\frac{\partial \rho\phi}{\partial t} + \frac{\partial \rho u_i \phi}{\partial x_i} = \frac{\partial}{\partial x_i} \left(\rho D \frac{\partial \phi}{\partial x_i} \right) + S, \quad (18.34)$$

where u is velocity, ρ is density, D is the diffusion coefficient, ϕ is the passive, and S is the passive source term.

18.8 Porous Media Source Modeling

CONVERGE solves for porous media when [*source.in > source > equation = POROUS*](#). In porous media, the flow occurs through a region of fine-scale geometrical structures which are too small to be numerically resolved within the overall simulation. Porous media modeling simulates these effects by converting them to distributed momentum resistances. Porous media modeling is performed with the assumption that, within the volume containing the distributed resistance, a local balance between pressure and resistance forces exists everywhere, such that

$$-K_i u_i = \frac{\partial p}{\partial \xi_i} \text{ (no summation on } i\text{)}, \quad (18.35)$$

where ξ_i ($i=1,2,3$) represents the mutually orthogonal orthotropic directions, K_i is the resistivity, and u_i is the superficial velocity of a fluid in direction ξ_i . The superficial velocity at any cross section through the porous medium is defined as the volume flow rate divided by the total cross-sectional area (*i.e.*, the area occupied by both fluid and solid). The resistivity is assumed to be a quasi-linear function of the superficial velocity magnitude $|u|$ of the form

Chapter 18: Source Modeling

Porous Media Source Modeling

$$K_i = \alpha_i |u| + \beta_i, \quad (18.36)$$

where α_i (kg/m^4) and β_i ($\text{kg}/\text{m}^3\cdot\text{s}$) are user-supplied coefficients representing the inertial resistivity and the viscous resistivity, respectively.

The inertial resistivity is defined as

$$\alpha = \frac{1}{2} \rho C_{inertial}, \quad (18.37)$$

where ρ is the density of the fluid and $C_{inertial}$ is the inertial resistivity factor (m^{-1}). The viscous resistivity is given by

$$\beta = \mu C_{viscous}, \quad (18.38)$$

where μ is the viscosity and $C_{viscous}$ is the viscous resistivity factor (m^{-2}). Depending on the nature of the resistances, resistivity factors can be isotropic (*i.e.*, uniform in all three directions) or orthotropic (*i.e.*, unique in each direction).

For a unidirectional flow with velocity u , Equations 18.35 and 18.36 can be combined and discretized as

$$dp = au^2 + bu, \quad (18.39)$$

where the pressure drop dp occurs over a distance L and a and b are given by

$$a = \alpha L \text{ and } b = \beta L. \quad (18.40)$$

Given a few sample data points (dp, u) and the material thickness L , you can calibrate α and β based on a curve fit to Equation 18.39.

Chapter 18: Source Modeling

Porous Media Source Modeling

You can either set α and β directly or set the inertial resistivity factor ($C_{inertial}$) and the viscous resistivity factor ($C_{viscous}$), which then will be used during the CONVERGE simulation to calculate α , β , and the resistivity (K_i) based on the actual flow conditions. The benefit of the second method is that $C_{inertial}$ and $C_{viscous}$ do not include density and viscosity and therefore do not change with temperature. When modeling the same porous media under different flow conditions, then, $C_{inertial}$ and $C_{viscous}$ do not need to be re-calibrated for each simulation condition.

CONVERGE Studio provides tools to calibrate either α and β or $C_{inertial}$ and $C_{viscous}$. For both methods, you must supply the data points (dp , u) and the thickness L . To calibrate $C_{inertial}$ and $C_{viscous}$, you also must supply the density and viscosity. Since $C_{inertial}$ and $C_{viscous}$ are dependent on the material, their values also can be obtained from an experimental database.

CONVERGE includes two options for determining the thermal conductivity of the porous region. If you set [source.in > source > porosity_data > eff_conductivity_flag = 0](#), CONVERGE will assume the thermal conductivity of the porous region is equal to the thermal conductivity of the fluid in the porous region. If you set [eff_conductivity_flag = 1](#), CONVERGE will calculate the effective thermal conductivity of the porous region as a function of the fluid and solid thermal conductivities ([Mazumder and Sengupta, 2002](#)), as follows:

$$k_{eff} = -2k_s + \left(\frac{\varepsilon}{2k_s + k_f} + \frac{1-\varepsilon}{3k_s} \right)^{-1}, \quad (18.41)$$

in which ε and k_s are user-supplied parameters ([source.in > source > porosity_data > conductive_porosity](#) and [source > porosity_data > solid_conductivity](#)) and k_f is the thermal conductivity of the fluid in the porous region.

To improve the prediction of temperatures in the porous region, you can optionally use either a local thermal equilibrium (LTE) model or a local thermal non-equilibrium (LTNE) model of energy transport in the porous medium. The LTE model is appropriate when the local temperature of the solid particles in the porous region (T_s) can be assumed to be equal to the local fluid temperature T_f . In this scenario, the transient term of the [energy transport equation](#) for the fluid becomes

$$\frac{\partial [\varepsilon \rho_f E_f + (1-\varepsilon) \rho_s E_s]}{\partial t}, \quad (18.42)$$

Chapter 18: Source Modeling

Porous Media Source Modeling

where ρ_f is the fluid density, E_f is the specific internal energy of the fluid, ε is the porosity of the porous medium, ρ_s is the solid density, and E_s is the specific internal energy of the solid. This term incorporates the effects of the thermal inertia of the solid. Because the fluid and solid are in local thermal equilibrium, the solid energy transient term can be converted into

$$\frac{\partial E_s}{\partial t} = \frac{c_p^s}{c_v^f} \frac{\partial E_f}{\partial t}, \quad (18.43)$$

where c_v^f is the specific heat of the fluid (solved internally) and c_p^s is the specific heat of the solid. To enable the LTE model, set `source > porosity_data > solid_transient_flag = LTE` in `source.in` and specify values for ε (`porosity_data > conductive_porosity`), ρ_s (`porosity_data > solid_density`), and c_p^s (`porosity_data > solid_specific_heat`).

In the LTNE model, the local temperatures of the fluid and solid can be different, and the energy transport equations for the fluid and solid are solved separately. These equations are coupled via source terms that represent the fluid-solid heat transfer. In the fluid energy equation, the source term is equal to

$$h_{sf} a (T_s - T_f), \quad (18.44)$$

where h_{sf} is the fluid-solid heat transfer coefficient and a is the specific surface area of the porous medium, given by the total area of the fluid-solid interface divided by the volume of the porous region. The source term for the solid energy equation has the same form as Equation 18.44 with the opposite sign.

To enable the LTNE model, set `source.in > source > porosity_data > solid_transient_flag = LTNE`. Specify values for ε (`source.in > source > porosity_data > conductive_porosity`), ρ_s (`source.in > source > porosity_data > solid_density`), c_p^s (`source.in > source > porosity_data > solid_specific_heat`), the solid thermal conductivity k_s (`source.in > source > porosity_data > solid_conductivity`), and the initial solid temperature (`source.in > source > porosity_data > solid_init_temp`), and configure your preferred method for calculating the effective heat transfer coefficient (`source.in > source > porosity_data > solid_htc_eff`). You can set $(h_{sf}a)$ to a constant value or calculate h_{sf} and a from one of several models, as detailed below in the table. If you use one of these models, you must specify the characteristic length scale d_p , which corresponds to the diameter of the solid particles in the porous medium.

If you use the effective conductivity from Equation 18.41 in conjunction with the LTNE model, CONVERGE uses k_{eff} for both the fluid and the solid.

Chapter 18: Source Modeling

Porous Media Source Modeling

Table 18.1: Models for calculating the effective heat transfer coefficient for LTNE heat transfer in porous media.

Model	Expressions for h_{sf} and a
Hwang et al. (1995)	$h_{sf} = 0.004 \left(\frac{d_{Nu}}{d_p} \right) \left(\frac{k_f}{d_p} \right) \text{Pr}^{0.33} \text{Re}^{1.35} \quad (\text{Re} < 75, \text{ where } d_{Nu} = 4\epsilon/a),$ $h_{sf} = 1.064 \left(\frac{k_f}{d_p} \right) \text{Pr}^{0.33} \text{Re}^{0.59} \quad (\text{Re} > 350),$ $a = \frac{20.346(1-\epsilon)\epsilon^2}{d_p}$
	For $75 < \text{Re} < 350$, CONVERGE calculates h_{sf} by interpolating between the values at $\text{Re} = 75$ and $\text{Re} = 350$.
Wakao et al. (1979)	$h_{sf} = \frac{k_f (2 + 1.1 \text{Pr}^{1/3} \text{Re}^{0.6})}{d_p},$ $a = \frac{6(1-\epsilon)}{d_p}$
Dixon and Cresswell (1979)	$h_{sf} = \left(\frac{d_p \epsilon}{0.2555 \text{Pr}^{1/3} \text{Re}^{2/3} k_f} + \frac{d_p}{10k_s} \right)^{-1},$ $a = \frac{6(1-\epsilon)}{d_p}$
Kuwahara et al. (2001)	$h_{sf} = \frac{k_f}{d_p} \left[\left(1 + \frac{4(1-\epsilon)}{\epsilon} \right) + \frac{1}{2} (1-\epsilon)^{1/2} \text{Re}^{0.6} \text{Pr}^{1/3} \right],$ $a = \frac{6(1-\epsilon)}{d_p}$

Particulate Filter Modeling

In exhaust aftertreatment simulations, you can model a diesel particulate filter or a gasoline particulate filter as a special type of porous media source. CONVERGE uses the particulate filter model of [Konstandopoulos et al., 2000](#) to account for changes in porosity and filtration efficiency that occur as soot particulates flow through the filter and form deposits on the porous substrate. This model also accounts for the differing effects on filtration efficiency from particulates of different sizes.

To identify a porous media source as a particulate filter, set [`source.in > source > porosity_data > particulate_filter > active = ON`](#) and include a [`particulate_filter.in`](#) file in your case setup. For

Chapter 18: Source Modeling

Porous Media Source Modeling

this type of porous medium, CONVERGE ignores the resistivity parameters in [source.in](#) and instead computes the resistivity as described below.

Because the fluid moves through the filter at low velocity, the inertial resistivity in Equation 18.36 is negligible and the resistivity can be modeled as

$$K_i = \beta_i = \mu C_{viscous}. \quad (18.45)$$

The viscous resistivity factor $C_{viscous}$ is related to the permeability of the filter by

$$C_{viscous} = \frac{1}{\kappa}. \quad (18.46)$$

The permeability κ decreases over time as soot deposits form on the filter wall. CONVERGE models this process with a transport equation for the soot concentration ϕ , given by

$$\frac{\partial(\rho\phi)}{\partial t} + (1-E) \frac{\partial(\rho u_i \phi)}{\partial x_i} = 0. \quad (18.47)$$

Equation 18.47 shows that the soot concentration is not conserved in the presence of a nonzero filtration efficiency E . CONVERGE calculates the efficiency from

$$E = 1 - e^{-\alpha}, \quad (18.48)$$

where

$$\alpha = \frac{4(1-\varepsilon)w}{\pi\varepsilon d_c} (E_D + E_R + E_I). \quad (18.49)$$

In Equation 18.49, ε is the local cell porosity, d_c is the local unit collector diameter, and w is the cell thickness. The terms E_D , E_R , and E_I represent the efficiency of filtration by diffusion, interception, and inertia, respectively. CONVERGE uses the method of [Opris and Johnson](#),

Chapter 18: Source Modeling

Porous Media Source Modeling

[1998](#) to calculate these quantities based on the soot particulate density, soot particulate size distribution, and filter properties specified in [*particulate_filter.in*](#).

CONVERGE updates ε and d_c at each time-step based on the deposited soot mass m_s , obtaining the latter by integrating the following equation:

$$\frac{\partial m_s}{\partial t} + E \frac{\partial (\rho u_i \phi)}{\partial x_i} = 0. \quad (18.50)$$

The local unit collector diameter at time t is then computed from

$$d_c(t) = 2 \left[\frac{3}{4\pi} \frac{m_s}{\rho_s} + \left(\frac{d_c(t-1)}{2} \right)^3 \right]^{1/3} \quad (18.51)$$

and the local porosity is computed from

$$\varepsilon(t) = 1 - \left(\frac{d_c(t)}{d_{c0}} \right)^3 (1 - \varepsilon_0). \quad (18.52)$$

These quantities determine the local permeability at time t according to

$$\kappa(t) = \kappa_0 \left[\frac{d_c(t)}{d_{c0}} \right]^2 \frac{f[\varepsilon(t)]}{f(\varepsilon_0)}, \quad (18.53)$$

where

$$f(\varepsilon) = \frac{\frac{2}{9} \left[2 - \frac{9}{5}(1-\varepsilon)^{1/3} - \varepsilon - \frac{1}{5}(1-\varepsilon)^2 \right]}{1-\varepsilon}. \quad (18.54)$$

The initial values of the unit collector diameter (d_{c0}), porosity (ε_0), and permeability (κ_0) are specified in [*particulate_filter.in*](#).

CONVERGE uses Equation 18.53 to update the porous resistivity (K_i) in the momentum source term at the frequency specified in [*particulate_filter.in*](#) > *flow_control* > *stream* >

Chapter 18: Source Modeling

Porous Media Source Modeling

mom_source_update_interval. After the resistivity has been updated, CONVERGE runs with a small fixed time-step for the period specified in [particulate_filter.in > flow_control > stream > coupling_period](#), which allows the flow to react to the change in resistivity due to soot loading. You can specify the time-step for the coupling period in [particulate_filter.in > flow_control > stream > coupling_dt_max](#).

18.9 Baffle Media Source Modeling

CONVERGE solves for baffle media when [source.in > source > equation = BAFFLE](#). Baffle media modeling is similar to porous media modeling, but is designed to model the pressure drop across a planar surface. The pressure drop is calculated according to

$$\Delta p = \rho (\alpha |v_n| + \beta) v_n, \quad (18.55)$$

where ρ is the fluid density, v_n is the velocity normal to the boundary, and $|v_n|$ is the magnitude of this velocity. The baffle model parameter α is dimensionless. The baffle model parameter β has units of m/s . CONVERGE uses a temporal type of *PERMANENT* for all baffle sources, regardless of the value specified in [source.in > source > temporal_control > type](#). Note that you must activate the [Rhe-Chow algorithm](#) in order to specify a baffle source.

18.10 Battery Heat Source Modeling

You can specify a battery heat source for a simulation according to the model proposed by [Lin et al. \(2014\)](#) by setting [source.in > source > equation = BATTERY](#). In this model, a battery is represented as an electrical network (see figure below) and the heat source is calculated based on the electrical current, voltages across resistor-capacitor (R-C) pairs, and series resistance (R_s). For this model, a positive current, I , represents discharging the battery and a negative current represents charging the battery.

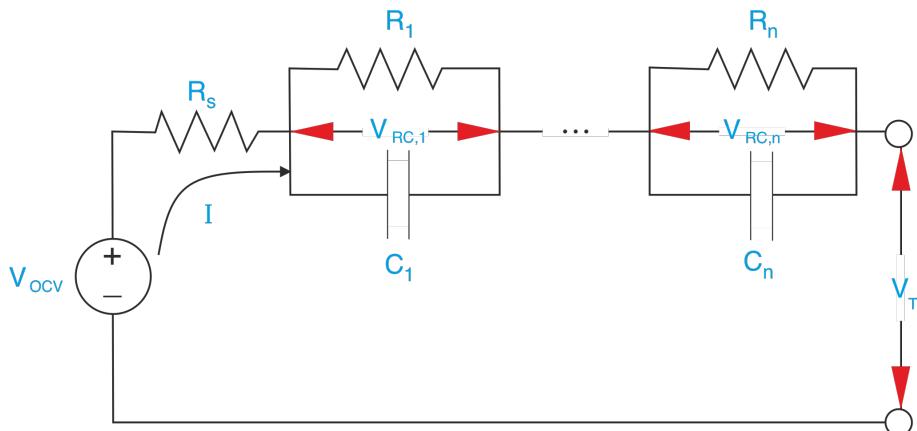


Figure 18.4: Battery represented as an electrical network.

Chapter 18: Source Modeling

Battery Heat Source Modeling

The terminal voltage for a circuit, V_T is given as

$$V_T = V_{OCV} - IR_S - \sum_{i=1}^n V_{RC,i}, \quad (18.56)$$

where V_{OCV} represents the open circuit voltage specified in [*source.in > source > battery_data > soc_vocv*](#), R_S is a series resistance and is supplied in a [*battery properties file*](#) (e.g., *battery_properties.dat*), i is the R-C pair index, n is the total number of R-C pairs ([*source.in > source > battery_data > n_rc_pairs*](#)), and $V_{RC,i}$ is the voltage drop across a series of parallel R-C circuits.

The state of charge (SOC) is calculated by

$$\frac{dSOC}{dt} = -\frac{1}{C_{battery}} I, \quad (18.57)$$

where $C_{battery}$ is the battery capacity (Ampere-hour) as specified in [*source.in > source > battery_data > battery_capacity*](#).

The dynamics of a single R-C pair are given as

$$\frac{dV_{RC,i}}{dt} = -\frac{1}{R_i C_i} V_{RC,i} + \frac{1}{C_i} I, \quad (18.58)$$

where R_i and C_i are the equivalent resistance and capacitance and are provided in the battery properties file. You can include 0, 1, 2, or 3 R-C pairs in the model via [*source.in > source > battery_data > n_rc_pairs*](#). While including more R-C pairs improves the model, it does so with increased computational cost and complexity.

R_s , R_v and C_i values typically vary with temperature, SOC, electrical current direction (charge/discharge). R_s , R_v and C_i values are looked up in the battery properties file for the given SOC value and mean temperature of the domain, and interpolated if necessary. If SOC or mean temperature in the simulation goes beyond the data that is given in the battery properties file, then R_s , R_v and C_i values for the closest available SOC or mean temperature is used.

The total heat generated, Q , is thus given as

$$Q = I(V_{OCV} - V_T), \quad (18.59)$$

which is reflected in the [thermodynamic output file](#).

18.11 Relaxation Zone Source Modeling

You can include relaxation zone sources in simulations with an active volume of fluid (VOF) model ([inputs.in](#) > *feature_control* > *vof_flag* = 1). Relaxation zones improve the numerical stability of simulations that involve surface wave generation and/or absorption ([Jacobsen et al., 2012](#)). This type of source is available for 2-species VOF simulations in which both species are incompressible.

Within a relaxation zone, CONVERGE computes the relevant fields as a weighted average of the CFD solution ϕ^{CFD} and a known theoretical solution ϕ^{wave} . For simulations that employ the [void fraction solution](#) (VFS) method, the relevant fields are velocity and void fraction. For simulations that employ the [individual species solution](#) (ISS) method, the relevant fields are velocity and species density.

As an example, consider the numerical wave tank in Figure 18.5. Surface waves are generated by spatially and temporally varying boundary conditions for the velocity and void fraction at the INFLOW boundary. The waves are absorbed at an OUTFLOW boundary with Neumann boundary conditions for pressure and velocity. In the relaxation zones near the INFLOW and OUTFLOW boundaries, CONVERGE blends ϕ^{CFD} with ϕ^{wave} based on a blending function w , which increases monotonically from $w = 0$ at the boundary to $w = 1$ at the end of the relaxation zone. Relaxing the CFD solution to the theoretical solution at the boundary gradually removes wave reflections that might otherwise cause the solution to become unstable.

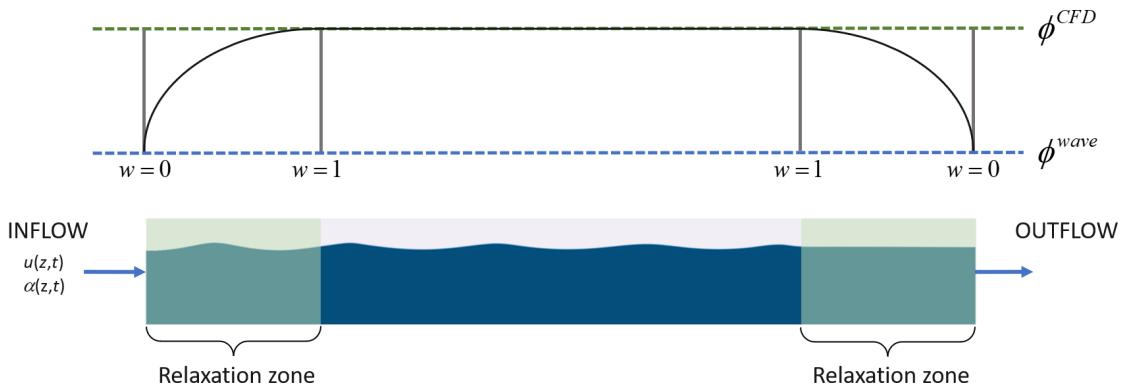


Figure 18.5: Relaxation zone setup for a numerical wave tank. Waves are generated at the INFLOW boundary and absorbed at the OUTFLOW boundary. In the relaxation zones, CONVERGE blends the CFD solution with the theoretical solution at the boundary.

The contribution from ϕ^{wave} is calculated differently depending on whether the VOF model uses the ISS method or the VFS method. Given the discretized transport equation for the variable ϕ , which has the general form

Chapter 18: Source Modeling

Relaxation Zone Source Modeling

$$A_{ij}\phi_j^{n+1} = b, \quad (18.60)$$

the CFD solution satisfies

$$A_{ij}\phi_j^{CFD} = b. \quad (18.61)$$

For simulations that employ the ISS method, the solution in the relaxation zone for iteration $n+1$ is simply given by

$$\phi_j^{n+1} = w\phi_j^{CFD} + (1-w)\phi_j^{wave}. \quad (18.62)$$

For simulations that employ the VFS method, we multiply Equation 18.62 by the matrix A to obtain

$$\begin{aligned} A_{ij}\phi_j^{n+1} &= wb + (1-w)A_{ij}\phi_j^{wave} \\ &= b + (1-w)A_{ij}(\phi_j^{wave} - \phi_j^{CFD}). \end{aligned} \quad (18.63)$$

Defining D as the diagonal part of A , given by

$$D_{ij} = \left(\frac{\rho}{dt} \right) \delta_{ij}, \quad (18.64)$$

Equation 18.63 can be approximated as

$$A_{ij}\phi_j^{n+1} \approx b + (1-w)D_{ij}(\phi_j^{wave} - \phi_j^n). \quad (18.63)$$

In this approach, CONVERGE solves for ϕ_j^{n+1} directly from Equation 18.63, without solving for ϕ_j^{CFD} .

For both methods, the blending function w is given by

$$w = 1 - \frac{\exp(s^p) - 1}{\exp(1) - 1}, \quad (18.65)$$

where s is the local coordinate in the relaxation zone and p is the blending function exponent.

Chapter 18: Source Modeling

Relaxation Zone Source Modeling

To add a relaxation zone source, set `source.in > source > equation = RELAXATION_ZONE`. This type of source can have either a box shape (`source.in > source > shape > type = BOX`) or a boundary shape (`source.in > source > shape > type = BOUNDARY`). For a boundary shape, specify the distance by which the relaxation zone extends from the boundary (`source.in > source > shape > distance_from_boundary`).

In `source.in > source > relaxation_zone_data`, specify parameters for the blending function w (`relaxation_zone_data > blending_function`) and the theoretical solution ϕ^{wave} (`relaxation_zone_data > zone_field_data`). If the theoretical solution consists of a constant velocity and water level, as shown in the example OUTFLOW boundary in Figure 18.5, you can specify those values directly in `relaxation_zone_data > zone_field_data > velocity` and `relaxation_zone_data > zone_field_data > water_level`. Alternatively, you can set the theoretical solution equal to the fields specified in `initialize.in` or define the theoretical solution in a user-defined function (UDF).

Chapter



19

Conjugate Heat Transfer Modeling

19 Conjugate Heat Transfer Modeling

Conjugate heat transfer (CHT) is when heat transfer occurs simultaneously within and between fluid and solid regions. The predominant mode of heat transfer in fluids is convection, while the predominant mode of heat transfer in solids is conduction. CHT models are available in CONVERGE to solve the heat transfer in both the fluid and solid regions. CHT coupling is also available for [CONVERGE + GT-SUITE co-simulations](#).

The [1D CHT](#) model offers a simplified approach for predicting the temperature in thin solid layers near the fluid-solid interface. This model is suitable for solids with a small thermal penetration depth (*i.e.*, high thermal resistance).

Some solids require a [3D CHT](#) model. Ideally, 3D CHT simulations will involve simultaneous heat transfer calculations for the solid and fluid phases. This approach is practical for some types of simulations but impractical for others. For example, the time-scale for the solid phase heat transfer is orders of magnitude larger than the time-scale for the fluid phase heat transfer. Thus it is not always practical to run the numerical simulation entirely in a coupled transient manner. CONVERGE offers a novel solution: [super-cycling](#) to solve 3D CHT problems in a computationally efficient manner.

In a CONVERGE + GT-SUITE co-simulation, you can use [CHT coupling](#) to predict the surface temperature field on a WALL boundary. CONVERGE simulates the fluid and GT-SUITE uses a 3D finite element solver to simulate the solid. The two solvers exchange surface data at the WALL boundary.

19.1 1D CHT

CONVERGE includes a 1D conjugate heat transfer (CHT) model for heat transfer in thin solid layers between a fluid and a solid. When the thermal penetration depth is small, a 1D model can accurately resolve temperatures in the solid in the direction normal to the wall. This model is available for WALL boundaries with Dirichlet or law-of-the-wall boundary conditions.

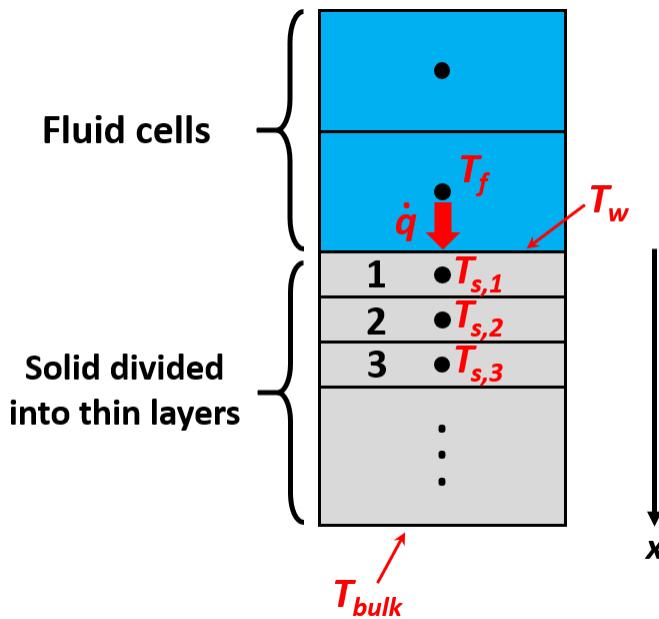


Figure 19.1: 1D CHT model.

Figure 19.1 above shows a schematic of the 1D CHT model. At each time-step, CONVERGE solves the energy transport equation for the fluid based on the wall temperature (T_w) from the previous time-step. CONVERGE then calculates the heat flux for near-wall fluid cells from

$$\dot{q} = k \frac{\partial T}{\partial x_i} = k \frac{(T - T_w)}{(x - x_w)} n_i \quad (19.1)$$

for Dirichlet boundary conditions, or from the appropriate [heat transfer model](#) for law-of-the-wall boundary conditions. CONVERGE also calculates a heat transfer coefficient (HTC) based on the heat flux, given by

$$\text{HTC} = \text{htc_scale_factor} \cdot \left(\frac{\dot{q}}{T_f - T_w} \right), \quad (19.2)$$

where T_f is the temperature of the fluid cell and `htc_scale_factor` is user-specified (`cht1d.in > boundaries > boundary > htc_scale_factor`).

Next, CONVERGE solves the 1D transient heat transfer equation for the solid, given by

$$c\rho \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right), \quad (19.3)$$

where ρ is the density of the solid, c is the specific heat, and k is the thermal conductivity. The values of the HTC and T_f are used to define a [convection boundary condition](#) at the solid-fluid interface. The boundary condition on the solid side is a user-specified solid bulk temperature T_{bulk} . CONVERGE divides the wall into thin solid layers, as shown in Figure 19.1, and uses Equation 19.2 to solve for the temperature in each layer (and to update T_w). Because the problem is one-dimensional, the system of equations is tri-diagonal and can be solved efficiently with the Thomas algorithm.

To enable 1D CHT, set `boundary.in > boundary_conditions > boundary > temperature > value = CHT1D` and include a `cht1d.in` file in your Case Directory. Specify T_{bulk} , the number of solid layers, htc_scale_factor , and other model parameters in `cht1d.in`.

When a CHT model (either 1D or 3D) is enabled, you also must specify `boundary.in > boundary_conditions > boundary > cht_resolution_level` for each WALL boundary, including those on the forward and reverse sides of an INTERFACE boundary. This parameter n controls the resolution of the CHT interpolation points at which CONVERGE calculates the energy coupling. If set to *AUTO*, the distance between CHT interpolation points is given by $base_dx/2^n$, where

$$n = \min(\text{boundary embedding level} + 2, n_{max}) \quad (19.4)$$

and n_{max} is the embedding level for the smallest cells in the domain. If `boundary.in > boundary_conditions > boundary > cht_resolution_level` is set to an integer value, then

$$n = \min(cht_resolution_level, n_{max}) \quad (19.5)$$

For example, if `boundary.in > boundary_conditions > boundary > cht_resolution_level = AUTO` and the boundary embedding level is equal to n_{max} , so that the cells on the boundary are the size of the smallest cells in the domain, then there will be one CHT interpolation point per boundary fluid node, as shown above in Figure 19.1.

19.2 3D CHT

CONVERGE calculates 3D conjugate heat transfer at the interface between a solid region and a fluid region. To complete the setup for 3D CHT, you must define the solid and fluid materials and their physical properties, assign the appropriate regions to solid and fluid streams, and set up the INTERFACE boundary at which the CHT coupling occurs. The following sections describe the required setup for materials, streams, and boundaries.

CONVERGE offers several time-control methods for 3D CHT. If you choose transient or steady-state, CONVERGE solves the fluid and solid together with the same time-step for the entire domain. If you choose [super-cycling](#) (recommended for simulations with combustion),

Chapter 19: Conjugate Heat Transfer Modeling

3D CHT

CONVERGE iterates between fully-coupled transient and steady-state solution methods for the solid only and uses the transient solver for the fluid.

Solid Properties

For a CHT simulation, you must define one or more solid species in [*species.in*](#). Figure 19.2 below gives an example of [*species.in*](#).

```
version: 3.1
---
liquid:
- icl14h30
solid:
- iron
```

Figure 19.2: Excerpt of [*species.in*](#) with a solid species.

CONVERGE looks in [*solid.dat*](#) for properties of any species listed in [*species.in*](#) > *solid*. Figure 19.3 shows an excerpt of a [*solid.dat*](#) file for a CHT case that includes iron.

```
!   solid species name
!   melting point (k)
!   Temperature density      specific heat    conductivity
!   k          (N.s/m^2)     (j/kg.k)        (w/m.k)
iron
690.0
0.0000E+00  7.85E+003  5.61E+002  4.27E+001
1.0000E+01  7.85E+003  5.61E+002  4.27E+001
2.0000E+01  7.85E+003  5.61E+002  4.27E+001
.
.
```

Figure 19.3: An example [*solid.dat*](#) file.

Fluid materials and their properties are specified in the [reaction mechanism](#), [thermodynamic data](#), [*species.in*](#), [*gas.dat*](#), and [*liquid.dat*](#) files.

Fluid and Solid Streams

In [*stream_settings.in*](#), all fluid regions must be assigned to a fluid stream (*stream_list* > *stream* > *solid_flag* = 0) and all solid regions must be assigned to a solid stream (*solid_flag* = 1). Figure 19.4 shows an example [*stream_settings.in*](#) file that contains one fluid stream and one solid stream.

```
version: 3.1
---

stream_match_species_settings:      0
post_file_option:                  COMBINED
stream_list:
  - stream:
    id:                            0
    name:                          fluid-stream
    solid_flag:                   0
    stream_overwrite:             ""
    region_list:
      - region:
        id:                            0
        name:                          swirler-combustor
      - region:
        id:                            1
        name:                          casing
  - stream:
    id:                            1
    name:                          solid-stream
    solid_flag:                   1
    stream_overwrite:             ""
    region_list:
      - region:
        id:                            2
        name:                          solid-wall
```

Figure 19.4: An example [stream_settings.in](#) file for a conjugate heat transfer case.

Boundary Conditions

This section describes how to set up an [INTERFACE](#) boundary and a [WALL](#) boundary for a CHT simulation.

INTERFACE Boundary

An [INTERFACE boundary](#) consists of a single layer of triangles but has two unique sets of boundary conditions: one for each side of the INTERFACE. For conjugate heat transfer applications, an INTERFACE boundary allows you to calculate the heat transfer between a fluid region and a solid region.

Figure 19.5 below shows an example geometry, and Figure 19.6 shows an excerpt of a [boundary.in](#) file for that geometry. Boundary 2 is the INTERFACE boundary, boundary 3 is the forward boundary (the fluid side), and boundary 4 is the reverse boundary (the solid side). To allow heat transfer through the INTERFACE, CONVERGE couples the forward and reverse boundaries (available only for fluid-solid or solid-solid INTERFACE boundaries). This coupling takes place via the temperature boundary condition for the forward and reverse boundaries ([boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [temperature](#) > [value = COUPLED](#)). The resolution of the [CHT interpolation points](#) at which CONVERGE calculates the energy coupling is independent of the grid and is specified in [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [cht_resolution_level](#).

Chapter 19: Conjugate Heat Transfer Modeling

3D CHT Boundary Conditions

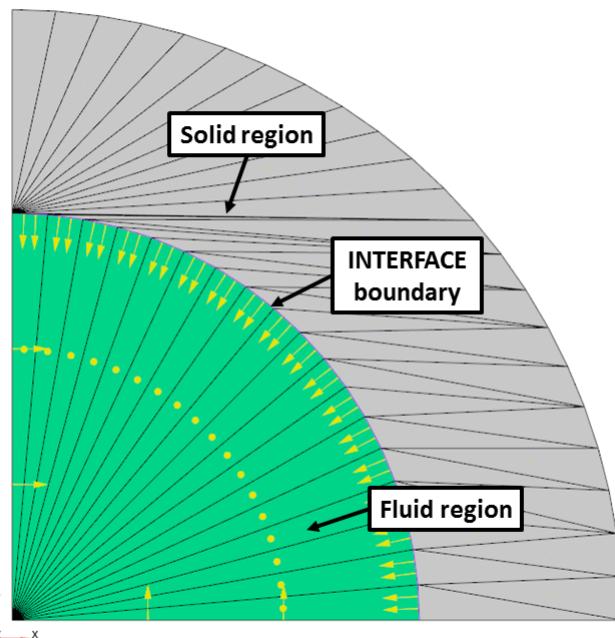


Figure 19.5: Normal vectors pointing towards the fluid region (region 0).

```
version: 3.1
---

boundary_conditions:
  - boundary:
      id: 1
      type: WALL
      name: Outer wall
      region: 1
      motion: STATIONARY
      geometry_motion: FIXED
      roughness:
        height: 0
        constant: 0
      velocity:
        type: Law_of_Wall
        value: [0, 0, 0]
      pressure:
        type: Neumann
      temperature:
        type: Flux
        value: -10000
      species:
        type: Neumann
      passive:
        type: Neumann
      turbulence:
        tke:
          type: Neumann
          value: 0
        eps:
          type: Dirichlet
          value: AUTO
  - boundary:
      id: 2
```

Chapter 19: Conjugate Heat Transfer Modeling

3D CHT Boundary Conditions

```
type: INTERFACE
name: Solid fluid interface
disconnect: 0
forward:
  id: 3
  type: WALL
  name: Fwd side of INTERFACE Solid fluid interface (ID = 2)
  region: 0
  motion: STATIONARY
  geometry_motion: FIXED
  roughness:
    height: 0
    constant: 0
  velocity:
    type: Law_of_wall
    value: [0, 0, 0]
  pressure:
    type: Neumann
  temperature:
    type: Law_of_wall
    value: COUPLED
    heat_model: 0
    contact_resistence: 0
  species:
    type: Neumann
  passive:
    type: Neumann
  turbulence:
    tke:
      type: Neumann
      value: 0
    eps:
      type: Dirichlet
  cht_resolution_level: AUTO
reverse:
  id: 4
  type: WALL
  name: Rev side of INTERFACE Solid fluid interface (ID = 2)
  region: 1
  motion: STATIONARY
  geometry_motion: FIXED
  roughness:
    height: 0
    constant: 0
  velocity:
    type: Dirichlet
    value: [0, 0, 0]
  pressure:
    type: Neumann
  temperature:
    type: Dirichlet
    value: COUPLED
    contact_resistence: 0
  species:
    type: Neumann
  passive:
    type: Neumann
  turbulence:
    tke:
      type: Neumann
      value: 0
    eps:
      type: Dirichlet
  cht_resolution_level: AUTO
```

Figure 19.6: An excerpt of [boundary.in](#) with heat flux into the outer wall, the INTERFACE boundary, and the accompanying virtual boundaries.

CONVERGE imposes thermal continuity across the INTERFACE via

Chapter 19: Conjugate Heat Transfer Modeling

3D CHT Boundary Conditions

$$[T_b]_F = [T_b]_S \quad (19.6)$$

and

$$[\text{Heat flux}]_F = [\text{Heat flux}]_S, \quad (19.7)$$

where the subscripts F and S denote fluid and solid, respectively, and T is the temperature. The temperatures and heat fluxes of the solid and the fluid regions are consistent across the entire area of the INTERFACE boundary.

When CONVERGE generates the grid, it treats solid-solid and solid-fluid interfaces as if they were in perfect contact. In reality, at the microscopic level, there are irregularities and defects in the solid surfaces that prevent perfect thermal contact. As a result, there is a small air gap between the two solids or the solid and fluid that restricts heat flux. In order to account for the small gap between solids, CONVERGE includes a thermal contact resistance model. The equation for the heat flux balance at the interface between two solid boundaries is a function of the contact resistance:

$$q_1(T_1, T_{1i}) = q_2(T_2, T_{2i}) = \frac{T_{2i} - T_{1i}}{R_c''}, \quad (19.8)$$

where q_j is the heat flux (in W/m^2) at the interface for solid j , T_j is the temperature at the first interior point near the interface for solid j , T_{ji} is the interface temperature for solid j , and R_c'' is the contact resistance for a unit area of the interface in $\text{K}\cdot\text{m}^2/\text{W}$. To enable the contact resistance model between coupled *forward* and *reverse* boundaries, set [boundary.in > boundary_conditions > boundary > temperature > value = COUPLED](#) and specify a value for [boundary.in > boundary_conditions > boundary > temperature > contact_resistance](#).

If [boundary.in > boundary_conditions > boundary > disconnect = 0](#) for an INTERFACE boundary, then the INTERFACE triangles cannot be disabled and no fluid flow is allowed between the two regions. Heat transfer across the INTERFACE boundary is allowed. This setup is typical for a conjugate heat transfer simulation of heat transfer between fluid and a solid piston. When $disconnect = 0$, both the forward and reverse virtual boundaries must be [WALL](#) boundaries.

If [boundary.in > boundary_conditions > boundary > disconnect = 1](#), CONVERGE treats the INTERFACE triangles as [disconnect triangles](#). This configuration works only for an INTERFACE between two fluid regions. CONVERGE will allow fluid flow across the INTERFACE boundary and you can set up OPEN and CLOSE events in [events.in](#) to control the flow. OPEN events will allow both mass and heat to move across the boundary. CLOSE events will prohibit mass and heat flow across the boundary. When $disconnect = 1$, both the forward and reverse virtual boundaries must be [SYMMETRY](#) boundaries.

WALL Boundary

In a CHT simulation, a [WALL boundary](#) contains the portion of the solid that is not in contact with the secondary material. There are multiple options for the WALL temperature boundary conditions. For example, to specify a flux boundary condition, set [boundary.in > boundary_conditions > boundary > temperature = FLUX](#) and [boundary.in > boundary_conditions > boundary > temperature > value](#) to the amount of heat per surface area (in W/m^2) applied uniformly to the boundary (see boundary 1 in Figure 19.5). A negative value indicates energy entering the solid while a positive value indicates energy exiting the solid.

19.3 GT-SUITE CHT Coupling

You can run a CONVERGE + GT-SUITE co-simulation that includes CHT coupling at WALL boundaries. GT-SUITE simulates the solid using a 3D finite element solver. Heat exchange between CONVERGE and GT-SUITE occurs at the 2D WALL surface. At each time-step, GT-SUITE first sends the surface temperature field to CONVERGE. Then, CONVERGE sends updated reference temperature and heat transfer coefficient fields to GT-SUITE.

To use GT-SUITE CHT coupling, complete the following setup:

- Set [inputs.in > feature_control > cosimulation_flag = 1](#).
- Set [cosimulation.in > GT-SUITE > scheme = GT_LEAD](#).
- For the WALL boundaries at which fluid-solid heat transfer occurs, set [boundary.in > boundary_conditions > boundary > temperature > type = DIRICHLET or LAW_OF_WALL](#). Set the corresponding [temperature > value](#) to GT-SUITE.

You can use GT-SUITE CHT coupling with or without GT-SUITE flow coupling. The latter requires at least one [GT-SUITE boundary](#).

Chapter



20

Fluid-Structure Interaction Modeling

20 Fluid-Structure Interaction Modeling

CONVERGE computes external forces, such as aerodynamic and gravitational forces, and moments on an object to model fluid-structure interaction (FSI). The aerodynamic forces are computed by numerical integration of pressure and shear stress over the surface of the object.

CONVERGE offers two types of FSI objects. Rigid FSI objects translate and rotate through the flow according to the imposed forces and moments. Beam FSI objects change their shape in response to imposed fluid forces according to a deformable structure solver.

Using the FSI technique in concert with the cut-cell grid generation method and [Adaptive Mesh Refinement \(AMR\)](#), you can easily simulate spring-loaded ball valves, flexible reed valves, wind turbines, and other complex engineering structures.

20.1 Force and Moment Calculations

The flow solver calculates the force, $F_{fluid-i}$, by integrating the stress tensor over the surface of the FSI object, as

$$F_{fluid-i} = \int_S (-P\delta_{ij} + \sigma_{ij})n_j dS, \quad (20.1)$$

where n_i is the normal vector, P_i is the pressure, and σ_{ij} is the [viscous stress tensor](#). As described below in the [Equations of Motion](#) section, $F_{fluid-i}$ is used to calculate the translational motion of the FSI object.

The total moment ($M_{fluid-i}$) is calculated from the flow solver by integrating the stress tensor over the surface of the FSI object as

$$M_{fluid-i} = \int_S \varepsilon_{ijk} r_j (-P\delta_{kl} + \sigma_{kl})n_l dS, \quad (20.2)$$

where ε_{ijk} is the Levi-Civita symbol and r_j is the distance of the center of mass from the surface boundary. As explained in Equation 20.4 in the [Equations of Motion](#) section, the total moment ($M_{fluid-i}$) is used to calculate the angular motion of the FSI object.

20.2 Equations of Motion: Rigid Body Dynamics

CONVERGE solves the governing equation for the translational motion of the center of mass, which is given by Equation 20.3:

Chapter 20: Fluid-Structure Interaction Modeling

Equations of Motion: Rigid Body Dynamics

$$\sum F_{\text{external-}i} + F_{\text{fluid-}i} = m\ddot{x}_i, \quad (20.3)$$

where m is the mass and \ddot{x}_i is the acceleration of the center of mass in the i -th direction of the FSI object. The $F_{\text{external-}i}$ forces, which are summed, include the applied forces (specified in [fsi.in](#)) and the body forces ([inputs.in](#) > *property_control* > *gravity* or other forces on the FSI object as specified via [user_fsi.c](#)) on the FSI object.

The governing equation for the rotational motion of the center of mass in the inertial coordinate system is given by

$$\sum M_{\text{external-}i} + M_{\text{fluid-}i} = I_{ij}\dot{\omega}_j + \varepsilon_{ijk}\omega_j I_{kl}\omega_l, \quad (20.4)$$

where ω_i is the angular velocity in the i -th direction and I_{ij} is the moment of inertia tensor of the FSI object. The $M_{\text{external-}i}$ moments, which are summed, include the applied moments (specified in [fsi.in](#)) and moments due to other external forces on the FSI object. The rotational position is updated on a quaternion-based modeler where the quaternion (q) is defined by

$$q = \cos \frac{\theta}{2} + i \sin \frac{\theta}{2} + j \sin \frac{\theta}{2} + k \sin \frac{\theta}{2}, \quad (20.5)$$

where θ is the angle and (i, j, k) is the axis vector about which the rotation occurs.

In FSI cases that also incorporate volume of fluid (VOF) modeling with [wall adhesion](#), the FSI equations of motion include this wall adhesion force. These forces are calculated according to

$$F_{\text{adhesion}} = -l_{\text{contact}} \cdot \sigma_{\text{st}} \cdot (e_t \cos \theta_w + n \sin \theta_w), \quad (20.6)$$

where l_{contact} is the contact length in the cell, σ_{st} is the surface tension, e_t is the tangent unit vector, n is the normal unit vector, and θ_w is the contact angle. The contact length is calculated as the length of the intersection line of the PLIC liquid surface plane and the boundary face.

Implicit FSI Coupling Method

An object accelerating or decelerating through a fluid has additional inertial forces acting on it. This concept may be thought of in simplified terms as arising due to additional fluid acceleration/deceleration induced by the object. This extra inertial force is frequently modeled in terms of an added mass on the object. This added mass, m_{added} , scales with the density ratio between the solid and the fluid, *i.e.*,

$$m_{\text{added}} \sim \frac{\rho_f}{\rho_s} m_{\text{obj}}, \quad (20.7)$$

where ρ_f and ρ_s are the density of the fluid and solid, respectively, and m_{obj} is the mass of the object. The added mass effect, therefore, is more prominent when the density of the fluid approaches the density of the solid, or when the density of the fluid exceeds the density of the solid. In these situations, explicit coupling between the fluid and the rigid object may lead to instabilities or trouble with convergence. Thus, the tighter implicit FSI coupling in CONVERGE should be employed to successfully run these simulations.

Without implicit coupling, in a single time-step CONVERGE first uses the CFD solver to determine the force, then uses the FSI solver to determine the position of the object. The implicit method couples the CFD solver with the FSI solver, iterating between the two in a single time-step until convergence (or until the user-specified maximum number of iterations is completed). In the implicit method, the pressure forces are implicitly coupled and the viscous forces are explicitly coupled.

Implicit FSI coupling is implemented only for rigid FSI objects. To use implicit FSI coupling, specify parameters in the [*fsi.in*](#) > *implicit_fsi* settings block.

Aitkens dynamic under-relaxation can be performed on the force calculations to avoid convergence issues for complex cases. This type of under-relaxation uses the convergence history from loop $k-1$ to loop k in a single time-step to determine the under-relaxation factor, ω . The under-relaxation factor is determined by

$$\omega = \min(\omega_k^{\max}, \max(\omega_k^{\min}, \omega_k)) \quad (20.8)$$

where

Chapter 20: Fluid-Structure Interaction Modeling

Equations of Motion: Rigid Body Dynamics

$$\omega_k = \frac{r_k(r_k - r_{k-1})}{|r_k - r_{k-1}|}. \quad (20.9)$$

In Equation 20.9, r_k is the residual for the k -th loop, given as

$$r_k = (a_k - a_{k-1}) \quad (20.10)$$

and r_{k-1} is the residual for the $k-1$ loop. The residual is based on the acceleration of the force (a). You can specify limits for the minimum and maximum relaxation factor via [*fsi.in* > *implicit_fsi* > *relaxation_factor*](#).

20.3 FSI Setup

To implement FSI modeling in a CONVERGE simulation, you must [prepare the surface geometry](#) in CONVERGE Studio and note the boundary IDs of the boundaries that define your FSI object(s). In the figure below, all of the boundaries that constitute part of the bird are boundaries used to define the FSI object. There is an additional boundary (not visible in the figure) that defines a box around the bird, but that boundary is not part of the FSI object.

After preparing the geometry, activate FSI modeling by setting [*inputs.in* > *feature_control* > *fsi_flag* = 1](#). Next you need to prepare [*boundary.in*](#) and [*fsi.in*](#) as described below.

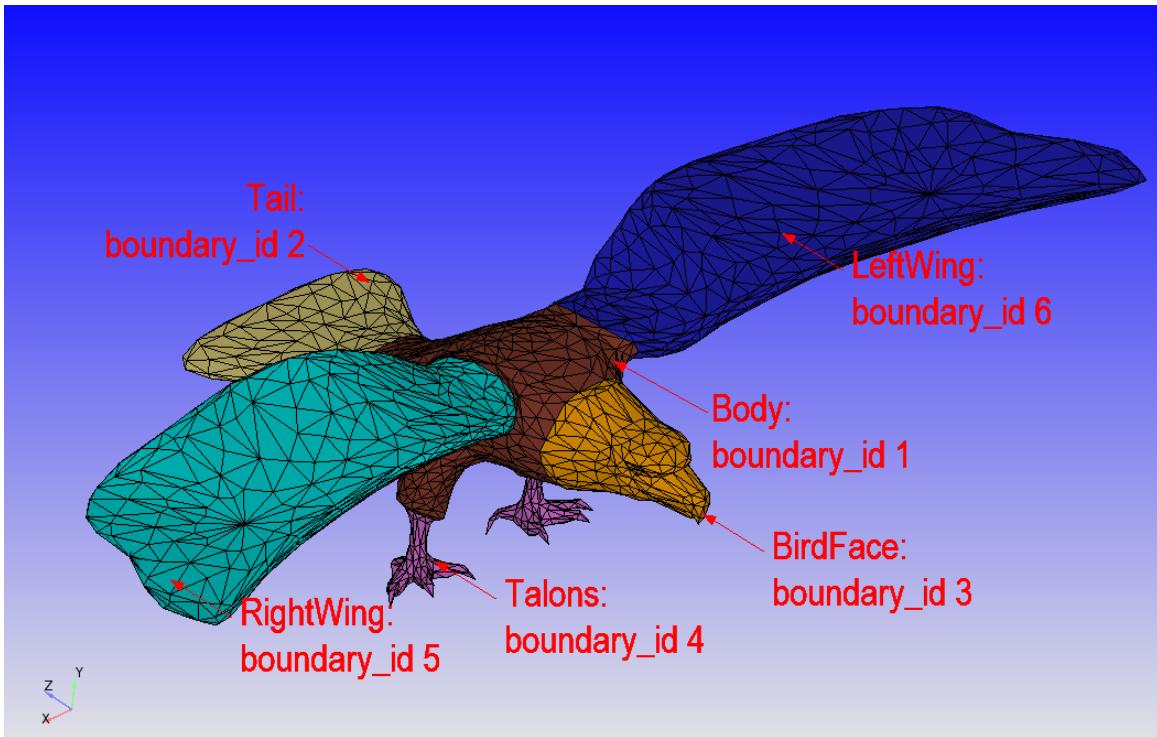


Figure 20.1: All of the example input files in this chapter pertain to this bird.

boundary.in

After preparing the geometry, you need to prepare the [*boundary.in*](#) file for your FSI case.

Figure 20.2 below shows an example [*boundary.in*](#) file for the flying bird shown in Figure 20.1.

Set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *type* = WALL, [*boundary.in*](#) > *boundary_conditions* > *boundary* > *motion* = FSI, and [*boundary.in*](#) > *boundary_conditions* > *boundary* > *geometry_motion* = MOVING for each boundary whose motion will depend solely on the fluid-structure interaction (*i.e.*, solely on the parameters defined in [*fsi.in*](#)). In the example below, Boundaries 1, 2, 3 and 4 are in this category. The motion of these WALL FSI MOVING boundaries will be dictated by Equations 20.8 and 20.9. For these boundaries, CONVERGE ignores any velocity vector or motion file name in the corresponding [*boundary.in*](#) > *boundary_conditions* > *boundary* > *velocity* settings block. Also, if [*boundary.in*](#) > *boundary_conditions* > *boundary* > *motion* = FSI, that boundary ID must be listed within an object in [*fsi.in*](#).

Some boundaries may be a part of an FSI object but may not have FSI as their motion type ([*boundary.in*](#) > *boundary_conditions* > *boundary* > *motion*). For example, boundaries 5 and 6 (RightWing and LeftWing) have [*boundary.in*](#) > *boundary_conditions* > *boundary* > *motion* = ROTATING because these boundaries have a rotating motion assigned to them by their respective velocity profiles. The rotation specified for these boundaries will have a moving rotation center, with the initial rotation center specified in [*boundary.in*](#) > *boundary_conditions* > *boundary* > *velocity* > *origin* ([0.052655 0.099025 -0.034354] for boundary 5). CONVERGE automatically calculates the current position of the rotation center based on how the

Chapter 20: Fluid-Structure Interaction Modeling

FSI Setup

boundary position is affected by the FSI forces. Note that boundaries 5 and 6 also will be affected by the fluid-structure interaction parameters defined in [fsi.in](#) because boundaries 5 and 6 are included in the list of boundaries in that file.

You cannot have [boundary.in](#) > *boundary_conditions* > *boundary* > *geometry_motion* = *FIXED* if [boundary.in](#) > *boundary_conditions* > *boundary* > *motion* = *FSI*.

```
version: 3.1
---
boundary_conditions:
  - boundary:
      id: 1
      type: WALL
      name: Body
      region: 0
      motion: FSI
      geometry_motion: MOVING
      roughness:
        height: 0
        constant: 0.5
      velocity:
        type: Dirichlet
      pressure:
        type: Neumann
      temperature:
        type: Dirichlet
        value: 300
      species:
        type: Neumann
      passive:
        type: Neumann
    - boundary:
        id: 2
        type: WALL
        name: Tail
        region: 0
        motion: FSI
        geometry_motion: MOVING
        roughness:
          height: 0
          constant: 0.5
        velocity:
          type: Dirichlet
        pressure:
          type: Neumann
        temperature:
          type: Dirichlet
          value: 300
        species:
          type: Neumann
        passive:
          type: Neumann
    - boundary:
        id: 3
        type: WALL
        name: BirdFace
        region: 0
        motion: FSI
        geometry_motion: MOVING
        roughness:
          height: 0
          constant: 0.5
        velocity:
          type: Dirichlet
        pressure:
          type: Neumann
        temperature:
          type: Dirichlet
```

Chapter 20: Fluid-Structure Interaction Modeling

FSI Setup

```
        value: 300
        species:
          type: Neumann
        passive:
          type: Neumann
      - boundary:
          id: 4
          type: WALL
          name: Talons
          region: 0
          motion: FSI
          geometry_motion: MOVING
          roughness:
            height: 0
            constant: 0.5
          velocity:
            type: Dirichlet
          pressure:
            type: Neumann
          temperature:
            type: Dirichlet
            value: 300
        species:
          type: Neumann
        passive:
          type: Neumann
      - boundary:
          id: 5
          type: WALL
          name: RightWing
          region: 0
          motion: ROTATING
          geometry_motion: MOVING
          roughness:
            height: 0
            constant: 0.5
          velocity:
            type: Dirichlet
            origin: [0.052655, 0.099025, -0.034354]
            rotation_axis: [0, 0, 1]
            rotation_speed: right_motion.in
          pressure:
            type: Neumann
          temperature:
            type: Dirichlet
            value: 300
        species:
          type: Neumann
        passive:
          type: Neumann
      - boundary:
          id: 6
          type: WALL
          name: LeftWing
          region: 0
          motion: ROTATING
          geometry_motion: MOVING
          roughness:
            height: 0
            constant: 0.5
          velocity:
            type: Dirichlet
            origin: [-0.050097, 0.099026, -0.034362]
            rotation_axis: [0, 0, 1]
            rotation_speed: left_motion.in
          pressure:
            type: Neumann
          temperature:
            type: Dirichlet
            value: 300
        species:
          type: Neumann
```

Chapter 20: Fluid-Structure Interaction Modeling

FSI Setup

```
    passive:
        type: Neumann
  - boundary:
      id: 7
      type: WALL
      name: DomainLimits
      region: 0
      motion: STATIONARY
      geometry_motion: FIXED
      roughness:
        height: 0
        constant: 0.5
      velocity:
        type: Dirichlet
        origin: [0, 0, 0]
      pressure:
        type: Neumann
      temperature:
        type: Dirichlet
        value: 300
      species:
        type: Neumann
      passive:
        type: Neumann
```

Figure 20.2: The [boundary.in](#) file for the flying bird example. The geometry for this case is [shown above](#).

fsi.in

After preparing [boundary.in](#), you need to prepare [fsi.in](#). The [fsi.in](#) file specifies the number of FSI objects; the boundaries that constitute each FSI object; and information related to the spatial coordinates, applied moments and forces, and translational and rotational constraints. The file also sets flags for FSI spring and stiction options.

Figure 20.3 shows an [fsi.in](#) file for the flying bird case.

```
version: 3.1
---
rigid_fsi_objects:
  - object:
      object_id: fsiobject1
      boundary_id: [1, 2, 3, 4, 5, 7]
      gti_fsi_flag: 0
      fsi_start_time: 0.0
      center_of_mass_flag: GEO CENTER
      center_of_mass: [0.0, 0.0, 0.0]
      mass_flag: 0
      object_mass: 200.0
      object_density: 10.0
      object_vel_init: [0.0, 0.0, 0.0]
      object_rot_vel_init: 0.0
      rot_vel_init_axis: [1.0, 0.0, 0.0]
      applied_force: [0.0, 0.0, 0.0]
      applied_moment: 0.0
      applied_moment_axis: [0.0, 0.0, 0.0]
      moment_of_inertia_flag: 0
      moment_of_inertia_1: [1.0, 0.0, 0.0]
      moment_of_inertia_2: [0.0, 1.0, 0.0]
      moment_of_inertia_3: [0.0, 0.0, 1.0]
      1dof_constraint_flag: OFF
      axis_point: [0.0, 0.0, -1.0]
      axis: [0.0, 1.0, 0.0]
      min_displacement: -0.2
      max_displacement: 3.25
      constrained_mom_of_inertia: 1.0
#-----
```

Chapter 20: Fluid-Structure Interaction Modeling

FSI Setup

```
# forces option
#-----
spring_flag:          0
stiction_flag:        0
```

Figure 20.3: The [fsi.in](#) file for the flying bird example. The geometry for this case is [shown above](#).

The flying bird example uses zero applied force. You can specify a constant applied force directly in [fsi.in](#), or you can specify the name of tabulated applied force file (e.g., [fsi_force.in](#)).

Multi-Stream Considerations

You can define FSI objects that are in contact with multiple [streams](#). For example, consider a sphere that is submerged in water (stream 0) and contains a mixture of air and water (stream 1). As shown below in Figure 20.4, there are several ways to set up this FSI object.

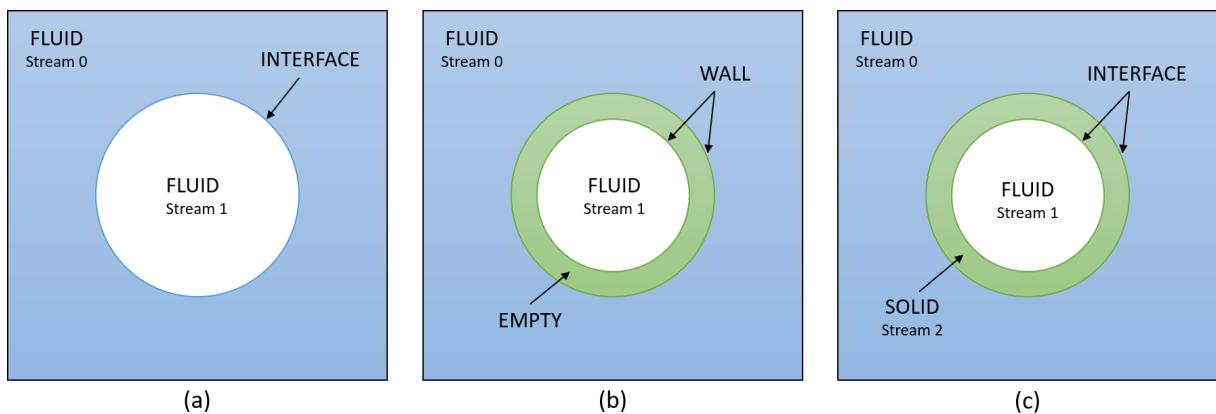


Figure 20.4: Possible configurations for a spherical FSI object in contact with two fluid streams.

For a thin-walled sphere, create a rigid FSI object consisting of one INTERFACE boundary between the two fluid streams (Figure 20.4a).

For a thick-walled sphere, create a rigid FSI object consisting of two boundaries. If the solid material enclosed by those boundaries is not of interest, mark each boundary as a WALL boundary (Figure 20.4b). CONVERGE will treat the space between the boundaries as an empty volume (except when calculating the mass, center of mass, or moment of inertia for the FSI object). If you want to include the behavior of the solid in your simulation (e.g., for [conjugate heat transfer](#) applications), mark each boundary as an INTERFACE boundary and set up a solid stream in the space between them (Figure 20.4c).

When you create FSI objects that are in contact with multiple streams, you must use the same values for [fsi.in](#) > `implicit_fsi` > `implicit_fsi_flag` and [inputs.in](#) > `property_control` > `gravity` in all streams. This is true for both fluid and solid streams. Additionally, when you create an FSI object that includes a fluid-fluid INTERFACE boundary, as in Figure 20.4a, you must specify the mass, center of mass, and moment of inertia for that object (i.e., set `mass_flag` = 1, `center_of_mass_flag` = 2, and `moment_of_inertia_flag` = 1 in [fsi.in](#) > `rigid_fsi_objects` > `object`).

`fsi_object_<name>.out`

Chapter 20: Fluid-Structure Interaction Modeling

FSI Setup

CONVERGE writes output data for each FSI object to an [*fsi_object_<name>.out*](#) file, where *<name>* is the object ID of the FSI object (as specified in [*fsi.in*](#) > * > *object_id*). For example, the output file name might be *fsi_object_anchor.out* or *fsi_object_3.out* (the *object_id* is a string, not a number).

20.4 FSI Beam

CONVERGE incorporates a special beam FSI object type that solves a simplified 1D finite element problem as part of a coupled FSI analysis. When you specify this object type, CONVERGE subdivides a long, thin object (*e.g.*, a reed valve) into a one-dimensional set of solid material elements. CONVERGE solves the deflection of these elements in two dimensions according to local fluid forces, specified material properties, and the classical Euler-Bernoulli beam equations.

CONVERGE writes FSI beam monitor point information to [*beam_<name>_monitor_point_<ID>.out*](#).

FSI Beam Inputs

In CONVERGE, a beam FSI object is characterized by several configuration types and parameters. You must specify physical dimensions and material properties, direction of deflection, damping coefficients, fluid load type, the beam support type, and the number of material elements in [*fsi.in*](#). You can optionally specify an external load, an initial deflection, a contact model, and the type of FSI beam solver.

The beam flexure model requires that you specify the beam length, breadth, height, cross-sectional area, density, Young's modulus, and moment of inertia. These can be constant-valued along the length of the beam, or you can provide a beam profile file. If you provide a beam profile file, the number of entries does not need to match the number of material elements you specify in [*fsi.in*](#).

CONVERGE supports a variety of beam FSI support configurations. You can specify one of three beam support types: cantilevered (zero displacement and zero rotation at the fixed end), simply supported (zero displacement at both ends), or doubly-clamped (zero displacement and zero rotation at both ends).

FSI beam motion is driven by a combination of fluid loading and optional external loading. CONVERGE directly calculates the fluid loading from the local pressure field and the FSI beam shape profile. You can prescribe an external loading of uniform or variable load, or a force or moment applied at the end of the beam.

You can impose an initial deflection of the beam based on a prescribed load. This can be either a static load or a load which varies in time (prescribed by a file), and it can be uniform in space or applied as a force or moment at the end of the beam.

You can also use a contact model to disallow beam FSI motion beyond set limits. The contact model can have user-defined constraints, in which you specify boundaries or planes through

which the beam cannot move, or it can be a resting contact model, in which the beam cannot move in a direction opposite to the direction of deflection.

CONVERGE offers two methods for solving the beam equations of motion. For simulations without a contact model, we recommend using the Hilber-Hughes-Taylor (HHT) method ([Hilber et al., 1977](#)), which is combined with the HYPRE implementation of the [BiCGSTAB](#) linear solver to solve the beam motion implicitly at each time-step of the fluid solver. For simulations with an active contact model, we recommend using the implementation of Moreau's midpoint time-stepping (MMT) scheme described in [Liakou et al., 2017](#). With the latter method, CONVERGE calculates different time-steps for the beam solver and the fluid solver and solves the beam motion explicitly. This method is required for simulations with a resting contact model. If you set `fsi.in > beam_solver_settings > beam_solver = AUTO`, CONVERGE will automatically select the recommended method based on your case setup.

Beam FSI Equations of Motion

The two-dimensional deflection of a long, thin deformable beam oriented along the x axis can be written

$$u = -z \frac{\partial w}{\partial x} \approx -z \theta, \quad (20.11)$$

where u is the displacement in x and w the displacement in z . The strain ε and stress σ are calculated as

$$\varepsilon = \frac{\partial u}{\partial x} = -z \kappa \quad (20.12)$$

and

$$\sigma = E \varepsilon = -E z \kappa, \quad (20.13)$$

where E is Young's modulus and κ is the curvature of the beam. The Lagrangian of the system is written

$$\frac{d}{dt} \frac{d\mathcal{L}}{dt} - \frac{\partial \mathcal{L}}{\partial x_i} = 0, \quad (20.14)$$

Chapter 20: Fluid-Structure Interaction Modeling

FSI Beam

where

$$\mathcal{L} = T - \Pi, \quad (20.15)$$

where T is the kinetic energy and Π is the potential energy. The potential energy is the sum of the bending energy U and the external work W , the latter including both conservative and non-conservative (*i.e.*, damping) work,

$$\Pi = U - (W^c + W^{nc}). \quad (20.16)$$

The total potential energy can alternately be written

$$\Pi = \frac{1}{2} \int_0^L \int_A \varepsilon \sigma dA dx - \int_0^L f w dx + \int_0^L f^d w dt, \quad (20.17)$$

where L is the total length of the beam, f is the force per unit length acting on the beam, f^d is a damping force, and A is the cross-sectional area. The kinetic energy is written

$$T = \frac{1}{2} \int_0^L \rho (d_t w)^2 A dx, \quad (20.18)$$

where ρ is the beam density. With the moment of inertia I , the total potential and kinetic energies for the beam can be written

$$\Pi = \frac{1}{2} \int_0^L \frac{d^2 w}{dx^2} EI \frac{d^2 w}{dx^2} dx - \int_0^L (f + f^d) w dx \quad (20.19)$$

and

Chapter 20: Fluid-Structure Interaction Modeling

FSI Beam

$$T = \frac{1}{2} \int_0^L \rho \frac{dw}{dt} dx. \quad (20.20)$$

Beam Finite Element Model

CONVERGE divides the beam into a user-defined number of elements. Each element consists of two nodes, each with two degrees of freedom (w and θ). The nodes of each interior element coincide with the nodes of the neighboring elements. At any point along the beam, the displacement can be expressed in terms of nodal values and shape functions,

$$w(\zeta) = N_i q_i^e, \quad (20.21)$$

where

$$\zeta = \frac{2}{l_e} x - 1, \quad (20.22)$$

q_i^e is the i -th component of the nodal vector $[w_1, \theta_1, w_2, \theta_2]$, and l_e is the length of the element. The shape functions are Hermite polynomials,

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\zeta)^2(2+\zeta), \\ N_2 &= \frac{1}{8}L(1-\zeta)^2(1+\zeta), \\ N_3 &= \frac{1}{4}(1+\zeta)^2(2-\zeta), \\ N_4 &= -\frac{1}{8}L(1+\zeta)^2(1-\zeta). \end{aligned} \quad (20.23)$$

Substituting the w expression into Equation 20.14, the Lagrangian can be rewritten as

$$\mathcal{L}^e = \frac{dq_i^e}{dt} M_{ij}^e + q_i^e K_{ij}^e q_j^e - F_i^e q_i^e + \frac{dq_j^e}{dt} C_{ij}^e q_j^e, \quad (20.24)$$

Chapter 20: Fluid-Structure Interaction Modeling

FSI Beam

where the mass matrix M_{ij}^e is calculated as

$$M_{ij}^e = \int_0^1 \rho A N_i N_j d\zeta, \quad (20.25)$$

the stiffness matrix K_{ij}^e is calculated as

$$K_{ij}^e = \int_0^1 EI \frac{dN_i}{d\zeta} \frac{dN_j}{d\zeta} d\zeta, \quad (20.26)$$

and the force vector F_i^e is calculated as

$$F_i^e = \int_0^1 f N_i d\zeta. \quad (20.27)$$

The damping force C_{ij}^e is assumed to take the form

$$C_{ij}^e = c_k K_{ij}^e + c_m M_{ij}^e, \quad (20.28)$$

where c_k and c_m are prescribed damping coefficients in [fsi.in](#).

The beam equations of motion take the form

$$M_{ij} \frac{d^2 q_j}{dt^2} + C_{ij} \frac{dq_j}{dt} + K_{ij} q_j = F_i, \quad (20.29)$$

which CONVERGE solves with appropriate boundary conditions and the method specified in [fsi.in](#) > `beam_solver_settings` > `beam_solver`.

Beam Fluid-Structure Coupling

Chapter 20: Fluid-Structure Interaction Modeling

FSI Beam

At each time-step in the fluid solver, CONVERGE takes the structural solver displacements from the previous time-step as the new boundary condition, and the structural solver in turn uses the forces from the fluid solver to compute the external load. This method is stable for cases when the added mass effect is small (*i.e.*, the solid density is much greater than the fluid density).

CONVERGE uses the Cohen-Sutherland clipping algorithm to cut the solid-surface triangles into conformal beam element polygons. Forces are assumed to be constant within each element.

20.5 FSI Events

There are two types of FSI events: FSI with rigid-body motion and FSI with a beam bending model. These events are implemented in a similar way to [standard OPEN and CLOSE events](#). In contrast to standard OPEN and CLOSE events, though, FSI events can be defined by either translation distances (for rigid or beam FSI events) or rotation angles (for rigid FSI events) rather than times. In case of conflict, FSI events override standard events.

To set up an FSI event, set `events.in > events > event > type` to `RIGID_FSI` or `BEAM` and include the subsequent parameters specific to that event.

20.6 FSI Stiction

CONVERGE includes a stiction model to approximate the interaction force that must be overcome to put two static objects into relative motion. To use the stiction model, specify a rigid FSI object and a maximum interaction distance that is measured in either *meters* or *degrees* (for `fsi.in > stiction > displacement_type` = 0 or 1, respectively), along with a maximum stiction force and/or moment. CONVERGE treats the stiction force or moment as a linear decay between the maximum value (at zero distance) and zero (at the maximum distance), as shown in the example below in Figure 20.5.

Set `fsi.in > stiction_flag` = 1 to activate the FSI stiction model. CONVERGE will refer to the [`stiction.in`](#) input file for the stiction model parameters.

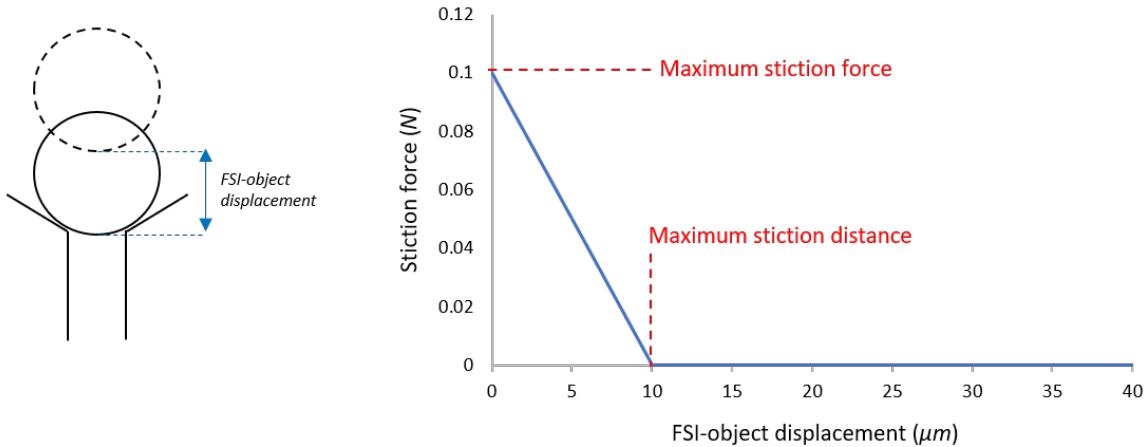


Figure 20.5: Example stiction force applied to a ball valve.

20.7 FSI Spring

You can use the CONVERGE FSI spring feature to model spring forces between a fixed object and a rigid FSI object. The model is an approximation of a damped linear coil spring, with specified stiffness, damping constant, length, and preload.

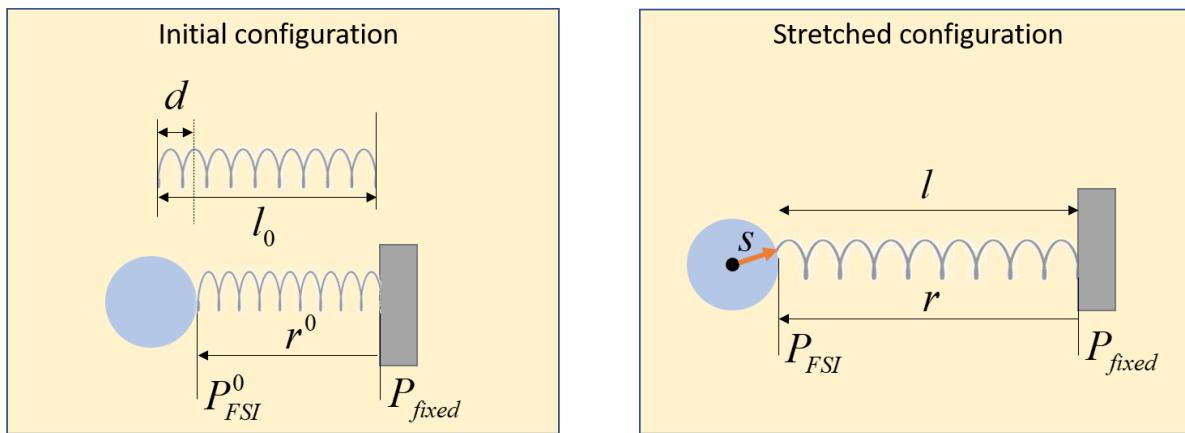


Figure 20.6: Example configuration for a rigid FSI object attached to a spring.

Figure 20.6 shows an example configuration for a spring that is attached to a rigid FSI object at one end and to a fixed object at the other end. For this configuration, Equation 20.3 becomes

$$m\ddot{x}_i = F_{fluid-i} + F_{spring-i}, \quad (20.30)$$

Chapter 20: Fluid-Structure Interaction Modeling

FSI Spring

while Equation 20.4 becomes

$$I_{ij}\dot{\omega}_j + \varepsilon_{ijk}\omega_j I_{kl}\omega_l = M_{fluid-i} + M_{spring-i}. \quad (20.31)$$

CONVERGE calculates the spring force from

$$F_{spring-i} = -\frac{k(l-l_0)}{l}r_i - cu_{FSI-i} \quad (20.32)$$

where r_i is the instantaneous displacement of the FSI endpoint with respect to the fixed endpoint in direction i , k is the spring constant, and c is the damping constant. The stretched length of the spring is defined as

$$l = \sqrt{r_i^0 r_i^0}. \quad (20.33)$$

The natural length of the spring is defined as

$$l_0 = \sqrt{r_i^0 r_i^0} + d, \quad (20.34)$$

where d is the initial deformation of the spring and

$$r_i^0 = P_{FSI-i}^0 - P_{fixed-i}. \quad (20.35)$$

The velocity used to calculate the damping term in Equation 20.32 is given by

$$u_{FSI-i} = u_{COR-i} + \varepsilon_{ijk}\omega_j s_k, \quad (20.36)$$

where u_{COR-i} is the velocity of the FSI object's center of rotation and the vector s_i points from the center of rotation to the instantaneous FSI endpoint P_{FSI-i} .

The moment applied by the spring is calculated from

$$M_{spring-i} = \varepsilon_{ijk} s_j F_{spring-k}. \quad (20.37)$$

To activate the FSI spring model, set `fsi.in > spring_flag = 1` and configure spring parameters in the `spring.in` file. If you configure a spring that is detached at one end (`spring.in > spring > detached_flag = 1`), there will be an initial separation between the FSI object and the spring (`spring.in > spring > detachment_distance`), and CONVERGE will apply the spring force and moment only when the FSI object is in contact with the spring (*i.e.*, $F_{spring-i} = M_{spring-i} = 0$ when the FSI object is not touching the spring).

If you are simulating a physical coil spring, you can add fidelity to the model by using subsprings, as shown below in Figure 20.7. In this configuration, CONVERGE distributes the stiffness and damping constants among N subsprings according to

$$k_{subspring} = \frac{k}{N} \quad c_{subspring} = \frac{c}{N} \quad (20.38)$$

and calculates the total spring force and moment as

$$F_{spring-i} = \sum_{n=1}^N F_{subspring-i}^n \quad M_{spring-i} = \sum_{n=1}^N M_{subspring-i}^n. \quad (20.39)$$

In `spring.in`, you specify a radius and azimuthal angle for each subspring, along with a zero-angle direction, as shown below. CONVERGE determines the fixed endpoint for each subspring by rotating the zero-angle vector about the spring axis, which is in the direction of the spring vector r_i^0 from Equation 20.35. A simulation with many subsprings will approximate an ideal coil spring.

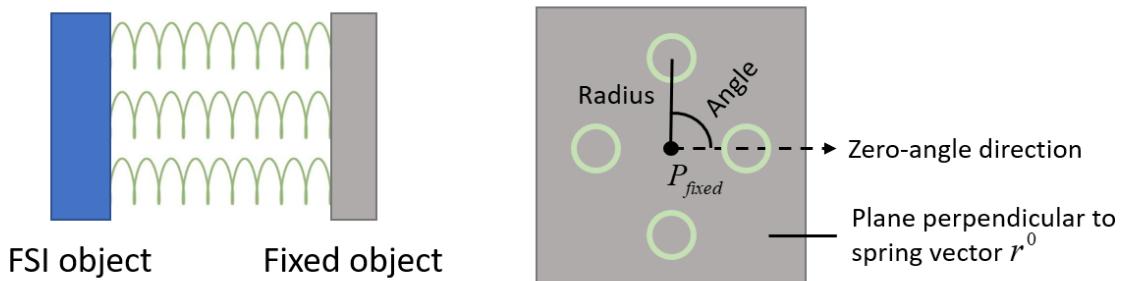


Figure 20.7: FSI spring divided into subsprings.

20.8 GT-SUITE/FSI Coupling

In addition to the basic fluid-structure interaction capability described previously in this chapter, CONVERGE can perform a simulation coupled with GT-SUITE.

In a CONVERGE + GT-SUITE simulation that includes FSI modeling, there are several case setup requirements.

- Set [*cosimulation.in*](#) > *GT-SUITE* > *scheme* = *GT LEAD* (this option directs GT-SUITE to advance in time prior to CONVERGE).
- Set [*fsi.in*](#) > *rigid_fsi_objects* > *object* > *gti_fsi_flag* = 1.
- Set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *motion* = *FSI* as described previously in this chapter and [set up the GT-SUITE boundaries](#).

Finally, it is important to verify that [*fsi.in*](#) > *rigid_fsi_objects* > *object* > *object_id* matches the object name given in GT-SUITE.

20.9 Abaqus/FSI Coupling

In addition to the basic fluid-structure interaction capability described previously in this chapter, CONVERGE can perform a simulation coupled with the [Abaqus finite element solver](#). This approach offers the most general capability to analyze the interaction of the fluid flow with deformable solid boundaries.

Abaqus boundaries are defined as walls with a special motion type. Set [*boundary.in*](#) > *boundary_conditions* > *boundary* > *type* = *WALL*, [*boundary.in*](#) > *boundary_conditions* > *boundary* > *motion* = *ABAQUS*, and [*boundary.in*](#) > *boundary_conditions* > *boundary* > *geometry_motion* = *MOVING* for each boundary that will be subject to Abaqus analysis.

For a CONVERGE + Abaqus coupled simulation, [*cosimulation.in*](#) is required.

CONVERGE 3.1 supports co-simulation with Abaqus 2019-2022. You can couple CONVERGE with either Abaqus/Standard or Abaqus/Explicit. Abaqus/Explicit requires substantially less memory and disk space than Abaqus/Standard. However, CONVERGE can couple implicitly in time with Abaqus/Standard. The latter is much more

Chapter 20: Fluid-Structure Interaction Modeling

Abaqus/FSI Coupling

computationally stable for cases where the FSI object and surrounding fluid are of similar density.

Chapter



21

Radiation Modeling

21 Radiation Modeling

In addition to energy transport by conduction and convection, high-temperature systems may experience substantial energy transport due to radiation. In general, at any point within a non-opaque region (*i.e.*, a high-temperature gas), the local medium emits radiation and variously absorbs, scatters, and transmits incident radiation. Solid boundaries and other local media (*e.g.*, spray parcels or soot) also participate in emission, absorption, and scattering, as well as some directional effects (*e.g.*, smooth-wall specular reflection). The emission, absorption, and scattering coefficients are a function of temperature, radiation frequency, and species, as well as surface finish for solid boundaries and soot particle distribution functions. The unsteady radiation transport equation (RTE) is given as

$$\frac{1}{c} \frac{\partial I_\eta}{\partial t} + \frac{\partial I_\eta}{\partial s} = \kappa_\eta I_{b\eta} - \kappa_\eta I_\eta - \sigma_{s\eta} I_\eta + \frac{\sigma_{s\eta}}{4\pi} \int_{4\pi} I_\eta(\hat{s}_i) \Phi_\eta(\hat{s}_i, \hat{s}) d\Omega_i, \quad (21.1)$$

where c is the speed of light, I is the intensity, I_b is an emission term, s is the distance along a ray, \hat{s} is the unit vector along a ray, \hat{s}_i is a local surface normal, κ is an absorption coefficient, σ_s is a scattering coefficient, Φ is the scattering phase function, and Ω' is the solid angle. The subscript η indicates a wavenumber (*i.e.*, a frequency) dependence.

The full RTE given above is an extremely stiff integro-differential equation in five spatial dimensions. It is impractical to perform engineering analyses by solving the full RTE. CONVERGE offers several reduced-fidelity models and simplifying assumptions which allow you to model radiation and associated heat transfer at a reasonable computational cost. These are split into two broad categories: the [method of spherical harmonics](#) (also called the PN approximation) and the [finite volume method](#), a development of the method of discrete ordinates (also called the SN approximation). Both the spherical harmonic method and the finite volume method approximate the RTE as a set of partial differential equations, which CONVERGE then solves with appropriate discretizations and approximations.

Radiation modeling may be useful for gas turbine or diffusion flame simulations, among other applications. Because carbon-containing reaction products are more optically active than water vapor, radiation modeling is relatively more important for combustion of fuels with higher carbon content.

To model radiation, set `inputs.in > feature_control > radiation_flag = 1` in and include the `radiation.in` file in your case setup. You can set numerical constraints such as under-relaxation and convergence criteria by configuring the parameters in the `solver.in > Transport > radiation` settings block.

Chapter 21: Radiation Modeling

Spherical Harmonics Method

21.1 Spherical Harmonics Method

Spherical harmonic (or PN) methods transform the integro-differential radiative transfer equation (RTE) into a set of partial differential equations. Broadly, PN methods assume that the radiation intensity environment is relatively isotropic, and so invoke spherical symmetries to reduce the complexity of the transport equation. The N indicates the order of the method, with a higher order being more accurate and more expensive. Because mathematical complexity increases more rapidly than solution convergence, P1 methods are common. These solve a single elliptic PDE for radiation transport. Refer to [Modest \(2003\)](#) for more details on the general PN model derivation and formulation.

In the CONVERGE implementation, the P1 model solves a transport equation for each band,

$$\frac{\partial}{\partial x_j} \left(\frac{1}{\tilde{\kappa}_i} \frac{\partial G_i}{\partial x_j} \right) - 3\tilde{\kappa}_i G_i = -12\pi\tilde{\kappa}_i \tilde{I}_{bi}, \quad (21.2)$$

where G is the incident radiation and i indicates a band subscript and not a summation index. The blackbody intensity \tilde{I}_{bi} and Planck-mean absorption coefficient $\tilde{\kappa}_i$ are calculated as

$$\tilde{\kappa}_i = \frac{\int_{\Delta\eta_i} \kappa_\eta I_{b\eta} d\eta}{\int_{\Delta\eta_i} I_{b\eta} d\eta} \quad (21.3)$$

and

$$\tilde{I}_{bi} = \frac{1}{\Delta\eta_i} \int_{\Delta\eta_i} I_{b\eta} d\eta. \quad (21.4)$$

The P1 transport equation is subject to the Marshak boundary condition

$$\frac{2-\varepsilon}{\varepsilon} \frac{2}{3\tilde{\kappa}_i} \hat{n}_j \frac{\partial G_i}{\partial x_j} + G_i = 4\pi\tilde{I}_{bwi}, \quad (21.5)$$

Chapter 21: Radiation Modeling

Spherical Harmonics Method

where ε is the wall emissivity, \hat{n}_j is the surface normal vector, and \tilde{I}_{bwi} is the blackbody intensity at the wall temperature.

The P1 model's low computational expense makes it attractive for an engineering simulation, but you must ensure that the simplifying approximations are valid. In particular, the gas must be optically thick (*i.e.*, the absorption coefficient is large), and emission, absorption, and scattering must all be relatively isotropic. Many modern high-efficiency internal combustion engines satisfy these requirements.

All gray and nongray gas models are available for the P1 radiation model.

21.2 Finite Volume Method

The Discrete Ordinates (DO) method, which encompasses all optical thicknesses, was developed to solve non-isotropic problems with participating media. The computational cost depends on angular discretization and pixelation. As originally formulated, the DO method suffers several drawbacks, the most important being radiative energy non-conservation.

By using the finite volume method to discretize the angular terms as well as the Cartesian spatial terms, this non-conservation can be corrected. In the radiation FVM, the weight associated with each direction is the solid angle for that direction, in contrast to the quadrature weight of the traditional DO method.

This model can be used with either gray radiation or nongray radiation. For nongray radiation, CONVERGE includes a gray band model and the Weighted Sum of Gray Gases (WSGG) model.

FVM Governing Equations

The FVM model solves the radiative transfer equation (RTE) for a fixed set of discrete solid angles. Each angle is associated with a fixed Cartesian vector direction s_i . The quasi-steady RTE for an absorbing, emitting, and scattering medium at position r_i in the direction s_i is given by

$$\frac{\partial(I(r_i, s_i)s_i)}{\partial x_i} + (a + \sigma_s)I(r_i, s_i) = an_{RI}^2 \frac{\sigma T^4}{\pi} + \frac{\sigma_s}{4\pi} \int_0^{4\pi} I(r_i, s'_i)\Phi(s'_i s_i)d\Omega'. \quad (21.6)$$

In this equation, r_i is the position vector (and is a function of x , y , and z), s_i is the direction vector, s'_i is the scattering direction vector, a is the absorption coefficient, n_{RI} is the refractive index, σ_s is the scattering coefficient, σ is the Stefan-Boltzmann constant, I is the radiation intensity, T is the local temperature, Φ is the scattering phase function, and Ω' is the solid angle.

Chapter 21: Radiation Modeling

Finite Volume Method FVM Governing Equations

The radiation heat flux at the wall is computed by

$$J_w = \int_{\Omega, \mathbf{n} \cdot \mathbf{s} > 0} I \mathbf{n} \cdot \mathbf{s} d\Omega = \int_{\Omega, \mathbf{n} \cdot \mathbf{s} > 0} \varepsilon_w I_{bw} \mathbf{n} \cdot \mathbf{s} d\Omega + (1 - [\varepsilon_w + \tau_w]) \int_{\Omega, \mathbf{n} \cdot \mathbf{s} > 0} I |\mathbf{n} \cdot \mathbf{s}| d\Omega, \quad (21.7)$$

where J_w is the wall heat flux, I_{bw} is the blackbody intensity at the wall temperature, ε_w is the wall emissivity, and τ_w is the wall transmissivity.

Equation 21.6 can be written as a set of M partial differential equations,

$$e_j^l \frac{\partial I}{\partial x_j} = -(\sigma_a + \sigma_s) I^l + \sigma_a I_b + S^l, \quad (21.8)$$

where e_j is the direction cosine, σ_a is the absorption coefficient, σ_s is the scattering coefficient, and l is the counter from 1 to M , the number of discrete solid angles. S is given as

$$S^l = \frac{\sigma_s}{4\pi} \left[\sum_{l'=1}^M w^{l'} \Phi^{ll'} l' + \sum_B \Phi^{lB} l^B \right]. \quad (21.9)$$

Spatial Discretization

The spatial discretization to solve the radiative transfer equation (RTE) is two-fold.

1. Finite-volume in Cartesian space (x, y, z) for radiation intensity as given by Equation 21.8 in the [Discrete Ordinates Governing Equations](#) section.
2. Discrete solid angles for each direction vector (s_i) of intensity I .

Angular Discretization

There are two possible schemes for angular discretization.

Set `radiation.in > radiation_model = FVM` to implement the standard finite volume method (FVM). In this model the polar and azimuthal angles are subdivided uniformly in N_θ and N_ϕ directions, respectively, with a total number of $N_\theta N_\phi$ control angles per octant. This scheme is shown in the figure below.

Alternatively, set `radiation.in > radiation_model = FTnFVM` to use a variation of the finite volume method called FTnFVM. In this model the polar angle in an octant is subdivided uniformly into a pair number N , and the azimuthal angle is divided uniformly in an octant

in the following sequence 1, 2, ..., N in each level of the polar angle. You can control the angular refinement for spherical discretization by [radiation.in > discretization > ordinates_in_octant > num_theta](#) and [radiation.in > discretization > ordinates_in_octant > num_phi](#). This scheme is shown below.

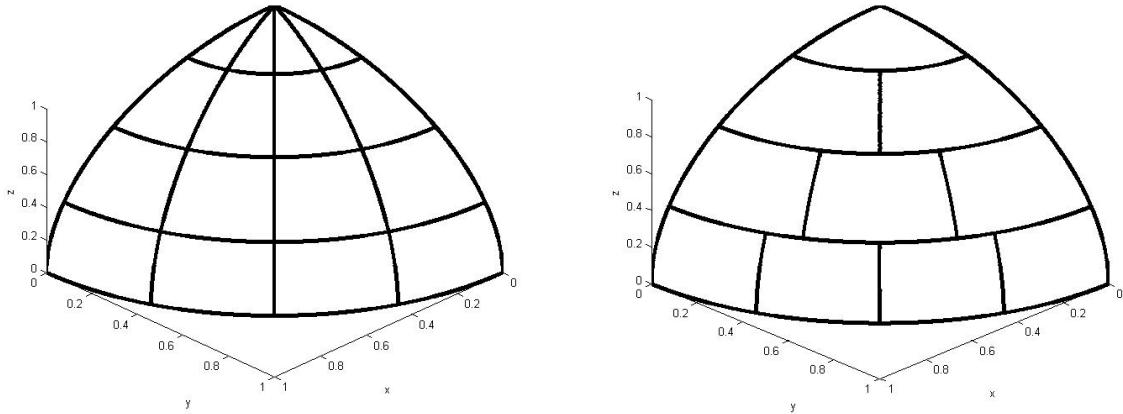


Figure 21.1: Angular discretization for the DO method. The image at left pertains to the FVM, while the image at right pertains to the FTnFVM.

Pixelation

Because CONVERGE uses a Cartesian mesh, the angular discretization aligns with the control volume face. However, control volume faces do not always align with the angular discretization at boundaries in CONVERGE, leading to the problem of control angle overhang. As shown below in Figure 21.2, this problem results in control angles that are split by the boundary.

CONVERGE adds resolution by discretizing each control angle into $N_{\theta p}N_{\phi p}$ pixels. You can set the pixel resolution by [radiation.in > discretization > pixels_in_ordinate > num_theta_pix](#) and [radiation.in > discretization > pixels_in_ordinate > num_phi_pix](#).

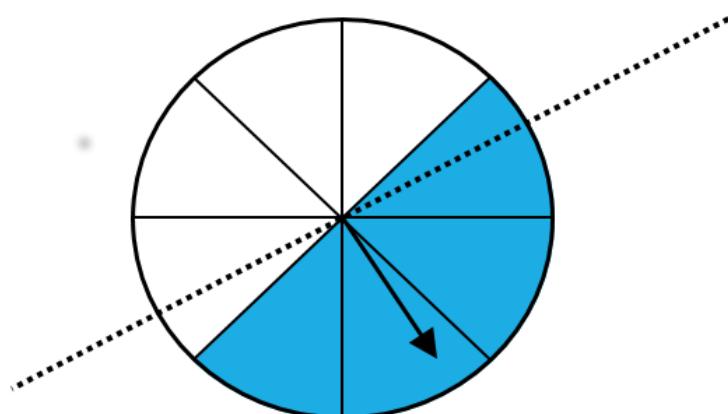


Figure 21.2: Control angle overhang.

Scattering Phase Function

There are five options for the scattering phase function (Φ). These options are controlled by [*radiation.in*](#) > *scattering* > *scatter_function* as follows:

- *scatter_function* = *OFF*: No scattering,
- *scatter_function* = *ISO*: Isotropic phase function,
- *scatter_function* = *ANISO*: Linear anisotropic phase function,
- *scatter_function* = *DELTA_EDD*: A Delta-Eddington phase function, and
- *scatter_function* = *UDF*: Use the *user_scattering_phase_function* user-defined function to define the scattering phase function (consult the CONVERGE 3.1 UDF Manual for more information).

For isotropic scattering (*i.e.*, scattering that is equally likely in all directions), set [*radiation.in*](#) > *scattering* > *scatter_function* = *ISO*.

The linear anisotropic phase function ([*radiation.in*](#) > *scattering* > *scatter_function* = *ANISO*) is given by

$$\Phi(s'_i s_i) = 1 + C_{aniso} s'_i s_i, \quad (21.10)$$

in which s_i and s'_i are the two direction vectors.

The Delta-Eddington function ([*radiation.in*](#) > *scattering* > *scatter_function* = *DELTA_EDD*) is given by

$$\Phi(s'_i s_j) = 2f_{DE}\delta(1 - s'_i s_i) + (1 - f_{DE})(1 + C_{aniso} s'_i s_i). \quad (21.11)$$

21.3 Gas Models

In general, gases absorb and emit radiation over relatively narrow wavelength bands specific to that gas. CONVERGE offers several simplified gas models. These are split into two classes: the [gray gas model](#) and [nongray gas models](#).

Gray Gas Model

The gray gas model is the simplest radiation model available. As the name suggests, the gray gas model treats the gas as though its properties (emissivity, absorption coefficient, etc.) are completely insensitive to radiation wavelength. In common parlance, the gas is colorless, or gray.

Chapter 21: Radiation Modeling

Gas Models Gray Gas Model

Activate the gray gas model by setting `radiation.in > nongray_radiation > nongray_model = GRAY`. You can either directly prescribe the absorption coefficient (`radiation.in > absorption > absorption_coeff_model = 0`) or direct CONVERGE to calculate the absorption coefficient as a function of gas-phase properties (`radiation.in > absorption > absorption_coeff_model = 1`). This assumes the absorption coefficient to be a function of the local concentration of H₂O, CO₂, and soot.

Nongray Gas Radiation Models

If you set `radiation.in > nongray_radiation > nongray_model = WSGG` or `BAND`, then CONVERGE will perform a nongray gas simulation. The two options ([Weighted Sum of Gray Gases model](#), [band model](#)) are described below.

Weighted Sum of Gray Gases Model

CONVERGE will use the Weighted Sum of Gray Gases (WSGG) model if `radiation.in > nongray_radiation > nongray_model = WSGG`. Specify the number of gray gases via `radiation.in > nongray_radiation > wsgg > num_gray_gases`. The total emissivity over the distance s in the WSGG model is given by

$$\varepsilon = \sum_{i=0}^I a_{\varepsilon,i}(T) \left(1 - e^{-\kappa_i p s}\right), \quad (21.12)$$

where $a_{\varepsilon,i}$ are the weighting factors for the fictitious gray gases, κ_i is the absorption coefficient of the i -th gray gas, p , is the sum of the partial pressures of all of the absorbing gases, and s is the path length. You can specify up to four gray gases.

The absorption coefficient for $i = 0$ is assigned a value of zero to account for windows in the spectrum between spectral regions of high absorption, and the weighting factor for $i = 0$ is given as ([Smith et al., 1982](#))

$$a_{\varepsilon,0} = 1 - \sum_{i=1}^I a_{\varepsilon,i}. \quad (21.13)$$

The temperature dependence of $a_{\varepsilon,i}$ is approximated by

$$a_{\varepsilon,i} = \sum_{j=1}^J b_{\varepsilon,i,j} T^{j-1}, \quad (21.14)$$

where $b_{\varepsilon,i,j}$ are the emissivity gas temperature polynomial coefficients. The polynomial coefficients $b_{\varepsilon,i,j}$ and κ_i are nearly constant with respect to ps (the pressure-path length product) and T . CONVERGE calculates initial polynomial coefficients and absorption coefficients assuming they are constant-valued. These coefficients are calculated for different relative pressures of the CO₂ and H₂O vapor, assuming that the total pressure is 1 *atm*. At 1 *atm*, these constant-valued coefficients are considered valid for $0.001 \leq ps \leq 10.0$ *atm-m* and $600K \leq T \leq 2400K$ ([Smith et al., 1982](#)).

For pressures higher than 1.1 *atm* or lower than 0.9 *atm*, CONVERGE locally scales the absorption coefficients in each cell. This scaling ([Edwards and Matavosian, 1984](#)) reduces the error associated with the constant-value assumption. The absorption coefficients are scaled according to

$$\kappa'_i = \kappa_i p_{atm}^m, \quad (21.15)$$

where p_{atm} is the pressure in *atm* and the pressure exponent m is a non-dimensional term that is a function of species partial pressures, total pressure, and temperature.

Band Model

If you set [*radiation.in* > nongray_radiation > nongray_model = BAND](#), CONVERGE will use a band model. For band models, you must provide the spectral band bounds and the absorption and scattering coefficients. For each spectral band, specify the lower spectral bound, the upper spectral bound, the absorption coefficient, and the scattering coefficient via the [*radiation.in* > nongray_radiation > band > band_properties](#) settings block.

21.4 Radiation/Spray Coupling

CONVERGE will couple radiation modeling and [*discrete phase modeling*](#) when [*radiation.in* > radiation_spray_coupling > rad_spray_coupling_flag = 1](#). This option captures the effect of parcels on radiation. The contribution is given by

$$\begin{aligned} \frac{\partial(I s_i)}{\partial x_i} + (a + a_p + \sigma_p) I(r_i, s_i) = \\ a n_{RI}^2 \frac{\sigma T^4}{\pi} + E_p + \frac{\sigma_p}{4\pi} \int_0^{4\pi} I(r_i, s_i) \Phi(s'_i s_j) n_i n_j d\Omega', \end{aligned} \quad (21.16)$$

where a_p is the absorption coefficient due to the presence of particulates and is given by

Chapter 21: Radiation Modeling

Radiation/Spray Coupling

$$a_p = \sum_{n=1}^N \varepsilon_{pn} \frac{A_{pn}}{V}. \quad (21.17)$$

The emission due to particulates, E_p , is given by

$$E_p = \sum_{n=1}^N \varepsilon_{pn} A_{pn} \frac{\sigma T_{pn}^4}{\pi V}, \quad (21.18)$$

where σ_p is the particle scattering factor, as follows:

$$\sigma_p = \sum_{n=1}^N (1 - \varepsilon_{pn}) \frac{A_{pn}}{V}, \quad (21.19)$$

where ε_{pn} , A_{pn} , T_{pn} are the emissivity, projected area, and temperature, respectively, of parcel n . In the above equations the summation is over N parcels in volume V . Specify the parcel emissivity (ε_{pn}) via [*radiation.in*](#) > *radiation_spray_coupling* > *parcel_emissivity*.

21.5 Radiation/Energy Coupling

As discussed in the [Iterative Algorithm section](#) of [Chapter 4 - Numerics](#), CONVERGE can solve the radiation transport equation (RTE) either inside or outside of the iterative loop.

To solve radiation outside of the iterative algorithm, set [*radiation.in*](#) > *radiation_energy_coupling* > *rad_energy_coupling_flag* = SEG. By default, CONVERGE will solve the RTE every time-step, but you can alter this behavior. Set [*radiation.in*](#) > *radiation_energy_coupling* > *rad_solve_frequency* to an integer larger than one to direct CONVERGE to solve the RTE every *rad_solve_frequency* time-steps.

To solve radiation inside of the iterative algorithm, set [*radiation.in*](#) > *radiation_energy_coupling* > *rad_energy_coupling_flag* = COUPLE. CONVERGE will solve the RTE within the iterative algorithm at each time-step. This option is more computationally expensive than solving it outside of the loop, but it may be more stable for some simulations.

21.6 Non-Transport Passives and Post-Processing

There are three [non-transport passives](#) related to radiation modeling: *RADIATION*, *RADIATION_SRC*, and *RAD_BOUND_FLUX*. CONVERGE does not require these non-transport passives to be defined in order to perform a simulation with radiation modeling.

Chapter 21: Radiation Modeling

Non-Transport Passives and Post-Processing

To direct CONVERGE to write cell-by-cell values that you need to post-process the data from a radiation simulation, you must include the *RADIATION*, *RADIATION_SRC*, and/or *RAD_BOUND_FLUX* non-transport passives in [*species.in*](#), and you must include the *radiation* variable in [*post.in*](#). The *RADIATION* non-transport passive is the incident radiation, which is calculated by the integral of intensity of radiation over a spherical surface:

$$\int_0^{4\pi} I(r_i, s_i) d\Omega. \quad (21.20)$$

The *RADIATION_SRC* non-transport passive is the radiation source term. The radiative heat flux is the *RAD_BOUND_FLUX* non-transport passive. If you include the *RAD_BOUND_FLUX* non-transport passive in [*species.in*](#), then you must specify the variable *bound_flux* in [*post.in*](#). The radiative heat flux at CONVERGE boundaries is given by

$$\int_0^{2\pi} I(r_i, s_i) s_i d\Omega. \quad (21.21)$$

Chapter



22

Aeroacoustic Modeling

22 Aeroacoustic Modeling

Sound is generated and transmitted directly by turbulent fluid motion and by aerodynamic forces interacting with boundaries. It can be strongly tonal (e.g., generated by periodic vortex shedding or rotating machinery) or broadband (e.g., generated by turbulence). The study of this generation and transmission, termed aeroacoustics, is of practical importance to many engineering systems. CONVERGE offers several techniques for performing computational aeroacoustic (CAA) studies.

Fundamentally, acoustic waves are weak longitudinal pressure/density disturbances, approximately inviscid, isentropic, and adiabatic, that propagate through a compressible material. In a liquid or gas, they obey the same governing Navier-Stokes equations that describe bulk fluid dynamics, and in principle they can be resolved by direct numerical simulation (DNS) or large eddy simulation (LES). However, acoustic waves have a number of properties that make a direct approach challenging for engineering applications.

Compared to bulk fluid dynamics, acoustic dynamics present wide ranges of scales of time, length, and energy. The human range of hearing is traditionally taken to be 20 Hz to 20 KHz; resolving the latter immediately implies a maximum time-step of no larger than 5e-5 seconds. In air at standard temperature and pressure, the speed of sound is approximately 340 m/s. Thus, a 20 KHz acoustic wave has a wavelength of about 1.7 cm. Nyquist sampling dictates a theoretical minimum of two samples per wavelength to resolve a wave, and Tam (2014) notes that traditional bulk-flow numerical methods typically require on the order of 20 grid points to resolve acoustic waves. At 20 KHz, this implies a base grid spacing on the order of 1 mm.

In contrast, the domain of interest for aeroacoustic modeling is often very large. The magnitude of an expanding acoustic wave diminishes approximately linearly with increasing distance, rather than the more familiar inverse-square relationship. Thus, acoustic waves can meaningfully propagate to distances greater than hundreds of meters, at least five orders of magnitude than the base grid spacing necessary to resolve them.

Further complicating matters, the magnitude of acoustic disturbances is very low compared to bulk fluid motion. [Thompson \(1988\)](#) cites the average threshold of human hearing as a power flux of 1e-12 W/m², while a normal conversational voice has a total power of about 1e-6 W. The range of human-relevant acoustic powers spans some 14 orders of magnitude, and some engineering systems must be designed with much greater acoustic powers in mind. Sound intensity is frequently five or six orders of magnitude weaker than variations in the mean flow. In practice, conventional computational fluid dynamics techniques are overwhelmed by numerical dissipation at acoustic intensities if the disturbances are to be transmitted over reasonable distances (direct Navier-Stokes transmission can be practical over short distances).

22.1 Direct Methods

The direct simulation of acoustic disturbances is computationally expensive, but it can be practical over short distances. Direct CAA requires either [LES](#) turbulence modeling or a full DNS. It also requires non-reflecting boundary conditions (*i.e.*, [NSCBC](#)).

Higher acoustic frequencies have shorter wavelengths and, thus, require finer meshes to resolve directly. The mesh frequency cutoff f_{MC} characterizes the highest frequency that can be resolved locally. It is given as

$$f_{MC} = \frac{\sqrt{\frac{2}{3}} k}{2\Delta}, \quad (21.22)$$

where k is the local turbulent kinetic energy and Δ is the characteristic local grid spacing. Specify [post.in](#) > *cells* > *general* > *acoustic_mesh_freq_cutoff* to output this quantity to *post*.h5*. This frequency cutoff should be treated as a rough guideline, not a guarantee of grid-converged acoustic propagation.

22.2 Reduced-Order Models

Reduced-order computational aeroacoustic models partially decouple the solution of the bulk fluid dynamics from the generation and propagation of acoustic waves. Generally, the coupling is one-way: acoustic sources are calculated from the fluid field calculated by a CFD solver, but there is no feedback from the acoustics to the fluid dynamics. A reduced-order CAA model must calculate these acoustic sources and process and/or transmit them, depending on the use case. Acoustic waves are linear and thus can be superimposed, and this property makes acoustic simulations practical.

CONVERGE offers two types of reduced-order CAA models, both based on integral formulations. Far-field models derive a wave equation for density fluctuations from the Navier-Stokes equations. This wave equation can be solved accurately to predict acoustic transmission over very long distances (*kilometers* or greater) and offers spectral information. Near-field models are designed to identify noise sources within the domain based on the local flow state (tke, eps, pressure, etc.), as well as estimate their acoustic power. They are broadband in nature, offering no spectral information. Both classes of models are one-way coupled with the CFD solution. Multiple aeroacoustic models can be specified simultaneously without having any effect on each other.

Both classes of reduced-order CAA models in CONVERGE rely on the acoustic analogy proposed by Lighthill ([1952](#), [1954](#)). This reformulates the Navier-Stokes equations as an inhomogeneous wave equation:

Chapter 22: Aeroacoustic Modeling

Reduced-Order Models

$$\frac{\partial^2 \rho'}{\partial t^2} - c^2 \frac{\partial^2 \rho'}{\partial x_i \partial x_i} = \frac{\partial^2 T_{ij}}{\partial x_i \partial x_j}, \quad (22.1)$$

where T_{ij} is the Lighthill stress tensor, the acoustic source,

$$T_{ij} = \rho u_i u_j - \sigma_{ij} + (p' - c^2 \rho') \delta_{ij}. \quad (22.2)$$

The viscous source term σ_{ij} is typically neglected, leaving

$$T_{ij} = \rho u_i u_j + (p' - c^2 \rho') \delta_{ij}. \quad (22.3)$$

If you specify [*acoustics.in*](#) > *acoustic_models* > *LIGHTHILL_STRESS_TENSOR*, CONVERGE will write this tensor to [*post*.h5*](#).

Far-Field Methods

Far-field models based on acoustic analogies use integrals of solved quantities on source surfaces or volumes. They can predict noise at distant receivers, both broadband and tonal (*i.e.*, frequency-dependent). They are computationally much less expensive than direct CAA.

Ffowcs Williams-Hawkins Equation

[Ffowcs Williams and Hawking \(1969\)](#) generalized Lighthill's analogy to describe the acoustic properties of a flow contained by surfaces in arbitrary motion. The Ffowcs Williams and Hawking equation takes the form

$$\frac{1}{c^2} \frac{\partial^2 p'}{\partial t^2} - \frac{\partial^2 p'}{\partial x_k \partial x_k} = S_T + S_L + S_Q, \quad (22.4)$$

Chapter 22: Aeroacoustic Modeling

Reduced-Order Models Far-Field Methods

$$\begin{aligned} S_T &= \frac{\partial}{\partial t} \left\{ [\rho_0 v_n + \rho(u_n - v_n)] \delta(f) \right\} \\ S_L &= -\frac{\partial}{\partial x_i} \left\{ [P_{ij} n_j + \rho u_i (u_n - v_n)] \delta(f) \right\} \\ S_Q &= \frac{\partial^2}{\partial x_i \partial x_j} \left\{ T_{ij} H(f) \right\}, \end{aligned} \quad (22.5)$$

where f defines the surface of the body such that $f < 0$ inside the body and $f > 0$ outside of it, $H(f)$ is the Heaviside function, u_n is the fluid velocity normal to the surface, $\delta(f)$ is the Dirac delta function, and S_T , S_L , and S_Q represent source terms defined below. This equation also introduces the velocity of the surface in the negative normal direction,

$$v_n = -\frac{\partial f}{\partial n}. \quad (22.6)$$

The first acoustic source term S_T is the thickness, or monopole-like, term. The monopole term can be imagined as an infinitesimally-small pulsating sphere. Physically, monopole-like behavior manifests from unsteady volumetric flow addition. The intensity of a monopole source is proportional to

$$I_{mono} = \rho \frac{L^2 V^3}{r^2} M, \quad (22.7)$$

where ρ is the fluid density, L^2 is the area of the noise emitter, r is the distance from the noise source, V is the fluctuating velocity, and M is the Mach number.

The second source term S_L is the loading, or dipole-like, term. The dipole term can be imagined as identical adjacent monopoles that are pulsating exactly out of phase, infinitesimally close together. Physically, dipole-like behavior manifests from unsteady momentum fluxes. Vortex shedding and flow separation result in dipole-like acoustic pressure sources, as does the noise radiated from a small obstruction in a turbulent crossflow. The intensity of a dipole source is proportional to

$$I_{dip} = \rho \frac{L^2 V^3}{r^2} M^3. \quad (22.8)$$

The third source term S_Q is the volume, or quadrupole-like, term. The quadrupole term can be envisioned as a square of four monopoles, each exactly out of phase with its two adjacent neighbors (*i.e.*, two of each phase), infinitesimally close together. Physically, quadrupole-like behavior manifests from the collision of fluid elements. Turbulent shear layers result in quadrupole-like acoustic pressure sources. The intensity of a quadrupole source is proportional to

$$I_{quad} = \rho \frac{L^2 V^3}{r^2} M^5. \quad (22.9)$$

At low subsonic Mach numbers, monopole-like sources typically dominate sound generation, followed by dipole-like sources, followed by quadrupole-like sources.

Ffowcs Williams-Hawkins Model

CONVERGE offers an advanced time integral solution of the Ffowcs Williams and Hawking equation described by [Casalino \(2003\)](#), building upon the retarded-time approaches of [Farrasat and Succi \(1983\)](#) and [Brentner \(1997\)](#). This will henceforth be termed the FW-H model.

This method calculates the sound generated within a control surface, then solves a wave propagation equation to a set of specified receiver points. The control surface can be a solid boundary, as in many legacy FW-H implementations, or a flow-through INTERFACE boundary some distance away from a solid body. A solid control surface takes into account the sources on and within the solid boundary, while a flow-through INTERFACE some distance from the solid boundary also takes into account the volume sources within the control surface. Receiver points are defined such that they have a position and velocity relative to the reference frame.

For each receiver point, CONVERGE separately calculates the contribution of the monopole and dipole sources within the control surface, then calculates the wave propagation to the receiver point. Due to the Mach number scaling described above, quadrupole sources are not included. The monopole contribution is calculated as

$$4\pi p'_T(\mathbf{x}, t) = \int_{f=0} \left[\frac{\rho_0 (\dot{U}_n + U_{\dot{n}})}{r (1 - M_r)^2} \right]_{ret} dS + \int_{f=0} \left[\frac{\rho_0 U_n (r \dot{M}_r + c (M_r - M^2))}{r^2 (1 - M_r)^3} \right] \quad (22.10)$$

where c is the far-field sound speed, ρ_0 is the far-field density, r is the distance to the receiver point, dots on quantities denote a time derivative, $[...]_{ret}$ indicates evaluation at the retarded time,

Chapter 22: Aeroacoustic Modeling

Reduced-Order Models Far-Field Methods

$$t_{ret} = t - \frac{|\mathbf{x} - \mathbf{y}(t_{ret})|}{c}, \quad (22.11)$$

\mathbf{x} the receiver position, \mathbf{y} the source position, and

$$\begin{aligned} U_i &= \left(1 - \frac{\rho}{\rho_0}\right) v_i + \frac{\rho u_i}{\rho_0} \\ P_{ij} &= (p - p_0) \delta_{ij} - \sigma_{ij} \\ L_i &= P_{ij} \hat{n}_j + \rho u_i (u_n - v_n) \\ U_n &= U_i \hat{n}_i \\ U_{\dot{n}} &= U_i \dot{\hat{n}}_i \\ \dot{U}_n &= \dot{U}_i \hat{n}_i \\ M_r &= M_i \hat{r}_i \\ \dot{M}_r &= \dot{M}_i \hat{r}_i. \end{aligned} \quad (22.12)$$

The dipole contribution is calculated as

$$\begin{aligned} 4\pi p_L'(\mathbf{x}, t) &= \frac{1}{c} \int_{f=0} \left[\frac{\dot{L}_r}{r(1-M_r)^2} \right]_{ret} dS \\ &\quad + \int_{f=0} \left[\frac{L_r - L_M}{r^2(1-M_r)^2} \right]_{ret} dS \\ &\quad + \frac{1}{c} \int_{f=0} \left[\frac{L_r (r \dot{M}_r + c(M_r - M^2))}{r^2(1-M_r)^3} \right]_{ret} dS, \end{aligned} \quad (22.13)$$

where

$$\begin{aligned} L_r &= L_i \hat{r}_i \\ \dot{L}_r &= \dot{L}_i \hat{r}_i \\ L_M &= L_i M_i. \end{aligned} \quad (22.14)$$

Specify [acoustics.in](#) > *acoustic_models* > *INTEGRAL_FWH* to activate this model. Specify far field properties, source details, and receiver details. CONVERGE writes output for each receiver to [acoustic_receiver <ID> stream <ID> pressure.out](#).

Near-Field Methods

Near-field reduced-order CAA models use solved turbulence quantities in the flow domain to identify noise sources. They cannot predict noise at distant receivers, and they can only predict broadband noise. They are the least computationally expensive approach to CAA modeling.

Proudman Model

The [Proudman model \(1952\)](#) approximates the generation of sound by isotropic turbulence by assuming a high Reynolds number, a low Mach number, and zero mean motion. The acoustic power per unit volume thus generated, P_A , is given by

$$P_A = \alpha \rho_o \frac{u_{RMS}^3}{l} \frac{u_{RMS}^5}{c^5}, \quad (22.15)$$

where α was proposed to be approximately 38 and c is the speed of sound. Within a RANS framework, we can restructure this equation based on our calculated turbulence variables k (the turbulent kinetic energy) and ϵ (the turbulent kinetic energy dissipation rate),

$$P_A = \alpha_\epsilon \rho \epsilon \left(\frac{\sqrt{2k}}{c} \right)^5, \quad (22.16)$$

where the model constant $\alpha_\epsilon = 0.1$ based on the DNS of [Sarkar and Hussaini \(1993\)](#). When this model is active, CONVERGE writes the acoustic power (either in W/m^3 or in dB/m^3) to [post*.h5](#). Specify [acoustics.in](#) > *acoustic_models* > *BROADBAND_PROUDMAN* to activate this model.

Curle Model

The [Curle model \(1955\)](#) approximates the generation of sound by flow over a solid surface. It is a special case of the Ffowcs Williams and Hawking analogy, making the additional assumption that the outer surface S is rigid and does not move. The generated acoustic power per unit area, P_A , is given by

$$P_A = \frac{B\rho}{c^3} \left(\frac{\tau \mu_t}{\rho \mu \sqrt{C_\mu}} \right)^3, \quad (22.17)$$

where ρ is the fluid density at the surface, c is the speed of sound, τ is the shear stress at the wall, μ and μ_t are the molecular and turbulent viscosities, and C_μ is the turbulence model

Chapter 22: Aeroacoustic Modeling

Reduced-Order Models Near-Field Methods

constant. In CONVERGE, the model constant B has a value of 6.6e-8 based on calibration by [Croaker et al.](#) (2011). When this model is active, CONVERGE writes the acoustic power (either in W/m^2 or in dB/m^2) to [*post*.h5*](#). Specify [*acoustics.in*](#) > *acoustic_models* > *BROADBAND_CURLE* to activate this model.

Chapter



23

Utilities

23 Utilities

CONVERGE includes a variety of utilities that complement the CONVERGE CFD solver.

23.1 Pre-Processing

CONVERGE offers utilities that can be used to pre-process input or data files prior to running a simulation.

Make Surface

The *make_surface* utility in CONVERGE (also available as the Make Surface tool in CONVERGE Studio) will automatically configure the [surface geometry file](#) (e.g., *surface.dat*) for engine sector cases. You may wish to simulate an axisymmetric engine sector, an example of which is shown below, rather than a full engine to reduce computational time.

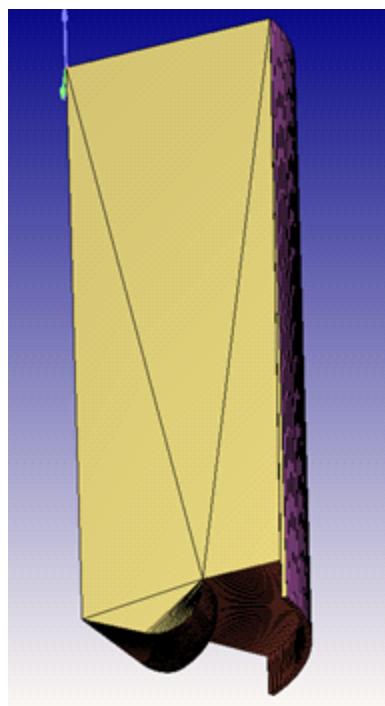


Figure 23.1: An engine cylinder sector geometry created automatically by the *make_surface* utility.

In sector cases, you need to specify values for the velocity and turbulence fields at the start of the calculation. To determine accurate velocity and turbulence field values, run CONVERGE with intake ports and valves included in the geometry from the intake valve opening time until all valves are closed. This simulation will model the complete induction event and give you accurate velocity and turbulence fields. Then use these velocity and turbulence values to create a *map*.h5* file to initialize the sector case via [mapping](#).

Although you can prepare a sector geometry in a CAD program or manually in CONVERGE Studio (*i.e.*, without using the Make Surface tool), the *make_surface* utility or the Make

Chapter 23: Utilities

Pre-Processing Make Surface

Surface tool will yield an engine sector geometry that is defect-free, boundary-flagged, and periodic face-matched.

The *make_surface* utility requires the [*makesurface.in*](#) file. Once you have prepared this file, save it to the Case Directory. From the command line, go to the Case Directory and enter

```
make_surface
```

to direct CONVERGE to generate *surface.dat*.

Mesh Convert

The *mesh_convert* utility in CONVERGE (also available in CONVERGE Studio) can convert meshes from CGNS (version 4.2.0, 3.3.0, or earlier) or OpenFOAM format to *surface.dat*.

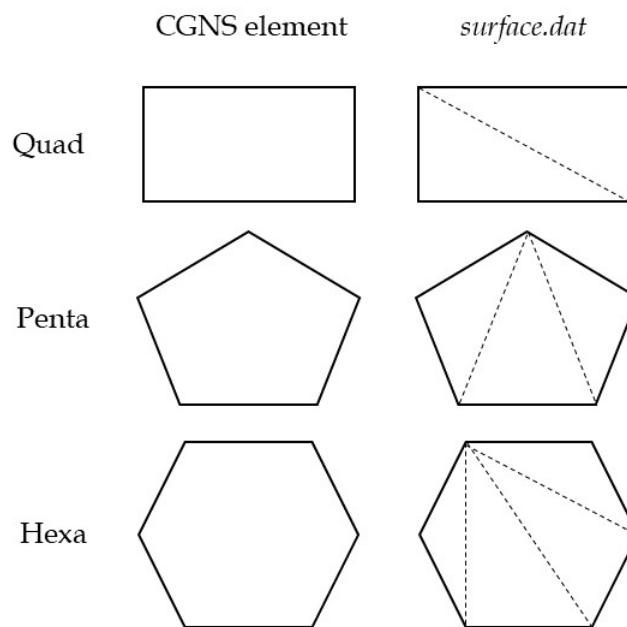


Figure 23.2: Example of the *mesh_convert* utility that creates a triangulated *surface.dat* from a CGNS mesh.

To convert a CGNS mesh to *surface.dat*, include the CGNS grid file in your Case Directory, and type the following command:

```
./mesh_convert --cgns --grid=grid_c.cgns --write_internal_elements=[0 or 1]  
--tag_boundaries=[0 or 1]
```

To convert an OpenFOAM mesh to *surface.dat*, a polyMesh directory that includes files for points, boundaries, and faces, is required. Enter the following command:

Chapter 23: Utilities

Pre-Processing Mesh Convert

```
./mesh_convert --foam --write_internal_elements=[0 or 1] --
tag_boundaries=[0 or 1]
```

If you want only the external triangles written to *surface.dat*, set *write_internal_elements*=0. If *write_internal_elements*=1, *surface.dat* will keep all of the triangles, and *surface_inlaid.dat* will have only internal triangles.

If you have tagged boundary elements in your starting file and want to see them in CONVERGE Studio, set *tag_boundaries*=1. If *tag_boundaries*=0, the external triangles will be unassigned and the interior triangles will be assigned to boundary ID 1.

Extract Profile

The *extract_profile* utility can extract a bowl or head profile from almost any geometry. This utility is useful for a geometry in which the bowl or head does not have straight radial lines along which to manually copy points for the profile.

To use the *extract_profile* utility, enter

```
<extract_profile executable> <file name>
```

at the command prompt. For example, if you are using the 64-bit Linux executable and your surface geometry file name is *surface.dat*, enter

```
extract_profile_linux_64 surface.dat
```

You must provide a geometry that is centered at $z = 0$. Any cut-plane through the geometry must form a closed loop. You will have to remove ports and valves from the geometry to form a closed loop.

When you execute *extract_profile*, it will obtain the x and z coordinate data from points along a straight line in the bowl or head, as shown below in Figure 23.3.

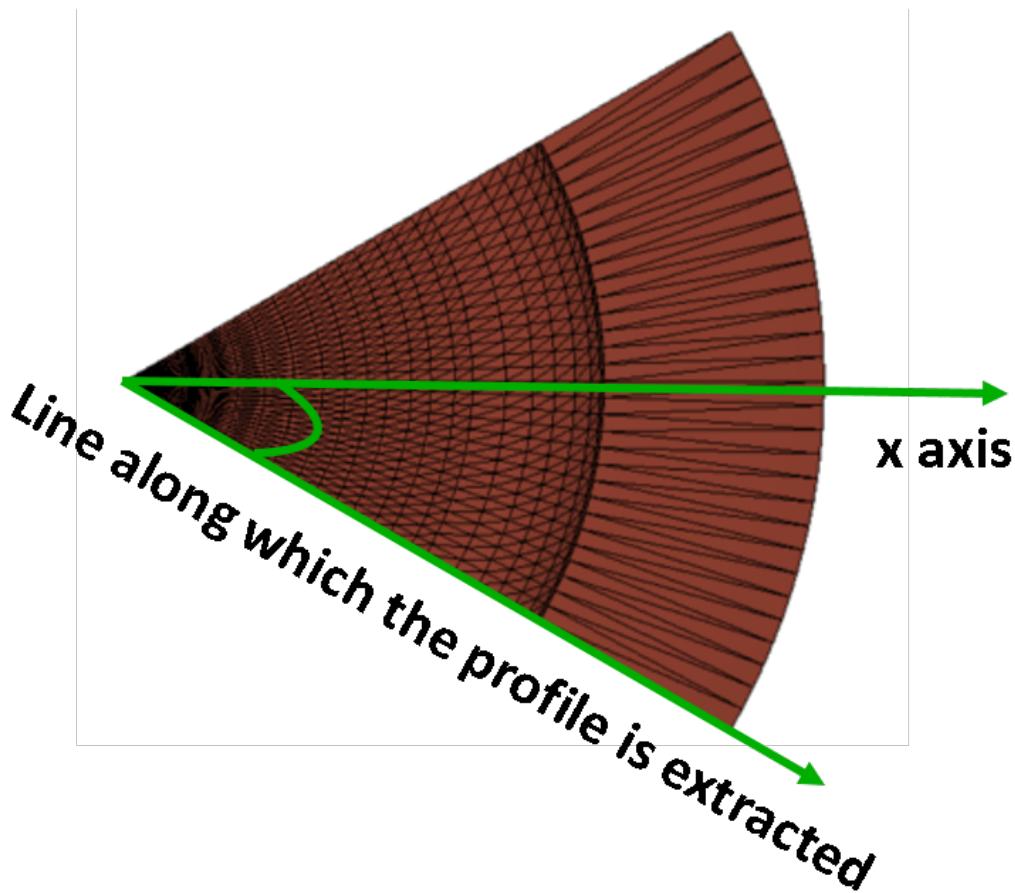


Figure 23.3: The bottom green line shows the line along which `extract_profile` obtains the x and z coordinate data for the profile.

Once you have obtained the x and z coordinates of the bowl or head profile, follow the instructions for running the [`make_surface`](#) utility to construct the complete geometry. Figure 23.4 shows a view of the extracted bowl profile in the xz plane. Figure 23.5 shows the complete bowl shape rendered in CONVERGE Studio.

Chapter 23: Utilities

Pre-Processing Extract Profile

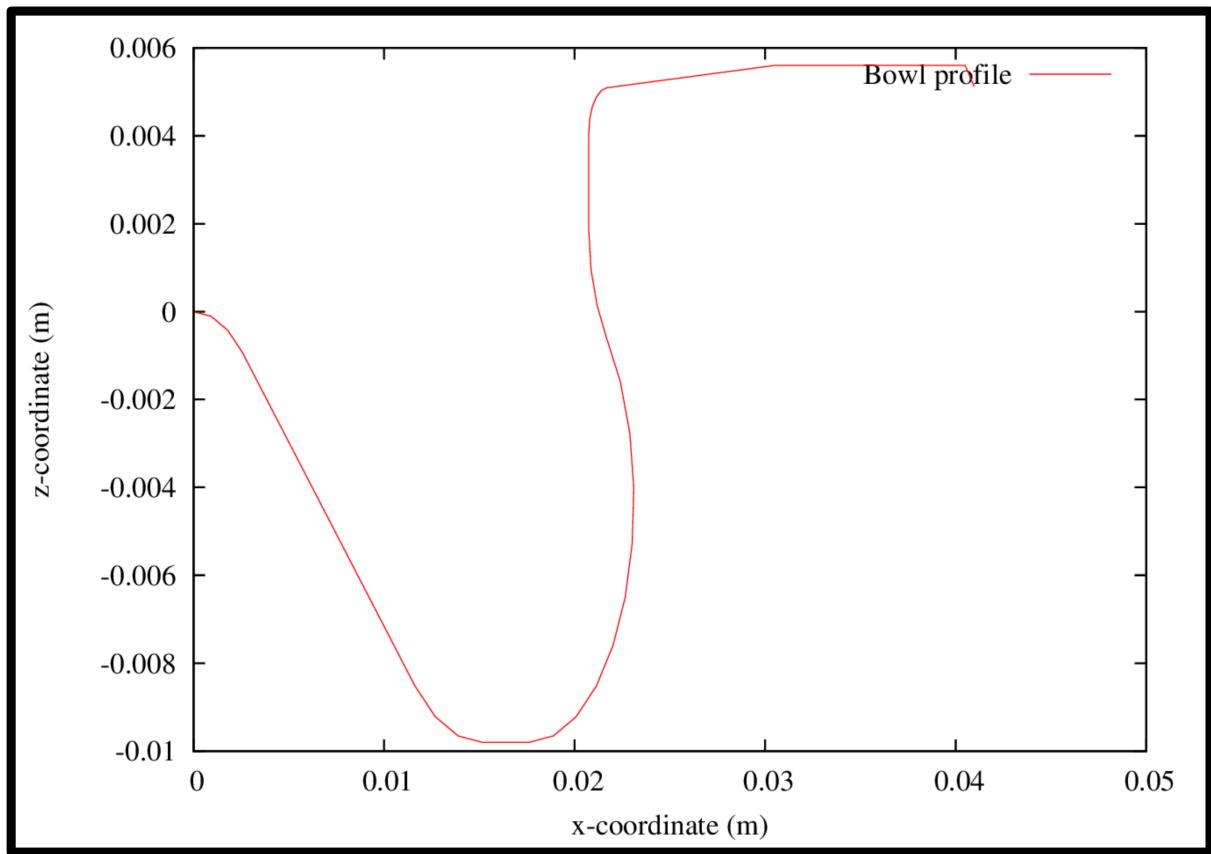


Figure 23.4: The extracted bowl profile in the xz plane.

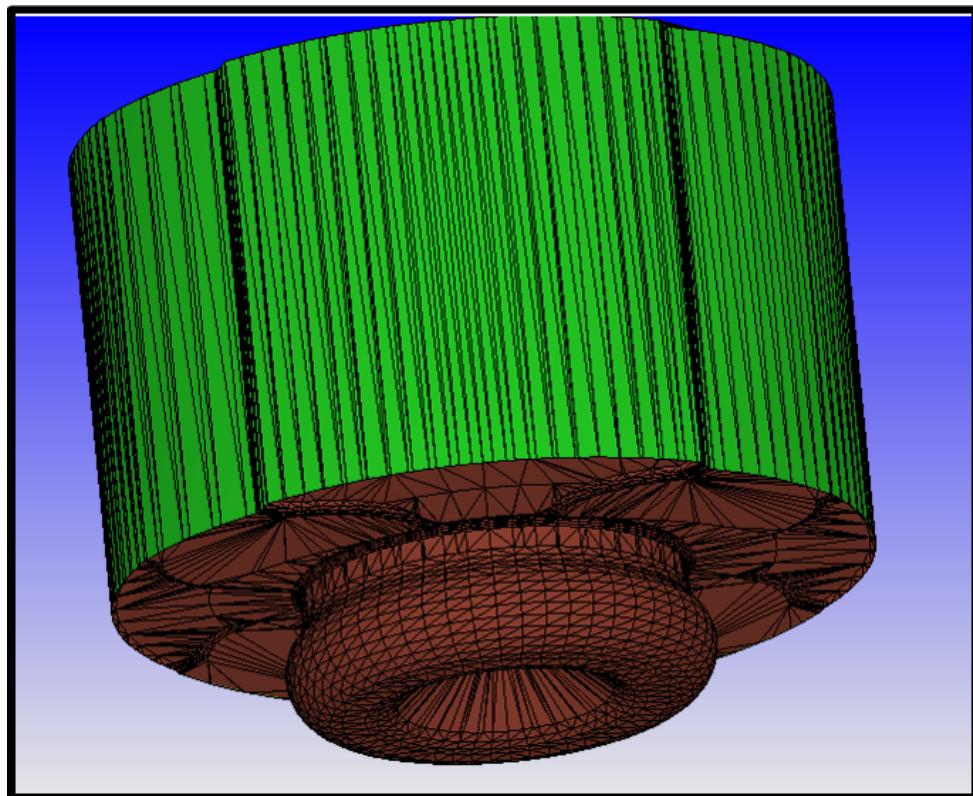


Figure 23.5: The complete geometry including the bowl shape rendered in CONVERGE Studio.

Thermodynamic File Cleanup

Use the *cleantherm* utility to condense your [thermodynamic data file](#) so that it lists only the species contained in the reaction mechanism file. To run this utility, go to your Case Directory and type the following command:

```
<CONVERGE executable> -u cleantherm
```

where `<CONVERGE executable>` is any CONVERGE executable (*e.g.*, converge-mpich).

CONVERGE will read the thermodynamic data and reaction mechanism files in the Case Directory and generate a *therm_cleaned.dat* file that contains only the species listed in the reaction mechanism file.

Mechanism File Cleanup

Use the *cleanmech* utility to condense your [reaction mechanism data file](#) so that it contains only the data required by CONVERGE. To run this utility, go to your Case Directory and type the following command:

```
<CONVERGE executable> -u cleanmech
```

where `<CONVERGE executable>` is any CONVERGE executable (*e.g.*, converge-mpich).

Chapter 23: Utilities

Pre-Processing Mechanism File Cleanup

CONVERGE will read the [reaction mechanism data file](#) in the Case Directory and generate a *mech_cleaned.dat* file that does not contain the comments and verbosity of the [reaction mechanism data file](#).

Transport File Cleanup

Use the *cleantransport* utility to condense your [transport data file](#) so that it lists only the species contained in the reaction mechanism file. To run this utility, go to your Case Directory and type the following command:

```
<CONVERGE executable> -u cleantransport
```

where <CONVERGE executable> is any CONVERGE executable (e.g., converge-mpich).

CONVERGE will read the [transport data file](#) and the [reaction mechanism data file](#) in the Case Directory and generate a *transport_cleaned.dat* file that contains only the species listed in the [reaction mechanism file](#), and does not contain comments.

Fluid Property Calculator

CONVERGE includes a utility that serves as an interface for the *CoolProp* fluid property library. Use this utility to compute custom tabulated properties to use as inputs for a CONVERGE simulation. Because the interface utility connects to the online *CoolProp* library, you must have a working internet connection.

In the [fluid_properties.in](#) file specify the fluid name and ranges of temperature (in K) and pressure (in MPa) for which to calculate fluid properties such as enthalpy, entropy, specific internal energy, and compressibility. Enter steps for temperature and pressure. The *CoolProp* interface utility will calculate the desired properties at intervals equal to the value for the step within the specified range.

To run the utility, execute the command `coolprop` from the directory that contains your [fluid_properties.in](#) file. The utility will access *CoolProp*, determine the fluid properties for the range of temperatures and pressures specified in [fluid_properties.in](#), and write the properties to [fluid_properties.dat](#).

You can use [fluid_properties.dat](#) in place of [liquid.dat](#) for a liquid simulation or in place of [gas.dat](#) and a [thermodynamic data file](#) (e.g., *therm.dat*) for a gas simulation. Set [inputs.in](#) > *property_control* > *tabular_fluid_prop_flag* = 1 and ensure that [inputs.in](#) > *property_control* > *real_gas_prop_flag*, [inputs.in](#) > *solver_control* > *species_solver*, and [inputs.in](#) > *property_control* > *llv_flag* are all set to 0. CONVERGE will take the fluid properties from [fluid_properties.dat](#) instead of calculating fluid properties via the equation of state.

Extract Sub-Mechanism

Use the *submech* utility to extract a smaller reaction mechanism from a larger reaction mechanism based on a list of species that you want to keep. The species must be listed in a *species_list.in* file, formatted as shown below in Figure 23.6. CONVERGE will extract the

Chapter 23: Utilities

Pre-Processing Extract Sub-Mechanism

listed species and associated reactions from the original [reaction mechanism file](#) and write them to a new [reaction mechanism file](#).

```
SPECIES_LIST
CH4
CO
CO2
H
H2O
N2
O2
OH
...
END
```

Figure 23.6: An example *species_list.in* file.

Save the *species_list.in* file to the Case Directory along with the original [reaction mechanism file](#) (*mech.dat*) and [thermodynamic data file](#) (*therm.dat*). To run the utility, enter

```
<CONVERGE executable> -u submech
```

where <CONVERGE executable> is any CONVERGE executable (e.g., *converge-mpich*).

CONVERGE will write the extracted reaction mechanism to *outputs_original/mech_ske.dat*.

23.2 Post-Processing

CONVERGE contains utilities to assist in post-processing simulation data.

Post Convert

To prepare cell-based output files for three-dimensional visualization, use the *post_convert* utility to convert the cell-by-cell output data into an appropriate format for Tecplot, FieldView, or other programs. You can also use this utility to create cell-by-cell data in column format in text files for selected variables. CONVERGE Studio contains an analogous *Post Convert* tool.

In the command line, go to the *output* subdirectory within your Case Directory and then type *post_convert*. CONVERGE will prompt you to:

1. Specify a case name, which will replace *post* in the file name for each of the output files that are written to the *output* directory. The original *post*.h5* files will remain unchanged.
2. Select the output type from the following options:
 - Tecplot for CONVERGE and Tecplot 360
 - General Mesh Viewer (GMV)
 - EnSight
 - Column format
 - Column format (select only one region)
 - FieldView
 - ParaView VTK inline binary format

- ParaView VTK ASCII format
- Average and RMS of selected post output files
- EnSight for GT-SUITE

Tecplot for CONVERGE and Tecplot 360 Format

- Select `yes` for boundary surface output. For many features, Tecplot's performance is enhanced with boundary surface output.
- Note that CONVERGE will add the prefix `DP_` to parcel-related parameters to distinguish them from fluid parameters for Tecplot.
- Note that older versions of `post_convert` can output only Tecplot 360-formatted files. These cannot be opened in Tecplot for CONVERGE. If you are having trouble opening Tecplot for CONVERGE output files, confirm that you are running the latest version of `post_convert`. Tecplot 360 can open Tecplot for CONVERGE-formatted files.
- Note that Tecplot for CONVERGE and Tecplot 360 version 2019 R1 and later can read CONVERGE `post*.h5` files directly. You may see minor performance benefits from converting to a Tecplot `*.plt` file, but for general use we recommend loading the `post*.h5` file directly into Tecplot.

Average and RMS of Selected Post Output Files

- Use this option to generate average and RMS values for velocity, pressure, temperature, vorticity, and other parameters. This option can also produce average and RMS values for a specific portion of the simulation.
 - Before running a simulation, you must include the following parameters in the `post.in` file: `logic_ijk` and `volume`. Otherwise CONVERGE cannot generate the average and RMS values.
 - The averaged information will be written to `post*.avg.h5`. Corresponding RMS quantities will be written to `post*.rms.h5`. The time and run number in the averaged and RMS file names will match the first file selected.
 - We recommend averaging data from simulations in which the geometries are the same. The cells in the averaged file will be based on the cells from the first file selected, so, if the geometries in the selected files are not the same, you should verify that the first file selected contains data from the largest geometry.
 - Averaging will not work with multi-stream cases, such as cases that employ conjugate heat transfer modeling.
3. Select which output files to convert. The utility lists all `post*.h5` files in a numbered list, an example of which follows:

```
[1] : post000061_-6.483057e+01.h5
[2] : post000062_-6.379913e+01.h5
[3] : post000063_-6.281712e+01.h5
[4] : post000064_-6.191693e+01.h5
[5] : post000065_-6.085729e+01.h5
```

The utility will prompt you for the files to convert. To select specific file(s), enter the file number(s) (e.g., in the example above, enter `1,3-5` to convert all files except [2]). To select files at a specified frequency, enter the number of the first file in the range, the

selection frequency, and the number of the last file in the range, separated by colons, (e.g., in the example above, enter `1:2:5` to convert files [1], [3], and [5]). You can also enter `all`, `first`, or `last`.

4. Select which variables to convert. The `post_convert` utility will list all of the variables in the first file to be converted and prompt you to select the variable(s) to be included in the converted files. Enter the variable numbers.
5. Select which parcel variables to convert. If the simulation contained parcels, the `post_convert` utility will prompt you to select the parcel variable(s) to be included in the converted files.

The `post_convert` utility automatically saves a `convert.in` file in the output directory. This text file contains the conversion information that you just entered. You can use this file to expedite your next `post_convert` process. For example, you may wish to repeatedly examine the latest output of a simulation that is currently running. Work through the prompts the first time that you use `post_convert` (and be sure to select the `last` file to convert). Thereafter, simply type `post_convert convert.in` to repeat the conversion process.

CONVERGE offers a limited capability to run `post_convert` in parallel from the command line. A parallel post-processing computation requires the same memory per core as a serial computation, but it runs much faster. The speed improvement is dependent on your network's speed and file system performance. You can expect to see good parallel `post_convert` performance on ten to twenty cores.

Like CONVERGE, the [parallel implementation](#) of `post_convert` is built on the Message Passing Interface (MPI) package.

Post Convert Version Check

Older versions of `post_convert` may not support all current functionality. To check the version of `post_convert` installed on your system, type `post_convert -v` or `post_convert --version` from the command line. We generally recommend replacing your installation of `post_convert` every time you install a new version of CONVERGE. Figure 23.7 shows an example `post_convert` version output.

```
post_convert -v

POST_CONVERT_30
Convergent Science Inc.
All rights reserved.

Build ID:      bef9779
Build Type:    RELEASE Build
Build Date:   Feb 25 2020
```

Figure 23.7: Sample `post_convert` version output.

HTC Mapping

You can use the heat transfer mapping (*htc_map*) utility to time-average near-wall data from a CONVERGE simulation and map the result to another surface file. For example, you might use *htc_map* to provide boundary conditions for a heat transfer analysis with a finite element analysis (FEA) code.

You can also use *htc_map* to transform a surface temperature file between two coordinate systems. The [*htc_map.in*](#) section in [Chapter 25 - Input and Data Files](#) describes this option.

To run *htc_map*, CONVERGE requires several files in the Case Directory: [*transfer.out*](#), [*htc_map.in*](#), and *htc_surface.stl* (the surface file geometry for finite element analysis), as well as the *htc_map* executable.

The two primary output files for heat transfer mapping are [*htc_triangles.map*](#) and [*htc_vertices.map*](#). CONVERGE also provides three-dimensional output for data visualizations in Tecplot, GMV, EnSight, and FieldView. These output files provide node data for each variable contained in [*transfer.out*](#).

Mapping CONVERGE Data to the FEA Solver

Typically the triangulation of a surface for FEA is different from the surface triangulation used in CONVERGE. The FEA surface will have higher resolution (*i.e.*, more triangles) in areas most relevant to the heat transfer analysis.

CONVERGE uses a search algorithm (*direct hits*, *grown points*, *neighbors*) to map the triangles in the [*transfer.out*](#) file to the FEA surface file. *Direct hits* indicate a CONVERGE grid point mapped directly to a FEA surface triangle. If the grid resolution in CONVERGE is fine, there may be multiple surface triangles mapped to a single FEA surface triangle. In this case, CONVERGE will use area-weighted averaging of the grid triangles to map onto the FEA triangle. If small FEA triangles have no corresponding grid points, CONVERGE initializes them by *neighbors* (neighboring cells). CONVERGE uses the *grown points* algorithm to initialize triangles that are not initialized by *direct points* or *neighbors*. These triangles are initialized with values that grow from neighbor triangles.

Note that the boundary IDs between the original surface triangle and the mapped FEA triangle must be consistent. CONVERGE will not map FEA triangles with inconsistent boundary IDs. Although it is not recommended, you can disable this requirement by setting [*htc_map.in* > *enforce_boundID_match* = 0](#).

After mapping, there may be triangles in the FEA surface file that are not initialized with CONVERGE boundary data. Uninitialized triangles may be the result of mistakes (*e.g.*, if the FEA surface is not aligned with the CONVERGE surface or if boundary IDs do not match). Alternatively, uninitialized triangles may be a normal result due to moving surfaces. For example, a moving piston will block portions of the liner, which will then remain uninitialized.

At any time-step when the number of uninitialized triangles is greater than zero, CONVERGE creates a *uninit_triangles_complete.dat* file. You can import both the FEA surface file and *uninit_triangles_complete.dat* into CONVERGE Studio, where the uninitialized triangles will be shown with red vertices.

Cyclic Averaging of Heat Transfer Data

In multi-cycle simulations, such as for internal combustion engines, CONVERGE calculates cyclic averages of heat transfer data that are spatially resolved for the FEA surface file.

The *transfer.out* file includes data at the interval specified by *inputs.in* > *output_control* > *twrite_transfer*. The data from one engine cycle (except the first set of data) are averaged to yield a single file with heat transfer data spatially resolved for the FEA surface file. For a four-stroke engine, the cycle is 720 *crank angle degrees*.

CONVERGE calculates the cycle-averaged heat transfer coefficient, \bar{h} , and the fluid temperature, \bar{T}_g , using the following equations:

$$\bar{h} = \frac{1}{t} \sum_i^t h_i \cdot \Delta t \quad (23.1)$$

and

$$\bar{T}_g = \frac{\sum_i^t h_i \cdot T_{g,i} \cdot \Delta t}{\sum_i^t h_i \cdot \Delta t}. \quad (23.2)$$

In these equations, h_i and $T_{g,i}$ are the instantaneous heat transfer coefficient and fluid temperature for each point at each *twrite_transfer* time interval (Δt), where t is in *seconds* or *crank angle degrees*.

CONVERGE averages these values in this manner to ensure that the calculated heat transfer is the same as the integration of the instantaneous values over the cycle. Note that the average fluid temperature calculated with this method is biased toward the times when the heat transfer coefficient is high, but, when multiplied by average heat transfer coefficient, it gives the calculated heat transfer. In the situation in which a triangle is not initialized during a crank angle interval (for instance, when the liner is blocked by the piston), the heat transfer coefficient is zero.

Proper Orthogonal Decomposition

The proper orthogonal decomposition (POD) is a spectral analysis technique used primarily to separate the large-scale and small-scale elements of turbulent flow (e.g., in a large eddy

Chapter 23: Utilities

Post-Processing Proper Orthogonal Decomposition

simulation). CONVERGE contains a utility that calculates the POD of arbitrary flow variables over arbitrary *post*.h5* files.

In the command line, go to the *output* directory and then type `pod`. CONVERGE will prompt you to:

1. Specify the file type to export.
2. Specify which files you want to include. The utility lists all *post*.h5* files in a numbered list, an example of which follows:

```
[1]: post000061_-6.483057e+01.h5
[2]: post000062_-6.379913e+01.h5
[3]: post000063_-6.281712e+01.h5
[4]: post000064_-6.191693e+01.h5
[5]: post000065_-6.085729e+01.h5
```

The utility will prompt you for the files to convert. To select specific file(s), enter the file number(s) (*e.g.*, in the example above, enter `1,3-5` to convert all files except [2]). To select files at a specified frequency, enter the number of the first file in the range, the selection frequency, and the number of the last file in the range, separated by colons, (*e.g.*, in the example above, enter `1:2:5` to convert files [1], [3], and [5]). You can also enter `all`, `first`, or `last`.

3. Select which variables to convert. The *pod* utility will list all of the variables in the first file to be converted and prompt you to select the variable(s) to be included in the converted files. Enter the variable numbers. Note that the cell variables *logic_ijk*, *level*, and *volume* are required for the POD calculation.

The *pod* utility create a [*pod.in*](#) file in the current directory. By default, *pod* will calculate as many modes as there are output files, which accounts for the entire spectral content in the domain. You can optionally output fewer POD modes by changing the value of [*pod.in* > pod modes output](#). To run the utility with this altered input file, type `pod pod.in`. You will not be prompted for additional inputs.

The utility will perform the necessary calculations in serial and output the results to a new sub-directory called *pod_results* (note this will overwrite a previous *pod_results* directory). There are several types of output files. There is a single [*energy_fraction.out*](#) file. This file contains the fraction of content contained in each mode of each variable selected, except for the required *logic_ijk*, *level*, and *volume* variables. Note that even if you selected fewer POD modes in [*pod.in*](#), the utility will calculate and print the energy fraction over the entire set of possible modes.

The *pod_results* directory will also contain a set of *postPODmod_mode<num>.h5* mode output files. These files are formatted identically to the *post*.h5* files that are read in by the *pod* utility, and you can post-process them with *post_convert*. The *postPODmod_mode00000.h5* file is the flow mean.

Optionally, you can direct the *pod* utility to reconstruct the flow from a set of *postPODmod_mode<num>.h5* files. Specify the number of modes used in the reconstruction calculation in [*pod.in*](#). By default, this *pod modes output* parameter is set to zero, which directs the utility to not perform the modal reconstruction. If you set this parameter to a positive integer, the utility will generate a set of corresponding *postPODmod_recon<num>.h5* files. These files are formatted identically to the *post*.h5* files, and you can post-process them with *post_convert*.

You can perform the POD analysis on slices or volumes, although the volume calculation may be quite slow. The runtime of the *pod* utility is approximately linear in the number of *post*.h5* files.

Sensitivity Output Convert

The sensitivity coefficient matrix (contained in the [*sens<case ID>.out*](#) files, which are written by the [0D chemistry utility](#) when *zero_d_solver.in* > *zero_d_sensitivity_control* > *sensitivity_flag* = FORWARD) contains sensitivity values for all species and for the temperature for each reaction in a sensitivity case. Use the *sens_convert* utility to normalize the sensitivity coefficient matrix values by the maximum Z_j value for the entire simulation.

In the command line, go to the *output* directory and type *sens_convert*. CONVERGE will prompt you as follows:

1. Specify which output files to convert. This utility lists all *sens** files in a numbered list, an example of which follows:

```
[1]: sens0.out  
[2]: sens1.out  
[3]: sens2.out  
[4]: sens3.out  
[5]: sens4.out
```

The utility will prompt you for the files to convert. To select specific file(s), enter the file number(s) (e.g., in the example above, enter 1, 3-5 to convert all files except [2]). To select files at a specified frequency, enter the number of the first file in the range, the selection frequency, and the number of the last file in the range, separated by colons, (e.g., in the example above, enter 1:2:5 to convert files [1], [3], and [5]). You can also enter *all*, *first*, or *last*.

2. Specify which variable(s) (*i.e.*, species and/or temperature) to convert. The utility lists all species variables and then the temperature variable in a numbered list, an example of which follows:

```
[1]: O2  
[2]: N2  
[3]: CO2  
[4]: H2O  
...  
[46]: C2H2  
[47]: CH2CO  
[48]: HCCO  
[49]: TEMP
```

Chapter 23: Utilities

Post-Processing Sensitivity Output Convert

The utility will prompt you for the variables to post-process. Note that the species variables and variable numbers in this list are identical to those written to [*species_info.dat*](#) during forward sensitivity analysis and that the list ends with TEMP. To select specific file(s), enter the variable number(s) (e.g., in the example above, enter 1, 48-49 to post-process O₂, HCCO, and TEMP).

The utility creates *output<case ID>* folders, each of which contains three files for each variable:

[*sens<case ID> var<number> <name>.out*](#),
[*sens<case ID> var<number> <name> neg.out*](#), and
[*sens<case ID> var<number> <name> pos.out*](#),

where <case ID> is the output folder name and the file name represents the order of the 0D cases in [*zero_d_cases.in*](#), <number> represents the variable number chosen in step 2 above, and <name> is the variable name (either the species name or TEMP for temperature).

23.3 Chemistry

CONVERGE offers a variety of chemistry-related utilities, which allow you to study reacting systems, manipulate mechanisms, and generate data tables needed for some simulations. Unlike a 3D CONVERGE simulation, a chemistry simulation does not need boundary information; a surface geometry file; or parameters for turbulence, spray, combustion, or other physical models.

Zero-Dimensional Chemistry Utilities

CONVERGE offers zero-dimensional (0D) chemistry utilities that can perform a variety of actions:

- [Constant volume and constant pressures 0D \(autoignition\) simulations](#) including [generation of a TKI table](#)
- [Forward sensitivity analysis](#) and [adjoint sensitivity analysis](#)
- [Variable volume](#) calculations
- [0D homogeneous charge compression ignition \(HCCI\) engine simulations](#)
- [Fuel research octane number \(RON\) and motor octane number \(MON\) estimation](#)
- [Plug flow reactor \(PFR\)](#) analysis
- [Well-stirred reactor \(WSR\)](#) analysis
- [Chemical equilibrium \(CEQ\)](#) calculations

These 0D simulations are single-cell calculations without CFD boundary conditions, and there is no variation in space.

The format for a serial execution of a 0D calculation is:

```
<CONVERGE executable> -u zerod
```

where <CONVERGE executable> is any CONVERGE executable (e.g., converge-mpich).

A sample command for a parallel execution (using 16 processors) of a 0D case is:

```
mpirun -n 16 converge-mpich -u zerod
```

Inputs for 0D Chemistry Utilities

All 0D chemistry simulations require a [thermodynamic data file](#) (e.g., *therm.dat*), a [reaction mechanism file](#) (e.g., *mech.dat*) that contains reaction data (*i.e.*, not only element and species data), and [zero_d_solver.in](#). A 0D cases file (either [zero_d_cases.in](#) or [zero_d_template.in](#)) is almost always necessary.

The numerics-related parameters for the 0D chemistry utility are located in [zero_d_solver.in](#). These controls include the nature of the solver, the tolerance values, and the input and output format (mass or mole fraction). Note that the time for integration specified via [zero_d_solver.in](#) > [zero_d_solver_control](#) > [case_time_limit](#) may not be the numerical upper time limit for integration, especially for ignition delay cases in which the specified time is much greater than the ignition delay. CONVERGE cuts off this upper time limit at the moment when the temperature no longer increases (close to equilibrium).

The initial conditions for temperature, pressure, and species are specified in [zero_d_cases.in](#) (or [zero_d_template.in](#)). These parameters are repeated for each case in [zero_d_cases.in](#). Note that CONVERGE re-normalizes the mole fractions internally, so the sum of the mole fractions of the reactants specified in [zero_d_cases.in](#) does not need to be one.

We recommend setting up 0D chemistry input files in the *Chemistry* module in CONVERGE Studio.

Constant Pressure or Volume Autoignition

Ignition delay in an engine is defined as the time between the start of injection and the start of combustion, which is indicated in a simulation as a detectable heat release. CONVERGE allows the combustible species to react exothermically in a constant volume bomb, a constant pressure reactor, or a constant temperature and pressure reactor, depending on the *case_type*. The time required for the temperature to increase by 400 K from its initial value is defined in CONVERGE as ignition delay, which is shown below in Figure 23.8.

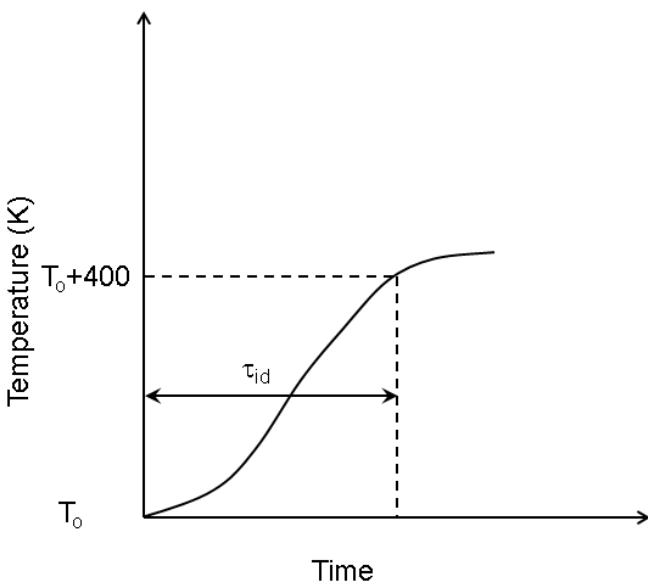


Figure 23.8: Ignition delay calculation in CONVERGE.

CONVERGE can optionally calculate the double ignition delay. When performing a double ignition delay calculation, the first ignition delay is the time at which the first derivative of temperature with respect to time is maximum and the second ignition delay is the time required for the temperature to increase by 400 K from its initial value.

You can use the zero-dimensional chemistry utility to generate ignition delay data for different combinations of temperature, pressure, and equivalence ratio of the fuel mixture. To calculate ignition delay data via the 0D chemistry utility, you must first save input data in two files: [zero_d_cases.in](#) (or [zero_d_template.in](#) if performing [TKI table generation](#), which is discussed in more detail below) and [zero_d_solver.in](#).

Set `zero_d_solver.in > zero_d_input_control > case_type` to `CONSTANT_VOLUME`, `CONSTANT_PRESSURE`, or `CONSTANT_TEMPERATURE_PRESSURE`. CONVERGE will use [zero_d_cases.in](#) and [zero_d_solver.in](#) to generate ignition delay data in [ignition_det.out](#). If you would like to calculate the double ignition delay, set `zero_d_solver.in > zero_d_output_control > double_ignition_delay_flag = 1`. In order to generate `zero_d_sol_case<case ID>.out` files, which contain detailed information for each case, set `zero_d_solver.in > zero_d_output_control > output_file_flag = 1`.

TKI Table Generation

To generate a tabulated kinetics of ignition (TKI) table file (`tki_table.h5`), which can be used for an [ECFM/ECFM3Z](#) simulation, set `zero_d_solver.in > zero_d_output_control > generate_tki_table_flag = 1` and use [zero_d_template.in](#) to configure the TKI table values and ranges. You can also use CONVERGE Studio to generate a TKI table file. See the CONVERGE Studio 3.1 Manual for more information.

To use a TKI table, set `combust.in > ecfm_model > auto_ignition > active = 1` (if using the ECFM model) or `combust.in > ecfm3z_model > auto_ignition > active = 1` (if using the ECFM3Z model) and then specify the TKI file name (e.g., `tki_table.h5`) via `combust.in > ecfm_model > auto_ignition > tki_table_filename` or `combust.in > ecfm3z_model > auto_ignition > tki_table_filename`, respectively.

Zero-Dimensional Forward Sensitivity Analysis

Given a general chemical reaction



we can define the total reaction rate, R , as

$$R = P(k_f c_A c_B - k_r c_C) , \quad (23.4)$$

in which k_f and k_r are the forward and reverse reaction rate constants, respectively; P is the sensitivity parameter; and c is the concentration of the reaction species. As the sensitivity parameter changes, the evolution of species concentration and temperature also changes. CONVERGE quantifies the rate at which the concentrations and temperature change with respect to the change in the sensitivity parameter as the sensitivity coefficient, S_{ij} , and calculates it as

$$S_{ij} = \frac{dZ_j}{dP_i} \quad (23.5)$$

in which Z_j is a matrix of all concentration and temperature output values from a 0D simulation. The 0D sensitivity analysis utility ranks the sensitivity coefficients for each case that is listed in `zero_d_cases.in`. If you know the sensitivity coefficient for each reaction, you can quantify how each reaction affects the resulting concentrations and temperature. You can then adjust the reaction pre-exponential factors, and thus the reaction rate, in the `reaction mechanism file` to improve your simulation results so that they more accurately reflect your data.

To activate forward sensitivity analysis (FSA) in CONVERGE, set `zero_d_solver.in > zero_d_sensitivity_control > sensitivity_flag = FORWARD`. FSA can be used only with constant pressure or constant volume cases (i.e., when `zero_d_solver.in > zero_d_input_control > case_type` is set to `CONSTANT_PRESSURE` or `CONSTANT_VOLUME`). To use FSA, you must deactivate TKI table generation (i.e., set `zero_d_solver.in > zero_d_output_control > generate_tki_table_flag = 0`). When FSA is active, it applies to all of the 0D cases with a case type of either constant pressure or constant volume (i.e., you cannot apply FSA to only certain cases in `zero_d_cases.in`). When you [run a 0D simulation](#) with sensitivity analysis, CONVERGE writes the sensitivity coefficient matrix at each time-step to `sens<case ID>.out`,

where `<case ID>` corresponds to the order of 0D cases from [`zero_d_cases.in`](#). Sensitivity analysis runs on all species and the temperature for each reaction in a sensitivity case. If the reaction mechanism is large, sensitivity analysis will be computationally expensive.

To post-process the [`sens<case ID>.out`](#) files(s), use the [`sens_convert`](#) utility.

Zero-Dimensional Adjoint Sensitivity Analysis

Given

$$\dot{y} = f(t, y) \text{ and } y(t_0) = y_0, \quad (23.6)$$

let

$$g = g(t, y, p) \quad (23.7)$$

be a derived function of the solution. To evaluate the sensitivity of this function with respect to the parameters at a given time t_f we must solve

$$\dot{\mu} = -\left(\frac{\partial f}{\partial y}\right)^* \mu \quad (23.8)$$

with the initial condition

$$\mu(t_f) = \left(\frac{\partial g}{\partial y}\right)_{t=t_f}^*. \quad (23.9)$$

Integrating backward from $t = t_f$ to $t = 0$, we obtain

$$\frac{dg(t_f)}{dp} = \mu^*(t_0)s(t_0) + g_p(t_f) + \int_{t_0}^{t_f} \mu^* f_p dt. \quad (23.10)$$

To obtain the sensitivity of temperature with respect to the parameters, the function g becomes temperature, and Equation 23.9 = {0,0,0,...1}. That is, the derivative of g with respect to the species concentrations is zero, and with respect to temperature is one. When

$$\frac{dg(t_f)}{dp} = \int_{t_0}^{t_f} \mu^* f_p dt, \quad (23.11)$$

the sensitivity matrix is a zero matrix at time $t = t_0$ and the derived function g is not a function of the parameter p .

While forward sensitivity analysis writes sensitivity data for all species for all time-steps internal to the CVODES solver, adjoint sensitivity analysis (ASA) provides data at the last time-step or at the ignition time for the species specified. Since it captures less information, adjoint sensitivity analysis is about one order of magnitude faster than forward sensitivity analysis. ASA can be useful when running ignition delay simulations, but it is important to note that ASA cannot be run with double ignition delay (*i.e.*, `zero_d_solver.in > zero_d_output_control > double_ignition_delay_flag` must be 0). Adjoint sensitivity analysis can be used only with constant pressure or constant volume cases (*i.e.*, when `zero_d_solver.in > zero_d_input_control > case_type` is set to `CONSTANT_PRESSURE` or `CONSTANT_VOLUME`). To use ASA, you must also deactivate TKI table generation (*i.e.*, set `zero_d_solver.in > zero_d_output_control > generate_tki_table_flag = 0`). When ASA is active, it applies to all of the 0D cases with a case type of either constant pressure or constant volume (*i.e.*, you cannot apply ASA to only certain cases in `zero_d_cases.in`).

To activate adjoint sensitivity analysis, set `zero_d_solver.in > zero_d_sensitivity_control > sensitivity_flag = ADJOINT`. List the variables for which you want to perform ASA via `zero_d_solver.in > zero_d_sensitivity_control > adjoint_control > adjoint_variables`. Tune the tolerance (`zero_d_solver.in > zero_d_sensitivity_control > adjoint_control > adjoint_rel_tol` and `adjoint_abs_tol`) to avoid a CVODES error.

CONVERGE writes results to ASA to `zero_d_adjoint_case<case ID>.out`. If `mechanism tune` is activated, CONVERGE also writes `zero_d_adjoint.out` and `zero_d_adjoint_rank.out`.

Variable Volume

To specify the volume of the reactor, set `zero_d_solver.in > zero_d_input_control > case_type` to `VARIABLE_VOLUME`, specify the name of the volume file in `zero_d_cases.in > zero_d_cases > case > volume_profile_filename`, and include the volume file in the Case Directory. CONVERGE will use the contents of `zero_d_cases.in` and `zero_d_solver.in` to calculate ignition delay data, which it writes to `ignition_det.out`. If you want CONVERGE to generate `zero_d_sol_case<case ID>.out` files with detailed information for each case, set `zero_d_solver.in > zero_d_output_control > output_file_flag = 1`.

Zero-Dimensional HCCI Engine

The zero-dimensional engine reactor is an application of a 0D variable volume case that can be used to quickly simulate single-cycle homogenous charge compression ignition (HCCI) engines. The 0D engine calculates a volume, V , at each time-step using the specified engine parameters based on the following equation from [Heywood \(1988\)](#):

$$\frac{V}{V_c} = 1 + \frac{r_c - 1}{2} \left[R_{engine} + 1 - \cos \theta - \sqrt{R_{engine}^2 + \sin^2 \theta} \right], \quad (23.12)$$

in which V_c is the clearance volume, r_c is the user-specified compression ratio, θ is the crank angle, and R_{engine} is the ratio of the connecting rod length to the crank radius. This ratio is defined as

$$R_{engine} = \frac{l}{a}, \quad (23.13)$$

in which l is the user-specified connecting rod length and a is crank-arm radius. The clearance volume, V_c , is defined as

$$V_c = \frac{r_c - 1}{V_d}, \quad (23.14)$$

and the maximum displaced volume, V_d , is calculated using the equation

$$V_d = \frac{\pi}{2} D^2 a, \quad (23.15)$$

in which D is the user-specified cylinder bore diameter.

The 0D engine utility then calculates the time derivative of Equation 23.12:

$$\frac{dV/V_c}{dt} = \frac{1}{2}(r_c - 1) \sin \theta \left[\sqrt{R_{engine}^2 - \sin^2 \theta} \right] \frac{d\theta}{dt}. \quad (23.16)$$

Based on the model described by [Heywood \(1988\)](#), the heat loss from the gas to the solid walls is calculated using the equation

$$Q_{wall} = h(T - T_{wall})A_s, \quad (23.17)$$

in which A_s is the area of cylinder (including the piston), T is the user-specified cylinder temperature ([zero_d_cases.in > zero_d_engine_cases > case > temperature](#)) at θ , and h is calculated based on the following global heat transfer coefficient equation:

$$h = \alpha L_{char}^{m-1} \frac{k}{\mu^m} p^m T^{-m} \nu^m, \quad (23.18)$$

in which α is the user-specified scaling factor specific to the engine geometry, L_{char} is based on the user-specified characteristic length ([zero_d_cases.in > zero_d_engine_cases > case >](#)

`heat_transfer_control > heat_transfer_model_control > heat_transfer_equation_control > length_type`), p is the pressure, T is the temperature, v is the characteristic velocity, m is a user-specified exponent (`zero_d_cases.in > zero_d_engine_cases > case > heat_transfer_control > heat_transfer_model_control > heat_transfer_equation_control > velocity_exponent`), and the transport properties k and μ can be expressed as

$$\begin{aligned} k &\propto T^{0.75} \\ \mu &\propto T^{0.75}. \end{aligned} \quad (23.19)$$

Equation 23.18 then becomes

$$h = \alpha L_{\text{char}}^{m-1} p^m T^{0.75-1.62-m} v^m. \quad (23.20)$$

Set `zero_d_solver.in > zero_d_input_control > case_type = ENGINE` to activate the engine case type, and include a `zero_d_engine_cases` settings block in `zero_d_cases.in`. You must specify engine parameters (`bore`, `stroke`, `connecting_rod`, `crank_offset`, `rpm`, and `compression_ratio`) in the `zero_d_cases.in > zero_d_engine_cases > engine_control` settings block. These parameters are defined as the identically named parameters in `engine.in`. Note that the 0D engine is assumed to be at top dead center at 0 crank angle degrees. You must ensure that a negative start time and a positive end time are listed.

You can specify an offset (crank + piston pin) via `zero_d_cases.in > zero_d_engine_cases > engine_control > crank_offset`.

Heat Transfer Correlations

Woschni Heat Transfer Correlation

The Woschni correlation ([Woschni, 1967](#)) uses the cylinder bore diameter, D , as the characteristic length and uses the mean piston speed to calculate the characteristic velocity. The reference conditions, T_r , p_r , and V_r are calculated at the engine start time.

The Woschni heat transfer correlation is described by the equation

$$h = D^{-0.2} p^{0.8} T^{-0.53} \left[C_1 \bar{S}_p + C_2 \frac{V_d T_r}{p_r V_r} (p - p_m) \right]^{0.8}, \quad (23.21)$$

in which $C_1 = 2.28$, $C_2 = 0.00324$, \bar{S}_p is the mean piston speed, p_m is the motored cylinder pressure, and r is the reference state. The motoring pressure p_m is defined by

$$p_m = \frac{V_r^\gamma}{V}, \quad (23.22)$$

where specific heat ratio γ is calculated by C_p / C_v .

Chang Heat Transfer Correlation

The 0D HCCI engine calculates the Chang correlation ([Chang et al., 2004](#)) via the same equations as the Woschni correlation, but the Chang correlation uses different coefficients.

Hohenberg Heat Transfer Correlation

The Hohenberg heat transfer correlation ([Hohenberg, 1979](#)) is calculated as

$$h = C_1 V^{-0.06} p^{0.8} T^{-0.4} \left[\bar{S}_p + C_2 \right]^{0.8}, \quad (23.23)$$

in which $C_1 = 130$, $C_2 = 1.4$, and V , the instantaneous volume, is used as the characteristic length. You do not need to specify a reference state for the Hohenberg correlation. Similar to the Woschni correlation, C_2 is non-zero only during the combustion and expansion stroke.

Annand Heat Transfer Correlation

The Annand heat transfer correlation ([Annand, 1963](#)) uses the mean piston speed as its characteristic velocity and the engine bore diameter as the characteristic length. This correlation is calculated via

$$h = C_1 D^{-0.3} p^{0.7} T^{0.316} \bar{S}_p^{-0.7} + C_2 \sigma \frac{T^4 - T_w^4}{T - T_w}, \quad (23.24)$$

in which σ is the Stefan-Boltzmann constant, C_1 varies from 350 to 800 and depends on the intensity of the charge motion, and C_2 is the radiation term. We recommend that you set C_2 to 0.58 and 0.0755 for diesel combustion and spark-ignition engines, respectively.

You can either specify a heat transfer coefficient profile through an input file (*i.e.*, set [`zero_d_cases.in > zero_d_engine_cases > case > heat_transfer_control > heat_transfer_option = 1`](#) and specify the file name in [`zero_d_cases.in > heat_transfer_control > htc_profile_filename`](#)) or activate a heat transfer model (the [Woschni](#), [Chang](#), [Hohenberg](#), or [Annand](#) correlation) (*i.e.*, set [`zero_d_cases.in > zero_d_engine_cases > case > heat_transfer_control = 2 or 3`](#)) to calculate the heat transfer correlations necessary for zero-dimensional engine simulations.

The table below lists the default values for various parameters, including the characteristic length, used in the four heat transfer coefficient models used when [`zero_d_cases.in > zero_d_engine_cases > case > heat_transfer_control = 2`](#). You can change the appropriate parameters in the [`zero_d_cases.in > heat_transfer_equation_control`](#) settings block when [`zero_d_cases.in > zero_d_engine_cases > case > heat_transfer_control = 3`](#).

Note that you can alter the exponent m (in Equation 23.22) by changing the values for density, characteristic length, temperature, and velocity. Some or all of these values are specified through the heat transfer correlations.

Table 23.1: Default values for parameters used by the heat transfer coefficient correlations in the `zero_d_cases.in` > `zero_d_engine_cases` > `case` > `heat_transfer_control` > `heat_transfer_model_control` > `heat_transfer_equation_control` settings block.

Parameters	Woschni	Chang	Hohenberg	Annand
Scaling factor (<i>scaling_factor</i>)	3.26	3.4	1.0	1.0
Characteristic length type (<i>length_type</i>)	<i>CYLINDER_BORE</i>	<i>CHAMBER_HEIGHT</i>	<i>VOLUME</i>	<i>CYLINDER_BORE</i>
Exponent for characteristic length (<i>length_exponent</i>)	-0.2	-0.2	-0.06	-0.3
Pressure exponent (<i>pressure_exponent</i>)	0.8	0.8	0.8	0.7
Temperature exponent (<i>temperature_exponent</i>)	-0.53	-0.73	-0.4	0.316
Velocity exponent (<i>velocity_exponent</i>)	0.8	0.8	0.8	0.7
C1 (<i>c1</i>)	2.28	2.28	130	0.25
C2 (<i>c2</i>)	0.00324	0.00054	1.4	0.576

RON/MON Calculator

You can use the zero-dimensional utility to calculate the research octane number (RON) and the motor octane number (MON) for a fuel. The calculated critical compression ratio (CCR) for the specified fuel is correlated against a reference CCR. A modified Woschni heat transfer correlation by [Chang et al., 2004](#) is used for calculating both RON and MON. Set `zero_d_solver.in` > `zero_d_input_control` > `case_type = ENGINE_*` to calculate RON and/or MON, and supply a `zero_d_cases.in` > `zero_d_engine_ron_mon_cases` settings block.

The reference CCR in CONVERGE is based on either the Lawrence Livermore National Laboratory (LLNL) Gasoline mechanism ([Mehl et al., 2011](#)) when `zero_d_solver.in` > `zero_d_input_control` > `ron_mon_table_input_flag = 0` or based on a [user-supplied correlation table](#) when `ron_mon_table_input_flag = 1`.

To determine the reference CCRs based on the LLNL Gasoline mechanism, a primary reference fuel (PRF) is used. A PRF is a mixture of the fuel components iso-octane and n-heptane, where PRF0 is pure n-heptane and PRF100 is pure iso-octane. The compression ratio of each PRF is varied and the lowest compression ratio at which the fuel autoignites is the CCR for that PRF composition, providing a correlation between PRF composition and CCR. Since these CCR values are used as a reference for the calculated CCR values of a specified fuel, they are referred to as reference CCR values and they are hard-coded in CONVERGE.

The utility then compares the calculated CCR of the specified fuel (*i.e.*, RON_CCR or MON_CCR in [ignition_det.out](#)) with the reference CCR values for a PRF. CONVERGE reports the PRF number of the reference CCR that most closely matches the calculated CCR for the specified fuel. If the calculated CCR lies between two reference CCR values, CONVERGE interpolates the data to determine the PRF. CONVERGE then reports the PRF of the reference CCR as the RON and MON values for that fuel in [ignition_det.out](#). If the calculated CCR of the fuel falls outside the range of the correlation with the reference fuel and mechanism, the utility will list a RON or MON value of 100. We recommend that this utility be used only for gasoline-based mixtures.

The table below lists engine parameters used to calculate the RON and MON values for the user-specified fuel composition. These parameters were derived from the standard test ASTM D2699 for RON and ASTM D2700 for MON.

Table 23.2: Engine parameters used in RON/MON calculations.

Parameters	RON	MON
RPM	600	900
Temperature (Celsius)	52	149
Pressure (atm)	1	1
Start time (crank angle degrees)	-146	-146
End time (crank angle degrees)	115	115
Bore (cm)	8.255	8.255
Stroke (cm)	11.43	11.43
Connecting rod length (cm)	25.4	25.4
Equivalence ratio	1	1

To calculate RON or MON values for a fuel, set [zero_d_solver.in](#) > zero_d_input_control > case_type = ENGINE_RON or ENGINE_MON, respectively. To calculate both RON and MON values, set [zero_d_solver.in](#) > zero_d_input_control > case_type = ENGINE_RON_MON. You must include a [reaction mechanism file](#), a [thermodynamic data file](#), and a [zero_d_solver.in](#) file, as well as a [zero_d_cases.in](#) file that includes the zero_d_engine_ron_mon_cases settings block.

User-Supplied Correlation Table

To generate a table that contains CCR values at various PRF that does not rely on the LLNL Gasoline mechanism in [Mehl et al., 2011](#), you must generate the correlation between PRF and reference CCR values prior to calculating CCR values for a specified fuel. Run a 0D simulation with [zero_d_solver.in](#) > zero_d_input_control > case_type = ENGINE_RON_MON_TABLE and include the [zero_d_solver.in](#) > zero_d_output_control > ron_mon_table_output_control settings block. You must also include a [reaction mechanism file](#).

and [thermodynamic data file](#). CONVERGE generates *ron_mon_table.dat* that you can then use in future RON/MON calculations. In order to read this user-generated table instead of the internal LLNL table, supply the user-generated table in the Case Directory and set [*zero_d_solver.in*](#) > *zero_d_input_control* > *ron_mon_table_input_flag* = 1 when calculating RON and/or MON.

Plug Flow Reactor

A plug flow reactor (PFR) can be used to simulate ignition delay times, emissions, and chemical reactions in cylindrical flows. A PFR model assumes that the flow through the reactor passes through a series of thin sections, which are called plugs. CONVERGE uses a Lagrangian particle approach to the PFR.

We define

$$\rho u A = \dot{m}, \quad (23.25)$$

in which A is the user-defined cross-sectional area of the reactor, ρ is the density, u is the user-specified axial velocity ([*zero_d_cases.in*](#) > *zero_d_cases* > *case* > *plug_flow_control* > *inlet_velocity*) and is the inline mass flow rate.

Kee et al. (2017) list the governing equations for the conservation of energy and momentum in a PFR as

$$\rho u c_p \frac{dT}{dz} = - \sum_{k=1}^n \dot{\omega}_k M W_k h_k \quad (23.26)$$

and

$$\rho u A \frac{du}{dz} = - \frac{d(\rho A)}{dz} - \tau_w P, \quad (23.27)$$

in which z is the axial location in the reactor; c_p is the specific heat capacity; T is the calculated temperature; τ_w is the wall-friction coefficient; P is the perimeter of the reactor; and, for each species k , $\dot{\omega}_k$ is the reaction rate, h_k is the enthalpy, and $M W_k$ is the molecular weight. The continuity equation for species k is

$$\rho u \frac{dY_k}{dz} = \dot{\omega}_k M W_k, \quad (23.28)$$

in which Y_k is the mass fraction for each species k .

Since the PFR does not account for a diffusion term, flow downstream from a point cannot affect the flow upstream, and thus the end of each plug could be matched to the beginning of the next. CONVERGE transforms the case from the location space (z) to the time space (t), and so Equations 23.26 and 23.28 become

$$\rho c_p \frac{dT}{dt} = -\sum_{k=1}^n \dot{\omega}_k M W_k h_k \quad (23.29)$$

and

$$\rho \frac{dY_k}{dt} = \dot{\omega}_k M W_k. \quad (23.30)$$

By performing a forward integration in time

$$z = \int_0^t u dt, \quad (23.31)$$

the calculated values of species, velocity, and temperature in the domain can be translated into a spatial description of the domain. Since the PFR can be used to predict the spatial behavior of chemical reactions and flow in the reactor, results from this 0D simulation can be compared to the temperature and species distribution in 1D reactor experiments.

CONVERGE offers a constant-pressure PFR case type in the 0D chemistry utility. To activate this option, set `zero_d_solver.in` > `zero_d_input_control` > `case_type` = `CONSTANT_PRESSURE_PLUG_FLOW_REACTOR` and include the appropriate `zero_d_cases.in` > `zero_d_cases` settings block.

Well-Stirred Reactor

A well-stirred reactor (WSR) is typically used to study highly turbulent or well-mixed phenomena. You can use the WSR to investigate phenomena that often limit combustion, such as extinction and ignition limits, for a reaction mechanism.

A well-stirred reactor is considered to be spatially uniform. The 0D utility solves the governing equation for the conservation of energy and species

$$\frac{dU_{sys}}{dt} = \dot{m} \sum_{k=1}^n Y_k^* h_k^* - \dot{m} \sum_{k=1}^n Y_k h_k - Q_{loss}, \quad (23.32)$$

in which U_{sys} is the internal energy of the system, \dot{m} is the mass flow rate, n is the number of species, and for each species k in the reactor, Y_k^* is the user-specified inlet species mass fraction, h_k^* is the enthalpy for each species at the inlet, and Y_k and h_k are the species mass fraction and enthalpy in the reactor. The term Q_{loss} is used to model heat transfer in the system using the equation

$$Q_{loss} = Ah_t(T - T_0), \quad (23.33)$$

in which A is the surface area of the reactor, h_t is the user-specified heat transfer coefficient for the reactor, T_0 is the user-specified ambient temperature, and T is the calculated temperature.

The 0D chemistry utility calculates the residence time scale (τ), which is defined as

$$\tau = \frac{\rho V}{\dot{m}^*}, \quad (23.34)$$

using the equation

$$\frac{dY_k}{dt} = \frac{(Y_k^* - Y_k)}{\tau} + \dot{\omega}_k \frac{MW_k}{\rho}, \quad (23.35)$$

in which $\dot{\omega}$ is the reaction rate and MW_k is the molecular weight of species k .

CONVERGE offers a WSR case type in the zero-dimensional chemistry utility. To activate the WSR, set `zero_d_solver.in > zero_d_input_control > case_type = WELL_STIRRED_REACTOR`, include the `zero_d_solver.in > well_stirred_reactor_control` settings block, and include the `zero_d_cases.in > zero_d_wsr_cases` settings block. You can activate `surface_chemistry` in conjunction with a WSR simulation by setting `zero_d_solver.in > well_stirred_reactor_control > surface_chemistry = 1`. If you activate surface chemistry, you must also provide a surface chemistry thermodynamic data file (e.g., `surface_therm.dat`) and a surface chemistry mechanism file (e.g., `surface_mech.dat`) in the `zero_d_solver.in > well_stirred_reactor_control` settings block.

Ignition and extinction correspond to a saddle-shaped bifurcation in a classical S-shaped steady-state curve. You can choose to identify the extinction limit or the extinction and ignition limits (i.e., set `zero_d_solver.in > well_stirred_reactor_control > find_turning = EXTINCTION` or `EXTINCTION_AND_IGNITION`, respectively). The 0D chemistry utility will apply a number of elemental equivalence ratios at the inlet to find a burning mixture and iteratively run a WSR simulation to determine the turning point based on the change in

the mass flow rate for the outlet species. Note that it is more computationally expensive to find the extinction time than the ignition time. CONVERGE writes the limiting residence time scale to [*extinction_limit.out*](#).

Chemical Equilibrium

You can use a chemical equilibrium (CEQ) solver to predict species concentrations at equilibrium. The zero-dimensional CEQ solver uses the same approach as that of the CEQ combustion model in CONVERGE ([Pope, 2003](#)).

Set [*zero_d_solver.in*](#) > [*zero_d_input_control*](#) > [*case_type*](#) to
[*CEQ_CONSTANT_ENTHALPY_PRESSURE*](#) or
[*CEQ_CONSTANT_TEMPERATURE_PRESSURE*](#) to enforce either constant enthalpy and pressure or constant temperature and pressure, respectively, and specify the case settings in [*zero_d_cases.in*](#).

This utility writes the results of a 0D CEQ simulation (species concentrations at equilibrium in mass fraction and mole fraction format) to [*zero_d_sol_case<case ID>.out*](#).

One-Dimensional Chemistry Utilities

CONVERGE includes both a [laminar freely propagating flamespeed calculation](#) and a [laminar counterflow calculation](#).

The format for a serial execution of a 1D calculation is:

```
<CONVERGE executable> -u oned_flame
```

where `<CONVERGE executable>` is any CONVERGE executable (e.g., `converge-mpich`).

A sample command for a parallel execution (using 16 processors) of a 1D case is:

```
mpirun -n 16 converge-mpich -u oned_flame
```

Inputs for 1D Chemistry Utilities

All 1D simulations require a [reaction mechanism file](#) (e.g., `mech.dat`) with reaction data for species and reaction details, a [thermodynamic data file](#) (e.g., `therm.dat`) for the species' thermodynamic constants, a gas properties file (`transport.dat` or `gas.dat`) for a list of diffusion coefficients, a 1D solver file ([*one_d_solver.in*](#)), and a 1D cases file (either [*one_d_cases.in*](#) or [*one_d_template.in*](#)). Even if you employ the [PISO solver](#) or the [hybrid approach](#), the 1D utility automatically generates any necessary CONVERGE inputs.

The controls for the numerics of the 1D chemistry utility are set in [*one_d_solver.in*](#). These controls include the flame type, tolerance values, and iteration frequency.

The initial conditions for temperature, pressure, and species are specified in [*one_d_cases.in*](#) (or [*one_d_template.in*](#)). These parameters are repeated for each case in [*one_d_cases.in*](#). Note that CONVERGE re-normalizes the mole fractions internally, so the sum of the mole fractions of the reactants specified in [*one_d_cases.in*](#) does not need to be one.

We recommend setting up 1D chemistry input files in the *Chemistry* module in CONVERGE Studio.

Laminar Freely Propagating Flamespeed Calculation

A premixed laminar flamespeed model at constant pressure is used to calculate the flamespeed of the combustion reaction using a freely propagating flame. CONVERGE models the one-dimensional flame in a channel with fixed cross-sectional area.

To calculate the laminar freely propagating flamespeed, set [*one_d_solver.in* > *one_d_general_control* > *case_type* = LAMINAR_FREELY_PROPAGATING_FLAME](#) and include the appropriate [*one_d_cases.in* > *one_d_cases*](#) settings block. You must specify the unburned fuel/oxidizer composition and temperature along with the channel pressure. There are three approaches for calculating the laminar freely propagating flamespeed:

- [Stand-alone steady-state solver \(Newton's method\)](#)
- [PISO solver: Calculation using the CONVERGE CFD solver in 1D](#)
- [A hybrid of the above approaches](#)

If you use the Newton or hybrid method for determining the laminar flamespeed, you can invoke [1D sensitivity analysis](#) to determine the flamespeed's sensitivity to the reactions' pre-exponential factors.

Newton's method of calculating laminar flamespeeds is also used to [generate tabulated laminar flamespeed \(TLF\) tables](#), which are required for certain combustion models.

Stand-Alone Steady-State Solver (Newton's Method)

CONVERGE solves for the burned fuel/oxidizer composition and temperature with the species and energy conservation equations using Newton's method. The species conservation equation is

$$\dot{m} \frac{dY_k}{dx} + \frac{d(\rho A Y_k V_k)}{dx} - \dot{S} = 0 \quad (23.36)$$

in which Y_k is the species mass fraction of the k -th species, ρ is the density, V_k is the diffusion velocity of the k -th species, A is the constant cross-sectional area of the channel, and \dot{S} is the source term. We define the mass flow rate, \dot{m} , as

$$\dot{m} = \rho u A \quad (23.37)$$

in which u is the velocity. The energy conservation equation is

$$\dot{m} \frac{dT}{dx} - \frac{1}{c_p} \frac{d}{dx} \left(\lambda A \frac{dT}{dx} \right) + \frac{A}{c_p} \sum \rho Y_k V_k c_{p,k} \frac{dT}{dx} + \dot{S} = 0 \quad (23.38)$$

in which T is the temperature, λ is the thermal conductivity of the mixture, $c_{p,k}$ is the constant-pressure heat capacity of the k -th species, and c_p is the constant-pressure heat capacity of the mixture. We define the flamespeed calculation model to have no heat loss. CONVERGE iteratively solves Equations 23.36 and 23.38 for values of Y_k and T until the user-defined tolerance is met.

CONVERGE calculates the Schmidt number of species in the burned region of a laminar freely propagating flame as

$$Sc = \frac{\mu}{\rho D_k} \quad (23.39)$$

where μ is the molecular viscosity, ρ is the density, and D_k is the species diffusivity in the burned region. To calculate the Schmidt number, set [one_d_solver.in](#) > [one_d_newton_control](#) > [output_control](#) > [schmidt_species_flag = 1](#).

To use Newton's method, set [one_d_solver.in](#) > [one_d_general_control](#) > [solver_control](#) > [solver_type = NEWTON](#) and configure the [one_d_solver.in](#) > [one_d_newton_control](#) settings block.

PISO Solver: CONVERGE Solver in 1D

The PISO solver approach uses the 3D CONVERGE CFD solver to run a 1D premixed laminar flamespeed case. The one-dimensional utility automatically maps 1D solver settings to the 3D PISO solver so that you do not need to specify as many inputs as for a full 3D CONVERGE CFD simulation. This approach is more robust—but more computationally expensive—than [Newton's method](#). CONVERGE monitors solution variables in [flamespeed.out](#) and [thermo.out](#) and, when both converge on the current grid, CONVERGE switches to a stricter criterion for [Adaptive Mesh Refinement](#) (AMR) to improve the solution accuracy.

Many of the parameters that you specify in [one_d_solver.in](#) for this approach are proxies for inputs that you would typically supply in files such as [inputs.in](#), [solver.in](#), or [amr.in](#) for a full 3D CONVERGE simulation. Since the 1D solver is a theoretical model, you need to specify only certain parameters while the 1D utility determines others from assumptions about the 1D model.

Using the PISO solver requires a procedure for anchoring the flame at a fixed location in the computational domain. Otherwise, the flame would progress through the domain and the results would not be useful. The flame location is the x location such that the temperature at the flame location is equal to T_{anchor} ([one_d_solver.in](#) > [one_d_general_control](#) > [solver_control](#) >

anchor_temp). After calculating the flame location, CONVERGE adjusts the inlet velocity u such that the flame is stationary:

$$u_{inlet}^{n+1} = u_{inlet}^n (1 - r) + rs_l, \quad (23.40)$$

where the superscript n indicates the previous time-step, the superscript $n+1$ indicates the current time-step, r is the velocity relaxation factor ([one_d_solver.in](#) > [one_d_piso_control](#) > [flame_anchoring_control](#) > [relax_velocity_factor](#)), and s_l is the estimated laminar flamespeed. To aid convergence, CONVERGE adds a source term to the x momentum equation. This source is

$$src_x = \Omega (\rho u_x - \rho_{unburned} u_{inlet}^n), \quad (23.41)$$

where Ω is a damping factor ([one_d_solver.in](#) > [one_d_piso_control](#) > [flame_anchoring_control](#) > [mass_flow_rate_damp_factor](#)).

To use the PISO solver, set [one_d_solver.in](#) > [one_d_general_control](#) > [solver_control](#) > [solver_type = PISO](#) and configure the [one_d_solver.in](#) > [one_d_piso_control](#) settings block.

To determine convergence at a particular grid level, CONVERGE uses the [steady-state monitor feature](#) to monitor the flamespeed that is written to [flamespeed.out](#) and the heat release rate that is written to [thermo.out](#). You do not have to manually configure the monitor steady-state feature because the 1D utility automatically activates and controls this procedure. You can, however, control several options via parameters in the [one_d_solver.in](#) > [one_d_piso_control](#) > [monitor_steady_state_control](#) settings block.

To improve the speed of the PISO solver, it is good practice to begin the 1D simulation on a coarse grid and specify multiple values for the [AMR sub-grid scale criterion](#). When the difference between the actual field and the resolved field in any part of the domain exceeds the sub-grid scale criterion, CONVERGE refines the grid in this location. The 1D utility monitors convergence via the steady-state monitor feature (as described above) to determine when to change to the next value of sub-grid scale. Use [one_d_solver.in](#) > [one_d_piso_control](#) > [amr_stages_control](#) > [amr_temp_stages](#) to specify the number of values for the AMR sub-grid scale criterion. On the lines below [amr_temp_stages](#), enter monotonically decreasing values for the temperature sub-grid scale criterion for each stage.

You can optionally specify a final stage for the PISO solver. That is, after the number of AMR sub-grid scale stages prescribed by [one_d_solver.in](#) > [one_d_piso_control](#) > [amr_stages_control](#) > [amr_temp_stages](#) has occurred, CONVERGE will perform one stage of the simulation with tighter tolerances and a lower CFL number and calculate a converged solution. Set [one_d_solver.in](#) > [one_d_piso_control](#) > [final_stage_control](#) > [active = 1](#) to activate the final simulation stage. In addition to specifying a final set of parameters for the steady-state monitor, you can also specify a maximum CFL number based on viscosity via [one_d_solver.in](#)

`> one_d_piso_control > final_stage_control > max_cfl_nu.` Typically, you decrease this CFL number to decrease the time-step and thus improve accuracy.

1D Hybrid Solver

The hybrid solver approach combines the previously described [Newton's method](#) and [PISO solver](#) approaches. The first portion of the simulation employs the PISO solver on a coarse grid. After the monitored variables converge for each value of sub-grid scale criterion in the CONVERGE CFD solver, the 1D utility maps the results to the stand-alone steady-state Newton solver for a faster solution on a fine mesh. Since the PISO solver approach is more robust, using the hybrid approach assists in obtaining a stable initial solution and then obtaining a fast solution on a finer mesh.

To use the hybrid approach, set `one_d_general_control > solver_control > solver_type = HYBRID` and configure the `one_d_newton_control` and the `one_d_piso_control` settings blocks in [`one_d_solver.in`](#).

One-Dimensional Sensitivity Analysis

You can run a sensitivity analysis with the 1D laminar freely propagating flamespeed solver to determine the flamespeed's sensitivity to the pre-exponential A factor for various reactions in the [reaction mechanism file](#) (e.g., `mech.dat`).

This feature is available only when using the stand-alone [1D Newton solver](#) or the [hybrid solver](#) (i.e., when [`one_d_solver.in`](#) > `one_d_general_control > solver_control > solver_type = NEWTON` or `HYBRID`).

To run a 1D sensitivity analysis, set [`one_d_solver.in`](#) > `one_d_newton_control > output_control > sensitivity_flag = 1`. When you run a 1D sensitivity analysis, the solver first calculates a converged flamespeed and then perturbs the pre-exponential factor for the reactions in the mechanism to detect the flamespeed sensitivity to this factor. You can then adjust the reaction pre-exponential factors, and thus the laminar flamespeed, in the [reaction mechanism file](#) to improve your simulation results so that they more accurately reflect your data.

When you activate the sensitivity analysis option, the 1D utility writes files that contain the sensitivity information for each case in the form [`one_d_sens_case<case ID>.out`](#). The `<case ID>` corresponds to the cases listed in [`one_d_cases.in`](#) based on the order in which the cases appear. If [mechanism tune](#) is activated, the utility also writes [`one_d_sens.out`](#), which consolidates information from `one_d_sens_case<case ID>.out` and contains the reactions from the [reaction mechanism file](#) sorted in decreasing order of sensitivity, and [`one_d_sens_rank.out`](#), which lists the combined ranking from all the 1D cases into one file.

Tabulated Laminar Flamespeed Calculation

To generate a [table of laminar flamespeeds](#) (`tlf_table.h5`) that can be used in an [ECFM](#), [ECFM3Z](#), [FGM](#), [G-Equation](#), or [SAGE with TFM](#) simulation, set [`one_d_solver.in`](#) > `one_d_general_control > output_control > generate_tlf_table_flag = 1` and use [`one_d_template.in`](#) to configure the TLF table values and ranges. Since generating a TLF table file is

computationally expensive, we recommend using a [reaction mechanism file](#) with no more than 500 species. You must use the [Newton solver](#) (i.e., `one_d_solver.in` > `one_d_general_control` > `solver_control` > `solver_type = NEWTON`) to generate a TLF table.

As an alternative to generating a TLF table from the command line, you can generate a TLF table using CONVERGE Studio. See the CONVERGE Studio 3.1 Manual for more information.

A TLF table file is an HDF5-formatted file that contains the calculated laminar flamespeed (*Output* > *sl*) as well as the laminar flame thickness (*Output* > *aux_data* > *flame_thickness*). For SAGE with TFM, you can use [sensor species](#). If sensor species are specified in `one_d_solver.in` > `one_d_general_control` > `output_control` > `sensor_species`, then the maximum reaction rate of $\dot{\Omega}_{sens,0}$ for the sensor species is also calculated and written to the TLF table (*Output* > *aux_data* > *Omega<number>*). You can specify multiple sensor species when generating a TLF table, but only a single sensor species can be specified in [combust.in](#) > *thickened_flame_model* > *flame_sensor_model* > *sensor_species* during the 3D simulation.

CONVERGE writes cases to the TLF table file when either 1000 cases have been calculated or all cases have been calculated. Upon initiation, the 1D premixed laminar flamespeed solver writes zeros to *Output* > *sl* in the TLF table file. If a case fails to converge before the case time limit is reached, or if the solver is terminated before a case has been calculated, *Output* > *sl* in the TLF table file will remain zero, CONVERGE prints a message to the [log file](#), and CONVERGE writes the case number and details to [tlf_table_failed_cases.out](#). If you restart the 1D laminar flamespeed solver with *tlf_table.h5* in the Case Directory, the solver will recalculate only those values that are zero (all previously completed calculations will remain unaltered).

To use a TLF table in a CONVERGE 3D simulation that invokes an appropriate combustion model, set `combust.in` > *sl_model* > *active* = 5 and specify the file name of the TLF table (e.g., *tlf_table.h5*) in `combust.in` > *sl_model* > *sl_option* > *tlf_table_filename*.

Laminar Counterflow Calculation

A laminar counterflow calculation is used to calculate the temperature and species between two inlet streams, as well as the strain rate. This model is based on a geometry of two circular inlet nozzles directed at each other. The configuration of the inlets creates an axisymmetric flow with a stagnation plane between the inlets. The streams can either be premixed, in which case there are two flames, one on each side of the central plane. The streams can also be non-premixed, in which one stream contains fuel and the other contains oxidizer. In this non-premixed geometry, a diffusion flame is produced. Here we present the governing equations as derived by [Kee et al. \(1989\)](#).

Figure 23.9 below shows the laminar counterflow geometry, in which the axial components of position and velocity refer to the plane that is perpendicular to the stagnation plane (*x* and *u*) while radial components refer to the plane parallel with the stagnation plane (*r* and *v*).

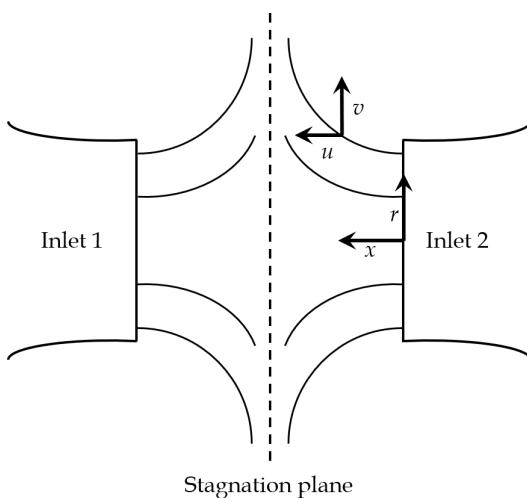


Figure 23.9: Laminar counterflow geometry. Two inlets are positioned opposite each other, producing an axisymmetric flow and a stagnation plane between the inlets. If Inlet 1 and/or 2 contains a fuel and oxidizer, a premixed flame is produced. If one inlet contains a fuel and the other inlet contains an oxidizer, a diffusion flame is produced.

CONVERGE solves for the burned fuel/oxidizer composition and temperature with the energy and species conservation equations using [Newton's method](#). Conservation of mass at steady-state is represented in cylindrical coordinates as

$$\frac{\partial}{\partial x}(\rho u) + \frac{1}{r} \frac{\partial}{\partial r}(\rho v r) = 0, \quad (23.42)$$

where u and v are the axial and radial velocity components, x and r are the axial and radial position components, and ρ is the mass density.

In order to define variables as functions of x , we use

$$G(x) = -\frac{\rho v}{r} \quad \text{and} \quad F(x) = \frac{\rho u}{2}, \quad (23.43)$$

and the continuity for the axial velocity (u) reduces to

$$G(x) = \frac{dF(x)}{dx}. \quad (23.44)$$

F and G are functions of x and thus ρ , u , T , and Y_k are also functions of x .

The radial momentum equation is satisfied by the eigenvalue

$$H = \frac{1}{r} \frac{\partial p}{\partial r} = \text{constant}, \quad (23.45)$$

where p is pressure and r is the radial distance.

The axial momentum equation is

$$H - 2 \frac{d}{dx} \left(\frac{FG}{\rho} \right) + \frac{3G^2}{\rho} + \frac{d}{dx} \left[\mu \frac{d}{dx} \left(\frac{G}{\rho} \right) \right] = 0 \quad (23.46)$$

The energy and species conservation equations are

$$\rho u \frac{dT}{dx} - \frac{1}{c_p} \frac{d}{dx} \left(\lambda \frac{dT}{dx} \right) + \frac{\rho}{c_p} \sum_k c_{p_k} Y_k V_k \frac{dT}{dx} + \frac{1}{c_p} \sum_k h_k \dot{\omega}_k = 0 \quad (23.47)$$

and

$$\rho u \frac{dY_k}{dx} + \frac{d}{dx} (\rho Y_k V_k) - \dot{\omega}_k MW_k = 0 \quad (23.48)$$

where MW_k is the species molecular weight and V_k is the diffusion velocity. CONVERGE solves equations 23.44 to 23.48 in order to determine F , G , H , T , and Y_k .

To perform a laminar counterflow calculation, set `one_d_general_control > case_type = LAMINAR_COUNTERFLOW_FLAME` and `one_d_solver.in > one_d_general_control > solver_control > solver_type = NEWTON`, configure the `one_d_solver.in > one_d_newton_control` settings block, and include the appropriate `one_d_cases.in > one_d_cases` settings block.

CONVERGE generates `one_d_strain_rate.out` for a laminar counterflow flame simulation, which provides the global strain rate of the fuel under the specified conditions.

Mechanism Reduction

Comprehensive kinetic combustion models incorporate hundreds of chemical species and thousands of reactions. These combustion models are too computationally expensive to use in most simulations in which the complementary physical processes (such as heat and mass transfer or gas motion) are also embodied in the simulations. The computational time taken

to obtain a numerical solution is typically proportional to N^2 , where N is the number of species, or proportional to n , where n is the number of reactions.

To make the mechanism computationally efficient, you must reduce the number of reaction species and reactions in a way that maintains solution accuracy. You must validate the reduced mechanisms at various stages of reduction by comparing the output with the reduced mechanism to the output with the comprehensive mechanism. The precision with which quantitative agreement between the reduced and full models is established determines the extent of the reduction that can be achieved. This precision is controlled by tests at different thresholds. The overall target becomes a balance between computational efficiency and accuracy of the reduced mechanism output.

CONVERGE uses a basic skeletal mechanism reduction in which all species deemed unimportant (along with their associated reactions) are removed from the original detailed mechanism. The skeletal mechanism reduction is based on Directed Relation Graph with Error Propagation and Sensitivity Analysis (DRGEPSA). The sections below describe the DRGEPSA methodology. You can efficiently generate a reduced or skeletal mechanism by running it on multiple machines in parallel.

Mechanism reduction can be performed with either 0D autoignition cases or 1D laminar flamespeed cases to generate a reduced mechanism: [*mech_ske.dat*](#). This reduced, or skeletal, mechanism is then used to generate the ignition delay data in [*ignition_ske.out*](#) or laminar flamespeed data in [*one_d_flamespeed_ske.out*](#). The ignition delay and laminar flamespeed data for the original mechanism are reported in [*ignition_det.out*](#) and [*one_d_flamespeed_det.out*](#), as described in previous sections for stand-alone 0D and 1D cases.

Mechanism Reduction

The format for a serial execution of a 0D or 1D mechanism reduction calculation is:

```
<CONVERGE executable> -u reduction
```

where <CONVERGE executable> is any CONVERGE executable (e.g., converge-mpich).

A sample command for a parallel execution (using 16 processors) of a 0D or 1D mechanism reduction calculation is:

```
mpiexec -n 16 converge-mpich -u reduction
```

Inputs for Mechanism Reduction

The controls for mechanism reduction are set in [*mechanism_reduction.in*](#). These controls include tolerance values and target species. CONVERGE uses [*zero_d_cases.in*](#) and [*zero_d_solver.in*](#) for 0D mechanism reduction (described in the [Zero-Dimensional Chemistry Utilities](#) section) and/or [*one_d_cases.in*](#) and [*one_d_solver.in*](#) for 1D mechanism reduction (described in the [One-Dimensional Chemistry Utilities](#) section). CONVERGE also requires [*mech.dat*](#). If you perform a 1D mechanism reduction, you must also include either [*transport.dat*](#) or [*gas.dat*](#).

Dynamic Mechanism Reduction

An alternative to reducing the mechanism prior to executing a SAGE simulation is to invoke the Dynamic Mechanism Reduction (DMR) option. DMR will reduce the mechanism during the SAGE simulation based on target weight and error propagation tolerance values you specify.

To activate DMR, you must first activate the [SAGE detailed chemistry solver](#) by setting `combust.in > sage_model > active = 1`. Configure the SAGE-related parameters as needed. Note that if you activate DMR in conjunction with [adaptive zoning](#), each bin will have its own reduced mechanism that will vary with time.

Next set `combust.in > sage_model > dmr_flag = 1`. CONVERGE will look for a file named `sage_dmr.in` in the Case Directory. When using DMR, you need to declare two non-transport [passives](#) in `species.in`, as shown in the following figure.

```
-  
-  
passive_nt:  
  - DMR_NUM_SPECIES  
  - DMR_NUM_REACTIONS
```

Figure 23.10: An excerpt of `species.in`.

The `dmr_mech_info.out` file contains statistical information about the species and reactions when DMR is active.

DRG (Directed Relation Graph)

[Lu and Law \(2005\)](#) devised an automatic mechanism reduction procedure based on the theory of directed relation graph (DRG). Each vertex in a DRG as shown in Figure 23.11 represents a species in the detailed mechanism. An edge from vertex A to vertex B exists only if the removal of species B would directly induce significant error to the production rate of species A. This effect is measured by the normalized contribution, r_{AB} , defined as:

$$r_{AB} \equiv \frac{\sum_{i=1,I} |\nu_{A,i} \omega_i \delta_{Bi}|}{\sum_{i=1,I} |\nu_{A,i} \omega_i|} \quad \delta_{Bi} = \begin{cases} 1, & \text{if the } i^{\text{th}} \text{ reaction involves species B} \\ 0, & \text{otherwise} \end{cases}, \quad (23.49)$$

where $\nu_{A,i}$ is the stoichiometric coefficient of species A in reaction i and ω_i is the net rate of a reversible reaction i . The net rate of a reversible reaction is the difference between the rates of the forward and the backward reactions.

The DRG starts as a set of species specified in the [reaction mechanism file](#), where each species is represented by a stand-alone vertex in Figure 23.11 below. Then the DRG uses a user-defined threshold, ε , for the normalized contribution, $0 < \varepsilon < 1$. The DRG uses this threshold to determine which species are retained in the reduced mechanism. The DRG is

developed by an iterative procedure by maintaining connections and the corresponding species for which $r_{AB} > \varepsilon$.

Figure 23.11 below is a simplified representation of a DRG. In this example, if you set $\varepsilon = 0.15$, species *G* and *J* from the detailed mechanism depicted in (a) get removed to reduce the mechanism to the mechanism shown in (b).

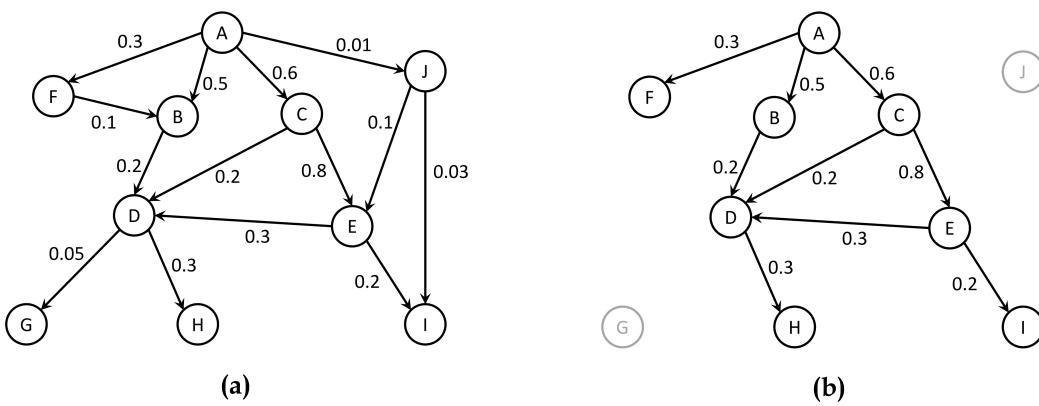


Figure 23.11: (a) Original DRG, (b) DRG reduced ($\varepsilon=0.15$).

DRGEP (Directed Relation Graph Error Propagation)

You can define a set of primary dependent species for species *A*, consisting of those species that appear explicitly in reactions involving *A*. If species *B* is not in the primary dependent set of *A*, then $r_{AB}=0$. However, species *C* interacting with species *A* through species *B* is necessary for *A* only if it is necessary for *B* and *B* is necessary for *A*. This indirect coupling is quantified by a path-dependent coefficient, $r_{AB,i}$. The path-dependent coefficient $r_{AB,i}$ is the product of the normalized contributions along path *i* between species *A* and *B*. The influence of species *B* on species *A* is characterized by coefficient R_{AB} , which is the maximum of the path-dependent coefficients and is known as the DRGEP interaction coefficient or local interaction coefficient:

$$r_{AB,i} = \prod_{XY \in i} r_{XY},$$

$$R_{AB} = \max_{\text{all paths } i} r_{AB,i}. \quad (23.50)$$

In the Directed Relation Graph with Error Propagation (DRGEP) method, the species selection procedure is based on the R_{AB} values instead of the r_{AB} values used in the DRG method. A connection is considered significant (and therefore is used in the mechanism) if the R_{AB} value is larger than the threshold you define as ε .

For example, consider the case in Figure 23.12, for the path from species *A* to *D*:

Chapter 23: Utilities

Chemistry Mechanism Reduction

$$R_{AD} \equiv \max_{\text{all paths } i} \{r_{AD,i}\} \equiv \max[(r_{AB}r_{BD}), (r_{AC}r_{CD}), (r_{AC}r_{CE}r_{ED})]. \quad (23.51)$$

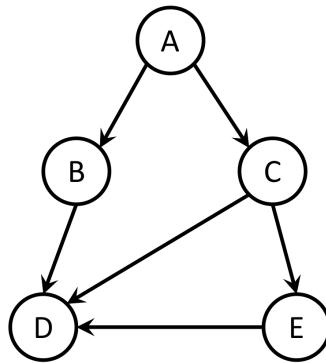


Figure 23.12: R_{AD} is the maximum of the path-dependent coefficients.

The maximum of the path-dependent coefficients for each species (with respect to A) as seen in Figure 23.11 (from the previous section) are shown below in Figure 23.13.

Species	RAi
A	1
B	0.5
C	0.6
D	0.12
E	0.48
F	0.3
G	0.006
H	0.036
I	0.096
J	0.01

Figure 23.13: Sample DRGEP interaction coefficients.

If you set $\varepsilon = 0.1$, then the species G, H, I , and J are removed from the original mechanism.

DRGEPSA (Directed Relation Graph Error Propagation with Sensitivity Analysis)

The Directed Relation Graph with Error Propagation and Sensitivity Analysis (DRGEPSA) algorithm ([Raju et al., 2012](#)) consists of two phases: (1) DRGEP and (2) Sensitivity Analysis (SA). Figure 23.14 shows a flow chart of the DRGEPSA algorithm.

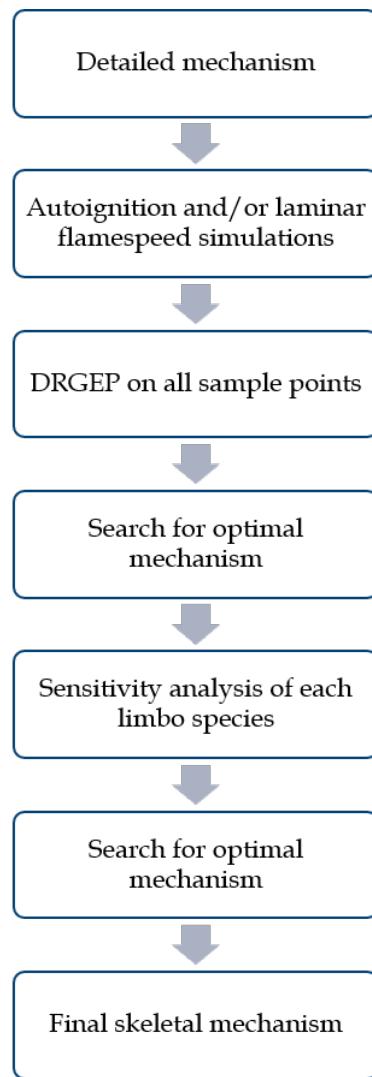


Figure 23.14: A flow chart depicting the DRGEPSA algorithm.

To begin the DRGEPSA process in CONVERGE, set up several [autoignition simulations](#) and/or [laminar flamespeed](#) calculations for the given range of initial temperature, pressure, and equivalence ratio conditions. For each case, a set of sample points are selected along the integration curve at 10 K intervals. For each sample point, the DRGEP coefficients are calculated based on a set of target species as specified in [mechanism_reduction.in](#). The maximum DRGEP interaction coefficient is termed the overall interaction coefficient (OIC).

Once the OICs for each species are determined, the species are ordered in descending fashion so that the species with the highest OIC values are at the top. Species are then divided into sets of size equal to the total number of species in the comprehensive mechanism divided by `drgep_control > search_iterations` (in [mechanism_reduction.in](#)). The first set of species is used to run autoignition simulations and/or laminar flamespeed calculations and the output is compared with the ignition delay and/or flamespeed calculated using the comprehensive mechanism. The error between the reduced mechanism

and comprehensive mechanism simulations is compared to the cut-off tolerance(s) ([*mechanism_reduction.in*](#) > *drgep_control* > *zero_d_control* > *ignition_delay_error* and/or [*mechanism_reduction.in*](#) > *drgep_control* > *one_d_control* > *flamespeed_error*). If the error does not satisfy the cut-off tolerance(s), then the next set of species is added to the first set and the autoignition simulations and/or laminar flamespeed calculations are re-run. This process continues, adding more sets of species in a cumulative fashion, until the error satisfies the cut-off tolerance(s). The optimal mechanism is chosen as the smallest mechanism whose error in the ignition delay and/or flamespeed calculations is within the user-specified tolerance limit.

Once the optimal mechanism is obtained from the DRGEP step, sensitivity analysis is performed to further reduce the size of the mechanism. In the sensitivity analysis phase, the OIC values of the species in the skeletal mechanism are arranged in ascending order and a fixed fraction ([*mechanism_reduction.in*](#) > *sensitivity_control* > *subset_species_analyzed*) of the species from the top of the list is chosen for sensitivity analysis. These species are the **limbo species**.

The error induced by each of the limbo species is calculated by removing this species from the DRGEP-generated skeletal mechanism. The error in the ignition delay and/or laminar flamespeed is then calculated for the resulting skeletal mechanism. This error is an indication of the sensitivity of the prediction of the ignition delay time/laminar flamespeed to the limbo species.

Now the species are arranged in ascending order based on this error. In the next step, the optimal skeletal mechanism is determined from the sensitivity analysis, which is accomplished by removing the species identified from the top of the list (*i.e.*, the species with the least error) one by one from the DRGEP-generated mechanism until the error generated by the resulting skeletal mechanism is less than the user-defined tolerance for ignition delay and/or laminar flamespeed. Thus the optimal skeletal mechanism is obtained. All the steps in the algorithm are executed sequentially.

In CONVERGE, mechanism reduction can be executed in parallel.

Mechanism Merge

Because most mechanisms are developed for specific fuels, it can be difficult to find a mechanism that includes all of the required fuel species. It may be convenient to merge two reactions or to develop a surrogate fuel and add extra species and reactions to a mechanism. The *mechmerge* utility can be used for either of these tasks.

It is important to use the *mechmerge* utility with care. The reaction path of the resulting mechanism may deviate from the original, and merging mechanisms can change quantities such as flamespeed and ignition delay. Due to the complexity of the mechanism merge utility, we recommend using CONVERGE Studio to run it. Refer to the Chemistry chapter in the CONVERGE Studio 3.1 Manual.

Running *mechmerge*

The *mechmerge* utility requires two [reaction mechanism files](#) (*mech1.dat* and *mech2.dat*) and two [thermodynamic data files](#) (*therm1.dat* and *therm2.dat*), all of which must be saved in the Case Directory.

You must follow a three-step process to merge mechanisms via *mechmerge*. In the first two steps, CONVERGE searches the *mech*.dat* and *therm*.dat* files for conflicts. After you resolve any conflicts, in the third step CONVERGE writes the final [reaction mechanism file](#) and [thermodynamic data file](#).

Find Species Conflicts

From the Case Directory, enter

```
<CONVERGE executable> -u mechmerge -s
```

or

```
<CONVERGE executable> -u mechmerge -s -2
```

where *<CONVERGE executable>* is any CONVERGE executable (e.g., *converge-mpich*). CONVERGE will write three output files: *mech_check1.dat*, *mech_check2.dat*, and *conflicts1.dat*. CONVERGE lists each species and reaction mechanism for the corresponding *mech*.dat* file in *mech_check1.out* and *mech_check2.out*. The ELEMENTS section of the *conflicts1.dat* contains a list of elements that are found in only one of the two *mech*.dat* files. The SPECIES section of *conflicts1.dat* differs based on which command you use. The first command populates *conflicts1.dat* with conflicts due to species that contain the same elements but have different names or species that have the same name but contain different elements. The second command (with the *-2* argument) populates *conflicts1.dat* with species that are found in only one of the *mech*.dat* files in addition to the previously listed conflicts.

Edit *conflicts1.dat* to resolve conflicts in the SPECIES section. (You do not need to make any changes to the ELEMENTS section.) If you do not resolve SPECIES conflicts, CONVERGE will write conflicting species to the final [reaction mechanism file](#) in the third step of *mechmerge*, which may lead to incorrect simulation results. Figure 23.15 below shows an example *conflicts1.dat* file with corrections. In the SPECIES section of *conflicts1.dat*, a comment character (!) precedes the conflict descriptions. After the conflict description, there is a list of each species that is part of the conflict. A 1/ (for *mech1.dat*) or a 2/ (for *mech2.dat*) and a number that matches the species number found in the corresponding *mech_check*.out* file precede the species name. For example, if 1/51/C12H25O2 is the species description, C12H25O2 is the species found in *mech1.dat* and is listed as 51 in *mech1_check.out*. If two species have the same name, one species must either be deleted or renamed because the final [reaction mechanism file](#) in CONVERGE cannot have any species with duplicate names.

To delete a species in *conflicts1.dat*, type /0 without any spaces after the species name, e.g., 2/175/C12H25/0. To rename a species, type a new name between / symbols, e.g., 2/16/T-CH2/CH2/.

Chapter 23: Utilities

Chemistry Mechanism Merge

```
ELEMENTS
HE
G
END
.

.

.

SPECIES
! mech1 species not found in mech2 and are not conflicting
1/51/C12H25O2
1/52/C12OOH
1/53/O2C12H24OOH
! mech2 species not found in mech1 and are not conflicting
2/14/C
2/15/CH
2/158/C7H13/0
2/180/HE/0
! ELEMENTS are the same between mech1 and mech2 species
1/11/CH2          2/16/T-CH2/CH2/
1/11/CH2          2/23/S-CH2
1/12/CH2*         2/16/T-CH2
1/12/CH2*         2/23/S-CH2/CH2*/
1/26/AC3H5        2/44/S-C3H5
1/26/AC3H5        2/47/A-C3H5/AC3H5/
1/26/AC3H5        2/54/T-C3H5
1/49/S3XC12H25   2/175/C12H25/0
! NAMES are the same between mech1 and mech2 species
END
```

Figure 23.15: Sample of conflicts and edits in the SPECIES section of *conflicts1.dat*.

Find Thermodynamic and Reaction Conflicts

Once you have edited the *conflicts1.dat* file, enter

```
<CONVERGE executable> -u mechmerge -r
```

or

```
<CONVERGE executable> -u mechmerge -r -2
```

With either command, CONVERGE writes the *conflicts2.dat* file. If you use the first command, *conflicts2.dat* lists thermodynamic and reaction conflicts based on edits from *conflicts1.dat*. If you use the second command (with the *-2* argument), *conflicts2.dat* does not list conflicts because CONVERGE uses *mech1.dat/therm1.dat* information as a default unless *mech2.dat* has new species and corresponding reactions. Since no conflicts are listed when using the second command, the remaining information in this section will refer to the *conflicts2.dat* file generated from the first command.

In the THERMO section of *conflicts2.dat*, CONVERGE writes the thermodynamic conflicts between duplicate species (*i.e.*, same name and elements) that have different LOW or HIGH thermodynamic coefficients. In the REACTIONS section, CONVERGE writes the reaction conflicts due to the species having the same names but have differences in the reaction coefficients for FORWARD, DUPLICATE, REVERSE, LOW, TROE, PLOG, FORD, LUMP, and THIRD-BODY reaction types. The *conflicts2.dat* file groups the reaction conflicts based on the type of reaction.

- Same species but different FORWARD rate coefficients

Chapter 23: Utilities

Chemistry Mechanism Merge

- Same species and forward rate coefficients but different REVERSE rate coefficients
- Same species and forward rate coefficients but different LOW rate coefficients
- Same species and forward rate coefficients but different TROE coefficients
- Same species and forward rate coefficients but different PLOG pressures and rate coefficients
- Same species and forward rate coefficients but different FORD species and coefficients
- Same species and forward rate coefficients but different LUMP rate coefficients
- Same species and forward rate coefficients but different THIRD-BODY species and coefficients
- Same species but different FORWARD rate coefficients but where one mech has a group of DUPLICATE reactions or where one mech has a group of only nonreversible (forward only, =>) reactions choose the single or group of reactions to be merged
- Same species but different FORWARD rate coefficients and are designated DUPLICATE
- Please delete which DUPLICATE group not wanted, the first reaction of each group is listed

Figure 23.16 shows an example conflict in the THERMO section. The conflict arises because the species CO, which has the same elements and number of elements in both mechanisms, is found to have different LOW and HIGH coefficients. Because the *therm1.dat* file lists the species as number 16, and the *therm2.dat* file lists the species as number 15, CONVERGE lists the conflicted species as [16][15/] in *conflicts2.dat*.

Chapter 23: Utilities

Chemistry Mechanism Merge

A species requires exactly one set of thermodynamic data. The *mechmerge* utility automatically uses *therm1.dat* data unless the first number in brackets has / added afterwards, instead of the second number. For example, to choose the thermodynamic data from *therm2.dat*, you must edit the species numbers in *conflicts2.dat* in Figure 23.16 as [16/] [15].

There should be no thermodynamic data conflicts for species with only the same elements. If there is a conflict listed, the *mechmerge* utility is not reliable. In this case, please contact the Convergent Science Applications team for assistance.

```
THERMO
!
! species with the same NAME and ELEMENTS
!
[16] [15/]
CO          C   10    1          G   300.00   5000.00 1000.00      1
         3.02507800E+00 1.44268900E-03-5.63082800E-07 1.01858100E-10-6.91095200E-15      2
-1.42683500E+04 6.10821800E+00 3.26245200E+00 1.51194100E-03-3.88175500E-06      3
5.58194400E-09-2.47495100E-12-1.43105400E+04 4.84889700E+00      4
CO          C   10    1          G   200.00   3500.00 1000.00      1
         2.71518561E+00 2.06252743E-03-9.98825771E-07 2.30053008E-10-2.03647716E-14      2
-1.41518724E+04 7.81868772E+00 3.57953347E+00-6.10353680E-04 1.01681433E-06      3
9.07005884E-10-9.04424499E-13-1.43440860E+04 3.50840928E+00      4
!
! species with only the same ELEMENTS, ERROR if any are listed
!
END
```

Figure 23.16: Sample conflict in the THERMO section of *conflicts2.dat*.

You edit the REACTIONS conflicts, as shown in Figure 23.17, with the same procedure as the THERMO section. Each reaction conflict has numbers in brackets that correspond to the reaction number found in the *mech_check*.out* files. Similar to the THERMO section, the *mechmerge* utility automatically uses the rate coefficients from *mech1.dat* unless the first number in brackets has / after it. Then, the *mechmerge* utility uses the *mech2.dat* rate coefficients for that particular reaction.

```
REACTIONS
!
! same species but different FORWARD rate coefficients
!
! same species and forward rate coefficients but different REVERSE rate coefficients
!

[71] [112/]
C2H2+OH=CH2CO+H           2.19000E-04   4.500   -1.00000E+03
    REV/   2.16100E-03   4.500   1.96600E+04 /
C2H2+OH=CH2CO+H           2.19000E-04   4.500   -1.00000E+03
!
! ...different LOW rate coefficients
!
! ...different TROE coefficients
!
! ...different PLOG pressures and rate coefficients
!
! ...different FORD species and coefficients
!
! ...different LUMP rate coefficients
!
! ...different THIRD BODY species and coefficients
!
```

Chapter 23: Utilities

Chemistry Mechanism Merge

```
[87] [44/]
H+O2 (+M)=HO2 (+M)                                1.47500E+12   0.600   0.00000E+00
    LOW/      6.36600E+20   -1.720   5.24800E+02 /
    TROE/     8.00000E-01   1.00000E-30   1.00000E+30 /
    H2/2.00/H2O/11.00/O2/0.78/CO/1.90/CO2/3.80/
H+O2 (+M)=HO2 (+M)                                1.47500E+12   0.600   0.00000E+00
    LOW/      3.50000E+16   -0.410   -1.11600E+03 /
    TROE/     5.00000E-01   1.00000E-30   1.00000E+30 /
    H2/2.00/H2O/12.00/CO/1.90/CO2/3.80/
!
! same species but different FORWARD rate coefficients but
! where one mech has a group of DUPLICATE reactions or
! where one mech has a group of only nonreversible (forward only, =>) reactions
! choose the single or group of reactions to be merged
!
[92] [51/]
!mech1 [92][93]
HO2+HO2=H2O2+O2                                4.20000E+14   0.000   1.19820E+04
    DUPLICATE
HO2+HO2=H2O2+O2                                1.30000E+11   0.000   -1.62930E+03
    DUPLICATE
HO2+HO2=H2O2+O2                                3.00000E+12   0.000   0.00000E+00
!
! same species but different FORWARD rate coefficients and are designated DUPLICATE
! please delete which DUPLICATE group not wanted, the first reaction of each group is listed
!
END
```

Figure 23.17: Sample of conflicts in the REACTIONS section of *conflicts2.dat*.

Generate Merged Mechanism

Finally, enter

```
<CONVERGE executable> -u mechmerge -m
```

or

```
<CONVERGE executable> -u mechmerge -m -2
```

to merge the two mechanisms. Include the `-2` argument only if you included the same argument in the previous step. With either command, the utility reads the mechanism and thermodynamic files (*mech1.dat*, *therm1.dat*, *mech2.dat*, and *therm2.dat*) along with the edited *conflicts1.dat* and *conflict2.dat* files. Without the `-2` argument, the utility joins together all of the original information along with the edits. With the `-2` argument, the utility joins together all of the duplicate reactions and all of the new reactions with new species found in *mech2.dat*. Both commands yield the final [reaction mechanism file](#) and [thermodynamic data file](#) that you can use to run a simulation.

Mechanism Tune

Most available chemical mechanisms are not validated under your specific simulation conditions. Also, ignition delay and laminar flamespeed data are not widely available for real fuel mixtures or dilute EGR mixtures. It is necessary, therefore, to adjust the reaction rates of selected reactions (by adjusting the pre-exponential factors) to match ignition delay or laminar flamespeed targets determined from experimental data. By doing so, you optimize the mechanism for specific operating points of interest. Tuning a mechanism in CONVERGE can be done with either CONGO or [NLopt, an open-source library for nonlinear optimization](#).

Mechanism Tune with CONGO

Since the setup of CONGO can be repetitive and time-consuming, you can use the mechanism tune utility as a pre-processor to set up your input files.

When [*mechanism_tune.in*](#) > *solver_type* = CONGO, the mechanism tune utility is a pre-processor for performing [CONGO genetic algorithm \(GA\) optimization](#) of the Arrhenius equation reaction A factors. During the GA optimization, the reaction A factors in the [reaction mechanism file](#) will be tuned based on user-specified target ignition delays and/or laminar flamespeeds.

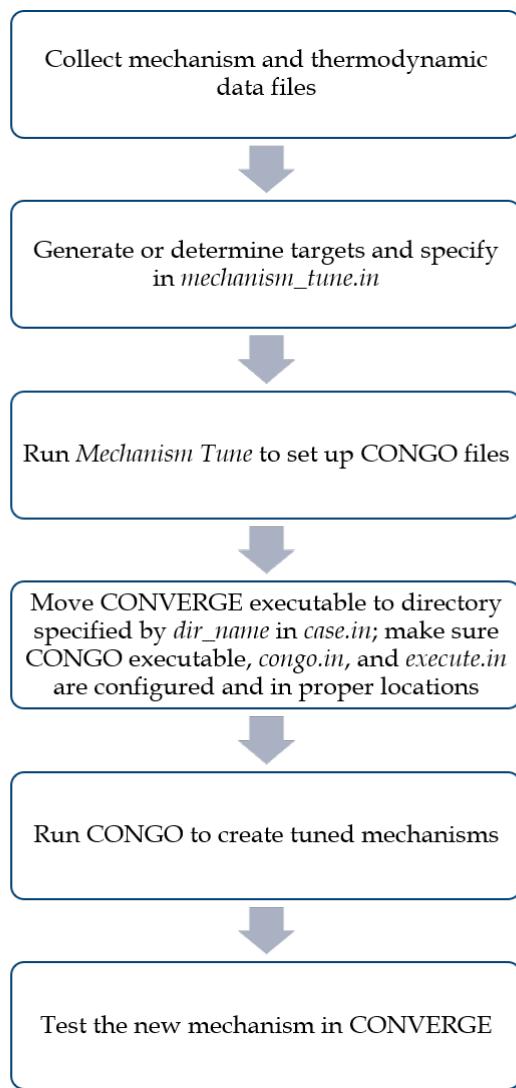


Figure 23.18: Workflow for mechanism tune and CONGO.

The mechanism tune utility runs the [zero-dimensional solver with adjoint sensitivity analysis \(ASA\)](#) and/or the [1D premixed laminar flamespeed solver](#) (based on Newton's

method) with sensitivity analysis. Then it combines the [*zero_d_adjoint_rank.out*](#) and/or the [*one_d_sens_rank.out*](#) files, respectively, to extract a user-specified number of the most sensitive reactions ([*mechanism_tune.in*](#) > [*one_d_control*](#) > [*num_reactions_to_modify*](#) or [*mechanism_tune.in*](#) > [*zero_d_control*](#) > [*num_reactions_to_modify*](#)) from the [*reaction mechanism file*](#) and lists these rankings and reaction numbers in [*mechanism_tune.out*](#). Mechanism tune then sets up files that will enable adjusting the A factors of each reaction listed in [*mechanism_tune.out*](#) along with a file ([*merit.in*](#)) that combines the targets for optimization specified in [*mechanism_tune.in*](#). The [*case.in*](#) and [*merit.in*](#) files are written into the directory specified by [*mechanism_tune.in*](#) > [*congo_control*](#) > [*directory_name*](#). Subsequently, [*inputs.in*](#), a modified [*reaction mechanism file*](#) (with [*markers*](#) listed for the relevant A factors), and copies of the [*thermodynamic data file*](#), [**_d_cases.in*](#), and [**_d_solver.in*](#) are written to a subdirectory of [*mechanism_tune.in*](#) > [*congo_control*](#) > [*directory_name*](#) specified by [*case.in*](#) > [*dir_name*](#).

After the utility has run, you must move the CONVERGE executable to the [*dir_name*](#) directory specified in [*case.in*](#) and then run the CONGO executable in order to tune your mechanism. Note that [*congo.in*](#), [*execute.in*](#), and all of the secondary CONGO files are required in the Case Directory in order to be moved to the proper subdirectories during the execution of the mechanism tune utility. If these files are not present in the Case Directory at the time of running mechanism tune, you will have to manually move them to the directory specified by [*mechanism_tune.in*](#) > [*congo_control*](#) > [*directory_name*](#) before running CONGO. During a CONGO run, your [*reaction mechanism file*](#) (e.g., *mech.dat*) and [*inputs.in*](#) will be edited by CONGO. CONGO finds these files based on the configuration of [*congo.in*](#). If you use files named *mech.dat* (recommended) and *inputs.in*, you must set *input_files* = 1 and *file_name* = *DEFAULT* in [*congo.in*](#). If you use a [*reaction mechanism file*](#) with name other than *mech.dat* (e.g., *mech2.dat*), you must set *input_files* = 2, *file_name* = *mech2.dat*, and *file_name* = *inputs.in* in [*congo.in*](#).

Mechanism Tune with CONGO Setup

First, choose to optimize your reaction mechanisms based on ignition delay targets (0D simulation), laminar flamespeed targets (1D simulation), or both. Choose the appropriate solver for your target, since you can modify the reactions in the [*reaction mechanism file*](#) based on 0D solver's adjoint sensitivity analysis (ASA), 1D Newton solver's sensitivity analysis, or both. Table 23.3 below shows the different combinations by which [*reaction mechanism files*](#) can be tuned using the mechanism tune utility. Each row lists the allowed solver(s) you can use for the mechanism tune utility that are compatible with the selected target(s) in CONGO. For example, if you want to use only laminar flamespeed targets to tune your mechanism in CONGO, you must use the 1D Newton solver with sensitivity analysis in the mechanism tune utility. If you want to use both ignition delay targets and laminar flamespeed targets in CONGO, you can use either the 0D solver with ASA, the 1D Newton solver with sensitivity analysis, or both. In the case where you choose to use only the 0D solver with ASA with ignition delay and laminar flamespeed targets, CONGO will adjust the reaction A factors for those reactions determined sensitive according to the 0D solver with ASA while monitoring both ignition delay and laminar flamespeed targets.

Chapter 23: Utilities

Chemistry Mechanism Tune

Table 23.3: Compatibility of CONGO targets with mechanism tune solvers.

	0D solver with adjoint sensitivity analysis	1D Newton solver with sensitivity analysis	0D solver with adjoint sensitivity analysis and 1D solver with sensitivity analysis
Ignition Delay Targets	Yes	No	No
Laminar Flamespeed Targets	No	Yes	No
Both Ignition Delay Targets and Laminar Flamespeed Targets	Yes	Yes	Yes

Mechanism Tune with NLOpt

When `mechanism_tune.in > solver_type = NLOPT`, the mechanism tune utility adjusts the Arrhenius equation reaction A factors and/or the activation energy to tune the [reaction mechanism file](#) to your specified target ignition delays and/or laminar flamespeeds. Note that with `mechanism_tune.in > solver_type = NLOPT`, you are actually tuning the mechanism (*i.e.*, with this solver type, mechanism tune is not a pre-processor for CONGO). Set the NLOpt model in `mechanism_tune.in > nlopt_control > model_type` and specify parameters in the `nlopt_control` settings block. When tuning a reaction in [P-Log format](#), the reaction P-log A factors are multiplied by the same multiplier as the A factors. Note that CONVERGE does not support mechanism tuning of reactions in Chebyshev format.

For more information about the implemented algorithms, visit https://nlopt.readthedocs.io/en/latest/NLOpt_Algorithms/.

Group Dependency Control in Mechanism Tune with NLOpt

If your simulation's deviation displays a trend with temperature (*e.g.*, deviation increases/decreases with temperature), you can use the group dependency control feature to more quickly tune your mechanism. CONVERGE will analyze the temperature range of all your cases from `one_d_cases.in` and generate four cases (a case at high temperature, a case at low temperature, and two cases near the middle temperature). Based on that analysis, the most sensitive reactions from the four cases are grouped into up to five groups based on their contribution to the simulation (*e.g.*, one group may have a promoting effect on flamespeed that is more obvious at higher temperatures, while another group may have an inhibiting effect that is more obvious at lower temperatures). All the reactions within a group are then tuned by the same factor. Tuning an entire group of reactions is quicker than tuning the individual reactions within the group.

Note that this feature is only available when running mechanism tune with the 1D laminar flamespeed solver. To activate this feature, set `mechanism_tune.in > nlopt_control > main_reaction_control > group_dependency_control > active = 1` and `group_dependency_control > type = TEMPERATURE`.

Running Mechanism Tune

The format for a serial execution of a mechanism tune calculation is:

```
<CONVERGE executable> -u mechture
```

where `<CONVERGE executable>` is any CONVERGE executable (*e.g.*, `converge-mpich`).

Inputs for Mechanism Tune Utility

The required input files are dependent on your choices above. You will need:

- a [reaction mechanism file](#) (*e.g.*, `mech.dat`)
- a [thermodynamic data file](#) (*e.g.*, `therm.dat`)
- [`transport.dat`](#) or [`gas.dat`](#)
- [CONGO files](#) (`congo.in`, `execute.in`), the CONGO executable, and any system files listed in `execute.in` [only when `mechanism_tune.in > solver_type = CONGO`])
- [`mechanism_tune.in`](#)
- solver files ([`zero_d_cases.in`](#) and [`zero_d_solver.in`](#) for the 0D solver, [`one_d_cases.in`](#) and [`one_d_solver.in`](#) for the 1D Newton solver)

The [`mechanism_tune.in`](#) file specifies the information required to run the utility and lists the targets toward which the reaction mechanism will be tuned. Note that if you use the 0D solver, you must specify [`mechanism_tune.in > ignitiondelay`](#) targets. If you use the 1D Newton solver, you must specify [`mechanism_tune.in > flamespeed`](#) targets.

Outputs for Mechanism Tune Utility

When this utility is finished, it will write [`mechanism_tune.out`](#), which specifies the reaction numbers of the reactions that will be modified during the GA as well as setup files for a CONGO run. You must then move the CONVERGE executable to the `dir_name` directory specified in `case.in` and then run the CONGO executable to tune your reaction mechanism. Note that you can open and edit the generated CONGO input files ([`case.in`](#) and [`merit.in`](#)) in CONVERGE Studio or in an editor prior to running CONGO if you wish to change target ([`merit.in`](#)) or reaction ([`case.in`](#)) parameters.

Surrogate Blender

A multi-component surrogate fuel is a mixture of simple fuels whose properties (*e.g.*, cetane number, flamespeed) approximate those of a complex target fuel. Key components of the target fuel can be mixed to create a surrogate, and the composition of the mixture may vary depending on which properties you want to approximate. To help you determine the optimal mixture composition for your problem, CONVERGE Studio includes a surrogate blender tool.

The surrogate blender assumes that mixture properties are a function of its components' properties and mole fractions. In other words,

$$\phi_s = \sum_i \phi_i x_i, \quad (23.52)$$

where ϕ_s is any property of the mixture, ϕ_i is the same property of component i , and x_i is the mole fraction of component i . The blender minimizes f , which is defined as

$$f = \sum_j w_j R_j^2, \quad (23.53)$$

where w_j is the importance (or weight) of property j and R_j is the error, defined as

$$R_j = T_j - \sum_i \phi_{ij} x_i, \quad (23.54)$$

where T_j is the value of property j for the target fuel T . The problem is subject to two constraints: x_i must be greater than zero and the sum of x_i must equal 1. The blender uses the [NLOpt optimization package](#) to optimize the molar concentrations for the fuel components. For more information on the models and algorithms used in the surrogate blender, see [Ghosh et al., 2005](#); [Jameel et al., 2016](#); and [Jameel et al., 2018](#).

For instructions on how to run the blender, please refer to the CONVERGE Studio 3.1 Manual. After running, you must convert the resultant mole fractions to mass fractions for use in a CONVERGE simulation.

The surrogate blender can import and export [`blender.in`](#) files and can export [`blender.out`](#) files. The [`blender.in`](#) file is used only to store the configuration for the surrogate blender and is not used as an input file for a command line execution of CONVERGE. CONVERGE Studio is the only way to run the surrogate blender tool.

Pathway Flux Analysis (PFA)

Chemical kinetic mechanisms attempt to model complex chemical reactions that proceed through a reaction network. Visualizing the reaction network can aid in understanding the chemical processes as well as the kinetic mechanisms. In order to visualize the kinetics and the contributions of various pathways, the pathway flux analysis (PFA) utility calculates the reaction rates based on the kinetic mechanism, relevant species concentrations in 0D or 1D solution files, and settings specified in the PFA settings file ([`pfa.in`](#)). For each species, there can be multiple consumption reactions and their contribution percentages are calculated by dividing each reaction rate by the total consumption rate.

PFA is applicable to 0D and 1D simulations. First, run a 0D simulation in which [`zero_d_solver.in`](#) > [`zero_d_output_control`](#) > [`output_file_flag` = 1](#) or run a 1D simulation in which [`one_d_solver.in`](#) > [`one_d_general_control`](#) > [`output_control`](#) > [`output_file_flag` = 1](#). These settings will direct CONVERGE to write the 0D ([`\(zero_d_sol_case<case ID>.out\)`](#)) or 1D ([`\(one_d_sol_case<case ID>.out\)`](#)) solution files, which PFA requires. PFA also requires the [reaction mechanism file](#) and [thermodynamic data file](#) and the [`pfa.in`](#) file to be saved in the Case Directory. To execute PFA, type:

```
<CONVERGE executable> -u pfa
```

where <CONVERGE executable> is any CONVERGE executable (e.g., converge-mpich). PFA generates output files for analyzing the species mole fractions ([spec_mol <number>.out](#)) and the reaction rates ([rate_tab <number>.out](#)). Use the *Chemistry* module of CONVERGE Studio to visualize the data in [rate_tab <number>.out](#) file. PFA also writes a [graphviz_input <number>.out](#) file for plotting a pathway diagram using CONVERGE Studio. Species are plotted as nodes and reactions are plotted as arrows with contribution percentages and reaction numbers as arrow labels. The width of the arrows is scaled to the absolute reaction rate, where thicker arrows represent faster reaction rates. The color of the arrows represents the contribution, where redder arrows are reactions that contribute more to the consumption of the reactant.

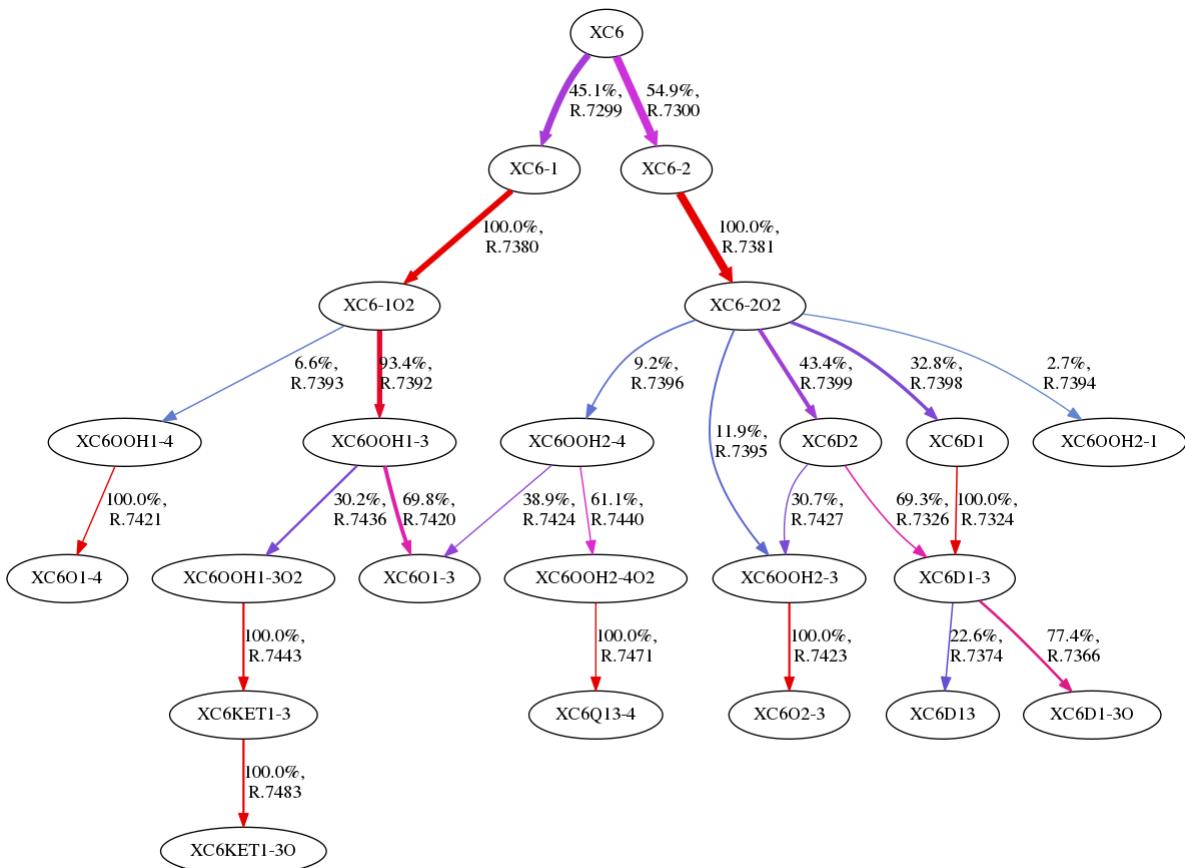


Figure 23.19: A diagram for PFA analysis performed at a single time-step. The width of the arrows is scaled to the absolute reaction rate (thicker arrows represent faster reaction rates). The color of the arrows correlate to the percentage of contribution to consumption (redder arrows represent higher contribution to the consumption of the species). The contribution percentage is provided next to the arrow along with the reaction number from the reaction mechanism file (*mech_check.out*).

PFA can be performed for a single time-step if `pfa.in > pfa_cases > cases > start` and `pfa.in > pfa_cases > cases > end` are identical. If `start` and `end` are identical but this value does not correspond to a value given in the solution file, the information is interpolated based on the two nearest time-steps.

PFA can also be performed over a period of time. If the span between `start` and `end` is shorter than the time-step in the solution file, an analysis will be performed at the mid-point between them. If the mid-point is not an exact time value in the solution file, data will be interpolated based on the `start` and `end` values. When PFA is performed over a period of time that is larger than the time-step in the solution file, the contribution percentages will be calculated based on an integrated contribution, which is calculated by multiplying the reaction rates by time-step sizes. In this case, the contribution values may not be accurate for time spans in which the concentrations of species are changing rapidly.

Chapter



24

File Overview

24 File Overview

This chapter gives an overview of the file types in CONVERGE. The figure below shows the CONVERGE workflow and where in the workflow you will encounter the various file types.

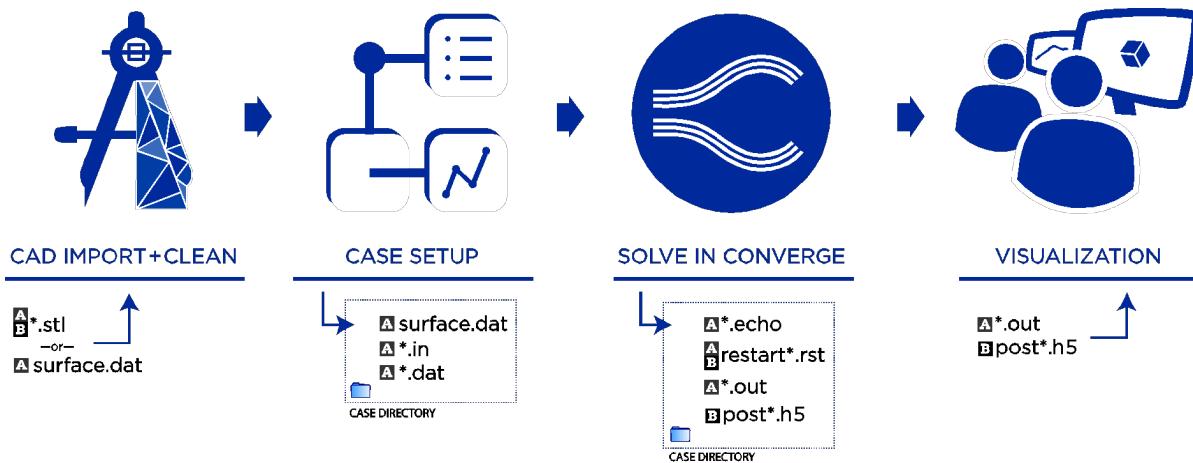


Figure 24.1: CONVERGE workflow and file types. A denotes ASCII files. B denotes binary files.

24.1 Input and Data File Overview

Before running a CONVERGE simulation, you need to prepare a set of input (**.in*) and data (**.dat*) files. The input files contain numerical inputs, model parameters, and boundary and initial conditions. The data files contain thermodynamic properties, chemical reaction data, and surface geometry information.

Most CONVERGE input and data files are [plain text-formatted in YAML](#), a data serialization language. You can create or modify the input and data files via text editor or you can work through the *Case Setup* process in the CONVERGE Studio graphical user interface and then export the input and data files. To ensure that the input and data files are formatted correctly, we recommend generating the files with CONVERGE Studio. Some CONVERGE features require setup in CONVERGE Studio.

All of the input and data files must be saved to the Case Directory or its subdirectories (the latter apply only to multi-stream simulations with [stream-based overrides](#)). If you start a simulation and a required file is missing, CONVERGE will exit and indicate which file is needed. We recommend creating a separate Case Directory for each case rather than overwriting files within an existing Case Directory.

At minimum, the Case Directory must contain [a surface geometry file](#) (e.g., *surface.dat*), [boundary.in](#), [initialize.in](#), [inputs.in](#), and [stream_settings.in](#). The [solver.in](#) file is also required for all simulations and, depending on the case setup, can be saved only to the Case Directory, only to stream-specific subdirectories, or to both the Case Directory and stream-specific

Chapter 24: File Overview

Input and Data File Overview

subdirectories. Additional input and data files may be required depending on the materials, models, and features in use. [Chapter 25 - Input and Data Files](#) contains detailed information about all of the available input and data files.

Check Inputs

To verify your input and data files without checking out a license from the server, type the following command in your Case Directory:

```
<CONVERGE executable> -u check_inputs
```

where `<CONVERGE executable>` is any CONVERGE executable (*e.g.*, `converge-mpich`).

CONVERGE will exit before starting the main solver but will write [screen output](#), the [log file](#), and the [active variables log](#).

Hidden Input Parameters

CONVERGE recognizes an optional *hidden.in* input file. You can use this file to set up parameters for features that are in the experimental phase. If a feature moves from experimental to official, then the relevant parameters are moved to other input files and documented in this manual. The *hidden.in* file is [YAML-formatted](#) in the same manner as other CONVERGE 3.1 input files, but there is no block structure (*i.e.*, all parameters are at the zero-indentation level). There are no required parameters in *hidden.in*, but there are dependencies between some parameters. If you have questions about *hidden.in*, please contact the Convergent Science Applications team.

24.2 Simulation Logs

During a simulation, CONVERGE generates [screen output](#), a [log file](#), a [list of variables](#), a [process ID log](#), and a latency log, all of which are described below.

Screen Output

During a simulation, CONVERGE writes information to the screen and to a log file (*e.g.*, `converge.log`). The screen output records the progress of the simulation while the log file contains more detailed information about the simulation. This section describes the screen output. The [log file](#) is described in the next section.

Figure 24.2 below shows the header portion of the screen output, which includes the CONVERGE version number.

Chapter 24: File Overview

Simulation Logs Screen Output

```
*****
* This software is Proprietary to Convergent Science Inc. (2008-2021)
* CONVERGE Internal Build 3.1.0
* Apr 21 2021
* MPI Version: MPICH 3.2.1
*
*
* CEQ equilibrium solver is included under license agreement with
* Ithaca Combustion Enterprise LLC.
*
* Adaptive zoning chemistry solver is included under license agreement with
* Lawrence Livermore National Security, LLC. All rights reserved.
*
*****
```

Figure 24.2: Header information in the screen output.

CONVERGE writes simulation status reports to the screen output. CONVERGE also writes to the screen output as it reads the input and data files, sets up the mesh, encounters situations for which a warning is justified, and writes output and restart files.

Figure 24.3 below shows example screen output for a case that terminated normally. You can redirect the screen output to a file (such as *log.out*). Note that the screen output contains hidden characters that allow it to be rendered in color on many terminals. If you redirect the screen output to a file, the file may show these characters (e.g., `ESC[35m`). If you view the screen output on the screen or in an appropriate text viewer (e.g., *cat*), these hidden characters will be correctly rendered.

```
reading stream_settings.in data from file stream_settings.in
Searching for stream 0 overwrite files at stream0
(Stream: 0)
For Stream 0, detected the following overwrite files:(Stream: 0)
turbulence.in(Stream: 0)
inputs.in(Stream: 0)
embedded.in(Stream: 0)

(Stream: 0)
Searching for stream 1 overwrite files at stream1
(Stream: 1)
For Stream 1, detected the following overwrite files:(Stream: 1)
turbulence.in(Stream: 1)
inputs.in(Stream: 1)
embedded.in(Stream: 1)

(Stream: 1)
Found the file 'inputs.in' at '/home/pipe_with_2_streams/stream0/inputs.in'.(Stream: 0)
Found the file 'inputs.in' at '/home/pipe_with_2_streams/stream1/inputs.in'.(Stream: 1)
reading data from file inputs.in
reading data from file inputs.echo(Streams: 0-1)
reading /home/pipe_with_2_streams/stream0/inputs.in data from
file /home/pipe_with_2_streams/stream0/inputs.in(Stream: 0)
reading /home/pipe_with_2_streams/stream1/inputs.in data from
file /home/pipe_with_2_streams/stream1/inputs.in(Stream: 1)
Stream: 0: turbulence_solver_flag = 1 in inputs.in; reading in turbulence.in.(Stream: 0)
reading turbulence.in data from file turbulence.in(Stream: 0)
Stream: 1: turbulence_solver_flag = 1 in inputs.in; reading in turbulence.in.(Stream: 1)
reading turbulence.in data from file turbulence.in(Stream: 1)
Stream 0: calculated log-law parameters in 4 iterations: E = 10.074425, yplus_transition =
11.266209 from law_kappa = 0.420000 and law_c = 5.500000
(Stream: 0)
Stream 1: calculated log-law parameters in 4 iterations: E = 10.074425, yplus_transition =
11.266209 from law_kappa = 0.420000 and law_c = 5.500000
```

Chapter 24: File Overview

Simulation Logs Screen Output

```
(Stream: 1)
creating pathname outputs_original/output

reading      species.in data from file species.in
There are 0 reactions and 0 species.
Reading thermodynamic information from therm.dat...(Stream: 0)
There are 0 reactions and 0 species.
Reading thermodynamic information from therm.dat...(Stream: 1)
reading      initialize.in data from file initialize.in(Streams: 0-1)
reading      solver.in data from file solver.in(Stream: 0)
reading      gas.dat data from file gas.dat(Stream: 0)
reading      gas.dat data from file gas.dat(Stream: 1)
reading      boundary.in data from file boundary.in
reading      surface.dat data from file surface.dat
reading      post.in data from file post.in(Stream: 0)
reading      embedded.in data from file embedded.in(Stream: 0)
reading      embedded.in data from file embedded.in(Stream: 1)
WARNING:
Stream group formed for FLUID streams [0, 1] connected by FLOW_THROUGH boundaries.
```

All equations are loosely coupled and see update from other streams in the group across PISO/SIMPLE iterations.

These streams can be solved together in one matrix for some equations if these settings are made the same:

- (1) Coupled streams should all be transient or all be steady

```
Calculating LES inflow BCs...
Writing post file output/post000001_+0.00000e+00.h5 .....
Writing post file output/post000001_+0.00000e+00.h5 complete
Writing restart file outputs_original/restart0001.rst .....(Stream: 0)
Finished writing restart file outputs_original/restart0001.rst
[ 1.00%] ncyc      1; Time    1.00000000e-06 (sec); dt=    1.25000000e-06 (dt_grow)
(Stream: 0)
[ 5.00%] ncyc      1(Stream: 1)
[ 2.25%] ncyc      2; Time    2.25000000e-06 (sec); dt=    1.56250000e-06 (dt_grow)
(Stream: 0)
[ 10.00%] ncyc     2(Stream: 1)
[ 3.81%] ncyc      3; Time    3.81250000e-06 (sec); dt=    1.95312500e-06 (dt_grow)
(Stream: 0)
[ 15.00%] ncyc     3(Stream: 1)
[ 5.77%] ncyc      4; Time    5.76562500e-06 (sec); dt=    2.441406250e-06 (dt_grow)
(Stream: 0)
[ 20.00%] ncyc     4(Stream: 1)
[ 8.21%] ncyc      5; Time    8.207031250e-06 (sec); dt=    3.051757813e-06 (dt_grow)
(Stream: 0)
[ 25.00%] ncyc     5(Stream: 1)
[ 11.26%] ncyc     6; Time    1.125878906e-05 (sec); dt=    3.814697266e-06 (dt_grow)
(Stream: 0)
[ 30.00%] ncyc     6(Stream: 1)
[ 15.07%] ncyc     7; Time    1.507348633e-05 (sec); dt=    4.768371582e-06 (dt_grow)
(Stream: 0)
[ 35.00%] ncyc     7(Stream: 1)
[ 19.84%] ncyc     8; Time    1.984185791e-05 (sec); dt=    5.960464478e-06 (dt_grow)
(Stream: 0)
[ 40.00%] ncyc     8(Stream: 1)
[ 25.80%] ncyc     9; Time    2.580232239e-05 (sec); dt=    6.483109872e-06 (dt_cfld)
(Stream: 0)
[ 45.00%] ncyc     9(Stream: 1)
[ 32.29%] ncyc    10; Time    3.228543226e-05 (sec); dt=    6.739759353e-06 (dt_cfld)
(Stream: 0)
[ 50.00%] ncyc    10(Stream: 1)
[ 39.03%] ncyc    11; Time    3.902519161e-05 (sec); dt=    7.070972390e-06 (dt_cfld)
(Stream: 0)
[ 55.00%] ncyc    11(Stream: 1)
[ 46.10%] ncyc    12; Time    4.609616400e-05 (sec); dt=    7.471319632e-06 (dt_cfld)
(Stream: 0)
```

Chapter 24: File Overview

Simulation Logs Screen Output

```
[ 60.00%] ncyc      12(Stream: 1)
Writing post file output/post000002_+5.35675e-05.h5 .....
Writing post file output/post000002_+5.35675e-05.h5 complete
[ 53.57%] ncyc      13; Time      5.356748363e-05 (sec); dt=    7.936784639e-06 (dt_cfld)
(Stream: 0)
[ 65.00%] ncyc      13(Stream: 1)
[ 61.50%] ncyc      14; Time      6.150426827e-05 (sec); dt=    8.464811495e-06 (dt_cfld)
(Stream: 0)
[ 70.00%] ncyc      14(Stream: 1)
[ 69.97%] ncyc      15; Time      6.996907977e-05 (sec); dt=    9.054296785e-06 (dt_cfld)
(Stream: 0)
[ 75.00%] ncyc      15(Stream: 1)
[ 79.02%] ncyc      16; Time      7.902337655e-05 (sec); dt=    9.705379089e-06 (dt_cfld)
(Stream: 0)
[ 80.00%] ncyc      16(Stream: 1)
[ 88.73%] ncyc      17; Time      8.872875564e-05 (sec); dt=    1.041910464e-05 (dt_cfld)
(Stream: 0)
[ 85.00%] ncyc      17(Stream: 1)
[ 99.15%] ncyc      18; Time      9.914786028e-05 (sec); dt=    1.119732344e-05 (dt_cfld)
(Stream: 0)
[ 90.00%] ncyc      18(Stream: 1)
Writing post file output/post000003_+1.10345e-04.h5 .....
Writing post file output/post000003_+1.10345e-04.h5 complete
[100.00%] ncyc      19; Time      1.103451837e-04 (sec); dt=    1.119732344e-05 () (Stream:
0)
[ 95.00%] ncyc      19(Stream: 1)
[100.00%] ncyc      20; Time      1.103451837e-04 (sec); dt=    1.119732344e-05 () (Stream:
0)
[100.00%] ncyc      20(Stream: 1)
Writing post file output/post000004_+1.10345e-04.h5 .....
Writing post file output/post000004_+1.10345e-04.h5 complete
Writing map file map_1.103452e-04.h5 .....(Stream: 0)
Writing map file map_2.000000e+01.h5 .....(Stream: 1)
Writing map file map_1.103452e-04.h5 complete(Stream: 0)
Writing map file map_2.000000e+01.h5 complete(Stream: 1)
Writing map file map_bound1_1.103452e-04.h5 .....(Stream: 0)
Writing map file map_bound5_2.000000e+01.h5 .....(Stream: 1)
Writing map file map_bound1_1.103452e-04.h5 complete(Stream: 0)
Writing map file map_bound5_2.000000e+01.h5 complete(Stream: 1)
Writing restart file outputs_original/restart0002.rst .....(Stream: 0)
Finished writing restart file outputs_original/restart0002.rst
normal termination
Program used 8.595353 seconds.
```

Figure 24.3: An example of the screen output.

Log File

During a simulation, CONVERGE writes information to the [screen](#) (described previously) and to a log file (*e.g.*, *converge.log*). The screen output records the progress of the simulation while the log file contains more detailed information about the simulation. This section describes the log file and how to [adjust the level of detail](#) in this file.

By default, CONVERGE names the log file *converge.log*. To specify a different name for your log file, in the command line type

```
export CONVERGE_LOG_FILE=name
```

where *name* is the desired file name. Then CONVERGE will generate a log file called *<name>.log* and save it to the Case Directory.

Chapter 24: File Overview

Simulation Logs Log File

In this section, we will refer to the log file as *converge.log*. If your output directory contains a file named *converge.log*, when you run a new simulation CONVERGE will name the old file *converge.back* and write a new *converge.log* file. If your output directory contains a file named *converge.back*, when you run a new simulation CONVERGE will write a new *converge.log* file. If your output directory contains files named *converge.log* and *converge.back*, when you run a new simulation CONVERGE will rename *converge.log* to *converge.back* (your old *converge.back* file will be lost) and will write a new *converge.log* file.

The log file output is buffered. As a result, some portion of the log file may not be written if the simulation terminates abnormally. Figure 24.4 below shows the header portion of the log file, which includes license information, the time and date, the host name, and the CONVERGE version number.

```
successfully checked out 1 parent and 0 child licenses
Wed Apr 21 15:28:24 2021
vgl01.converge.global

*****
* This software is Proprietary to Convergent Science Inc. (2008-2021)
*           CONVERGE Internal Build 3.1.0
*           Apr 21 2021
*           MPI Version: MPICH 3.2.1
*
* CEQ equilibrium solver is included under license agreement with
*   Ithaca Combustion Enterprise LLC.
*
* Adaptive zoning chemistry solver is included under license agreement with
*   Lawrence Livermore National Security, LLC. All rights reserved.
*
*****
```

Figure 24.4: Header information in the log file.

CONVERGE writes information to the log file as the input and data files are read for each stream. Figure 24.5 below provides an example. CONVERGE prints additional information for some input files, such as the turbulence model coefficients ([turbulence.in](#)), the number of reactions in the [reaction mechanism file](#), or [disconnect triangle](#) setup ([events.in](#)). You can use this information to verify that your simulation is using the intended input and data files.

```
reading stream_settings.in data from file stream_settings.in
Searching for stream 0 overwrite files at stream0
(Stream: 0)
For Stream 0, detected the following overwrite files:(Stream: 0)
turbulence.in(Stream: 0)
inputs.in(Stream: 0)
embedded.in(Stream: 0)

(Stream: 0)
Searching for stream 1 overwrite files at stream1
(Stream: 1)
For Stream 1, detected the following overwrite files:(Stream: 1)
turbulence.in(Stream: 1)
inputs.in(Stream: 1)
embedded.in(Stream: 1)

(Stream: 1)
Found the file 'inputs.in' at '/home/pipe_with_2_streams/stream0/inputs.in'.(Stream: 0)
```

Chapter 24: File Overview

Simulation Logs Log File

```
Found the file 'inputs.in' at '/home/pipe_with_2_streams/stream1/inputs.in'.(Stream: 1)
reading data from file inputs.in
reading data from file inputs.echo(Streams: 0-1)
reading /home/pipe_with_2_streams/stream0/inputs.in data from
file /home/pipe_with_2_streams/stream0/inputs.in(Stream: 0)
reading /home/pipe_with_2_streams/stream1/inputs.in data from
file /home/pipe_with_2_streams/stream1/inputs.in(Stream: 1)
...
```

Figure 24.5: Excerpt of input and data file information in the log file.

After printing this input data, CONVERGE loops over streams and regions, partitioning the grid and performing a [load balance](#) for each region. CONVERGE prints the imbalance factor to the log file as shown in Figure 24.6. The imbalance factor after each load balance should be approximately 100.

```
stream0- Performing Load Balance for stream 0.
stream0- Imbalance Factors for Stream 0 :
stream0-           for all cells is : 191.11
stream0- Load Balance Completed for stream 0.
stream1- Performing Load Balance for stream 1.
stream1- Imbalance Factors for Stream 1 :
stream1-           for all cells is : 195.06
stream1- Load Balance Completed for stream 1.
stream0- Imbalance Factors for Stream 0 :
stream0-           for all cells is : 100.00
stream1- Imbalance Factors for Stream 1 :
stream1-           for all cells is : 100.00
```

Figure 24.6: Load balance information in the log file. This example has two streams with one region in each stream.

After the load balance operations, CONVERGE begins iterating. CONVERGE writes information about each iteration in each stream to the log file. You can control the level of detail in the log file via [inputs.in](#) > [output_control](#) > [log_level](#). The least detailed option is [log_level = 0](#). Levels 1, 2, and 3 each add successively more information.

As the simulation progresses, CONVERGE writes solution information for each cycle, labeled by the cycle number `ncyc`. CONVERGE reports solution information separately for each stream. When the solution recovers due to non-convergence, CONVERGE goes back to the previous time-step but the cycle number continues incrementing. Thus, `ncyc` may exceed the number of time-steps.

Figure 24.7 shows an example iteration for [inputs.in](#) > [output_control](#) > [log_level = 0](#). After the cycle number, CONVERGE prints timing information, which varies depending on the temporal control type. After the cycle number, CONVERGE writes `time` followed by the simulation time. Then CONVERGE prints `crank` and the current crank angle. If you entered simulation times in `seconds`, CONVERGE converts them to `crank angle degrees` using the engine geometry and speed (in `RPM`). If you entered simulation times in `crank angle degrees`, CONVERGE converts them to `seconds`. This way, the log file contains both `seconds` and `crank angle degrees`. CONVERGE then prints the current time-step size (in `seconds`). Finally, CONVERGE writes `time-step limit` followed by the active [time-step limiter](#).

Chapter 24: File Overview

Simulation Logs Log File

CONVERGE solves the conservation equations at each time-step with an [iterative algorithm](#) (e.g., [PISO](#) or [SIMPLE](#)). This iterative process accounts for the coupling between different flow variables by using the solution of one equation to update the inputs to the next. The iterative algorithm compares the solution from one iteration to the next, and determines whether to perform another iteration based on the tolerance (e.g., *PISO > tol_scale* in *solver.in*) and minimum and maximum iteration count (e.g., [solver.in](#) > *PISO > piso_itmin* and *PISO > piso_itmax*). If the solution does not converge before reaching the specified maximum iteration count, CONVERGE will continue the iterative algorithm. If the solution converges in a number of iterations between the maximum iteration count and twice that count, CONVERGE writes *dt_piso* for the time-step limiter and reduces the time-step size. If the solution does not converge within twice the specified maximum iteration count, CONVERGE recovers (re-solves the iteration) using a smaller and more stable time-step.

For streams that include the [steady-state monitor](#), after the time-step limiter, CONVERGE will provide information about the variables that you are monitoring. For each variable, the convergence status (converged or not converged) is followed by the current and target mean difference and standard deviation difference. Note that the steady-state monitor is available for both steady-state and transient streams.

After the solution information, CONVERGE prints the CFL criteria for the current time-step for each stream, which are given below.

Table 24.1: CFL numbers printed to the log file.

Log file entry	Definition
MAX CFL	Convection CFL number (sometimes simply known as CFL number) that is based on the value of inputs.in > temporal_control > max_cfl_u .
MAX VISCOSITY CFL	Diffusion CFL number that is based on the value of inputs.in > temporal_control > max_cfl_nu .
MAX CONDUCTION CFL	Conduction CFL number that is based on the value of inputs.in > temporal_control > max_cfl_nu .
MAX MASS DIFFUSION CFL	Mass diffusion CFL number based on the species diffusion coefficient.
MAX MACH CFL	Speed of sound CFL number, based on the value of inputs.in > temporal_control > max_cfl_mach .

Finally, CONVERGE estimates the percentage of the simulation that has been completed in each stream and repeats the cycle number and some time control information.

Chapter 24: File Overview

Simulation Logs Log File

```
NCYC =      6:  
    time= 1.125878906e-05, dt= 3.051757813e-06, time-step limit = dt_grow  
    dt= 3.051757813e-06, time-step limit = dt_grow  
MAX CFL= 1.5488e-03, MAX VISCOSITY CFL= 2.3433e-03, MAX CONDUCTION CFL= 2.8173e-03, MAX  
MASS DIFFUSION CFL= 3.0042e-03, MAX MACH CFL = 3.4484e+00(Stream: 0)  
MAX CFL= 1.5586e-03, MAX VISCOSITY CFL= 7.7073e-01, MAX CONDUCTION CFL= 8.5641e-01, MAX  
MASS DIFFUSION CFL= 9.8812e-01, MAX MACH CFL = 3.4484e+00(Stream: 1)  
[ 11.26%] ncyc       6; Time 1.125878906e-05 (sec); dt= 3.814697266e-06 (dt_grow)  
(Stream: 0)  
[ 30.00%] ncyc       6 (Stream: 1)
```

Figure 24.7: An example solution cycle for a two-stream simulation at [inputs.in > output_control > log_level = 0](#).

Figure 24.8 shows an example iteration for [inputs.in > output_control > log_level = 1](#). This log level adds information about the step-by-step convergence of the [iterative algorithm](#) as well as final convergence of turbulence, species, and passives.

During the case setup process, you specify a tolerance for the momentum and pressure correction equations ([solver.in > Transport > \[mom/pres\] > tol](#)). During the simulation, CONVERGE writes Ustar and pstar for each stream followed by the number of iterations necessary to solve each equation and the associated convergence error.

```
NCYC =      3:  
    time= 3.812500000e-06, dt= 1.562500000e-06, time-step limit = dt_grow  
    dt= 1.562500000e-06, time-step limit = dt_grow  
Ustar converged: iterations= 2 residual= 2.6574e-07  
Ustar converged: iterations= 4 residual= 4.1095e-06  
*****starting piso loop 1*****  
pstar converged: iterations= 2 error= 8.9070e-06(Stream: 0)  
pstar converged: iterations= 2 error= 2.6535e-05(Stream: 1)  
piso loop 1 error= 2.6535e-05  
*****starting piso loop 2*****  
density converged: iterations= 2 error= 1.3760e-05 jac= 0  
density converged: iterations= 2 error= 2.8808e-05 jac= 0  
sie converged: iterations= 2 error= 2.9409e-05 jac= 0  
  
pstar converged: iterations= 2 error= 2.7452e-06(Stream: 0)  
pstar converged: iterations= 2 error= 4.9562e-06(Stream: 1)  
piso loop 2 error= 2.9409e-05  
*****starting piso loop 3*****  
density converged: iterations= 1 error= 4.4584e-06 jac= 1  
density converged: iterations= 1 error= 7.4441e-06 jac= 1  
sie converged: iterations= 1 error= 7.5930e-06 jac= 1  
  
pstar converged: iterations= 2 error= 2.9249e-06(Stream: 0)  
pstar converged: iterations= 2 error= 6.7332e-07(Stream: 1)  
piso loop 3 error= 7.4441e-06  
turbulence converged: iterations= 6 error_tke= 9.7833e-04 omega_tke= 7.0000e-01(Stream: 0)  
turbulence converged: iterations= 2 error_eps= 1.3707e-05 error_tke= 1.1961e-04 omega_eps= 7.0000e-01 omega_tke= 7.0000e-01  
MAX CFL= 7.8686e-04, MAX VISCOSITY CFL= 1.2042e-03, MAX CONDUCTION CFL= 1.4474e-03, MAX  
MASS DIFFUSION CFL= 1.5438e-03, MAX MACH CFL = 1.7656e+00(Stream: 0)  
MAX CFL= 7.9422e-04, MAX VISCOSITY CFL= 3.9614e-01, MAX CONDUCTION CFL= 4.4018e-01, MAX  
MASS DIFFUSION CFL= 5.0788e-01, MAX MACH CFL = 1.7656e+00(Stream: 1)  
[ 3.81%] ncyc       3; Time 3.812500000e-06 (sec); dt= 1.953125000e-06 (dt_grow)  
[ 15.00%] ncyc       3 (Stream: 1)
```

Figure 24.8: An example solution cycle for a two-stream simulation at [inputs.in > output_control > log_level = 1](#).

Chapter 24: File Overview

Simulation Logs Log File

Figure 24.9 shows an example iteration for [inputs.in](#) > *output_control* > *log_level* = 2. This log level adds additional convergence information within each loop of the iterative algorithm, including additional information about solving detailed chemistry. It also prints timing information. The timing information includes the total computational time for the cycle and the time taken by other major parts of the solution process such as load balancing and solving the transport equations. The times are listed in absolute (*seconds*) and relative (percentage of the computational time for the cycle) terms. At the end of the simulation, CONVERGE writes a summary of the computational time for the entire simulation in the same format as that for the timing information after each cycle.

If CONVERGE detects local instability for a component (x, y, or z) of density, velocity, or pressure, it reduces the spatial accuracy for that component from second-order to first-order upwind. The parameters *upwind_count_density*, *upwind_count_velocity*, and *upwind_count_temperature* display the number of components for which CONVERGE reduced the spatial accuracy for the density, velocity, or temperature equations, respectively. If CONVERGE solves the majority of the components with first-order upwind, the results likely will be less accurate and it is advisable to check your simulation parameters. If [solver.in](#) > *Numerical_Schemes* > *flux_limiter* > *global* = *STEP*, CONVERGE writes the upwind counts for the density and temperature equations. If [solver.in](#) > *Numerical_Schemes* > *flux_limiter* > *mom* = *STEP*, CONVERGE writes the upwind count for the momentum equation.

```
stream0-
ncyc=      3, time=    3.812500000e-06, stream0- dt=    1.562500000e-06stream0- , time-step
limit = dt_growstream0-
stream1-
ncyc=      3, stream1- dt=    1.562500000e-06stream1- , time-step limit = dt_growstream1-
stream0-          0 upwind_count_velocity
stream0-          0 upwind_count_density
stream0- Ustar iterations= 1 residual= 8.7182e-04
stream0- Ustar iterations= 2 residual= 2.8436e-07
stream0- Ustar converged: iterations= 2 residual= 2.8436e-07

stream0-          0 upwind_count_temperature
stream1-          0 upwind_count_velocity
stream1-          0 upwind_count_density
stream1- Ustar iterations= 1 residual= 1.4103e-03
stream1- Ustar iterations= 2 residual= 1.1232e-04
stream1- Ustar iterations= 3 residual= 2.0674e-05
stream1- Ustar iterations= 4 residual= 4.1090e-06
stream1- Ustar converged: iterations= 4 residual= 4.1090e-06

stream1-          0 upwind_count_temperature
*****starting piso loop  1*****
Initial L2 norm of residual: 1.221275e+01, relative residual: 1.000000e+00
-----
Iteration No    Residual    Relative Residual
    1        8.166713e-01    6.687041e-02
    2        1.394967e-01    1.142222e-02
stream0- pstar converged: iterations= 2 error= 8.9041e-06
Initial L2 norm of residual: 2.267175e+01, relative residual: 1.000000e+00
-----
Iteration No    Residual    Relative Residual
    1        2.994544e+00    1.320826e-01
    2        2.881880e-01    1.271132e-02
stream1- pstar converged: iterations= 2 error= 2.6535e-05
piso loop  1 error= 2.6535e-05
*****starting piso loop  2*****
stream0-    density iteration= 1, residual= 1.813389e-06, jac= 0
```

Chapter 24: File Overview

Simulation Logs Log File

```
stream0-    density iteration= 2, residual= 6.801346e-10, jac= 0
stream0- density converged: iterations= 2 error= 1.3754e-05 jac= 0
stream1-    density iteration= 1, residual= 2.477730e-06, jac= 0
stream1-    density iteration= 2, residual= 7.576266e-10, jac= 0
stream1- density converged: iterations= 2 error= 2.8809e-05 jac= 0
    sie iteration= 1, residual= 2.532629e-06, jac= 0
    sie iteration= 2, residual= 2.013574e-07, jac= 0
sie      converged: iterations= 2 error= 2.9409e-05 jac= 0

Initial L2 norm of residual: 3.794588e+00, relative residual: 1.000000e+00
-----
Iteration No    Residual      Relative Residual
    1            3.037170e-01    8.003951e-02
    2            5.535579e-02    1.458809e-02
stream0- pstar  converged: iterations= 2 error= 2.8231e-06
Initial L2 norm of residual: 6.166527e+00, relative residual: 1.000000e+00
-----
Iteration No    Residual      Relative Residual
    1            6.872667e-01    1.114512e-01
    2            7.883656e-02    1.278460e-02
stream1- pstar  converged: iterations= 2 error= 4.9565e-06
piso loop  2 error= 2.9409e-05
*****starting piso loop  3*****
stream0-    density iteration= 1, residual= 5.023203e-07, jac= 1
stream0- density converged: iterations= 1 error= 4.4584e-06 jac= 1
stream1-    density iteration= 1, residual= 3.935454e-07, jac= 1
stream1- density converged: iterations= 1 error= 7.4453e-06 jac= 1
    sie iteration= 1, residual= 4.581368e-07, jac= 1
sie      converged: iterations= 1 error= 7.5942e-06 jac= 1

Initial L2 norm of residual: 2.886637e+00, relative residual: 1.000000e+00
-----
Iteration No    Residual      Relative Residual
    1            2.564022e-01    8.882385e-02
    2            3.763436e-02    1.303744e-02
stream0- pstar  converged: iterations= 2 error= 2.9239e-06
Initial L2 norm of residual: 6.550090e-01, relative residual: 1.000000e+00
-----
Iteration No    Residual      Relative Residual
    1            7.170399e-02    1.094702e-01
    2            8.090602e-03    1.235189e-02
stream1- pstar  converged: iterations= 2 error= 6.7354e-07
piso loop  3 error= 7.4453e-06
stream0- turbulence iterations= 1 error_tke= 4.6726e-01 omega_tke= 7.0000e-01
stream0- turbulence iterations= 2 error_tke= 1.2733e-01 omega_tke= 7.0000e-01
stream0- turbulence iterations= 3 error_tke= 3.7145e-02 omega_tke= 7.0000e-01
stream0- turbulence iterations= 4 error_tke= 1.1040e-02 omega_tke= 7.0000e-01
stream0- turbulence iterations= 5 error_tke= 3.2989e-03 omega_tke= 7.0000e-01
stream0- turbulence iterations= 6 error_tke= 9.8741e-04 omega_tke= 7.0000e-01
stream0- turbulence converged: iterations= 6 error_tke= 9.8741e-04 omega_tke= 7.0000e-01
stream1- turbulence iterations= 1 stream1- error_eps= 1.9954e-03 stream1- error_tke=
1.7528e-03 stream1- omega_eps= 7.0000e-01 stream1- omega_tke= 7.0000e-01 stream1-
stream1- turbulence iterations= 2 stream1- error_eps= 1.4807e-05 stream1- error_tke=
1.1961e-04 stream1- omega_eps= 7.0000e-01 stream1- omega_tke= 7.0000e-01 stream1-
stream1- turbulence converged: iterations= 2 stream1- error_eps= 1.4807e-05 stream1-
error_tke= 1.1961e-04 stream1- omega_eps= 7.0000e-01 stream1- omega_tke= 7.0000e-
01 stream1-
stream0- Stream 0: MAX CFL= 7.8790e-04, MAX VISCOSITY CFL= 1.2042e-03, MAX CONDUCTION
CFL= 1.4474e-03, MAX MASS DIFFUSION CFL= 1.5438e-03, MAX MACH CFL = 1.7656e+00
stream0- time-step limit = dt_grow
stream1- Stream 1: MAX CFL= 7.9421e-04, MAX VISCOSITY CFL= 3.9614e-01, MAX CONDUCTION
CFL= 4.4018e-01, MAX MASS DIFFUSION CFL= 5.0788e-01, MAX MACH CFL = 1.7656e+00
stream1- time-step limit = dt_grow
Total Time Step Run Time: 0.144 seconds
Load Balance          = 0.000 seconds ( 0.00%)
Move Surface and Update Grid = 0.004 seconds ( 2.78%)
    Update Auxiliary Faces = 0.000 seconds ( 0.00%)
Solving Transport      = 0.103 seconds (71.52%)
    Energy                 = 0.013 seconds ( 9.03%)
    Pressure                = 0.018 seconds (12.50%)
    Turbulence              = 0.017 seconds (11.80%)
```

Chapter 24: File Overview

Simulation Logs Log File

```
Surface triangulation = 0.000 seconds ( 0.00%)
Updating Boundary Conditions = 0.008 seconds ( 5.56%)
Writing Output = 0.000 seconds ( 0.00%)
    .out files = 0.000 seconds ( 0.00%)
Post = 0.000 seconds ( 0.00%)
Restart = 0.000 seconds ( 0.00%)
stream0- [ 1.27%] stream0- ncyc 3; Time 3.812500000e-06 (sec); dt=
1.953125000e-06 (dt_grow)stream0-
stream1- [ 10.00%] stream1- ncyc 3stream1-
```

Figure 24.9: An example solution cycle for a two-stream simulation at [inputs.in > output_control > log_level = 2](#). Note this example does not include species-by-species convergence within the PISO loop.

CONVERGE includes a [inputs.in > output_control > log_level = 3](#) option, but this is primarily intended for internal development purposes rather than general use. It is functionally equivalent to `log_level = 2`.

If the values of [inputs.in > temporal_control > max_cfl_u](#), [temporal_control > max_cfl_nu](#), and [temporal_control > max_cfl_mach](#) change during a cycle, in the log file CONVERGE records the new values for that cycle for each region. If the values do not change during a cycle, these values are not printed.

Active Variables Log

At the start of a simulation, CONVERGE writes a list of all variables with which the user can interact to an `active_variables.log` file. These variables are ones that are related to [Adaptive Mesh Refinement](#), that can be [mapped](#), that can be written to a [post*.h5 file](#), that are written to a [restart file](#), that can be used in user-defined functions (UDFs), or that are [transported](#) during the simulation.

If your output directory contains an `active_variables.log`, CONVERGE will overwrite that file when starting a new simulation.

CONVERGE will write this log file if you run [check inputs](#).

Figure 24.10 below shows an excerpt from `active_variables.log`. Each `stream` has its own section, and within each section variables are listed alphabetically. CONVERGE lists cell-centered variables and parcel variables in this file.

Chapter 24: File Overview

Simulation Logs Active Variables Log

```
Key:
  a: available for amr
  m: available for import during mapping
  p: available for inclusion in post
  r: available for import during restart
  u: available for use in UDF
  t: transported variable

---
Streams:
  Stream_00:
    Cell_Center_Variables:
      ACENTRIC_FACTOR:
        avail: [p]
      ACOUSTIC_MESH_FREQ_CUTOFF:
        avail: [p]
      ALPHA:
        avail: [p]
      ANISO_CONDUCTIVITY_FLAG:
        avail: [p]
      ANISO_CONDUCTIVITY_MULT:
        avail: [p]
      AREA:
        avail: [p]
      ASPECT_RATIO:
        avail: [p]
```

Figure 24.10: An excerpt from an *active_variables.log* file.

Process ID Log

At the beginning of the simulation, CONVERGE writes process ID information to *mpi_map.log*. This file maps each rank to its associate process ID number and physical host. This log file is primarily intended to be used in concert with [*memory_usage.out*](#) to identify how much memory is being used on each host.

The information in *mpi_map.log* is useful for performing detailed timing comparisons (an identical CONVERGE case setup may record very different wall times on different hosts). It may also help identify a misbehaving host.

Figure 24.11 below shows an example *mpi_map.log*. Each rank has its own row, with a unique combination of process ID and host name. Note that process IDs are unique on a host, but the same process ID may be in use on two different hosts.

Rank	PID	Hostname
0	26067	quad0023.convergecfd.com
1	26385	quad0041.convergecfd.com
2	27245	quad0026.convergecfd.com
3	10746	quad0023.convergecfd.com
4	26068	quad0041.convergecfd.com
5	26386	quad0026.convergecfd.com
6	27246	quad0023.convergecfd.com
7	10747	quad0041.convergecfd.com
8	26069	quad0026.convergecfd.com
9	26387	quad0023.convergecfd.com
10	27247	quad0041.convergecfd.com
11	10748	quad0026.convergecfd.com

Figure 24.11: An example *mpi_map.log* file.

Latency Log

On Linux systems, at the beginning of the simulation, CONVERGE calculates a representative network latency between rank 0 and every other rank. CONVERGE writes the minimum, maximum, and average latency to the screen output and writes each rank's latency to *mpi_latency.log*. This information is useful for diagnosing network problems.

Figure 24.11 below shows an example *mpi_latency.log*. Each rank has its own row.

```
# Computed latency from quad004.converge.global with rank 0 to
# Rank      Latency(microseconds)          Hostname
1           2.765e+01                  #quad002.converge.global
2           2.581e+01                  #quad018.converge.global
3           3.122e+01                  #quad012.converge.global
```

Figure 24.12: An example *mpi_latency.log* file.

You can perform this latency check and write *mpi_latency.log* without actually performing a simulation by passing the `-u check_system` argument at the command line. CONVERGE will not check out a license when performing this test.

24.3 Output File Overview

There are five types of output files: [echo](#) (`*.echo`), [restart](#) (`restart*.rst`), [map](#) (`map_*.h5`), [cell-averaged output](#) (`*.out`), and [cell-by-cell post](#) (`post*.h5`). CONVERGE creates output directories for these files within the Case Directory, as shown below in Figure 24.1.

For simulations not restarted from a [restart](#) file, CONVERGE writes output files to the `outputs_original` directory and its subdirectories. For a restarted simulation, CONVERGE creates a new set of output files rather than overwriting the files created in the original run. CONVERGE writes these files to the `outputs_restart<number>` directory and its subdirectories, where `<number>` is the value specified via `inputs.in > simulation_control > restart_number`.

For all simulations (both original and restarted), CONVERGE writes [echo](#) files to same directory that contains the corresponding input files (either the Case Directory or a stream-specific input directory).

Chapter 24: File Overview

Output File Overview

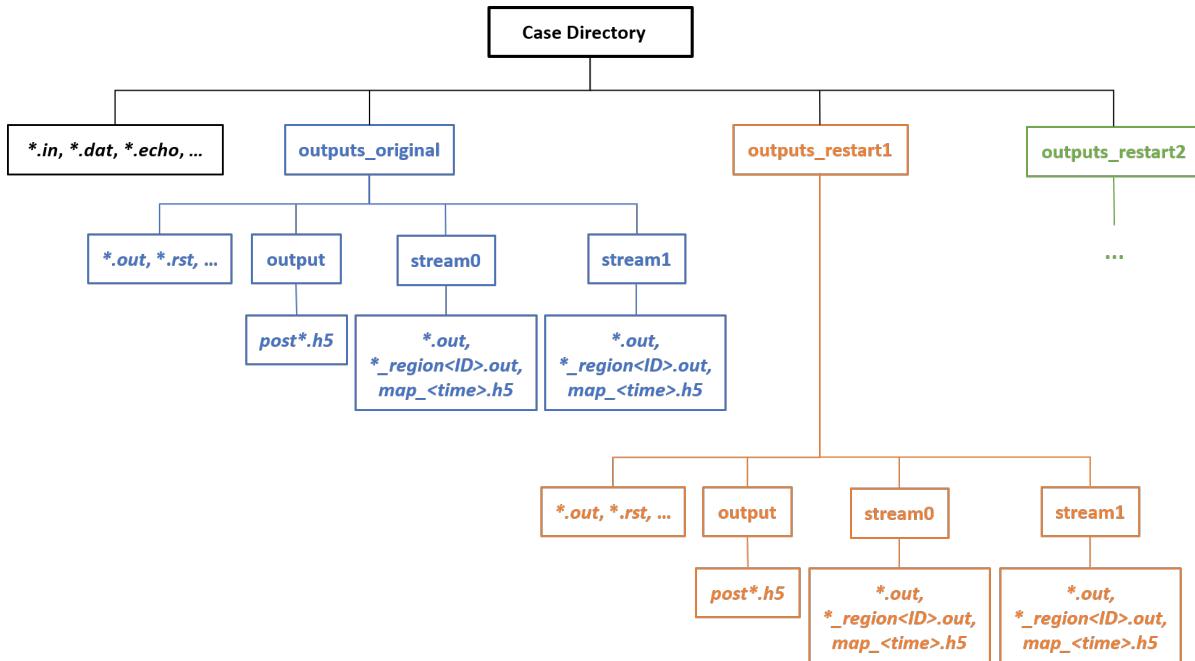


Figure 24.13: Example output directory structure for a two-stream simulation with `stream_settings.in > post_file_option = COMBINED`. Note that stream-specific input directories are not shown in this diagram.

The following table summarizes the types of files written to each location in this directory structure.

Chapter 24: File Overview

Output File Overview

Table 24.2: Contents of CONVERGE Case Directory and its output directories.

Directory	Contents
Case Directory and stream-specific input directories	Input (*.in) files, data (*.dat) files, and echo (*.echo) files. CONVERGE overwrites the echo files for each run of the simulation.
<i>outputs_original</i> or <i>outputs_restart<number></i>	Log (*.log) files, restart (restart*.rst) files, and cell-averaged output files containing data for the entire domain (e.g., <i>memory_usage.out</i>).
<i>output</i>	Cell-by-cell post (post*.h5) files for simulations with <i>stream_settings.in</i> > <i>post_file_option</i> = COMBINED or STREAM_BASED. If <i>post_file_option</i> = STREAM_BASED, there are separate <i>output</i> directories located under the <i>stream<ID></i> directories for each stream.
<i>stream<ID></i>	Stream-specific map files (<i>map_*.h5</i>), cell-averaged output files that are averaged over the stream or individual regions, and stream-specific <i>output</i> directories (for simulations with <i>stream_settings.in</i> > <i>post_file_option</i> = STREAM_BASED). If the stream contains one region, CONVERGE writes only the files that are averaged over the stream (e.g., <i>thermo.out</i>). If the stream contains multiple regions, CONVERGE writes files that are averaged over the stream (e.g., <i>thermo.out</i>) as well as files that are averaged over each region (e.g., <i>thermo_region0.out</i> , <i>thermo_region1.out</i> , etc.). CONVERGE always creates a separate <i>stream<ID></i> directory for each stream, even if there is only one stream in the simulation.
<i>output_steady</i> (not shown in diagram)	Cell-by-cell post (post*.h5) files containing data for the steady-state streams (for simulations with <i>stream_settings.in</i> > <i>post_file_option</i> = SEPARATE_STEADY_TRANSIENT).
<i>output_transient</i> (not shown in diagram)	Cell-by-cell post (post*.h5) files containing data for the transient streams (for simulations with <i>stream_settings.in</i> > <i>post_file_option</i> = SEPARATE_STEADY_TRANSIENT).

Echo Files

CONVERGE writes a *.echo file for each *.in file (e.g., the echo file for *combust.in* is *combust.echo*). Echo files give you the ability to review the input parameters and identify problems with the case setup. For example, if a *.in file contains a parameter that does not appear in the corresponding *.echo file, the parameter must have been formatted incorrectly and thus CONVERGE could not read it.

CONVERGE does not write *.echo files for *.dat files or auxiliary input files.

The *.echo files are saved to the Case Directory and (if applicable) to stream-specific input directories. Note that write permissions for the directory must be set such that these files can be created. When re-running CONVERGE from the same directory, the *.echo files will be overwritten without warning.

Restart Files

During a simulation, CONVERGE writes restart files (*.rst) and saves them to the Case Directory. These files allow you to resume a stopped simulation from a specific simulation time.

CONVERGE writes restart files at a frequency specified by [*inputs.in* > output_control > twrite_restart](#). Another parameter, [*inputs.in* > output_control > num_restart_files](#), indicates how many of these files CONVERGE will store in the *outputs_original* or *outputs_restart<number>* directory. It may be useful to set this parameter greater than one. For example, if CONVERGE writes a restart file after a simulation has started to diverge, then you can go back to an earlier restart file.

These restart files are named *restart<restart number>.rst*, and CONVERGE creates the restart files in numerical order. For example, if [*inputs.in* > output_control > num_restart_files = 3](#), the restart files found in the Case Directory might be *restart0021.rst*, *restart0022.rst*, and *restart0023.rst*. After CONVERGE has written *num_restart_files* restart files, it will remove the lowest numbered restart file when it writes another restart file. If CONVERGE writes 9,999 restart files, the next file will be *restart0001.rst*.

In addition to these restart files, CONVERGE provides [the option to save other restart files, written at specified times](#), to the *outputs_original* or *outputs_restart<number>* directory. These additional files are named according to the times at which they are written: *restart_<simulation time or CAD>.rst*. These additional restart files do not count toward the user-specified number of restart files to be saved (given by *num_restart_files*), and they will not be overwritten or deleted by CONVERGE.

Map Files

CONVERGE writes map files at the end of each simulation and, if applicable, at additional times specified in [*write_map.in*](#). CONVERGE writes *map_<time>.h5* files for all simulations, *map_parcel_<time>.h5* files for simulations that include [discrete phase modeling](#), and *map_bound<boundary ID>_<time>.out* files for simulations with [INFLOW or OUTFLOW boundaries](#).

CONVERGE writes distinct map files to each stream-specific subdirectory (*stream0*, *stream1*, etc.). Each file contains mapping data for only one stream. To initialize a new simulation from these data, you can either keep the data in separate files or merge multiple stream-specific *map_*.h5* files into a single map file. For more information about the mapping process and the considerations for multi-stream mapping, refer to the [Mapping](#) section.

Cell-Averaged Output Files

CONVERGE writes spatially averaged (*i.e.*, cell-averaged) output to fixed-width, column-formatted, ASCII `*.out` files. CONVERGE generates certain `*.out` files only if the corresponding model has been activated (*e.g.*, `liquid_parcel.out` will not be generated if the simulation does not include liquid parcel modeling).

Each line of data in the cell-averaged output files represents the values at a single cycle or time-step. Through `inputs.in` > `output_control` > `twrite_files`, you control the frequency with which data are written to the output files.

Every cell-averaged output file begins with a header composed of five rows that each begin with the character `#`. The header specifies the quantity and units for each column as well as the CONVERGE version (version number and release date). The first column in each of these output files contains time in either *seconds* (if `inputs.in` > `simulation_control` > `crank_flag = 0`) or *crank angle degrees* (if `crank_flag` is non-zero). The second column contains the number of cycles.

As shown previously in [Figure 24.12](#) and [Table 24.1](#), CONVERGE saves `*.out` files in different locations depending on whether they are averaged over the entire domain, a single stream, or a single region, and also depending on whether the simulation was started from a `restart` file. The file name format for most domain- and stream-averaged files is `<file name>.out`. For an individual region, the file name format for most files is `<file name>_region<region ID>.out`. If you rerun a simulation from the same Case Directory, CONVERGE will automatically overwrite the existing `*.out` files.

[Chapter 26 - Output Files](#) contains detailed information about each `*.out` file.

The CONVERGE Studio 3.1 Manual describes how to use the *Line Plotting* module to create two-dimensional plots of data from the `*.out` files.

Cell-By-Cell Post Output Files

CONVERGE writes individual cell quantities for all cells in the domain to separate, binary-format files at a user-specified frequency. You can use these cell-by-cell post output (`post*.h5`) files in post-processors such as Tecplot, GMC, EnSight, and Fieldview. Tecplot for CONVERGE, a version of Tecplot 360, is included with CONVERGE Studio.

For multi-stream simulations, you can combine the data for all streams into a single set of post files, write a separate set of post files for each stream, or write separate sets of post files for steady-state and transient streams (`stream_settings.in` > `post_file_option`). CONVERGE writes the post files to different directories depending on the value of `post_file_option`.

The file names for the post files are descriptive in nature. Each name contains the output number as well as the time or crank angle at which it was written. The output number appears first in the file name, so an alphabetical listing of the directory contents will be chronologically correct. The file names are formatted as follows: `post<output`

Chapter 24: File Overview

Output File Overview Cell-By-Cell Post Output Files

number>_<output time>.h5. For example, the first five post files for a simulation might be as follows:

```
post000001_-2.500000e+00.h5  
post000002_-5.783817e-01.h5  
post000003_1.424717e+00.h5  
post000004_3.432804e+00.h5  
post000005_5.444896e+00.h5
```

The first output time from a run is always the initialized field, whether it is a normal run, a restart run, or a mapping run.

Post Output Metadata

In addition to the Eulerian field variables and Lagrangian parcel data specified in *post.in*, CONVERGE writes simulation domain metadata to *post*.h5*. This includes information about boundaries, streams, cell pairing, and connectivity. This information is generally not of interest for engineering simulations, but it may be useful for identifying problems in a Case Setup or for UDF development. You can view this metadata using utilities provided by The HDF Group, such as HDFView.

Chapter



25

Input and Data Files

25 Input and Data Files

This chapter describes the CONVERGE input (**.in*) and data (**.dat*) files. Most of these files are for use in a 3D CONVERGE simulation, and the order in which these files are presented in this chapter mirrors the order of CONVERGE Studio's *Case Setup* dock. This final sections of this chapter describe input files associated with CONVERGE utilities: chemistry, heat transfer mapping, and CONGO (optimization and model interrogation).

Most CONVERGE input files are plain text-formatted in YAML, a data serialization language. The YAML format uses the text structure of the file to provide information to the input file parser without excessive use of syntax elements. YAML-formatted CONVERGE input files can include repeated sections, optional sections and parameters, one-line lists, and some degree of arbitrary ordering. Parameter names (e.g., *temporal_type*) are case-insensitive, as are most values assigned to parameters (e.g., *CYCLIC* or *cyclic* or any other capitalization is allowed).

If a parameter is set equal to a file name (e.g., *inputs.in > property_control > acentric_factor = crit_cond.dat*), then the file saved to the input directory must match that name exactly (e.g., continuing the *crit_cond.dat* example, the file in the input directory must be named *crit_cond.dat*, not *Crit_Cond.dat* or *CRIT_COND.DAT*). In previous versions of CONVERGE, file names sometimes had to be enclosed within quotation marks. In CONVERGE 3.1, you do not need to enclose a file name within quotation marks as long as the file name does not contain any spaces or other special characters.

Sometimes you need to specify a list, e.g., in [*skip_species.in*](#) you must specify a list of species to keep. YAML allows this information to be in one of two formats. The figure below shows the list in flow format, enclosed by square brackets. CONVERGE Studio prints flow-formatted lists in line with the parameter name.

```
species_to_keep: [CH4, OH, CO, CH2O]
```

Figure 25.1: A species list in flow format.

Figure 25.2 shows the same list in block format. In block format, the level of indentation indicates the block structure, and the hyphen indicates a list of one or more elements. Blocks may enclose sub-blocks (e.g., a species block inside of an AMR block). Note that the number of spaces defining a block is arbitrary, but it must be identical within each block. If any tabs are present in an input file, CONVERGE will convert each tab to three spaces at runtime.

```
species_to_keep:
  - CH4
  - OH
  - CO
  - CH2O
```

Figure 25.2: A species list in block format.

Although you can edit input and data files directly, we strongly recommend that you set up your case in CONVERGE Studio and then export the input and data files to the Case Directory. Some parameters are optional, but some are required, and using CONVERGE Studio helps to ensure that all files contain the required elements and are formatted correctly. In addition, CONVERGE Studio can provide recommended values for many different types of cases.

In this chapter, parameters that are optional in all simulations are marked with an asterisk (*) in the table that describes that parameter. Note that additional parameters may be optional depending on your particular case settings.

CONVERGE allows blank rows in all files and ignores all rows starting with #. CONVERGE Studio may look for some information in rows beginning with # when importing files.

Recommended values vary for different types of cases. To see recommended values for a particular type of case, load the relevant example case in CONVERGE Studio.

25.1 Spatial and Temporal Profiles

CONVERGE allows many parameters to be a constant value or to vary according to a spatial and/or temporal profile. A profile is a file that specifies the values of a parameter at specific locations and/or times. The use of profiles is especially common for parameters in [*boundary.in*](#).

To use a profile, you must define it in a file (e.g., *velocity.in*), include the file in your case setup, and specify the file name for the appropriate parameter (e.g., [*boundary.in*](#) > *boundary_conditions* > *boundary* > *velocity* > *value = velocity.in*). You do not need to put the file name in quotation marks unless it contains a space or other special character.

There is a standard format for each type of profile, as shown in the examples in Figures 25.3-25.7. We recommend setting up these files in CONVERGE Studio to ensure proper formatting.

Spatial Profile Format

Figure 25.3 shows an example of a spatial profile. In this example, the spatially varying parameter is temperature. You can use the parameters in rows 2-7 to scale, translate, or rotate the coordinates listed in the profile.

Row 9 must contain a time value followed by *second* (if [*inputs.in*](#) > *simulation_control* > *crank_flag* = 0) or *crank* (if *crank_flag* is non-zero). You can specify any time value. Because only one time is listed, there is no temporal variation. CONVERGE will use the the specified spatial profile at every time-step.

The profile contains a finite number of entries for discrete locations. At locations not listed in the profile, CONVERGE uses the value of the parameter from the nearest entry.

Chapter 25: Input and Data Files

Spatial and Temporal Profiles Spatial Profile Format

```
SPATIAL
1.0 scale_xyz
0.0 trans_x
0.0 trans_y
0.0 trans_z
x rot_axis
0.0 rot_angle

0.0 crank
x      y      z      temperature
-0.0196 0.0356 -0.0907 475.0
-0.022 0.0369 -0.0907 467.0
-0.0223 0.0347 -0.0907 479.0
-0.0161 0.0338 -0.0907 491.0
-0.0153 0.0381 -0.0907 479.0
.
.
```

Figure 25.3: Excerpt of a spatial temperature profile.

Table 25.1: Parameters in rows 2-7 of a spatial profile.

Parameter	Description
<i>scale_xyz</i>	Scaling factor. CONVERGE multiplies the x, y, and z coordinates by this value.
<i>trans_x</i>	Distance by which each point is translated in the x direction (<i>m</i>).
<i>trans_y</i>	Distance by which each point is translated in the y direction (<i>m</i>).
<i>trans_z</i>	Distance by which each point is translated in the z direction (<i>m</i>).
<i>rot_axis</i>	Axis of rotation (x, y, or z).
<i>rot_angle</i>	Angle by which each point is rotated about the axis (<i>degrees</i>).

Temporal Profile Format

A temporal profile contains a finite number of entries at discrete times, with the latter given in *seconds* (if *inputs.in* > *simulation_control* > *crank_flag* = 0) or in *crank angle degrees* (if *crank_flag* is non-zero). To set the value of the parameter at simulation times between these entries, CONVERGE interpolates between the values from the two nearest entries (for parameters in *boundary.in*) or uses the value from the nearest earlier entry (for parameters in files other than *boundary.in*).

A temporal profile can have sequential or cyclic timing. Figure 25.4 shows an example with sequential timing. In this example, the temporally varying parameter is pressure and the times are specified in *seconds*. For simulation times earlier than the first entry, CONVERGE uses the value from the first entry. For times later than the last entry, CONVERGE uses the value from the last entry.

For *steady-state* streams, the header of the time column must be *second*, as shown in Figure 25.4. However, the actual unit of time is the pseudo time-step used by the steady-state solver. The appropriate pseudo time-scale for the temporal variation will be case-dependent. For temporally varying temperature or pressure boundary conditions, note that CONVERGE will not perform automatic grid scaling until after the boundary condition has ceased to vary. Manual grid scaling or AMR, however, will still be applied.

Chapter 25: Input and Data Files

Spatial and Temporal Profiles Temporal Profile Format

```
TEMPORAL
SEQUENTIAL
second    pressure
0.0      60000.0
0.0005   60000.0
0.001    60000.0
0.0015   70000.0
.
.
```

Figure 25.4: Excerpt of a temporal pressure profile with sequential timing.

Figure 25.5 shows an example of a temporal profile with cyclic timing. In this example, the temporally varying parameter is [inputs.in](#) > *output_control* > *twrite_post* and the times are specified in *crank angle degrees*. The period of the cycle is 720 *crank angle degrees*. For simulation times earlier than the first entry or later than the last entry, CONVERGE repeats the cycle according to the specified period.

```
TEMPORAL
CYCLIC  720
crank   twrite_post
-360.0  10.0
-20.0   2.0
+150.0  10.0
+360.0  10.0
```

Figure 25.5: A temporal *twrite_post* profile with cyclic timing.

Spatial and Temporal Profile Format

For parameters that vary in both space and time, the format of the profile is similar to that of a [spatial profile](#), but with additional entries for additional times. The temporal variation can be sequential or cyclic.

At simulation times between entries, CONVERGE uses the data from the nearest earlier entry. At locations not listed in the profile, CONVERGE uses the data provided for the nearest point in space.

Figure 25.6 shows an example of a spatial and temporal profile with sequential timing. The parameters in rows 2-7 are the same as those for a [spatial profile](#), described above in Table 25.1. For simulation times earlier than the first entry, CONVERGE uses the data from the first entry. For times later than the last entry, CONVERGE uses the data from the last entry.

Chapter 25: Input and Data Files

Spatial and Temporal Profiles Spatial and Temporal Profile Format

```
SPATIAL
1.0      scale_xyz
-0.2001  trans_x
0.1001  trans_y
0.0      trans_z
z        rot_axis
0.0      rot_angle

0.0 crank
x      y      z      pressure
0.0    0.0    0.0    1.01324e5
10.0   10.0   10.0   1.00000e5

0.2 crank
x      y      z      pressure
0.0    0.0    0.0    1.11524e5
10.0   10.0   10.0   1.01100e5
.
.
```

Figure 25.6: Excerpt of a spatial and temporal pressure profile with sequential timing.

Figure 25.7 shows an example of a spatial and temporal profile with cyclic timing. The period of the cycle is 2.0 *seconds*. The parameters in rows 2-7 are the same as those for a [spatial profile](#), described above in Table 25.1. For simulation times earlier than the first entry or later than the last entry, CONVERGE repeats the cycle according to the specified period.

```
SPATIAL_CYCLIC 2.0
1.0      scale_xyz
0       trans_x
0.0     trans_y
0.0     trans_z
z        rot_axis
0.0      rot_angle

0.0 second
x      y      z      temperature
0.0    0.0    0.0    400
10.0   10.0   10.0   500

1.0 second
x      y      z      temperature
0.0    0.0    0.0    500
10.0   10.0   10.0   600

2.0 second
x      y      z      temperature
0.0    0.0    0.0    400
10.0   10.0   10.0   500
```

Figure 25.7: A spatial and temporal temperature profile with cyclic timing.

25.2 Applications

This section describes the applications-related input files. Refer to the Internal Combustion Engine example cases in CONVERGE Studio for more information about simulating internal combustion engines.

Engine Parameters: engine.in

For an engine simulation, set [inputs.in](#) > *simulation_control* > *crank_flag* = 1 and define engine-related parameters in the *engine.in* file.

Chapter 25: Input and Data Files

Applications Engine Parameters: `engine.in`

```
version: 3.1
---

bore:          0.086
stroke:        0.09
connecting_rod: 0.18
crank_offset:   0.0
rpm:           3000.0
swirl:          0.0
swirl_profile: 3.11
zhead:          0.0
piston_id:      1
liner_id:       2
head_id:        3
crevice_flag:    0
```

Figure 25.8: An example `engine.in` file.

Table 25.2: Format of `engine.in`.

Parameter	Description	Typical value
<code>bore</code>	Engine cylinder bore (<i>m</i>).	
<code>stroke</code>	Engine cylinder stroke (<i>m</i>). Twice the crank radius.	
<code>connecting_rod</code>	Engine connecting rod length (<i>m</i>).	
<code>crank_offset</code>	Engine crank offset (<i>m</i>).	
<code>rpm</code>	Engine speed (<i>rev/min</i>). Specify a file name for a case with variable RPM .	
<code>swirl</code>	Initial swirl ratio.	0.0 (when the gas exchange is simulated)
<code>swirl_profile</code>	Swirl profile parameter. Minimum value is 0.0 (for wheel flow). Maximum value is 3.83 (for zero velocity at the wall).	3.11
<code>zhead</code>	Cylinder head position (<i>z</i> coordinate, in <i>m</i>). Used for velocity initialization.	0.0
<code>piston_id</code>	Boundary ID of piston (from boundary.in).	
<code>liner_id</code>	Boundary ID of cylinder liner (from boundary.in).	
<code>head_id</code>	Boundary ID of head (from boundary.in).	
<code>crevice_flag</code>	0 = No crevice model, 1 = Crevice model (requires crevice.in).	

Crevice Model: `crevice.in`

A crevice model can simulate the flow between the piston, piston rings, and cylinder. This model is an alternative to resolving the crevice regions in the CFD grid directly, which can result in smaller cells and, therefore, smaller time-steps. To use the crevice model, the cylinder axis must be aligned with the *z* axis. To activate the crevice model, set [engine.in](#) > `crevice_flag` = 1 and include the `crevice.in` file in your case setup.

The crevice model in CONVERGE is based on the work of [Namazian and Heywood \(1982\)](#). This model, which is illustrated below in Figure 25.9, includes gas flow passages, N rings ($N \geq 2$), and $2N + 1$ crevice regions. Region 1 is at the cylinder pressure, while region $2N + 1$, which is just above the oil ring, is at the crankcase pressure. The crevice model will calculate the pressures in each region and the blowby rate, the mass flow rate from regions $2n - 1$ and $2n$ to region $2n + 1$ for the n -th ring.

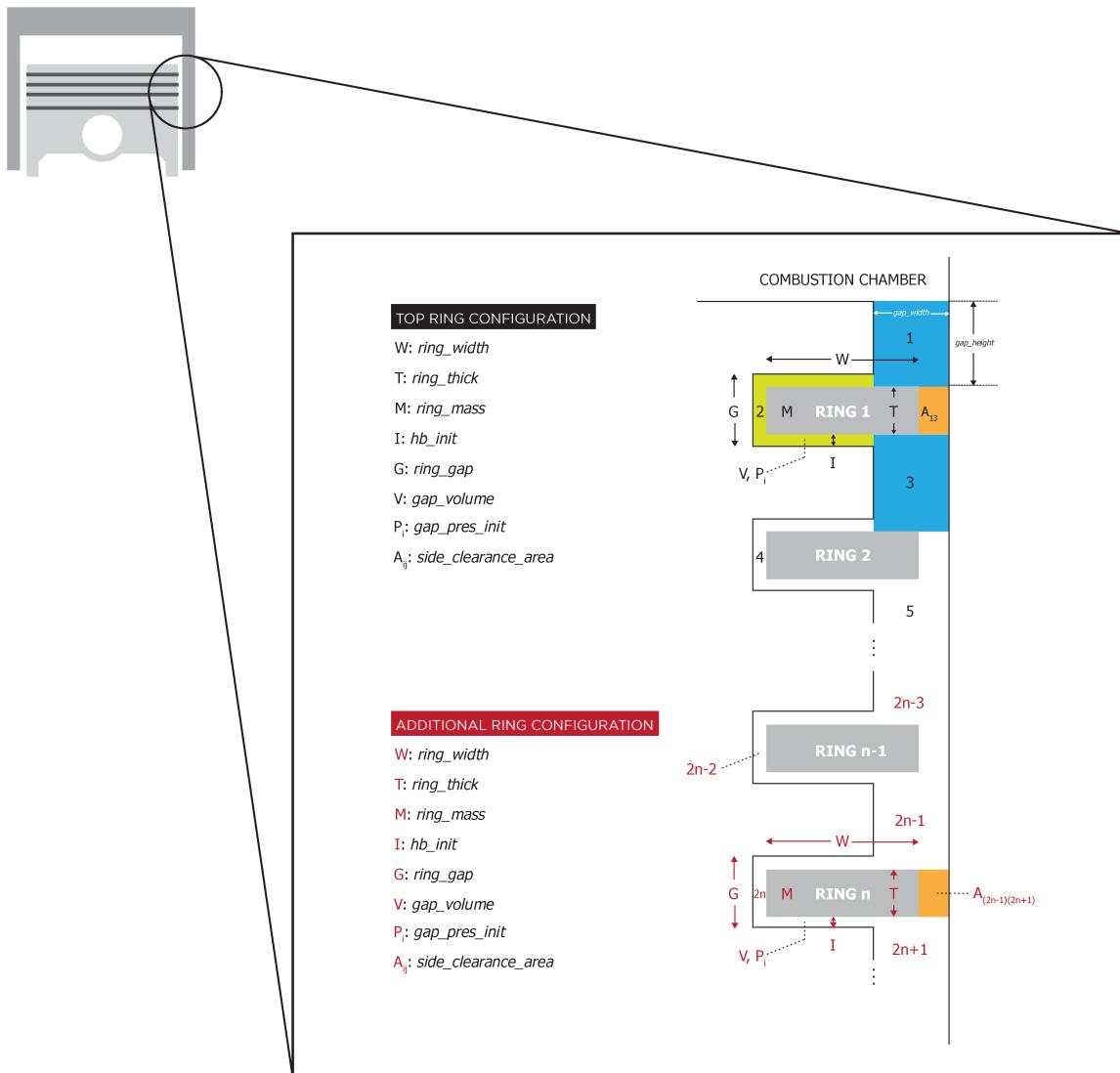


Figure 25.9: Illustration of gas flow passages, rings and regions for the crevice model.

According to Namazian and Heywood, for a two ring crevice model, the continuity equations for regions 2, 3, and 4 can be written as

$$\frac{m_{o(2)}}{P_{o(2)}} \frac{dP_{(2)}}{dt} = \dot{m}_{(1)(2)} - \dot{m}_{(2)(3)}, \quad (25.1)$$

$$\frac{m_{o(3)}}{P_{o(3)}} \frac{dP_{(3)}}{dt} = \dot{m}_{(1)(3)} + \dot{m}_{(2)(3)} - \dot{m}_{(3)(4)} - \dot{m}_{(3)(5)}, \quad (25.2)$$

and

$$\frac{m_{o(4)}}{P_{o(4)}} \frac{dP_{(4)}}{dt} = \dot{m}_{(3)(4)} - \dot{m}_{(4)(5)}, \quad (25.3)$$

where $P_{(i)}$ is the pressure in region i , $P_{o(i)}$ is the initial static pressure in region i , and $m_{o(i)}$ is the initial mass in region i . For each additional ring n , CONVERGE solves two more equations:

$$\frac{m_{o(2n-1)}}{P_{o(2n-1)}} \frac{dP_{(2n-1)}}{dt} = \dot{m}_{(2n-3)(2n-1)} + \dot{m}_{(2n-2)(2n-1)} - \dot{m}_{(2n-1)(2n)} - \dot{m}_{(2n-1)(2n+1)}, \quad (25.4)$$

and

$$\frac{m_{o(2n)}}{P_{o(2n)}} \frac{dP_{(2n)}}{dt} = \dot{m}_{(2n-1)(2n)} - \dot{m}_{(2n)(2n+1)}. \quad (25.5)$$

Finally, $\dot{m}_{(i)(j)}$ is the mass flow rate between regions i and j through the ring gap or ring side clearances. Namazian and Heywood show that the flow rate through the ring side clearances (e.g., $\dot{m}_{(1)(2)}$ and $\dot{m}_{(2)(3)}$) can be written as

$$\dot{m}_{(i)(j)} = \frac{Ah^2}{24W_r} \frac{1}{\mu RT} (P_i^2 - P_j^2), \quad (25.6)$$

where W_r is the channel length (which is the width of the ring of interest), h is the channel width (which is either the height below or above the ring, depending on the mass flow rate of interest), A is the area normal to the flow (approximated as $2\pi(bore/2)h$), T is the crevice region temperature, R is the ideal gas constant, μ is the viscosity evaluated at the crevice

Chapter 25: Input and Data Files

Applications Crevice Model: crevice.in

region temperature, and P_i and P_j are the upstream and downstream pressures, respectively. Equation 25.6 above is used to approximate the mass flow rates in Equations 25.1 to 25.5.

As shown by Namazian and Heywood, the mass flow rates through the ring gaps (e.g., $\dot{m}_{(1)(3)}$ and $\dot{m}_{(3)(5)}$) are calculated by the following orifice flow equation:

$$\dot{m}_{(i)(j)} = C_d A_{(i)(j)} \rho c \eta, \quad (25.7)$$

where C_d is the discharge coefficient, $A_{(i)(j)}$ is the ring end-gap area between regions i and j , ρ is the gas density, c is the speed of sound, and η is the compressibility factor given by

$$\eta = \frac{2}{\gamma - 1} \sqrt{\left(\frac{P_j}{P_i}\right)^{2/\gamma} - \left(\frac{P_j}{P_i}\right)^{(\gamma+1)/\gamma}} \quad \left(\frac{P_j}{P_i} > 0.52\right) \quad (25.8)$$

or

$$\eta = \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma+1}{2(\gamma-1)}} \quad \left(\frac{P_j}{P_i} \leq 0.52\right), \quad (25.9)$$

where P_i is the upstream pressure, P_j is the downstream pressure, $P_j < P_i$, and γ is assumed to be 1.4. Equations 25.3 through 25.5 are solved using a fourth-order Runge-Kutta method. Once values for P_2 , P_3 , and P_4 are obtained, $\dot{m}_{(1)(2)}$ and $\dot{m}_{(1)(3)}$ are re-calculated from Equations 25.6 and 25.7, respectively. CONVERGE uses these mass flow rates to determine the amount of mass entering and leaving the cylinder.

You can set `crevice.in > ring_motion_flag = 1` to include the motion of the rings in the crevice model calculation. If `ring_motion_flag = 0`, the initial ring positions are used throughout the simulation.

When `ring_motion_flag = 1`, CONVERGE must determine the forces on the rings. We use an equation of motion for a ring similar to that of [Namazian and Heywood \(1982\)](#), given by

$$M_r \frac{d^2 h}{dt^2} = F_p + F_f + F_i, \quad (25.10)$$

where h is the top ring side-clearance of ring n , F_p is pressure force, F_f is friction force, and F_i is inertial force. We do not include the resistance of the squeezed oil.

Chapter 25: Input and Data Files

Applications Crevice Model: crevice.in

Because the total force on the ring is constant for a given time-step, we can integrate Equation 25.10 from the beginning to the end of the time-step to obtain

$$\begin{aligned}\frac{dh}{dt} &= h' + \left(\frac{F_p + F_f + F_i}{M_r} \right) \Delta t \\ h &= \left(\frac{F_p + F_f + F_i}{M_r} \right) \frac{(\Delta t)^2}{2} + h' \Delta t + h_0,\end{aligned}\tag{25.11}$$

where h_0 and h' are the values of h and dh/dt , respectively, at the beginning of the time-step.

CONVERGE uses the analytical solution in Equation 25.11 to update the position of the ring at each time-step. The pressure force is given by

$$F_p = A_n \frac{P_{2n-1} - P_{2n+1}}{2}\tag{25.12}$$

for ring n , where A_n is the surface area of the top (or bottom) of ring n . The friction force is given by

$$F_f = P_n \pi d_n T_n f\tag{25.13}$$

for ring n , where d_n is the diameter of ring n and T_n is the thickness of ring n . The friction coefficient f is given by

$$f = 4.8 \sqrt{\mu_{oil} U_p / (P_n T_r)}\tag{25.14}$$

for ring n ([Reitz and Kuo, 1989](#)). In Equation 25.14, U_p is the instantaneous piston velocity and μ_{oil} is the viscosity of the oil, which is given by [Namazian and Heywood \(1982\)](#) as

$$\mu_{oil} = \exp \left[\left(\frac{1036}{T - 178} \right) - 9.84 \right],\tag{25.15}$$

where T is the crevice region temperature. The units of μ_{oil} are $N\text{-s}/m^2$.

Finally, the inertia force is given by

$$F_i = -M_{r,n} a_p,\tag{25.16}$$

Chapter 25: Input and Data Files

Applications Crevice Model: crevice.in

where $M_{r,n}$ is the mass of the ring n and a_p is the piston acceleration ([Reitz and Kuo, 1989](#)).

After calculating the ring motion and crevice region pressures, CONVERGE can link the solution to the cylinder by writing an expression in the form of Equations 25.1 through 25.3 for first zone of the crevice:

$$\frac{m_{o(1)}}{P_{o(1)}} \frac{dP_{(1)}}{dt} = \dot{m}_{(0)(1)} - \dot{m}_{(1)(2)} - \dot{m}_{(1)(3)}, \quad (25.17)$$

where $\dot{m}_{(0)(1)}$ is the mass flow rate from the cylinder region (region 0) to first zone of the crevice (zone 1 in Figure 25.9). Equation 25.17 can be rewritten as

$$\dot{m}_{(0)(1)} = \frac{m_{o(1)}}{P_{o(1)}} \frac{P_{cyl}^{n+1} - P_{cyl}^n}{dt} + \dot{m}_{(1)(2)} + \dot{m}_{(1)(3)}, \quad (25.18)$$

where P_{cyl}^{n+1} is the cylinder pressure at the new time-step and P_{cyl}^n is the cylinder pressure at the previous time-step. CONVERGE uses Equation 25.18 to calculate the relevant source terms for mass, momentum, and energy in the crevice cells.

Table 25.3 below matches the crevice model parameters in the equations above with their corresponding parameters in *crevice.in*. This file is described in full below.

Table 25.3: Comparison of crevice model and *crevice.in* parameters.

Crevice model parameter	<i>crevice.in</i> parameter
C_d	<i>cdis_crev</i>
W_{r1}	<i>top_ring > ring_width</i>
T_{r1}	<i>top_ring > ring_thick</i>
M_{r1}	<i>top_ring > ring_mass</i>
A_{13}	<i>top_ring > side_clearance_area</i>
$P_{o(1)}$	<i>top_ring > gap_pres_init</i>
$W_{r,n}$	<i>additional_rings > ring > ring_width</i>
$T_{r,n}$	<i>additional_rings > ring > ring_thick</i>
$M_{r,n}$	<i>additional_rings > ring > ring_mass</i>
$A_{(2n-1)(2n+1)}$	<i>additional_rings > ring > side_clearance_area</i>

Chapter 25: Input and Data Files

Applications Crevice Model: crevice.in

Crevice model parameter	<i>crevice.in</i> parameter
$P_{o(i)}$	<i>additional_rings > ring > gap_pres_init</i>
$P_{(2N+1)}$	<i>pcrank</i>
T	<i>temp_crev</i>

If *engine.in* > *crevice_flag* = 1, you must include the *crevice.in* file in your case setup. Figure 25.11 below shows an example file. Tables 25.4 - 25.13 below describe the format of *crevice.in*.

CONVERGE uses the piston and liner boundary IDs in the *cylinder_info* settings block to identify the crevice cells (*i.e.*, the cells to which the crevice source and sink terms are applied), as shown below in Figure 25.10. First, CONVERGE identifies all cells that share faces with the piston and the liner. Then, CONVERGE identifies cells for which the distance from the piston and the distance from the liner (as measured from the cell center) are both less than twice the crevice width.

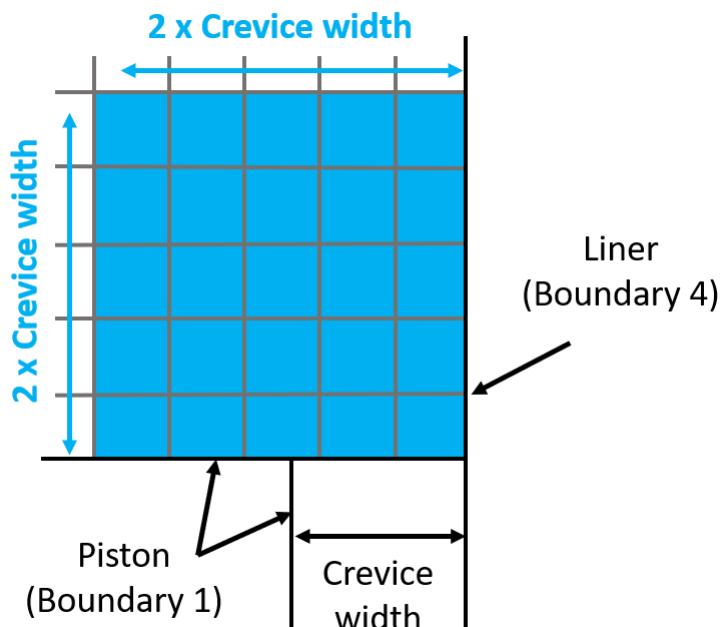


Figure 25.10: The definition of crevice cells. In this example, the piston is Boundary 1 and the liner is Boundary 4. Crevice cells are colored in blue.

```
version: 3.1
---
cylinder_info:
  - cylinder:
    piston_boundary_id: 1
    liner_boundary_id: 4
sector_angle: 360
ring_motion_flag: 0
```

Chapter 25: Input and Data Files

Applications Crevice Model: crevice.in

```
cdis_crev:          0.86
temp_crev:          433
crevice_init_species:
    N2:             0.6787
    CO2:            0.17768
    H2O:            0.0828
    O2:             0.06082
crank_case_species:
    N2:             0.6787
    CO2:            0.17768
    H2O:            0.0828
    O2:             0.06082
crevice_dummy_frac: 1
pcrank:            101325
top_land:
    gap_height:     0.01196
    gap_width:      0.000425
top_ring:
    ring_mass:      0.051
    ring_width:     0.0041
    ring_thick:     0.0031
    ring_gap:       0.0021
    side_clearance_area: 9.01e-6
    gap_volume:     1.2e-6
    gap_pres_init:  100002
    hb_init:        1.1e-20
additional_rings:
- ring:
    land:
        volume:        1.3e-6
        pres_init:     100003
        ring_mass:     0.052
        ring_width:    0.0042
        ring_thick:    0.0032
        ring_gap:      0.0022
        side_clearance_area: 9.02e-6
        gap_volume:    1.4e-6
        gap_pres_init: 100004
        hb_init:       1.2e-20
- ring:
    land:
        volume:        1.5e-6
        pres_init:     100005
        ring_mass:     0.053
        ring_width:    0.0043
        ring_thick:    0.0033
        ring_gap:      0.0023
        side_clearance_area: 9.03e-6
        gap_volume:    1.6e-6
        gap_pres_init: 100006
        hb_init:       1.3e-20
- ring:
    land:
        volume:        1.7e-6
        pres_init:     100007
        ring_mass:     0.054
        ring_width:    0.0044
        ring_thick:    0.0034
        ring_gap:      0.0024
        side_clearance_area: 9.04e-6
        gap_volume:    1.8e-6
        gap_pres_init: 100008
        hb_init:       1.4e-20
```

Figure 25.11: An example *crevice.in* file.

Chapter 25: Input and Data Files

Applications Crevice Model: crevice.in

Table 25.4: Parameter definitions in *crevice.in*.

Parameter	Description
<i>cylinder_info</i>	This settings block specifies the cylinder(s) that use the crevice model.
- <i>cylinder</i>	Repeat this sub-block for each cylinder.
<i>piston_boundary_id</i>	Piston boundary ID for the cylinder.
<i>liner_boundary_id</i>	Liner boundary ID for the cylinder.
<i>sector_angle</i>	Sector angle (<i>degrees</i>) for an engine sector simulation. Set to 360 for a non-sector simulation.
<i>ring_motion_flag</i>	0 = Do not include ring motion, 1 = Include ring motion.
<i>cdis_crev</i>	Orifice flow discharge coefficient.
<i>temp_crev</i>	Temperature (K) of the crevice region. Typically set to the cylinder liner boundary temperature.
<i>crevice_init_species</i> *	This settings block specifies the initial species in the crevice.
< <i>species name</i> >	Mass fraction of species. Repeat this line for each species. Each species must be on a separate line.
<i>pcrank</i>	Crankcase pressure (Pa).
<i>top_land</i>	
<i>gap_height</i>	Distance (m) between the top of the piston and the top of the top ring (ring 1) gap. For this parameter and subsequent parameters, see Figure 25.10 above.
<i>gap_width</i>	Distance (m) between the top of the piston and the liner/cylinder wall.
<i>top_ring</i>	
<i>ring_mass</i>	Mass (kg) of the top ring (ring 1).
<i>ring_width</i>	Width (m) of the top ring (ring 1).
<i>ring_thick</i>	Thickness (m) of the top ring (ring 1).
<i>ring_gap</i>	Distance (m) between the top and bottom of the top ring gap.
<i>side_clearance_area</i>	Flow area (m ²) between regions 1 and 3. You can specify a constant value or the name of a file (e.g., <i>area13.in</i>) with a temporally varying profile.
<i>gap_volume</i>	Volume (m ³) of region 2.
<i>gap_pres_init</i>	Initial pressure (Pa) in region 2. Typically set to the intake valve closing (IVC) pressure.
<i>hb_init</i>	Initial position (m) of the top ring (ring 1) with respect to the bottom of the top ring gap. Typical value: 0.0.
<i>additional_rings</i>	

Chapter 25: Input and Data Files

Applications Crevice Model: crevice.in

Parameter	Description
- <i>ring</i>	This settings sub-block specifies the conditions for additional rings in the crevice model. You must include at least one ring in the <i>additional_rings</i> settings block. Repeat this sub-block for each additional ring.
<i>land</i>	This settings sub-block specifies the initial volume and pressure of the land (<i>i.e.</i> , the common area between two rings).
<i>volume</i>	Volume (m^3) of region $2n-1$.
<i>pres_init</i>	Initial pressure (Pa) in region $2n-1$. Typically set to the intake valve closing (IVC) pressure.
<i>ring_mass</i>	Mass (kg) of ring n .
<i>ring_width</i>	Width (m) of ring n .
<i>ring_thick</i>	Thickness (m) of ring n .
<i>ring_gap</i>	Distance (m) between the top and bottom of the ring n gap.
<i>side_clearance_area</i>	Flow area (m^2) between regions $2n - 1$ and $2n + 1$. You can specify a constant value or the name of a file (<i>e.g.</i> , <i>area35.in</i>) with a temporally varying profile.
<i>gap_volume</i>	Volume of region $2n$.
<i>gap_pres_init</i>	Initial pressure (Pa) in region $2n$. Typically set to IVC.
<i>hb_init</i>	Initial position (m) of the ring n with respect to the bottom of the ring n gap. Typical value: 0.0.

* Note: Blocks or parameters indicated with an asterisk are optional.

Non-Engine RPM: rpm.in

For a *crank angle-based* simulation that is not an engine simulation, set [*inputs.in*](#) > *simulation_control* > *crank_flag* = 2 and include an *rpm.in* file.

```
version: 3.1
---
rpm:    2000
```

Figure 25.12: An example *rpm.in* file.

Table 25.5: Format of *rpm.in*.

Parameter	Description
<i>rpm</i>	Piston speed (<i>rev/min</i>). Specify a file name for a case with variable RPM .

Variable RPM: var_rpm.in

For an engine case with a variable RPM, specify a file name for *rpm* in [*engine.in*](#), as shown below in Figure 25.13.

Chapter 25: Input and Data Files

Applications Variable RPM: var_rpm.in

```
.  
.crank_offset: 0.0  
rpm: var_rpm.in  
swirl: 0.0  
..
```

Figure 25.13: An excerpt of *engine.in* that includes activation of the variable RPM option.

For a non-engine case with a variable RPM, specify a file name for *rpm* in [*rpm.in*](#), as shown below in Figure 25.14.

```
version: 3.1  
---  
rpm: var_rpm.in
```

Figure 25.14: An example *rpm.in* file that includes activation of the variable RPM option.

For both engine and non-engine cases, if you specify a file name for *rpm*, you must include a file with that name in your case setup. Figure 25.15 shows an example of a file you can use to specify different *rpm* values for different simulation times. The first row contains the keyword *TEMPORAL*, and the second row contains the keyword *SEQUENTIAL* or *CYCLIC*. If *CYCLIC*, the period must follow, also on the second row. The third row contains two column headings: *crank* and *rpm*. Subsequent rows should contain RPM values at various times. You can specify RPM values for multiple cycles if desired.

```
TEMPORAL  
SEQUENTIAL  
crank rpm  
-720.0 800.0  
-360.0 1000.0  
-180.0 1100.0  
0.0 1200.0  
90.0 1300.0  
180.0 1400.0  
270.0 1500.0  
. .
```

Figure 25.15: An example of a variable RPM file (e.g., *var_rpm.in*).

For variable RPM cases with spray injection, if the rate-shape is constant ([*parcels.in* > *injectors* > *injector* > *injector_control* > *rate_shape* = *CONSTANT*](#)), then CONVERGE automatically generates a [*rateshape_inj<num>.in*](#) file for each injector. For variable RPM cases, you can include [tabular profiles for injection start time, injection duration, and total mass injected](#).

Generic Motion: *motion_sets.in*

Use the *motion_sets.in* file to specify various types of complex boundary motion for a WALL boundary. This file allows you to easily configure and reuse boundary motion in complex cases. To use motion sets, set [*inputs.in* > *feature_control* > *motion_sets_flag* = 1](#) and include a *motion_sets.in* file in your case setup. You must also specify the appropriate option(s) in [*boundary.in*](#).

Chapter 25: Input and Data Files

Applications Generic Motion: motion_sets.in

First, specify the the motion configuration in *motion_sets.in*. After you specify the motion parameters, you must direct CONVERGE to apply this motion to the appropriate moving boundary or boundaries. For the relevant boundary, set the velocity boundary condition as [boundary.in](#) > [boundary_conditions](#) > [boundary](#) > [velocity](#) > [value = MOTION_CONFIG](#). Supply the motion object name (*motion_sets.in* > *motion_object* > *motion_name*) for [boundary.in](#) > [boundary](#) > [velocity](#) > [motion_name](#) to use the motion object for the boundary. If [inputs.in](#) > [simulation_control](#) > [crank_flag = 2](#) and you choose *MOTION_CONFIG* and supply a *CRANK_SLIDER* object in *motion_sets.in*, CONVERGE generates the piston position table for a non-engine application.

```
version: 3.1
---

- motion_object:
    motion_name:          object1
    mechanism:            CRANK_SLIDER
    slider_axis:          [0.0, 0.0, 1.0]
    crank_radius:         0.08255
    connecting_rod:       0.263
    crank_offset:         0.0
    speed_ratio:          1.0

- motion_object:
    motion_name:          object2
    mechanism:            ROTATION
    rotation_origin:      [0.0, 0.0, 0.0]
    rotation_axis:         [0.0, 0.0, -1.0]
    rotation_speed:        27000.0

- motion_object:
    motion_name:          object3
    mechanism:            PROGRESSIVE_CAVITY
    number_of_lobes:       2
    rotor_axis:           [0.0, 0.0, 1.0]
    rotor_radius:          0.004039
    rotor_center:          [0.0, 0.004039, 0.0]
    housing_center:        [0.0, 0.0, 0.0]

- motion_object:
    motion_name:          object4
    mechanism:            DRILL_BIT
    rpm_shaft:             180
    rpm_drill_bit:         300
    shaft_axis:            [0.0, 1.0, 0.0]
    shaft_center:          [0.0, 0.0, 0.0]
    drill_bit_axis:        [3.535529E-1, -7.071038E-1, -6.123761E-1]
    drill_bit_center:      [4.674246E-2, 1.629737E-1, -8.096035E-2]
```

Figure 25.16: An example *motion_sets.in* file.

Table 25.6: Format of the *motion_object* settings block.

Parameter	Description
- <i>motion_object</i>	This settings block specifies the mechanism parameters. Repeat this block for each motion object.
<i>motion_name</i>	Name of the motion object.
<i>mechanism</i>	Motion mechanism:

Chapter 25: Input and Data Files

Applications Generic Motion: motion_sets.in

Parameter	Description
	CRANK_SLIDER = Generate a piston motion table (requires boundary.in > boundary_conditions > boundary > motion = TRANSLATING and boundary_conditions > boundary > geometry_motion = MOVING), DRILL_BIT = Generate a motion profile for drill bit motion (requires inputs.in > simulation_control > crank_flag to be non-zero and boundary.in > boundary_conditions > boundary > motion = ARBITRARY), PROGRESSIVE_CAVITY = Generate a motion profile for progressive cavity motion (requires inputs.in > simulation_control > crank_flag to be non-zero and boundary.in > boundary_conditions > boundary > motion = ARBITRARY), ROTATION = Supply a custom rotation (requires boundary.in > boundary_conditions > boundary > motion = ROTATING and boundary_conditions > boundary > geometry_motion = MOVING).
#for CRANK_SLIDER mechanism	
slider_axis	Vector along which the piston slides.
crank_radius	Half the length of the cylinder stroke (m).
connecting_rod	Connecting rod length (m).
crank_offset	Piston wrist pin offset (m).
speed_ratio	RPM multiplier.
#for DRILL_BIT mechanism	
rpm_shaft	Shaft RPM (RPM).
rpm_drill_bit	Drill bit RPM (RPM).
shaft_axis	Direction of shaft axis. Must be non-zero.
shaft_center	Any point on the main shaft axis (m).
drill_bit_axis	Direction of drill bit axis. Must be non-zero.
drill_bit_center	Any point on the drill bit axis (m).
#for PROGRESSIVE_CAVITY mechanism	
number_of_lobes	Specify the number of lobes for hypocycloidal motion in the progressive cavity.
rotor_axis	Rotor axis vector. Must be non-zero and orthogonal to the vector between the <i>rotor_center</i> and <i>housing_center</i> .
rotor_radius	Rotor radius at one end of the progressive cavity (m). If <i>number_of_lobes</i> = 2, <i>rotor_radius</i> must be equal to the distance between <i>rotor_center</i> and <i>housing_center</i> .

Chapter 25: Input and Data Files

Applications Generic Motion: motion_sets.in

Parameter	Description
<i>rotor_center</i>	Rotor center at one end of the progressive cavity (m).
<i>housing_center</i>	Center of the housing measured at one end of the progressive cavity (m).
#for ROTATION mechanism	
<i>rotation_origin</i>	Boundary rotation origin.
<i>rotation_axis</i>	Axis about which the boundary rotates. Must be non-zero.
<i>rotation_speed</i>	Rotation speed (RPM).

Region-Specific Setup Considerations for Engine Modeling

For a multiple-cylinder simulation, assign each cylinder to a unique [region](#). It is useful to have each cylinder be its own region because you can activate many models or features on a region-by-region basis. Region-dependent features include [Adaptive Mesh Refinement](#) (AMR), the [SAGE detailed chemical kinetics solver](#), temporally varying [CFL numbers](#), [source terms](#), [initialization](#), and [output customization](#). Table 25.6 contains notes to keep in mind while setting up a multi-cylinder simulation.

Table 25.7: : Setup considerations for a multiple-cylinder simulation.

Input file	Notes
events.in	Specify an event between each pair of adjacent regions. For adjacent regions in which no valves are present, you need specify only an OPEN event, which will allow flow at all times between these regions. Remember to add the phase lag while specifying OPEN and CLOSE events for adjacent regions containing valves.
source.in	Specify a source to the transport equations of energy, momentum, turbulence, species, and passives. <ul style="list-style-type: none">• Use an energy source to model spark events by defining the source as a geometric space (e.g., BOX, SPHERE, CYLINDER, or REGION).• If you do not directly model combustion in a cylinder, the heat release in that cylinder can be modeled using either cylinder pressure or heat release rate. A REGION source works well.• You can distribute sources within a region.• The source start and end times and the coordinates will differ for each cylinder so as to correspond with the phase angle and rotation angle for each cylinder.
parcel_introduction.in	Specify injection times (<i>injectors > injector > injector_control > start_time</i> and <i>injectors > injector > injector_control > duration</i> in parcel_introduction.in) and injector locations (<i>injectors > injector > injector_control > position</i> in parcel_introduction.in) that are consistent with the phase angle and location of each cylinder.
combust.in	Simulate combustion in one or more cylinders and use energy sources for the rest. <ul style="list-style-type: none">• Activate adaptive zoning to reduce the computational time.• Region-dependent combustion is an option for several combustion models. (For example, if you are using the SAGE detailed chemistry solver, you can turn off

Chapter 25: Input and Data Files

Applications Region-Specific Setup Considerations for Engine Modeling

Input file	Notes
	SAGE in the intake and exhaust ports to save computational time.) Control this option via <i>general_control > region_flag</i> in <i>combust.in</i> . Note that the start and end times for the combustion model for each region must correspond to the phase lag angle for each cylinder.
	<ul style="list-style-type: none">• If you are using the CTC model, you can invoke region-specific reset times and periods. Specify a file name (e.g., ctc_init_time.in) for <i>ctc_model > init_time</i> in <i>combust.in</i> and include the <i>ctc_init_time.in</i> file in the Case Directory.• If you are using the G-Equation model, you can invoke region-specific reinitialization times. Specify a file name (e.g., g_eqn_init_time.in) for <i>g_eqn_model > init_value</i> in <i>combust.in</i> and include the <i>g_eqn_init_time.in</i> file in the Case Directory.
initialize.in	If you do not specify an initial value for a passive or a mass fraction of a species in initialize.in , CONVERGE will assign it a value of 0.
embedded.in	Specify the start and end times (<i>embedding > start_time</i> and <i>embedding > end_time</i> in embedded.in) to be consistent with the corresponding cylinder phase lag angles for injector and spark plug embedding. Since each cylinder is in a unique location, you must ensure each embedding location (<i>embedding > x_center</i> in embedded.in) corresponds to the location of the associated cylinder.
amr.in	Activate AMR on a region-by-region basis. <ul style="list-style-type: none">• Specify the regions and/or boundaries on which CONVERGE will invoke AMR.• Ensure that <i>amr_groups > amr_group > amr_* > start_time</i> and <i>amr_groups > amr_group > amr_* > end_time</i> in amr.in are consistent with the phase lag angle for each cylinder.
max_cfl_u.in	Activate region-specific maximum convective CFL numbers. When the exhaust valve opens, the gas velocity in the exhaust port is very high. The time-step calculated based on the maximum CFL number may be very small, resulting in significantly longer runtimes. If the accuracy of the results in the exhaust port is less important, increase the maximum CFL criteria for the exhaust port region to reduce the runtime. Specify a file name (e.g., max_cfl_u.in) for <i>temporal_control > max_cfl_u</i> in inputs.in and include that file in the Case Directory.

25.3 Materials

This section describes the input and data files that contain properties for the various species in your CONVERGE simulation.

CONVERGE assumes that ideal gas mixture assumptions are valid when determining the properties of gas mixtures, and thus CONVERGE evaluates gas mixture properties as

$$\begin{aligned} e &= \sum_{m=1}^M e_m Y_m, & h &= \sum_{m=1}^M h_m Y_m, & s &= \sum_{m=1}^M s_m Y_m, & c_p &= \sum_{m=1}^M c_{p,m} Y_m \\ c_v &= \sum_{m=1}^M c_{v,m} Y_m, & MW &= \sum_{m=1}^M MW_m X_m, & \text{and } \rho &= \sum_{m=1}^M \rho_m, \end{aligned} \tag{25.19}$$

Chapter 25: Input and Data Files

Materials
.....

where the summation is over all M species, h is the enthalpy, s is the entropy, X_m is the species mole fraction, Y_m is the species mass fraction, c_p is the specific heat at constant pressure, c_v is the specific heat at constant volume, MW is the molecular weight, and ρ_m is the species density.

Gas Properties: `gas.dat`

CONVERGE generates gas properties from species-specific polynomial coefficients. These values, which must be present for each gas-phase species in your simulation, are stored in a [thermodynamic data file](#) (e.g., `therm.dat`) or a [tabular thermodynamic data file](#) named `tabular_therm.dat`.

You must specify additional gas properties in either a `gas.dat` file (if `inputs.in > property_control > species_diffusion_model = 0` or `2`) or a [transport.dat](#) file (if `species_diffusion_model = 1`).

The `gas.dat` file lists gas viscosity and conductivity data for a single gas species. This file contains gas viscosity and conductivity data in 10 K-increments starting from 0 K. The `gas.dat` data must go at least to the `inputs.in > property_control > max_temp` or to 5000 K, whichever is lower. If the `max_temp` exceeds 5000 K and if the `gas.dat` file contains data only to 5000 K, CONVERGE will perform a zero-order extrapolation to obtain gas property data up to the maximum temperature.

Figure 25.17 below shows a sample `gas.dat` file, which must contain three columns: temperature in K, viscosity ($N\cdot s/m^2$), and conductivity ($W/m^2\cdot K$).

```
# temperature    viscosity    conductivity
0.0000E+00      7.06E-06     9.2000E-3
1.0000E+01      7.06E-06     9.2000E-3
.
.
5.0000E+03      1.52E-04     7.1800E-01
```

Figure 25.17: An example `gas.dat` file.

For simulations with multiple gas species, you can set `inputs.in > property_control > individual_properties_flag = 1` if you want to use different gas properties for certain species. Create separate files for those species in the same format as `gas.dat` and list those files and species in [`species_transport.in`](#). For any species not listed in [`species_transport.in`](#), CONVERGE uses gas properties from the main `gas.dat` file. CONVERGE calculates mass-based averages of the properties of individual species to determine the properties of the gas mixture.

Gas Properties: `transport.dat`

CONVERGE generates gas properties from species-specific polynomial coefficients. These values, which must be present for each gas-phase species in your simulation, are stored in a [thermodynamic data file](#) (e.g., `therm.dat`) or a [tabular thermodynamic data file](#) named `tabular_therm.dat`.

Chapter 25: Input and Data Files

Materials Gas Properties: *transport.dat*

You must specify additional gas properties in either a *gas.dat* file (if *inputs.in* > *property_control > species_diffusion_model = 0 or 2*) or a *transport.dat* file (if *species_diffusion_model = 1*).

The *transport.dat* file is CHEMKIN-formatted.

CO	1	98.100	3.650	0.000	1.950	1.800
H2O	2	572.400	2.605	1.844	0.000	4.000
H2	1	38.000	2.920	0.000	0.790	280.000
CO2	1	244.000	3.763	0.000	2.650	2.100
O2	1	107.400	3.458	0.000	1.600	3.800
H2O2	2	107.400	3.458	0.000	0.000	3.800
OH	1	80.000	2.750	0.000	0.000	0.000
HO2	2	107.400	3.458	0.000	0.000	1.000
H	0	145.000	2.050	0.000	0.000	0.000
O	0	80.000	2.750	0.000	0.000	0.000
AR	0	136.500	3.330	0.000	0.000	0.000
N2	1	97.530	3.621	0.000	1.760	4.000
HE	0	10.200	2.576	0.000	0.000	0.000

Figure 25.18: Sample *transport.dat* file.

Table 25.8: Format of *transport.dat*.

Column	Quantity
1	Species name.
2	0 = Molecular geometry is monatomic, 1 = Molecular geometry is linear, 2 = Molecular geometry is nonlinear.
3	Lennard-Jones potential.
4	Lennard-Jones collision diameter.
5	Dipole moment.
6	Polarizability.
7	Rotational relaxation collision number.

Gas Properties: *schmidt_species.dat*

For a simulation with gases, CONVERGE requires a *schmidt_species.dat* file if *inputs.in* > *property_control > species_diffusion_model = 2*. This file must contain two columns: species name and molecular Schmidt number for that species. Each gas species in *gas.dat* must have a corresponding entry in *schmidt_species.dat*.

Chapter 25: Input and Data Files

Materials Gas Properties: schmidt_species.dat

C3H7	1.2653320783
HNCO	0.933961938407
CH2 (S)	0.661398279649
N2	0.714380158937
HNO	0.747652972768
CH3OH	0.88357023439
HOCH	0.933961938407
HCNO	0.933961938407
HCNN	0.597785737274
CH2O	0.862908457977
CH2OH	0.883070064629
O2	0.746891806368

Figure 25.19: An example *schmidt_species.dat* file.

Species-Dependent Gas Properties: *species_transport.in*

If *inputs.in* > *property_control* > *individual_properties_flag* = 1, CONVERGE uses gas properties from the files listed in *species_transport.in* for the corresponding species listed in *species_transport.in*. CONVERGE uses gas properties from the main *gas.dat* file for any gas species not listed in *species_transport.in*. To determine the properties of the gas mixture, CONVERGE calculates mass-based averages of the properties of individual species.

```
version: 3.1
---
species_with_transport_prop:
  O2: O2_gas.dat
  H2: H2_gas.dat
```

Figure 25.20: An example *species_transport.in* file.

Table 25.9: Format of *species_transport.in*.

Parameter	Description
<i>species_with_transport_prop</i>	
< <i>species name</i> >	Species name and name of the file that contains gas properties for this species. This file must be in the same format as <i>gas.dat</i> .
	Repeat for each gas species with gas properties different from those in <i>gas.dat</i> .

Species-Dependent Critical Properties of Gases: *crit_cond.dat*

For a simulation with multiple gas species, you can include species-dependent critical temperatures, critical pressures, and acentric factors. These data are used only when the equation of state is not the ideal gas law (the equation of state is set via *inputs.in* > *property_control* > *eos_flag*). To include species-dependent data, specify a file name (e.g., *crit_cond.dat*) instead of a value for *inputs.in* > *property_control* > *crit_temp*, *property_control* > *crit_pres*, and/or *property_control* > *acentric_factor*. For a simulation that includes composite species, CONVERGE will calculate the critical properties of the composite from the species-specific critical properties in *crit_cond.dat*.

Chapter 25: Input and Data Files

Materials Species-Dependent Critical Properties of Gases: crit_cond.dat

The following figure shows an excerpt of an [inputs.in](#) file in which *crit_temp*, *crit_pres*, and *acentric_factor* each specify a file name instead of a value. In this example the *crit_cond.dat* file will contain the species-dependent gas property data.

```
.  
.  
property_control:  
    gas_compressible_flag:      1  
    liquid_compressible_flag:   0  
    lhv_flag:                  0  
    tabular_fluid_prop_flag:   0  
    eos_flag:                  REDLICH_KWONG  
    real_gas_prop_flag:        0  
    max_reduced_pres:          6.0  
    crit_temp:                 crit_cond.dat  
    crit_pres:                 crit_cond.dat  
    acentric_factor:           crit_cond.dat  
    species_diffusion_model:  0  
    prandtl_turb:              0.9  
    schmidt_turb:              0.78  
    min_temp:                  10.0  
    max_temp:                  60000.0  
    max_visc:                  10.0  
    gravity:                   [gx.dat, gy.dat, gz.dat]  
.  
.
```

Figure 25.21: An excerpt of an *inputs.in* file in which a file name (*crit_cond.dat*) instead of a value is listed for the critical temperature, critical pressure, and acentric factor.

The first (uncommented) row of *crit_cond.dat* contains the keyword *DEFAULT* followed by a critical temperature, critical pressure, and acentric factor. Any gas not listed by name in this file will be assigned these default properties.

#	species	crit_temp	crit_pres	acentric_factor
DEFAULT		133.0	3.77e+6	0.035
AR		150.8	4.87e+6	0.001
HE		5.19	2.27e+5	-0.365
H2		33.18	1.3e+6	-0.216
N2		126.19	3.3978e+6	0.039
O2		154.58	5.043e+6	0.025
CH4		190.6	4.61e+6	0.011
CH2O		408	6.59e+6	0.253
C2H2		308.3	6.138e+6	0.19
C2H3		308.3	6.138e+6	0.19
C2H4		282.5	5.06e+6	0.089
C3H4		402.4	5.63e+6	0.215
C6H12		504.03	3.14e+6	0.28
C6H6		562.1	4.89e+6	0.212
C7H16		540.2	2.74e+6	0.349
C10H22		617.8	2.11e+6	0.489
NC10H22		617.8	2.11e+6	0.489
NC12H26		658.2	1.80e+6	0.562
CO		134.45	3.49875e+6	0.066
CO2		304.18	7.38e+6	0.2373
HO2		728.0	22.0e+6	0.0
H2O		647.0	22.064e+6	0.344
H2O2		728.0	22.0e+6	0.0
NO		180.0	6.48e+6	0.588

Figure 25.22: An example species-dependent gas property data file (e.g., *crit_cond.dat*).

Chapter 25: Input and Data Files

Materials Species-Dependent Critical Properties of Gases: crit_cond.dat

Table 25.10: Format of the species-dependent gas property data file (e.g., crit_cond.dat).

Column	Parameter	Units
1	species name	N/A
2	<i>crit_temp</i>	K
3	<i>crit_pres</i>	N/m ²
4	<i>acentric_factor</i>	N/A

Liquid Properties: liquid.dat

When simulating one or more liquids in CONVERGE, you must specify liquid properties in a data file called *liquid.dat*. You can specify multiple liquids in the same *liquid.dat* file to generate a database of liquids. Specify all data for one liquid before specifying data for the next liquid.

CSI Gasoline Surrogate

Convergent Science has created a gasoline surrogate known as CSI_Gasoline_v1. The composition of this surrogate is 50% isoctane, 35% decane, 10% pentane, and 5% dodecane. You can load the *liquid.dat* data for this surrogate in CONVERGE Studio (go to *Case Setup > Materials > Liquid simulation > Predefined liquids*). We do not recommend specifying these four components separately as the spray for an [injector](#).

```
version: 3.1
---

!REFERENCE YAWS
H2O molecular_weight 18.0153
! PowerIdx ConsistentIdx YieldStress(N/m2) SolidViscosity(m2/s)
NON_NEUTONIAN 1 1 1 1
! RefPressure[Pa] RefDensity[kg/m3] BulkModulus[Pa]
Compressible 101325 1025.76 1.9e+09
!
Critical_temp 647.13
!Temperature Viscosity Surf tension Heat of vapo Vapor pres
![K] [N*s/m^2] [N/m] [J/kg] [Pa]
0.000000e+00 1.774387e-03 7.858852e-02 2.423018e+06 6.106160e+02
1.000000e+01 1.774387e-03 7.858852e-02 2.423018e+06 6.106160e+02
2.000000e+01 1.774387e-03 7.858852e-02 2.423018e+06 6.106160e+02
3.000000e+01 1.774387e-03 7.858852e-02 2.423018e+06 6.106160e+02

Conductivity Density Specific heat
[W/ (m*K)] [kg/m^3] [J/(kg*K)]
5.704524e-01 1.049947e+03 4.235123e+03
5.704524e-01 1.049947e+03 4.235123e+03
5.704524e-01 1.049947e+03 4.235123e+03
5.704524e-01 1.049947e+03 4.235123e+03
```

Figure 25.23: An excerpt of a *liquid.dat* file. In this example, the formatting is slightly altered for readability. In reality, temperature, viscosity, surface tension, heat of vaporization, vapor pressure, conductivity, density, and specific heat data are all on the same line.

Chapter 25: Input and Data Files

Materials Liquid Properties: liquid.dat

Table 25.11: Format of *liquid.dat*.

Column	Quantity	Units
1	Temperature	K
2	Viscosity	N·s/m ²
3	Surface tension	N/m
4	Latent heat of vaporization	J/kg
5	Vapor Pressure	Pa
6	Conductivity	W/m·K
7	Density	kg/m ³
8	Specific heat capacity	J/kg·K

The first row contains the name of the liquid species (*e.g.*, H₂O_L) and the species molecular weight. Note that in CONVERGE 2.4 and earlier versions, *liquid.dat* may not include the molecular weight. If you import an old *liquid.dat* file into CONVERGE Studio 3.1, CONVERGE Studio will set the missing molecular weights to 0.0 and issue a warning message.

After the liquid name, there are three sections that contain the liquid data. The [non-Newtonian section](#) and the [compressible section](#) are required for non-Newtonian and compressible liquids, respectively, while the [critical temperature section](#) is required for all liquids.

Non-Newtonian Section

The non-Newtonian section of *liquid.dat* consists of one line and is required only for non-Newtonian liquids (*i.e.*, only if the liquid species is listed under LIQUID_NON_NEWTIONIAN in *species.in*). You can use the Herschel-Bulkley model or the Carreau model to simulate non-Newtonian liquids.

Herschel-Bulkley Model

The Herschel-Bulkley model assumes that viscosity varies as a function of shear rate and that the liquid will not flow until it experiences a certain yield stress. (For Newtonian liquids, CONVERGE assumes that viscosity is a function of temperature.) Equation 25.20 below gives the expression for shear stress used in this model:

$$\tau = \tau_0 + K\gamma^n, \quad (25.20)$$

where τ_0 is the yield stress, K is the consistency index, γ is the shear rate, and n is the power index. If the shear stress applied to the fluid is less than the yield stress, CONVERGE assigns the fluid a high viscosity equivalent to the solid viscosity.

Chapter 25: Input and Data Files

Materials Liquid Properties: liquid.dat

To specify a non-Newtonian liquid governed by the Herschel-Bulkley model, include a single line in *liquid.dat* starting with the keyword *HERSCHEL_BULKLEY_MODEL* or the keyword *NON_NEWTONIAN*, followed by the power index, the consistency index, the yield stress (N/m^2), and the solid viscosity ($Pa\cdot s$). Additionally, specify the species name below *LIQUID_NON_NEWTONIAN* in *species.in*.

Carreau Model

The Carreau model is an empirical relation and assumes that viscosity (μ) is a function of shear rate (γ). At low shear rates, the fluid behaves with an effectively constant viscosity, μ_0 . At intermediate shear rates, the fluid viscosity exhibits power law behavior, controlled by the power law index n and the time constant λ . At high shear rates, the fluid behaves again with an effectively constant viscosity, μ_∞ . Viscosity is expressed as

$$\mu = \mu_\infty + (\mu_0 - \mu_\infty)[1 + (\lambda\gamma)^2]^{(n-1)/2}. \quad (25.21)$$

To specify a non-Newtonian liquid governed by the Carreau model, include a single line in *liquid.dat* starting with the keyword *CARREAU_MODEL*, followed by the low shear rate viscosity ($Pa\cdot s$), the high shear rate viscosity ($Pa\cdot s$), the power index, and the time constant (s). Additionally, specify the species name below *LIQUID_NON_NEWTONIAN* in *species.in*.

Compressible Section

The compressible section of *liquid.dat* consists of one line and is required only when your simulation includes a compressible liquid (*i.e.*, if *inputs.in* > *property_control* > *liquid_compressible_flag* = 1). CONVERGE calculates the density of the compressible liquid according to

$$\rho = \rho_{ref} e^{\left(\frac{P - P_{ref}}{B}\right)}, \quad (25.22)$$

where ρ_{ref} is the reference density, P_{ref} is the reference pressure, and B is the bulk modulus.

To specify these parameters for a compressible liquid, include a single line in *liquid.dat* with the keyword *compressible*, followed by the reference pressure (Pa), the reference density (kg/m^3), and the bulk modulus (Pa).

Critical Temperature Section

The critical temperature section of *liquid.dat* is required for all liquids. This section contains liquid properties in 10-Kelvin intervals from 0 K to the critical temperature. If liquid properties are independent of temperature, simply specify the same properties for each temperature in the list.

The first line of this section consists of the keyword *critical_temp* and the critical temperature (T_{crit}) of the liquid (in K). Each row thereafter consists of the temperature and the

Chapter 25: Input and Data Files

Materials Liquid Properties: liquid.dat

corresponding viscosity, surface tension, latent heat of vaporization, vapor pressure, conductivity, density, and specific heat. The number of entries in this section must equal $\text{int}(T_{\text{crit}}/10) + 2$. For example, if $T_{\text{crit}} = 540.30 \text{ K}$, CONVERGE requires 56 entries, from 0 to 550 K, in 10 K intervals.

In a multi-component liquid parcel, mass-averaged viscosity, mass-averaged surface tension, mass-averaged specific heat of vaporization, mass-averaged thermal conductivity, and volume-averaged density are calculated for each parcel. CONVERGE solves the temperature for the parcel according to the governing equations. CONVERGE calculates the vapor pressure based on the mole fraction of the components. If you activate the liquid parcel boiling model ([parcels.in > parcel_list > parcel > evap_control > drop > boiling = ON](#)), latent heat of vaporization is mass-averaged over the parcel. Otherwise, latent heat of vaporization is calculated for each species.

Species Data and Reaction Mechanism: mech.dat

The reaction mechanism file lists the gas-phase elements and species used in the simulation. If your simulation invokes the [SAGE detailed chemical kinetics solver](#), then this file must also include reaction data. Note that the species names and information defined in the reaction mechanism file must be consistent with the species names in [initialize.in](#).

The name of the reaction mechanism file (e.g., *mech.dat*) in your case setup must match [inputs.in > mechanism_filename](#).

```
ELEMENTS
  h  c  o  n
END

SPECIES
  c7h16    o2    n2    co2    h2o
  co      h2    ch4   c2h2   c2h4
  .
  .
END

REACTIONS
  c7h16 + h = c7h15-1 + h2    5.600e+07    2.0    7667.0
  c7h16 + h = c7h15-2 + h2    4.380e+07    2.0    4750.0
  .
END
```

Figure 25.24: An excerpt of a reaction mechanism file. Note that the species names must NOT be preceded by #.

If the reactions section is not required (e.g., if your simulation does not invoke the SAGE detailed chemistry solver), you can insert an END statement immediately after the REACTIONS statement. CONVERGE will ignore all text after the END statement, and thus you do not need to delete the reaction data from the file.

The three numbers that follow a reaction define, respectively, the pre-exponential factor A_i (in centimeters, grams, or seconds, depending on the reaction), the temperature exponent β_i (dimensionless), and the activation energy E_i (cal/mol) in the Arrhenius equation

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

$$k_{i,f} = A_i T^{\beta_i} \exp\left(\frac{-E_i}{RT}\right) \quad (25.23)$$

for the forward rate constant of the i -th reaction, $k_{i,f}$

To change the activation energy units to *kJ/mole*, include the statement `KJOULE/MOLE` in the same line as the `REACTIONS` statement.

Although the `=` symbol in the reactions in Figure 25.23 indicates that the reactions are reversible, typically only the forward rate constant is defined in the reaction mechanism file. CONVERGE will calculate the reverse rate constant $k_{i,r}$ as

$$k_{i,r} = \frac{k_{i,f}}{K_{i,c}}, \quad (25.24)$$

where $K_{i,c}$ is the equilibrium constant determined from thermodynamic properties. Alternatively, you can specify different options (reversible, pressure-dependent, etc.) for reactions. These options are described in the sections below.

Reverse Reaction Option

Use the reverse reaction option to specify the reverse rate constant for a reaction in a reaction mechanism file. After the reaction row, add a row beginning with the keyword `REV` followed by values for A_i , β_i , and E_i , as shown in the following figure.

```
REACTIONS
.
.
c7h16 + h = c7h15-1 + h2      5.600e+07   2.0      7667.0
REV / 4.80e+12    0   1.143e+04 /
.
.
END
```

Figure 25.25: An example of the `REV` option in a reaction mechanism file.

Third-Body Reaction Option

Some reactions require a third body for the reaction to proceed. A third body is often needed in dissociation or recombination reactions such as



In such a reaction involving K chemical species, the rate of progress for the i -th reaction is given by

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

$$q_i = \left(\sum_{k=1}^K (\alpha_{ki}) [X_k] \right) \left(k_{i,f} \prod_{k=1}^K [X_k]^{\nu'_{ki}} - k_{i,r} \prod_{k=1}^K [X_k]^{\nu''_{ki}} \right), \quad (25.26)$$

where ν_{ki} are integers representing stoichiometric coefficients, the superscript ' indicates *forward* stoichiometric coefficients, the superscript " indicates *reverse* stoichiometric coefficients, $[X_k]$ is the molar concentration of the k -th species, and $k_{i,f}$ and $k_{i,r}$ are the forward and reverse rate constants, respectively, for the i -th reaction. If all of the species in the mixture contribute equally as third bodies, then $\alpha_{ki} = 1$ for all k_i and the first factor in the previous equation is the total concentration of the mixture

$$[M] = \sum_{k=1}^K [X_k] \quad (25.27)$$

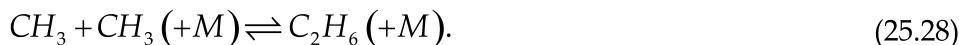
In reality, some species act more efficiently as third bodies than others, and thus you need to specify the non-unity values of the enhancement factor α_{ki} in the reaction mechanism file. After the reaction row, add a row specifying the species that enhances the efficiency of the reaction, followed by the enhancement factor, as shown in the following figure. You can enter up to 30 enhanced third bodies.

```
REACTIONS
.
.
h + oh + M = h2o + M          0.160e+23    -2.0      0.0
h2o/5/
.
.
END
```

Figure 25.26: An example of the third-body reaction option in a reaction mechanism file.

Pressure-Dependent Reaction Options

Under certain conditions, some reaction rate expressions depend on both pressure and temperature. In low-pressure and high-pressure ranges, different forms of these reaction rate expressions may apply. During pressure and temperature conditions between the two limits (low and high pressure), the rate expressions become complicated. To denote a reaction that is in this range, write the reaction with (+M), as shown,



In reaction mechanism files, CONVERGE allows the [Lindemann](#), [Troe](#), [SRI](#), [P-Log](#), and [Chebyshev](#) methods to represent rate expressions in this range.

Lindemann Format

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

For pressure-dependent reactions, Arrhenius rate parameters are required for both the high- and low-pressure-limiting cases. The Lindemann form ([Lindemann et al., 1922](#)) of the rate coefficient blends these two sets of parameters to produce a single pressure-dependent rate expression. In Arrhenius form, the low-pressure limit (k_0) and high-pressure limit (k_∞) are as follows:

$$k_0 = A_0 T^{\beta_0} \exp\left(-\frac{E_0}{RT}\right) \quad (25.29)$$

and

$$k_\infty = A_\infty T^{\beta_\infty} \exp\left(-\frac{E_\infty}{RT}\right). \quad (25.30)$$

The rate constant at any pressure is then taken to be

$$k = k_\infty \left(\frac{P_r}{1 + P_r} \right) F, \quad (25.31)$$

where the reduced pressure P_r is given by

$$P_r = \frac{k_0 [M]}{k_\infty} \quad (25.32)$$

and $[M]$ is the concentration of the mixture, which may include enhanced third-body efficiencies. In the Lindemann form, F is unity in Equation 25.31.

To use the Lindemann form for a reaction, add a row after the reaction row with either the keyword *LOW*, followed by the values of A_0 , β_0 , and E_0 , or the keyword *HIGH*, followed by the values of A_∞ , β_∞ , and E_∞ . CONVERGE will then use the values in the reaction row for the opposite limit. In the example below, the reaction row contains A_∞ , β_∞ , and E_∞ , while the following row contains A_0 , β_0 , and E_0 .

```
REACTIONS
.
.
o + co (+M) = co2 (+ M)           1.800e+10   0.0      2385.00
LOW / 6.020E+14  0.000  3000.00
.
.
END
```

Figure 25.27: An example of the Lindemann option in a reaction mechanism file.

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

Troe Format

In the Troe form ([Gilbert et al., 1983](#)), F in Equation 25.31 is given by

$$\log F = \left[1 + \left[\frac{\log P_r + c}{n - d(\log P_r + c)} \right]^2 \right]^{-1} \log F_{cent}. \quad (25.33)$$

The constants in Equation 25.33 are

$$c = -0.4 - 0.67 \log F_{cent}, \quad (25.34)$$

$$n = 0.75 - 1.27 \log F_{cent}, \quad (25.35)$$

$$d = 0.14, \quad (25.36)$$

and

$$F_{cent} = (1 - \alpha) \exp(-T / T^{***}) + \alpha \exp(-T / T^*) + \exp(-T^{**} / T). \quad (25.37)$$

To use the Troe format, first add a row in the [Lindemann format](#). Then, add a row with the keyword **TROE**, followed by values for the parameters α , T^{***} , T^* , and T^{**} , as shown in the following figure.

```
REACTIONS
.
.
h + ch2o(+ M) = ch3o (+ M)      5.400e+11    0.454      2600.0
LOW / 2.200E+30   -4.800  5560.00/
TROE / 0.7580    94.00   1555.00   4200.00 /
.
.
END
```

Figure 25.28: An example of the Troe option in a reaction mechanism file.

SRI Format

The blending function F in Equation 25.31 in SRI format ([Kee et al., 1989](#)) is given by

$$F = d \left[a \exp\left(\frac{-b}{T}\right) + \exp\left(\frac{-T}{c}\right) \right]^x T^e, \quad (25.38)$$

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

where

$$X = \frac{1}{1 + \log^2 P_r}. \quad (25.39)$$

To use the SRI format, first add a row in the [Lindemann format](#). Then, add a row with the keyword *SRI*, followed by values for the parameters *a*, *b*, *c*, *d*, and *e*, as shown in the following figure. The default values for *d* and *e*, if not specified, are *d*=1 and *e*=0.

```
REACTIONS
.
.
ch3 + ch3(+ M) = h      + c2h5(+M)   4.989e+12   0.099      10600.0
HIGH / 3.80E-7 4.838 7710. /
SRI / 1.641 4334 2725 /
.
.
END
```

Figure 25.29: An example of the SRI option in a reaction mechanism file.

P-Log (Pressure-Dependent Arrhenius Rate Expressions) Format

When using the P-Log format ([Goodwin et al., 2012](#)), you must specify different rate parameters for discrete pressures within the pressure range of interest. For each set of parameters, add a row beginning with the keyword *PLOG*, followed by the pressure value (in *bar*) and the three Arrhenius reaction rate parameters, as shown in the following figure. These rows must be listed in ascending order by pressure, and you must provide at least two rows with different pressure values.

```
REACTIONS
.
.
h + h2o2 = oh + h2o      1.000E+13  0.000      3600.00
PLOG / 1.00000e-1 1.000E+13 .000 3600.00/
PLOG / 1.00000e+0 1.000E+13 .000 3600.00/
PLOG / 1.00000e+1 1.000E+13 .000 3600.00/
.
.
END
```

Figure 25.30: An example of the P-Log option in a reaction mechanism file.

When computing the actual reaction rate, CONVERGE will determine the rate parameters using logarithmic interpolation of the specified rate constants, at the current pressure from the simulation. If the current pressure is within 1% of one of the pressures for which you provided rate constants, then CONVERGE will use that set of rate parameters directly. However, if the current pressure is between the pressure points provided, then CONVERGE will determine the rate by a linear interpolation of *ln k* as a function of *ln P* (natural logarithms) by the following relation:

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

$$\ln k = \ln k_i + (\ln k_{i+1} - \ln k) \frac{\ln P - \ln P_i}{\ln P_{i+1} - \ln P}. \quad (25.40)$$

To calculate the rate of the reaction for a pressure lower than any of those provided, CONVERGE uses the rate parameters you specified for the lowest pressure. Likewise, in calculating the rate of the reaction for a pressure higher than any of those you specified, CONVERGE will use the rate parameters provided for the highest pressure.

You can optionally specify two sets of rate parameters for each pressure point to capture the complex temperature dependencies of certain reaction rates, such as those derived from quantum chemistry calculations. The following figure shows an excerpt from an example reaction mechanism file based on the work of [Goldsmith et al. \(2015\)](#):

```
REACTIONS
.
.
C2H3+O2=C2H3OO 3.41E+39 -8.040 14360
PLOG / 1.00E-02 1.55E+24 -5.450 9662 /
PLOG / 1.00E-02 1.78E-09 4.150 -4707 /
PLOG / 1.00E-01 3.48E+56 -15.010 19160 /
PLOG / 1.00E-01 2.36E+22 -4.520 2839 /
PLOG / 1.00E+00 3.34E+61 -15.790 20150 /
PLOG / 1.00E+00 6.13E+28 -5.890 3154 /
PLOG / 1.00E+01 4.16E+48 -11.210 16000 /
PLOG / 1.00E+01 3.48E+28 -5.370 3636 /
PLOG / 1.00E+02 3.41E+39 -8.040 14360 /
PLOG / 1.00E+02 1.03E+27 -4.720 3680 /
.
.
END
```

Figure 25.31: An example of the P-Log option with two sets of rate parameters for each pressure point.

In this case, each pressure point has two available rate constants, k_1 and k_2 . CONVERGE adds them to obtain the rate constant for each pressure point:

$$k = k_1 + k_2 = A_1 T^{\beta_1} \exp\left(-\frac{E_{a_1}}{RT}\right) + A_2 T^{\beta_2} \exp\left(-\frac{E_{a_2}}{RT}\right) \quad (25.41)$$

CONVERGE then interpolates between neighboring values of k as described in Equation 25.40. However, for reactions that are identified as [duplicate reactions](#), CONVERGE first interpolates k_1 and k_2 separately and then adds the interpolated values. The latter method is appropriate for duplicate reactions because they can proceed through different paths.

Chebyshev (Polynomial-Based) Reaction Rate Format

For selected reactions, you can use the bivariate Chebyshev polynomial to determine a phenomenological rate coefficient $k(T,P)$ ([Goodwin et al., 2012](#)) as

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

$$\log k(T, P) = \sum_{t=1}^{N_T} \sum_{p=1}^{N_p} \alpha_{tp} \phi_t(\tilde{T}) \phi_p(\tilde{P}), \quad (25.42)$$

where α_{tp} are the constants defining the rate, $\phi_n(x)$ is the Chebyshev polynomial of the first kind of degree n evaluated at x , and

$$\tilde{T} \equiv \frac{2T^{-1} - T_{\min}^{-1} - T_{\max}^{-1}}{T_{\max}^{-1} - T_{\min}^{-1}} \quad (25.43)$$

and

$$\tilde{P} \equiv \frac{2\log P - \log P_{\min} - \log P_{\max}}{\log P_{\max} - \log P_{\min}} \quad (25.44)$$

are a reduced temperature and reduced pressure, respectively, that map the ranges (T_{\min}, T_{\max}) and (P_{\min}, P_{\max}) to $(-1, 1)$. Note that the Chebyshev polynomials are not defined outside the interval $(-1, 1)$. Therefore, you should not extrapolate rates outside the range of temperatures and pressures defined for each reaction.

To use this option, add a row with the keywords *TCHEB* and *PCHEB*, followed by values for T_{\min} and T_{\max} and P_{\min} and P_{\max} , respectively. Begin the next row with the keyword *CHEB*, followed by the number of columns and rows, respectively, in the coefficient matrix α . Then, list the values of α in this row and subsequent rows, all labeled with the keyword *CHEB*. The following figure shows an example Chebyshev reaction in a reaction mechanism file.

```
REACTIONS
.
.
h + h2o2 = oh + h2o          1.000E+13   0.000      3600.00
TCHEB / 300 2000/ PCHEB / 0.132 132./
  CHEB / 8 4      1.2451E+01  7.3554E-01 -1.7244E-02 -8.1648E-04 -3.5023E+00/
  CHEB / 7.9339E-01 1.6780E-02 -7.1629E-04 -1.8341E+00 2.3633E-01 1.7728E-02/
  CHEB / 5.2831E-04 -7.3236E-01  3.8689E-03  4.4442E-03 2.8688E-04 -2.5432E-01/
  CHEB / 6.8284E-03 -3.9174E-03 -9.5956E-04 -1.3544E-01 1.2097E-02 1.7585E-03/
  CHEB / -4.1806E-04 2.8834E-02  3.8520E-03  1.3198E-03 4.6928E-04 -8.9989E-02/
  CHEB / -1.0474E-03 -3.0340E-04 1.0396E-04/
.
.
END
```

Figure 25.32: An example of the Chebyshev (polynomial-based) option in a reaction mechanism file.

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

Duplicate Reaction Option

Occasionally two or more reactions use the same set of reactants and products but proceed through distinctly different processes. To include reactions with the same reactants and products but different Arrhenius parameters, add a row with the keyword **DUPLICATE** (or **DUP**) for each line (including the first) that contains the reaction, as shown in the following figure.

```
REACTIONS
.
.
h2 + o2 = 2oh          1.700e+13    0.0      47780.0
DUPLICATE
h2 + o2 = 2oh          1.000e+13    0.0      47000.0
DUPLICATE
.
.
END
```

Figure 25.33: An example of the duplicate reaction option in a reaction mechanism file.

Isomer Lumping Reaction Option

CONVERGE can lump together isomer species during the mechanism reduction process. In the species section of the reaction mechanism file, use the prefix *ism* to indicate a lumped isomer species. In the reactions section of the reaction mechanism file, include the keyword **LUMP** in the row below any reaction that includes a lumped isomer species, followed by values of the constant α (defined below) for each lumped isomer species, in the order that the species appear in the reaction row. The **LUMP** keyword is necessary because the presence of a lumped isomer species requires a change in the calculation of the reaction rate. Note that you can apply other reaction options to these reactions, as shown in the following figure, where the lumping reaction option is combined with the reverse reaction option.

```
REACTIONS
.
.
AC5H10 + H = ism001 + H2          3.376E+05    2.36E+00    2.07000E+02
LUMP / 6.61365E-01 /
REV / 4.35200E+06    2.10000E+00    2.03300E+04 /
.
.
END
```

Figure 25.34: An example of the lumping reaction option in a reaction mechanism file.

The following sections describe the methods used to calculate reaction rates for different types of lumped reactions. In these sections, c_j is the concentration of species j (j represents the X, Y, Z, *ismX*, and *ismY* species in the following equations), α represents the specific constant associated with a lumped reaction (Lu and Law, 2008), and rest of the symbols and notations have their usual meaning.

Type 1

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

The reaction has a lumped isomeric species ismX and Y as reactants to form Z, as shown below.

```
ismX + Y =Z  
/LUMP/alphaX/
```

Figure 25.35: An excerpt of a reaction mechanism file showing TYPE 1 lumped reaction.

The reaction rates (q) for the lumped reactions are calculated as

$$q_{f_lumped} = \frac{q_f}{\alpha_X} \quad (25.45)$$

and

$$q_{r_lumped} = q_r \quad (25.46)$$

where

$$q_f = c_{ismX} c_Y A T^\beta \exp\left(-\frac{E_a}{RT}\right) \quad (25.47)$$

and

$$q_r = c_Z A_{rev} T^\beta \exp\left(-\frac{E_a}{RT}\right). \quad (25.48)$$

Type 2

The reaction has lumped isomeric species ismX and ismY as reactants to form Z, as shown in the following figure.

```
ismX+ismY = Z  
/LUMP/alphaX/alphaY
```

Figure 25.36: An excerpt of a reaction mechanism file showing TYPE 2 lumped reaction.

The reaction rates (q) for the lumped reactions are calculated as given by

$$q_{f_lumped} = \frac{q_f}{\alpha_X \alpha_Y} \quad (25.49)$$

and

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

$$q_{r_lumped} = q_r \quad (25.50)$$

where

$$q_f = c_{ismX} c_{ismY} A T^\beta \exp\left(-\frac{E_a}{RT}\right) \quad (25.51)$$

and

$$q_r = c_Z A_{rev} T^\beta \exp\left(-\frac{E_a}{RT}\right). \quad (25.52)$$

Type 3

The reaction has X and Y as reactants to form an isomeric species Z, as shown in the following figure.

```
X+Y = ismZ  
/LUMP/alphaZ/
```

Figure 25.37: An excerpt of a reaction mechanism file showing TYPE 3 lumped reaction.

The reaction rates (q) for the lumped reactions are calculated as given by

$$q_{f_lumped} = q_f \quad (25.53)$$

and

$$q_{r_lumped} = \frac{q_r}{\alpha_Z}, \quad (25.54)$$

where

$$q_f = c_X c_Y A T^\beta \exp\left(-\frac{E_a}{RT}\right) \quad (25.55)$$

and

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

$$q_r = c_{ismZ} A_{rev} T^\beta \exp\left(-\frac{E_a}{RT}\right). \quad (25.56)$$

Type 4

The reaction has an isomeric species X and Y as reactants to form an isomeric species Z, as shown in the following figure.

```
ismX+Y = ismZ  
/LUMP/alphaX/alphaZ/
```

Figure 25.38: An excerpt of a reaction mechanism file showing TYPE 4 lumped reaction.

The reaction rates (q) for the lumped reactions are calculated as given by

$$q_{f_lumped} = \frac{q_f}{\alpha_X} \quad (25.57)$$

and

$$q_{r_lumped} = \frac{q_r}{\alpha_Z}, \quad (25.58)$$

where

$$q_f = c_{ismX} c_Y A T^\beta \exp\left(-\frac{E_a}{RT}\right) \quad (25.59)$$

and

$$q_r = c_{ismZ} A_{rev} T^\beta \exp\left(-\frac{E_a}{RT}\right). \quad (25.60)$$

Fractional Order (FORD) Reaction Option

The fractional order (*FORD*) reaction option allows you to modify stoichiometric coefficients to account for fractional order reactions. The non-integer *FORD* value overrides the stoichiometric coefficients in

$$q_r = k_f \prod_{m=1}^M [X_m]^{v'_{m,r}} - k_r \prod_{m=1}^M [X_m]^{v''_{m,r}}, \quad (25.61)$$

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

where $v'_{m,r}$ and $v''_{m,r}$ are the stoichiometric coefficients for the reactants and products, respectively, for species m ; $[X_m]$ is the molar concentration of species m ; k_f and k_r are the forward and reverse rate constants, respectively; and q_r is the rate of progress for the r -th reaction.

To indicate a fractional order reaction, add one row for each species with a fractional order, as shown in the following figure. In each row, enter the keyword *FORD*, followed by the species and its fractional order.

```
REACTIONS
.
.
IC8H18 + OH => C8H17 + H2O      2.000E+13   0.00      3000.0
FORD / IC8H18    0.9 /
FORD / C8H17    0.8 /
.
.
END
```

Figure 25.39: An example of the fractional order reaction option in a reaction mechanism file.

User-Defined Reaction Rate

Some detailed chemistry simulations might include reactions with rate parameters not in Arrhenius format. For these reactions, you can define reaction rates via user-defined functions (UDFs). Refer to the CONVERGE 3.1 UDF Manual for more information about reaction rate UDFs.

To identify these reactions in the reaction mechanism file, add a row with the keyword *USER/<number>/* after the reaction row, as shown in the following figure. You must include values for the three Arrhenius rate parameters in the reaction row, but CONVERGE will not use these values. The number listed after the *USER* keyword must also be appended to the name of the *CONVERGE_REACTION_RATE* macro in your UDF.

```
REACTIONS
.
.
c7h16 => 3c2h4 + ch3 + h      0          0.0        0.0
user/1/
.
.
END
```

Figure 25.40: An example of the user-defined reaction rate option in a reaction mechanism file.

Ionization Reactions

Ionization occurs during combustion in gasoline and diesel engines. CONVERGE can simulate gas ionization reactions when you add electron constants to the reaction mechanism and thermodynamic data files.

Chapter 25: Input and Data Files

Materials Species Data and Reaction Mechanism: mech.dat

In the elements section of the reaction mechanism file, you must include the electron species name with the electron mass, e/5.447e-4/, in *atomic mass units*. In the species section, you must also include the electron species name, e, and, in the reactions section, you must include any ionization reactions.

```
elements
  h   c   o   n   e/5.447E-4/
end

species
c7h16    o2          n2          co2          h2o
co        h2          ho2         oh
h2o2     ho2         oh
h         o
no        n           ch
e         hco+
end

reactions
  hco+ + h2o      = h3o+      + co          1E+16 -0.09      0
  h3o+ + e       = h2o       + h           2.291E+18 -0.5      0
end
```

Figure 25.41: A sample reaction mechanism file for an ionization reaction.

In the thermodynamic data file, you must list the thermodynamic coefficients for the electron (e), which is a massless gas, as well as any ionized species (e.g., H₃O⁺) that are part of a ionization reaction.

```
thermo
  300.000 1000.000 5000.000
e               GAS      L 6/88e 1      0      0      0g  200.000
5000.000 1000.      1
  0.25000000E+01 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 2
  -0.74537500E+03-0.11720813E+02 0.25000000E+01 0.00000000E+00 0.00000000E+00 3
  0.00000000E+00 0.00000000E+00-0.74537500E+03-0.11720813E+02 0.00000000E+00 4
h3o+          ATcT Ah  3o  1e  -1      0g  298.150  5000.000
1000.      1
  2.49647765E+00 5.72844840E-03 -1.83953239E-06 2.73577348E-10 -1.54093917E-14 2
  7.16244227E+04 7.45850493E+00 3.79295251E+00 -9.10852723E-04 1.16363521E-05 3
  -1.21364865E-08 4.26159624E-12 7.14027518E+04 1.47156927E+00 7.25739701E+04 4
end
```

Figure 25.42: Sample thermodynamic data file for an ionization reaction.

Custom Fluid Properties: fluid_properties.in and fluid_properties.dat

CONVERGE includes a utility that serves as an interface for the [CoolProp fluid property library](#). This utility requires the *fluid_properties.in* file.

```
R245fa  fluid_name
200.0  temp_min
500.0  temp_max
2.0    temp_step
0.0    pres_min
10.0   pres_max
0.02   pres_step
```

Figure 25.43: Example *fluid_properties.in* file.

Chapter 25: Input and Data Files

Materials Custom Fluid Properties: *fluid_properties.in* and *fluid_properties.dat*

Table 25.12: Format of *fluid_properties.in*.

Parameter	Description
<i>fluid_name</i>	Name of the fluid for which to calculate properties.
<i>temp_min</i>	Minimum temperature (K) of the temperature range.
<i>temp_max</i>	Maximum temperature (K) of the temperature range.
<i>temp_step</i>	Temperature interval (K) for the steps between <i>temp_min</i> and <i>temp_max</i> .
<i>pres_min</i>	Minimum pressure (MPa) of the pressure range.
<i>pres_max</i>	Maximum pressure (MPa) of the pressure range.
<i>pres_step</i>	Pressure interval (MPa) for the steps between <i>pres_min</i> and <i>pres_max</i> .

The utility writes the *fluid_properties.dat* file. The header information in *fluid_properties.dat* includes the species name and molecular weight, the number of temperature and pressure intervals, and the number of output columns.

```
tabular_fluid_properties
# species_name    molar_mass [kg/mol]
R134a          1.0203200e-01
# num_temp      num_pres   num_column
128            500        13
# temperature    pressure     density      specific_vol   intern_energy
# [K]           [Pa]         [kg/m3]     [m3/kg]       [J/kg]
2.00000e+02   2.00000e+04  1.51048e+03  6.62026e-04  1.07380e+05
2.02000e+02   2.00000e+04  1.50582e+03  6.64496e-04  1.09864e+05

enthalpy      entropy      Cv          Cp          sound_speed
[J/kg]        [J/kg/K]    [J/kg/K]    [J/kg/K]    [m/s]
1.07404e+05  6.07305e+02  8.56083e+02  1.57345e+03  9.67633e+02
1.81393e+05  6.11669e+02  8.03106e+02  1.20815e+03  9.54749e+02

viscosity     conductivity  compressibility
[Pa*s]        [W/m/K]     -
8.67423e-04  1.27949e-01  8.42674e-04
8.22352e-04  1.26650e-01  8.07615e-04
.
```

Figure 25.44: An excerpt of a *fluid_properties.dat* file.

Table 25.13: Format of *fluid_properties.dat*.

Column	Quantity	Units
1	Temperature	K
2	Pressure	Pa
3	Density	kg/m ³
4	Specific Volume	m ³ /kg
5	Internal Energy	J/kg
6	Enthalpy	J/kg

Chapter 25: Input and Data Files

Materials Custom Fluid Properties: fluid_properties.in and fluid_properties.dat

Column	Quantity	Units
7	Entropy	J/kg-K
8	Specific Heat at Constant Volume	J/kg-K
9	Specific Heat at Constant Pressure	J/kg-K
10	Speed of Sound	m/s
11	Viscosity	N-s/m ²
12	Conductivity	W/m-K
13	Compressibility	[non-dimensional]

Solid Properties: solid.dat

When simulating one or more solids in CONVERGE, you must specify the solid material properties in a data file called *solid.dat*. The melting point, density, specific heat capacity, and thermal conductivity are required for all solid simulations. The electrical conductivity is required only if [*inputs.in* > feature_control > electric_potential_flag = 1](#).

```
metal
1700.0
0.00e+00 7.85e+00 5.61e+00 4.00e+01 4.00e+03
1.00e+01 7.85e+00 5.61e+00 4.00e+01 4.00e+03
2.00e+01 7.85e+00 5.61e+00 4.00e+01 4.00e+03
3.00e+01 7.85e+00 5.61e+00 4.00e+01 4.00e+03
4.00e+01 7.85e+00 5.61e+00 4.00e+01 4.00e+03
5.00e+01 7.85e+00 5.61e+00 4.00e+01 4.00e+03
6.00e+01 7.85e+00 5.61e+00 4.00e+01 4.00e+03
7.00e+01 7.85e+00 5.61e+00 4.00e+01 4.00e+03
.
.
```

Figure 25.45: An example *solid.dat* file. The first row is the name of the solid species (in this case, *metal*).

Chapter 25: Input and Data Files

Materials Solid Properties: solid.dat

Table 25.14: Format of *solid.dat*.

Column	Quantity	Units
1	Temperature	K
2	Density	kg/m ³
3	Specific heat capacity	J/kg-K
4	Thermal conductivity	W/m-K
5*	Electrical conductivity	S/m

*Required only when [inputs.in](#) > feature_control > electric_potential_flag = 1.

Anisotropic Conductivity: *aniso_cond.in*

To model anisotropic conduction of heat or electricity, set [inputs.in](#) > feature_control > aniso_cond_flag = 1 and include an *aniso_cond.in* file in your case setup.

The generalized form of the thermal or electrical flux in anisotropic media is given by

$$q_i = K_{ij} \frac{\partial \phi}{\partial x_j}, \quad (25.62)$$

where ϕ is the relevant scalar quantity (temperature or electric potential) and K_{ij} is the thermal or electrical conductivity. For isotropic media, $K_{ij} = K \delta_{ij}$ and Equation 25.62 reduces to the form that appears in the [energy transport equation](#) and [Ohm's law](#) (for the latter, the flux q_i is equivalent to the current density J_i , with the opposite sign).

In parts of the domain where the conductivity is anisotropic, CONVERGE calculates the conductivity as

$$K_{ij} = K \cdot M_{ij}, \quad (25.63)$$

where the dimensionless multiplier matrix M_{ij} is defined in *aniso_cond.in* and K is the temperature-dependent conductivity based on the relevant *.dat file (e.g., [solid.dat](#)).

Chapter 25: Input and Data Files

Materials Anisotropic Conductivity: aniso_cond.in

```
version: 3.1
---

- anisotropic_conductivity:
  description: battery 1
  equation: ENERGY
  type: ORTHOGONAL
  m_vec: [1.0, 0.01, 10.0]
  e_1: [1, 1, 0]
  e_2: [0, 0, 1]
  shape:
    type: REGION
    target_regions: [0, 1]
  moving_control:
    moving_flag: STATIONARY

- anisotropic_conductivity:
  description: battery 2
  equation: ELECTRICITY
  type: CYLINDRICAL
  m_radial: 10.0
  m_tangential: 1.0
  m_axial: 1.0
  x_orig: [0, 0.176, 0.028]
  e_axial: [1, 0, 0]
  shape:
    type: CYLINDER
    x1_center: [0, 0.176, 0.028]
    x2_center: [0.05, 0.176, 0.028]
    radius1: 0.05
    radius2: 0.06
    target_regions: [0]
  moving_control:
    moving_flag: TRANSLATE
    translate_velocity: [0.01, 0, 0]
    max_displace: 0.1

- anisotropic_conductivity:
  description: battery 3
  equation: ENERGY_ELECTRICITY
  type: BIAXIAL
  m_in: 10.0
  m_through: 1.0
  e_through: [1, 0, 0]
  shape:
    type: BOX
    x_center: [0.1, 0.1, 0.]
    x_size: [0.05, 0.176, 0.028]
    target_regions: [0]
  moving_control:
    moving_flag: ROTATE
    rotate_center: [0, 0, 0]
    rotate_axis: [1, 0, 0]
    rotate_speed: 30

- anisotropic_conductivity:
  description: battery 4
  equation: ENERGY
  type: GENERAL
  m_1_vec: [1.0, 10, 0]
  m_2_vec: [10.0, 1, 0.5]
  m_3_vec: [0, 0.5, 1]
  shape:
    type: SPHERE
    x_center: [0.1, 0.1, 0.]
    radius: 0.5
    target_regions: [0]
  moving_control:
    moving_flag: MOVE_WITH_BOUNDARY
    boundary: 3
    boundary_side: REVERSE
```

Chapter 25: Input and Data Files

Materials Anisotropic Conductivity: aniso_cond.in

```
- anisotropic_conductivity:  
  description: battery 5  
  equation: ENERGY  
  type: user_anisodef  
  shape:  
    type: SPHERE  
    x_center: [0.1, 0.1, 0.]  
    radius: 0.5  
    target_regions: [0]  
  moving_control:  
    moving_flag: user_moveshape
```

Figure 25.46: An example *aniso_cond.in* file.

Table 25.15: Format of *aniso_cond.in*.

Parameter	Description
- <i>anisotropic_conductivity</i>	Settings block for a part of the domain with anisotropic conductivity. Repeat for each anisotropic domain.
<i>description</i> *	Description of the anisotropic domain.
<i>equation</i>	Equation(s) to which the anisotropic conductivity applies. <i>ENERGY</i> = Energy equation, <i>ELECTRICITY</i> = Electric potential equation, or <i>ENERGY_ELECTRICITY</i> = Both energy and electric potential equations.
<i>type</i>	Multiplier matrix definition type. <u>ORTHOGONAL</u> , <u>CYLINDRICAL</u> , <u>BIAXIAL</u> , GENERAL (specify each matrix component directly), or <i>user_<UDF name></i> (calculate matrix components from a user-defined function (UDF) based on the <i>CONVERGE_ANISO_COND</i> macro). Refer to the CONVERGE 3.1 UDF Manual for more information about UDFs.
<i>m_vec</i>	Vector containing the conductivity multipliers (m_1, m_2, m_3). Used only for <i>type</i> = <u>ORTHOGONAL</u> .
<i>e_1\$</i>	First orthogonal vector \mathbf{e}_1 . Used only for <i>type</i> = <u>ORTHOGONAL</u> .
<i>e_2\$</i>	Second orthogonal vector \mathbf{e}_2 . Used only for <i>type</i> = <u>ORTHOGONAL</u> .
<i>m_axial</i>	Conductivity multiplier in the axial direction (m_a). Used only for <i>type</i> = <u>CYLINDRICAL</u> .
<i>m_radial</i>	Conductivity multiplier in the radial direction (m_r). Used only for <i>type</i> = <u>CYLINDRICAL</u> .
<i>m_tangential</i>	Conductivity multiplier in the tangential direction (m_θ). Used only for <i>type</i> = <u>CYLINDRICAL</u> .
<i>x_orig</i>	Origin of cylindrical coordinate system. Used only for <i>type</i> = <u>CYLINDRICAL</u> .

Chapter 25: Input and Data Files

Materials Anisotropic Conductivity: aniso_cond.in

Parameter	Description
<i>e_axial\$</i>	Axial vector \mathbf{e}_a . Used only for <i>type</i> = CYLINDRICAL .
<i>m_in</i>	In-plane conductivity multiplier (m_{in}). Used only for <i>type</i> = BIAXIAL .
<i>m_through</i>	Through-plane conductivity multiplier ($m_{through}$). Used only for <i>type</i> = BIAXIAL .
<i>e_through\$</i>	Through-plane vector $\mathbf{e}_{through}$. Used only for <i>type</i> = BIAXIAL .
<i>m_1_vec</i>	First row of conductivity multiplier matrix (M_{11}, M_{12}, M_{13}). Used only for <i>type</i> = GENERAL .
<i>m_2_vec</i>	Second row of conductivity multiplier matrix (M_{21}, M_{22}, M_{23}). Used only for <i>type</i> = GENERAL .
<i>m_3_vec</i>	Third row of conductivity multiplier matrix (M_{31}, M_{32}, M_{33}). Used only for <i>type</i> = GENERAL .
<i>shape</i>	
<i>type</i>	Shape of the anisotropic domain. <i>BOX</i> , <i>SPHERE</i> , <i>CYLINDER</i> , or <i>REGION</i> .
<i>x_center</i>	Center of the anisotropic domain (x, y, and z coordinates). Used only for <i>shape > type</i> = <i>BOX</i> or <i>SPHERE</i> .
<i>x1_center</i>	Center of the first end circle (x, y, and z coordinates). Used only for <i>shape > type</i> = <i>CYLINDER</i> .
<i>x2_center</i>	Center of the second end circle (x, y, and z coordinates). Used only for <i>shape > type</i> = <i>CYLINDER</i> .
<i>radius</i>	Radius of the source (m). Used only for <i>shape > type</i> = <i>SPHERE</i> .
<i>radius_1</i>	Radius of the first end circle (m). Used only for <i>shape > type</i> = <i>CYLINDER</i> .
<i>radius_2</i>	Radius of the second end circle (m). Used only for <i>shape > type</i> = <i>CYLINDER</i> .
<i>x_size</i>	Size of the box in the x, y, and z directions (m). Used only for <i>shape > type</i> = <i>BOX</i> .
<i>target_regions</i>	List of region IDs to which the anisotropic conductivity applies. Required for <i>shape > type</i> = <i>REGION</i> , otherwise optional.
<i>moving_control</i>	

Chapter 25: Input and Data Files

Materials Anisotropic Conductivity: aniso_cond.in

Parameter	Description
<i>moving_flag</i>	STATIONARY = Shape does not move, TRANSLATE = Shape translates at the specified velocity, ROTATE = Shape rotates at the specified speed, MOVE_WITH_BOUNDARY = Shape moves with the specified boundary, or <i>user_<UDF name></i> (calculate motion from a user-defined function (UDF) based on the CONVERGE_SHAPE_MOVE macro). Refer to the CONVERGE 3.1 UDF Manual for more information about UDFs.
<i>boundary</i>	ID of the boundary with which the shape moves. Must be a WALL or WALL-type INTERFACE boundary. Used only for <i>moving_control > moving_flag = MOVE_WITH_BOUNDARY</i> .
<i>boundary_side</i>	FORWARD or REVERSE. Applies only if the boundary specified above is an INTERFACE boundary.
<i>translate_velocity</i>	Velocity of the shape (m/s). Used only for <i>moving_control > moving_flag = TRANSLATE</i> .
<i>max_displace*</i>	Maximum distance that the shape can move (m). Used only for <i>moving_control > moving_flag = TRANSLATE</i> .
<i>rotate_center</i>	Center of rotation (x, y, and z coordinates). Used only for <i>moving_control > moving_flag = ROTATE</i> .
<i>rotate_axis</i>	Axis of rotation. Used only for <i>moving_control > moving_flag = ROTATE</i> .
<i>rotate_speed</i>	Speed of rotation (degrees/s). Used only for <i>moving_control > moving_flag = ROTATE</i> .

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

§ It is not necessary to specify a vector with a magnitude of 1. CONVERGE automatically renormalizes the components so that the vector has unit magnitude.

With an ORTHOGONAL definition of the multiplier matrix M_{ij} (Figure 25.47), the conductivity has distinct values in each of three directions defined by the orthogonal unit vectors \mathbf{e}_1 , \mathbf{e}_2 , and \mathbf{e}_3 . In *aniso_cond.in*, you provide \mathbf{e}_1 and \mathbf{e}_2 and three conductivity multipliers (m_1 , m_2 , and m_3). CONVERGE calculates \mathbf{e}_3 from the cross product ($\mathbf{e}_3 = \mathbf{e}_1 \times \mathbf{e}_2$) and calculates the matrix components from

$$M_{ij} = m_1 e_{1i} e_{1j} + m_2 e_{2i} e_{2j} + m_3 e_{3i} e_{3j}. \quad (25.64)$$

Chapter 25: Input and Data Files

Materials Anisotropic Conductivity: aniso_cond.in

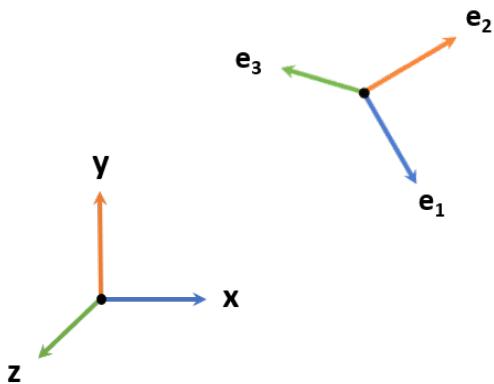


Figure 25.47: Unit vectors for **ORTHOGONAL** multiplier matrix definition.

With a *CYLINDRICAL* definition of M_{ij} (Figure 25.48), the conductivity has distinct values in the axial, radial, and tangential directions. In *aniso_cond.in*, you provide the axial unit vector \mathbf{e}_a and the coordinates of the origin, along with three conductivity multipliers (m_a , m_r , and m_θ). For a given point \mathbf{x} , CONVERGE calculates the radial unit vector \mathbf{e}_r as

$$\mathbf{e}_r = \frac{\mathbf{x} - \mathbf{x}_p}{|\mathbf{x} - \mathbf{x}_p|}, \quad (25.65)$$

where \mathbf{x}_p is the projection of \mathbf{x} onto \mathbf{e}_a , and calculates the tangential unit vector \mathbf{e}_θ from the cross product ($\mathbf{e}_\theta = \mathbf{e}_a \times \mathbf{e}_r$). The matrix components are given by

$$M_{ij} = m_a e_{ai} e_{aj} + m_r e_{ri} e_{rj} + m_\theta e_{\theta i} e_{\theta j}. \quad (25.66)$$

Chapter 25: Input and Data Files

Materials Anisotropic Conductivity: aniso_cond.in

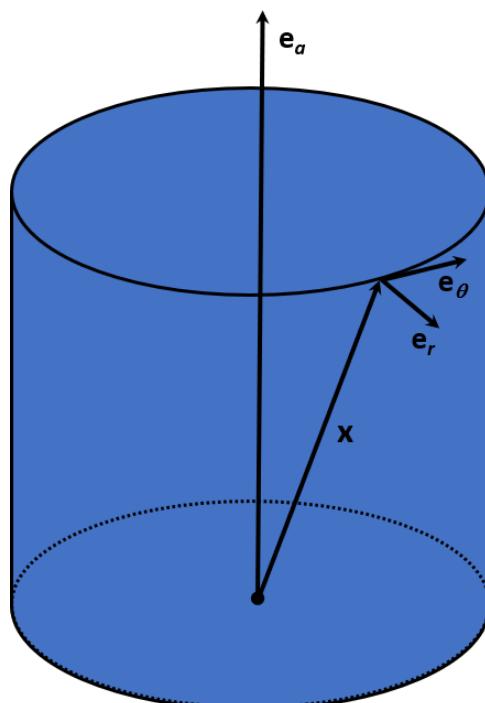


Figure 25.48: Unit vectors for CYLINDRICAL multiplier matrix definition.

With a BIAXIAL definition of M_{ij} (Figure 25.49), the conductivity has distinct values in the axial direction $e_{through}$ and in the plane perpendicular to $e_{through}$. In *aniso_cond.in*, you provide the through-plane unit vector $e_{through}$ and two conductivity multipliers ($m_{through}$ and m_{in}). CONVERGE randomly generates the first in-plane unit vector e_{in1} and calculates the second in-plane unit vector from the cross product ($e_{in2} = e_{through} \times e_{in1}$). The matrix components are given by

$$M_{ij} = m_{through} e_{through,i} e_{through,j} + m_{in} e_{in1,i} e_{in1,j} + m_{in} e_{in2,i} e_{in2,j}. \quad (25.67)$$

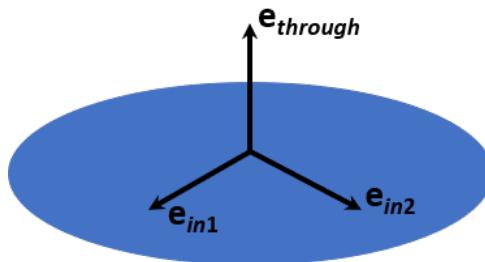


Figure 25.49: Unit vectors for BIAXIAL multiplier matrix definition.

Species Definition: species.in

Use the *species.in* file to define species that are not included in the [reaction mechanism file](#). The *species.in* file can contain discrete phase parcels; gas, liquid, non-Newtonian liquid, and solid species; [scalars](#); [passives](#); non-transport passives; internal passives; and ECFM3Z fuels. Typically, *species.in* includes discrete phase parcels; gas, liquid, and solid species; and passives and [non-transport passives](#).

Any gas-phase species that is defined in the [reaction mechanism file](#) does not need to be defined in *species.in*.

To direct CONVERGE to generate total mass, mass fraction, and other values for each species, use [inputs.in](#) > *output_control* > *species_output_flag*. To direct CONVERGE to write cell-species output for species and passives, use [post.in](#) > *cells* > *species_massfrac* and *cells* > *passive*, respectively.

```
version: 3.1
---

parcel:
  - parcel_species_name1
  - parcel_species_name2

liquid_non_newtonian:
  - liqNN_species_name1
  - liqNN_species_name2

liquid:
  - liq_species_name1
  - liq_species_name2

gas:
  - gas_species_name1
  - gas_species_name2

solid:
  - solid_species_name1
  - solid_species_name2

passive_nt:
  - passivent_name1
  - passivent_name2

scalar:
  scalar_name1: 0.78
  scalar_name2: 0.78

passive:
  passive_name1: 0.78
  passive_name2: 0.78

internal_passive:
  overwrite_schmidt_number:
    ipassive_name1: 0.78
    ipassive_name2: 0.78
  overwrite_solution_method:
    ipassive_name1: IN_PISO
    ipassive_name2: OUT_PISO

ecfm3z_fuel:
  - fuel_name1
  - fuel_name2
```

Figure 25.50: An example *species.in* file.

Chapter 25: Input and Data Files

Materials Species Definition: species.in

Table 25.16: Format of the *parcel*, *liquid_non_newtonian*, *liquid*, *gas*, *solid*, *passive_nt*, *scalar*, and *passive* settings blocks.

Parameter	Description
<code><species type>*</code>	<code>parcel</code> , <code>liquid_non_newtonian</code> , <code>liquid</code> , <code>gas</code> , <code>solid</code> , <code>passive_nt</code> , <code>scalar</code> , or <code>passive</code>
<code><species name></code>	Parcel name, non-Newtonian liquid species name, liquid species name, gas species name, solid species name, non-transport passive name, scalar name, passive name, or internal passive name.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

For *gas* species, the thermodynamic properties for the corresponding species must be included in the [thermodynamic properties data file](#) and the transport properties for gas species must be included in [gas.dat](#). For *parcel*, *liquid*, and *liquid_non_newtonian* species, the liquid properties must be included in [liquid.dat](#). For *solid* species, the solid properties must be included in [solid.dat](#).

For *scalar* and *passive* species, you must specify the Schmidt number of the species in the same row as the species name.

Many [combustion](#) and [emissions](#) models require model-specific passive species. Because their names and behaviors are generally defined by the model, CONVERGE defines these internal passives at runtime. Optionally, you can specify alternative behavior in the *internal_passive* settings block. If you define an internal passive in this block, you must specify the Schmidt number and also instruct CONVERGE whether you want the internal passive to be solved inside or outside of the [iterative algorithm](#) (e.g., PISO).

Table 25.17: Format of the *internal_passive* settings blocks.

Parameter	Description
<code>internal_passive</code>	
<code>overwrite_schmidt_number*</code>	
<code><species name></code>	Internal passive name and Schmidt number. Repeat for each internal passive.
<code>overwrite_solution_method*</code>	
<code><species name></code>	Internal passive name and iterative algorithm keyword (<i>IN_PISO</i> or <i>OUT_PISO</i>). Repeat for each internal passive. <i>IN_PISO</i> = Solve this internal passive inside the iterative algorithm, <i>OUT_PISO</i> = Solve this internal passive after the iterative algorithm.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

For the [ECFM](#) and [ECFM3Z](#) combustion models, if you are running a multi-fuel simulation, you must provide an *ecfm3z_fuel* settings block. This settings block tells CONVERGE which base species are treated as fuels in the simulation (and which correspond to the *BF_<fuel>* and *UMF_<fuel>* passives). If you are running a single-fuel simulation, this settings block is

Chapter 25: Input and Data Files

Materials Species Definition: species.in

optional. CONVERGE Studio will automatically generate this settings block if you have set up ECFM or ECFM3Z with multiple fuel species.

Table 25.18: Format of the *ecfm3z_fuel* settings block.

Parameter	Description
<i>ecfm3z_fuel</i> *	
< <i>species name</i> >	ECFM or ECFM3Z fuel species. Repeat for each fuel species.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Use the [inputs.in](#) > *output_control* > *species_output_flag* parameter to control what species mass information CONVERGE writes to output files.

Non-Transport Passives

Specify non-transport [passives](#) by including the name of each non-transport passive in the *passive_nt* settings block of *species.in*. You can use non-transport passives to numerically store conserved scalars in the domain. You can create custom names for passives and [manually initialize the values of these passives](#), or you can specify internal passives (passives that CONVERGE uses to track specific quantities).

Table 25.19: Description of internal passives that can be included in the *passive_nt* settings block.

Internal passive name	Description
<i>HC</i>	Hydrocarbons (for gas-phase-only hydrocarbons).
<i>HC_LIQUIDFUEL</i>	Hydrocarbons (for combined liquid- and gas-phase hydrocarbons).
<i>TUR_FLAMETHICKNE</i>	Turbulent flame thickness.
<i>SS</i>	
<i>CHEM_SRC</i>	Chemical source energy (in $J/s \cdot m^3$).
<i>G_EQN</i>	Required to use the G-Equation combustion model .
<i>TEMP_SGS</i> and <i>VEL_SGS</i>	The temperature and velocity sub-grid scales parameters (<i>temp_sgs</i> and <i>vel_sgs</i>). These values will be calculated if the non-transport passives are included, even if you are not using temperature-based or velocity-based AMR.
<i>CHEM_STIFF</i>	Invokes stiffness-based load balancing for a simulation that includes the SAGE detailed chemical kinetics solver .
<i>NUM_CELLS_IN_PAIR</i>	Tracks the number of cells that are paired with one another, which allows you to visualize which cells are paired during the simulation.
<i>CELL_PAIR_ID</i>	Tracks the unique index for cells that are part of the same cell pair.
<i>LSR_PARAM</i>	The length scale resolution parameter (Piscagila, 2013) monitors the quality of a large eddy simulation mesh or is used in AMR control. <i>LSR_PARAM</i> is defined in terms of grid spacing Δ and Kolmogorov length scale η :

Chapter 25: Input and Data Files

Materials Species Definition: species.in

Internal passive name	Description
$LSR = \frac{\Delta}{60\eta}$	

You can also keep track of turbulence statistics for transient simulations using non-transport passives. Refer to the [Non-Transport Passives for Turbulence Statistics](#) section below for more information.

User-Defined Global Variables as Non-Transport Passives

You can define global variables via a user-defined function. The list of definitions will refer to the list of *PASSIVE_NT*s named in *species.in*. User-defined global variables can be arbitrarily named. If you will be using both pre-defined and user-defined non-transport passives, we recommend that you list the user-defined non-transport passives first, which will make it easier to define them using the list in the UDF. Consult the CONVERGE 3.1 UDF Manual for details.

Refer to the [post.in](#) section of this chapter for information about how to generate output for non-transport passives.

Non-Transport Passives for Turbulence Statistics

CONVERGE contains predefined non-transport [passives](#) that you can use to track [turbulence-related statistics](#) in a transient simulation. To activate turbulence statistics, set [turbulence.in > Turbulence_statistics > turb_stat_flag = 1](#). Specify any of the predefined non-transport passives after *PASSIVE_NT* in *species.in*.

Table 25.20: Non-transport passives available for tracking turbulence statistics.

Turbulence statistics passive name	Description
<i>bar_Y.<speciesname></i>	Mean mass fraction of <speciesname> (e.g., <i>bar_Y.CH4</i>).*
<i>rms_Y.<speciesname></i>	Fluctuations (root mean square (RMS) mass fraction statistics) of <speciesname> (e.g., <i>rms_Y.CH4</i>).*
<i>bar_X.<speciesname></i>	Mean mole fraction of <speciesname> (e.g., <i>bar_X.CH4</i>).*
<i>rms_X.<speciesname></i>	Fluctuations (RMS mole fraction statistics) of <speciesname> (e.g., <i>rms_X.CH4</i>).*
<i>bar_passive.<passivename></i>	Mean passive quantity of <passivename> (e.g., <i>bar_passive.flow_tracer</i>).*
<i>rms_passive.<passivename></i>	Fluctuations (RMS statistics of the passive quantity) of <passivename> (e.g., <i>rms_passive.flow_tracer</i>).*
<i>bar_rho</i>	Mean density.

Chapter 25: Input and Data Files

Materials Species Definition: species.in

Turbulence statistics passive name	Description
<i>rms_rho</i> or <i>bar_rhop_rhop</i>	RMS density. You can specify either <i>rms_rho</i> or <i>bar_rhop_rhop</i> . $rms_rho = \sqrt{bar_rhop_rhop} = \sqrt{\rho'\rho'}$
<i>bar_p</i>	Mean pressure.
<i>rms_p</i> or <i>bar_pp_pp</i>	RMS pressure. You can specify either <i>rms_p</i> or <i>bar_pp_pp</i> . $rms_p = \sqrt{bar_pp_pp} = \sqrt{p'p'}$
<i>bar_T</i>	Mean temperature.
<i>rms_T</i> or <i>bar_Tp_Tp</i>	RMS temperature. You can specify either <i>rms_T</i> or <i>bar_Tp_Tp</i> . $rms_T = \sqrt{bar_Tp_Tp} = \sqrt{T'T'}$
<i>bar_u</i>	Mean velocity in the x direction.
<i>bar_v</i>	Mean velocity in the y direction.
<i>bar_w</i>	Mean velocity in the z direction.
<i>rms_u</i> or <i>bar_up_up</i>	RMS velocity (Reynolds stress) in the x direction. You can specify either <i>rms_u</i> or <i>bar_up_up</i> . $rms_u = \sqrt{bar_up_up} = \sqrt{u'u'}$
<i>rms_v</i> or <i>bar_vp_vp</i>	RMS velocity (Reynolds stress) in the y direction. You can specify either <i>rms_v</i> or <i>bar_vp_vp</i> . $rms_v = \sqrt{bar_vp_vp} = \sqrt{v'v'}$
<i>rms_w</i> or <i>bar_wp_wp</i>	RMS velocity (Reynolds stress) in the z direction. You can specify either <i>rms_w</i> or <i>bar_wp_wp</i> . $rms_w = \sqrt{bar_wp_wp} = \sqrt{w'w'}$
<i>bar_up_vp</i>	Mean $u'v'$ (Reynolds stress).
<i>bar_up_wp</i>	Mean $u'w'$ (Reynolds stress).
<i>bar_vp_wp</i>	Mean $v'w'$ (Reynolds stress).
<i>vol_frac</i>	Spray parcel volume fraction.

* Note that all species and passive names must match those specified in the [reaction mechanism file](#) or *species.in*.

Composite Species: composite.in

In some simulations, you may wish to create composite species, which are composed of multiple base species. You can have composites for gas, liquid, and liquid parcel phases. By creating a composite species, you can reduce the number of species that need to be solved.

Chapter 25: Input and Data Files

Materials Composite Species: composite.in

For example, you could define air as a composite of nitrogen, oxygen, water, and carbon dioxide, which would reduce the number of species from four to one.

To activate composite species, set [inputs.in](#) > *feature_control* > *composite_flag* = 1 and include a *composite.in* file in your case setup. You must include the name of the composite species in the [reaction mechanism file](#) or in [species.in](#).

```
version: 3.1
---

composites:
  - composite:
    composite_name: SPECIES_A
    phase: GAS_PHASE
    base_species:
      C8H18: 0.66667
      C2H2: 0.33333
  - composite:
    composite_name: SPECIES_B
    phase: GAS_PHASE
    base_species:
      C8H18: 1.00000
  - composite:
    composite_name: SPECIES_C
    phase: GAS_PHASE
    base_species:
      C8H18: 0.62500
      C2H3: 0.06250
      C3H5: 0.31250
  - composite:
    composite_name: SPECIES_D
    phase: LIQUID_PHASE
    base_species:
      C7H16: 1.00000
  - composite:
    composite_name: SPECIES_E
    phase: PARCEL_PHASE
    base_species:
      SPECIES_A: 0.33333
      C7H8: 0.66667
...
```

Figure 25.51: An example *composite.in* file.

Table 25.21: Format of *composite.in*.

Parameter	Description
<i>composites</i>	
- <i>composite</i>	This settings sub-block specifies the composite species parameters. Repeat this sub-block through <i>base_species</i> for each group.
<i>composite_name</i>	A unique name of a composite species. You must also include this name in the reaction mechanism file or <i>species.in</i> .
<i>phase</i>	Phase of the composite listed above: <i>GAS_PHASE</i> , <i>LIQUID_PHASE</i> , <i>PARCEL_PHASE</i> , or <i>SOLID_PHASE</i> . <i>PARCEL_PHASE</i> is available only for liquid parcels.
<i>base_species</i>	List of base species.

Chapter 25: Input and Data Files

Materials Composite Species: composite.in

Parameter	Description
<code><base species name></code>	Mass fraction of the specified base species. You must include thermodynamic data for each base species in the thermodynamic properties data file.
	Repeat this row for each base species for the composite species.

For a simulation that includes composite species, CONVERGE will calculate the critical properties of the composite from the [species-specific critical properties](#).

Thermodynamic Properties: `therm.dat`

CONVERGE generates gas properties from species-specific polynomial coefficients. These values, which must be present for each gas-phase species in your simulation, are stored in a thermodynamic data file (*e.g.*, `therm.dat`) or a tabular thermodynamic data file named [`tabular_therm.dat`](#).

The name of the thermodynamic data file (*e.g.*, `therm.dat`) must match [`inputs.in > thermodynamic_filename`](#). For simplicity, in this section we will assume that the thermodynamic data file is named `therm.dat`.

If both `therm.dat` and [`tabular_therm.dat`](#) are saved to the same input directory and if both of these files contain thermodynamic data for the same species, CONVERGE will use the data from `tabular_therm.dat`.

For each gas phase species in the file, `therm.dat` must contain the species name, the elemental composition of the species, the phase of the species, and temperature ranges over which a polynomial is fit to thermodynamic data. It is important to note that all columns and numbers in this file must be separated by spaces. Tabs cannot be used.

CONVERGE supports both the [NASA 7](#) and [NASA 9](#) formats for the thermodynamic data file. These two formats are described below.

NASA 7 Format

Chapter 25: Input and Data Files

Materials Thermodynamic Properties: therm.dat

Figure 25.52: An excerpt of a NASA 7-formatted thermodynamic data file. Note that the species names must NOT be preceded by #. The format has been slightly modified for ease of display; actual column numbers for each item are shown in parenthesis.

The first row contains the keyword *THERMO*. The second row specifies three temperatures: a low temperature, a common temperature, and a high temperature. These temperatures define the two ranges over which a polynomial is fit to thermodynamic data, as follows:

$$\frac{C_{p,m}^o}{R} = a_{1m} + a_{2m}T_m + a_{3m}T_m^2 + a_{4m}T_m^3 + a_{5m}T_m^4 \quad (25.68)$$

$$\frac{H_m^o}{RT_m} = a_{1m} + \frac{a_{2m}}{2} T_m + \frac{a_{3m}}{3} T_m^2 + \frac{a_{4m}}{4} T_m^3 + \frac{a_{5m}}{5} T_m^4 + \frac{a_{6m}}{T_m} \quad (25.69)$$

$$\frac{S_m^o}{R} = a_{1m} \ln T_m + a_{2m} T_m + \frac{a_{3m}}{2} T_m^2 + \frac{a_{4m}}{3} T_m^3 + \frac{a_{5m}}{4} T_m^4 + a_{7m} \quad (25.70)$$

The heat of formation is calculated by evaluating Equation 25.69 at 298.15 K. Therefore, the sensible enthalpy, H_{gen}° , is given by

$$H_m^S(T_m) = \left(a_{1m} + \frac{a_{2m}}{2} T_m + \frac{a_{3m}}{3} T_m^2 + \frac{a_{4m}}{4} T_m^3 + \frac{a_{5m}}{5} T_m^4 + \frac{a_{6m}}{T_m} \right) R T_m - H_m^o(298.15). \quad (25.71)$$

The third row specifies the species name (*h* in this example), followed by a date (120186), the atomic symbols and number of each atom type (*h* 1), the phase of the species (g for gas), three temperatures (low, high, and common), and the integer 1. This integer is not required by CONVERGE but is included to help distinguish the thermodynamic data for different species. The atomic symbols and numbers must contain exactly five characters for each element, with no spaces between elements (e.g., *h* $\square\square\square2*o* $\square\square\square1 for H₂O, where each \square represents a space). The three temperatures in this row are optional. If species-specific temperatures are not listed in this row, for this species CONVERGE will use the temperatures specified on row 2. The next three rows contain coefficient values for the$$

Chapter 25: Input and Data Files

Materials Thermodynamic Properties: therm.dat

polynomial fits for the standard state molar heat capacity at constant pressure C_p^o , the standard state molar enthalpy H^o , and the standard state molar entropy S^o , respectively, for the m -th species.

The fourth row specifies the coefficients a_1 through a_5 for the upper temperature interval, followed by the integer 2. The fifth row contains the coefficients a_6 and a_7 for the upper temperature interval and a_1 , a_2 , and a_3 for the lower temperature interval, followed by the integer 3. The sixth row contains the coefficients a_4 through a_7 for the lower temperature interval, followed by the integer 4. Repeat rows three through six for each species, followed by an *end* statement.

The original NASA 7 format supports up to four atomic symbols for each species. CONVERGE also supports the Chemkin version of this format ([Kee et al., 1989](#)) in which a fifth atomic symbol can be listed after the species-specific temperature ranges, as shown below. If a fifth element is included, it must be listed in columns 74-78.

Atomic symbol and number of atoms (up to four elements, 25-44)									
Low (46-55), high (56-65), and common (66-73) temperatures for this species (optional)									
Species name (1-18)									
THERMO Date (19-24)									Phase (45)
300.000	1000.000	5000.000	70590	H	7S	00	0C	2G	0300.00
2.46190400e+0	6.05916600e-3	-2.00497700e-6	3.13600300e-10	-1.93831700e-14					1
-6.49327000e+3	7.47209700e+0	2.20435200e+0	1.01147600e-2	-1.46526500e-5					2
1.44723500e-8	5.32850900e-12	-6.52548800e+3	8.12713800e+0						3
									4
.									
END									
									Atomic symbol and number of atoms (fifth element, 74-78, optional)

Figure 25.53: An excerpt of a thermodynamic data file in the Chemkin version of the NASA 7 format with a fifth element in columns 74-78.

NASA 9 Format

The NASA 9 format is specified in [McBride and Gordon \(1996\)](#).

Temperature ranges									
Phase (0 = Gas or 1 = Condensed, 52)									
Species name (1-24)									
Number of temperature ranges for this species (2)									
THERMO_NASA9	300.000	1000.000	5000.000	20000.000	0	0	0	0	
O2	2	0	2.00	0.00	0.00	0.00	0.00	0.00	
First temperature range (2-21)	2	0	2.00	0.00	0.00	0.00	0.00	0.00	
Coefficients a_1-a_5 (1-80)	300.000	1000.000	7	-2.0	-1.0	0.0	1.0	2.0	3.0
Coefficients a_6-a_8 (1-48)	0.0000000E+00	0.0000000E+00	3.21293600e+00	1.12748600e-03	-5.7561500e-07				
	0.0000000E+00	0.0000000E+00	0.0000000E+00	-1.0052490e+03	6.03473800e+00				
	1.31387700e-09	-8.7685540e-13	3.69757800e+00	6.13519700e-04	-1.25884200e-07				
	1000.000	5000.000	7	-2.0	-1.0	0.0	1.0	2.0	3.0
	0.0000000E+00	0.0000000E+00	0.0000000E+00	-1.2339300e+03	3.18916600e+00				
	1.77528100e-11	-1.1364350e-15	0.0000000E+00						
.									
END									
Atomic symbols and numbers (11-50)									Molecular weight (53-65)
Number of coefficients provided (23)									0.00 or heat of formation at 298.15 K (66-80)
Exponents (24-63)									$H^o(298.15\text{ K}) - H^o(0\text{ K})$ (66-80)
b_1 and b_2 (49-80)									
Data for second temperature range									

Figure 25.54: An excerpt of a NASA 9-formatted thermodynamic data file. Note that the species names must NOT be preceded by #. The format has been modified slightly for ease of display; actual column numbers for each item are shown in parenthesis.

Chapter 25: Input and Data Files

Materials Thermodynamic Properties: therm.dat

The first row contains the keywords *THERMO NASA9*. The second row specifies the temperature ranges over which a polynomial is fit to the thermodynamic data. Each adjacent pair of temperatures specify a range for which CONVERGE will accept data. Not all species must have data for each range. (Note that the NASA 7 format has only three temperatures on this line, so if CONVERGE reads a fourth temperature, it will expect the NASA 9 format.) The next set of rows repeat for each species until the end of the file is indicated by the keyword *end*.

The first row for each species specifies the species name (O2 in this example), followed by optional information. The second line specifies the number of temperature ranges for which data will be provided (in this example, two), the atomic symbols and number of each atom type (O 2.00), an integer to indicate the phase (0 for gas phase or 1 for condensed phase), and the molecular weight. The line ends with 0.000 or the heat of formation at 298.15 K in *J/mol*. The third row begins with the first temperature range, followed by the number of coefficients a_{im} provided for the polynomial fits, the temperature exponents in the empirical equation for $C_{p,m}^o$ and the standard state molar enthalpy $H_m^o(298.15\text{ K}) - H_m^o(0\text{ K})$.

The second row for each temperature range for each species specifies the coefficients a_1 through a_5 while the third row contains the coefficients a_6 through a_8 , as well as the integration constants b_1 and b_2 for enthalpy and entropy (these constants are optional). The third, fourth and fifth rows are repeated for each temperature range, and the entire block is repeated for each species.

Equations 25.72 through 25.74 ([Gordon and McBride, 1994](#)) specify how the standard state molar heat capacity at constant pressure ($C_{p,m}^o$), standard state molar enthalpy (H_m^o), and standard state molar entropy (S_m^o), respectively, are calculated as follows:

$$\frac{C_{p,m}^o}{R} = a_{1m}T_m^{-2} + a_{2m}T_m^{-1} + a_{3m} + a_{4m}T_m + a_{5m}T_m^2 + a_{6m}T_m^3 + a_{7m}T_m^4 \quad (25.72)$$

$$\frac{H_m^o}{RT_m} = -a_{1m}T_m^{-2} + a_{2m}T_m^{-1} \ln(T_m) + a_{3m} + \frac{a_{4m}}{2}T_m + \frac{a_{5m}}{3}T_m^2 + \frac{a_{6m}}{4}T_m^3 + \frac{a_{7m}}{5}T_m^4 + \frac{b_{1m}}{T_m} \quad (25.73)$$

$$\frac{S_m^o}{R} = -\frac{a_{1m}}{2}T_m^{-2} - a_{2m}T_m^{-1} + a_{3m} \ln(T_m) + a_{4m}T_m + \frac{a_{5m}}{2}T_m^2 + \frac{a_{6m}}{3}T_m^3 + \frac{a_{7m}}{4}T_m^4 + b_{2m}. \quad (25.74)$$

CONVERGE also accepts coefficients for the general form of the equations for the NASA 9 format if you do not have composite species or if you do not intend to use the CEQ solver. In the general form, Equations 25.72 through 25.74 become Equations 25.75 through 25.79:

Chapter 25: Input and Data Files

Materials Thermodynamic Properties: therm.dat

$$\frac{C_{p,m}^o}{R} = \sum_{i=1}^{n=8} a_{i,m} T^{c_i} \quad (25.75)$$

and

$$\frac{H_m^o}{RT_m} = \sum_{i=1}^{n=8} \frac{1}{T} \int_0^T a_{i,m} T^{c_i} dt + \frac{b_{1m}}{T_m}, \quad (25.76)$$

where

$$\begin{aligned} & \text{if } c_i = 1, \quad \frac{1}{T} \int_0^T a_{i,m} T^{c_i} dt = \frac{a_{i,m}}{T} \ln(T) \\ & \text{else,} \quad \frac{1}{T} \int_0^T a_{i,m} T^{c_i} dt = \frac{a_{i,m}}{c_i + 1} T^{c_i}; \end{aligned} \quad (25.77)$$

and

$$\frac{S_m^o}{R} = \sum_{i=1}^{n=8} \int_0^T a_{i,m} T^{c_i-1} dt + b_{2,m}, \quad (25.78)$$

where

$$\begin{aligned} & \text{if } c_i = 0, \quad \int_0^T a_{i,m} T^{c_i-1} dt = a_{i,m} \ln(T) \\ & \text{else,} \quad \int_0^T a_{i,m} T^{c_i-1} dt = \frac{a_{i,m}}{c_i} T^{c_i}. \end{aligned} \quad (25.79)$$

Calculation of Thermodynamic Properties

For both the NASA 7 and NASA 9 formats, CONVERGE calculates the Gibbs free energy, heat capacity at constant volume, and internal energy from

$$G_m^o = H_m^o - T_m S_m^o, \quad (25.80)$$

$$C_{vm}^o = C_{pm}^o - R, \quad (25.81)$$

and

Chapter 25: Input and Data Files

Materials Thermodynamic Properties: therm.dat

$$U_m^o = H_m^o - RT_m, \quad (25.82)$$

respectively, where G_m^o is the specific (per unit mass) Gibbs free energy, U_m^o is the specific internal energy, and R is the ideal gas constant.

Cleaning Utility

Use the [cleantherm](#) utility to condense your thermodynamic data file so that it lists only the species contained in the reaction mechanism file.

Thermodynamic Properties: tabular_therm.dat

Each of the gas-phase species in a CONVERGE simulation must have species-specific thermodynamic data in a non-tabular file (e.g., [therm.dat](#)) or a tabular file named *tabular_therm.dat*. If both a non-tabular file and a *tabular_therm.dat* file are saved to the same input directory and if both files contain thermodynamic data for the same species, CONVERGE will use the data from *tabular_therm.dat*.

The *tabular_therm.dat* file contains thermodynamic properties in 10 K-intervals. These data do not have to start or end at a certain temperature. CONVERGE will extrapolate if a simulation temperature is outside of the range covered by the data in this file.

```
THERMO_TABULAR @ T_ref = 298.15 K

#           name   phase   t_low      t_high   t_common
<species>  C7H16    G     300.000000  5000.000000  1391.000000
# elements
C 7   H 16
# temperature   enthalpy       csubp        entropy
0.000000    -3.5291068e+06  1.0354070e+02  2.7878508e+03
10.000000   -3.5291068e+06  1.0354070e+02  2.7878508e+03
20.000000   -3.5291068e+06  1.0354070e+02  2.7878508e+03
30.000000   -3.5291068e+06  1.0354070e+02  2.7878508e+03
.
.
4890.000000  1.7783302e+07  5.2057476e+03  1.5029023e+04
4900.000000  1.7835364e+07  5.2064243e+03  1.5039659e+04
4910.000000  1.7887430e+07  5.2070986e+03  1.5050274e+04

#           name   phase   t_low      t_high   t_common
<species>  O2      G     300.000000  5000.000000  1000.000000
# elements
O 2
# temperature   enthalpy       csubp        entropy
0.000000    -2.6120177e+05  8.3484247e+02  1.5680535e+03
10.000000   -2.5283873e+05  8.3775745e+02  3.4932715e+03
20.000000   -2.4444670e+05  8.4064459e+02  4.0748481e+03
.
.
END
```

Figure 25.55: An example *tabular_therm.dat* file.

As shown in the figure above, the first line of *tabular_therm.dat* contains the keyword **THERMO_TABULAR** and the reference temperature *T_ref*, which is the temperature (in K) at which the sensible enthalpy equals 0. The next line that is read by CONVERGE must list the

Chapter 25: Input and Data Files

Materials Thermodynamic Properties: tabular_therm.dat

keyword `<species>`, the chemical formula of the species, and the phase (G for gas). The G can be capitalized or lower case. The three temperatures (`t_low`, `t_high`, and `t_common`), which are described in the [therm.dat](#) section, are optional. If not included, CONVERGE will use the default values of $t_{low} = 300\text{ K}$, $t_{high} = 5000\text{ K}$, and $t_{common} = 1000\text{ K}$. The next line read by CONVERGE breaks down the species into its elemental components (e.g., for C7H16, there are 7 carbon atoms for every 16 hydrogen atoms).

The next lines include the species-specific thermodynamic data in ten-degree increments. Each line lists a temperature and the corresponding enthalpy (in J/kg), specific heat at constant pressure ($\text{J/(kg}\cdot\text{K)}$), and entropy ($\text{J/(kg}\cdot\text{K)}$) values. This list does not have to start at 0 K (e.g., the list could start at 20 K instead).

Thermodynamic data for subsequent species are formatted in a similar manner starting with the keyword `<species>`.

Lower Heating Value: lhv.in

If `property_control > lhv_flag = 0` in [inputs.in](#), CONVERGE calculates the lower heating value (LHV) from species data in the [thermodynamic data file](#). If `lhv_flag = 1`, CONVERGE will read user-specified LHVs for individual species from `lhv.in` and calculate a new NASA polynomial a_6 value. The original a_6 coefficient in the [thermodynamic data file](#) should not be modified for LHV because it is also used for other calculations. Figure 25.1 shows an example `lhv.in` file.

```
version: 3.1
---

- lhv_value: 4.6051879e+07
  lhv_fuel_species:
    C7H16: 0.5
    CH4: 0.5

- lhv_value: 4.4051879e+07
  lhv_fuel_species:
    C7H16: 0.35
    CH4: 0.65
```

Figure 25.56: An example `lhv.in` file.

Table 25.22: Format of `lhv.in`.

Parameter	Description
- <code>lhv_value</code>	The target lower heating value (J/kg).
	This settings block specifies LHV parameters. Repeat this block for each LHV.
<code>lhv_fuel_species</code>	
<code><species name></code>	Species mass or mole fraction.
	Repeat this line for each species to which the LHV applies. Each species must be on a separate line, and each species listed here must be included in the reaction mechanism file.

Chapter 25: Input and Data Files

Materials Lower Heating Value: lhv.in

CONVERGE will write the values used for LHV corrector calculations to [*lhv_info.out*](#).

Battery Properties: *battery_properties.dat*

For a simulation with a battery source (*i.e.*, [*source.in*](#) > *source* > *equation = BATTERY*), CONVERGE requires a battery properties data file (*e.g.*, *battery_properties.dat*) for each battery source. The battery properties file must contain the temperature, state of charge, and series resistance while discharging and charging the battery, in that order. For each resistor-capacitor (R-C) pair specified in [*source.in*](#) > *battery_data* > *n_rc_pairs*, you must include columns for the resistance while discharging and charging the battery, and capacitance while discharging and charging the battery.

#num_temp	num_SOC	num_column	R _s	R1	C1	charge	discharge
5	4	8	#Tm	SOC	Rs	charge	discharge
278	0.2	0.0189	0.0164	0.0410	0.0243	1082.0512	1663.1892
288	0.2	0.0133	0.0124	0.0267	0.0175	1492.3076	1965.7228
298	0.2	0.0105	0.0102	0.0185	0.0127	1948.7179	2263.0402
308	0.2	0.0090	0.0088	0.0138	0.0094	2379.4871	2591.6542
318	0.2	0.0080	0.0080	0.0109	0.0071	2784.6153	2857.6751
278	0.4	0.0190	0.0164	0.0349	0.0230	1487.1794	1950.0745
288	0.4	0.0133	0.0124	0.0224	0.0164	1841.0256	2221.3114
298	0.4	0.0105	0.0102	0.0157	0.0120	2215.3846	2466.4679
308	0.4	0.0090	0.0088	0.0116	0.0087	2579.4871	2722.0566
318	0.4	0.0080	0.0080	0.0093	0.0066	2912.8205	2967.2131
278	0.6	0.0189	0.0164	0.0307	0.0280	1789.7435	1772.7272
288	0.6	0.0133	0.0124	0.0198	0.0201	2138.4615	2101.3412
298	0.6	0.0105	0.0102	0.0138	0.0147	2482.0512	2419.5230
308	0.6	0.0090	0.0088	0.0101	0.0108	2810.2564	2727.2727
318	0.6	0.0080	0.0080	0.0079	0.0081	3117.9487	3008.9418
278	0.8	0.0189	0.0164	0.0289	0.0396	2010.2564	1183.3084
288	0.8	0.0133	0.0124	0.0186	0.0283	2348.7179	1652.7570
298	0.8	0.0105	0.0102	0.0129	0.0206	2692.3076	2111.7734
308	0.8	0.0090	0.0088	0.0095	0.0153	3046.1538	2560.3576
318	0.8	0.0080	0.0080	0.0075	0.0117	3364.1025	2982.8614

Figure 25.57: An example *battery_properties.dat* file.

The heading lists the number of temperatures, the number of SOCs, and the total number of columns in the file.

Table 25.23: Format of *battery_properties.dat*.

Column	Description
1	Temperature (<i>Kelvin</i>).
2	State of charge (SOC).
3	Ohmic resistance of the series during discharge (<i>ohm</i>).
4	Ohmic resistance of the series during charge (<i>ohm</i>).
5	Ohmic resistance of the resistor 1 during discharge (<i>ohm</i>).
6	Ohmic resistance of the resistor 1 during charge (<i>ohm</i>).
7	Capacitance of capacitor 1 during discharge (<i>Faraday</i>).

Chapter 25: Input and Data Files

Materials Battery Properties: battery_properties.dat

Column	Description
8	Capacitance of capacitor 1 during charge (<i>Faraday</i>).
9+	Repeat rows 5-8 for each pair of resistor and capacitor as specified by source.in > source > battery_data > n_rc_pairs . If source.in > source > battery_data > n_rc_pairs = 0 , do not include columns 3+.

Open Circuit Voltage: **vocv.dat**

For a simulation with a battery source (*i.e.*, [source.in > source > equation = BATTERY](#)), CONVERGE requires an open circuit voltage data file (*e.g.*, *vocv.dat*) if *source.in > battery_data > soc_vocv* contains a file name. The name of the open circuit voltage data file must match the name given in [source.in > battery_data > soc_vocv](#) and the file must be located in your case setup.

```
# SOC          Vocv-discharge    Vocv-charge
0.000411495444824 2.54316051018098 2.62871344499747
0.000790674493458 2.62700883293388 2.54574804679511
0.004018005259673 2.70267719658657 2.71437081882851
0.008345817037491 2.78449175549182 2.79324484429972
0.01123102488937 2.80903537476823 2.81582203825721
0.015009273266831 2.85403263377096 2.87149026799937
0.018444044519068 2.87857812403527 2.8924179639875
0.025055979179623 2.931759084114 2.95714481756809
0.029263573963613 2.95016773406526 2.97170832377169
0.037678763531593 3.00949676038445 3.03141662417848
0.043088528253866 3.03200100284951 3.0489895355353
0.056312397574978 3.0913337711445 3.10826622427231
0.077350371494928 3.14658778581679 3.17260323493423
...
...
```

Figure 25.58: An example *vocv.dat* file.

Table 25.24: Format of *vocv.dat*.

Column	Description
1	State of charge (SOC).
2	Open circuit voltage (V_{ocv}) for discharging battery (<i>volts</i>).
3	Open circuit voltage (V_{ocv}) for charging battery (<i>volts</i>).

25.4 Simulation Parameters

This section describes the [surface geometry file](#), the main input file ([inputs.in](#)), the [solver.in](#) file, and other general input files.

Surface Geometry: **surface.dat**

The [surface geometry information](#) must be contained in a *.dat file. The name of the surface geometry file (*e.g.*, *surface.dat*) in the Case Directory must match [inputs.in > surface_filename](#).

CONVERGE Studio generates a properly formatted surface geometry file when you export the simulation-ready geometry.

Chapter 25: Input and Data Files

Simulation Parameters Surface Geometry: surface.dat

Table 25.25: Surface geometry file format.

Row number	Row format	Description
1	<i>numverts_tot</i> <i>numverts</i> <i>numtriangles</i>	<i>numverts_tot</i> : Maximum vertex identification number. <i>numverts</i> : Number of vertices defined in this file. <i>numtriangles</i> : Number of surface triangles.
2 to (1+ <i>numverts</i>)	<i>vert_id vx vy vz</i>	Unique vertex identification number followed by the x, y, and z coordinates of the vertex.
(2+ <i>numverts</i>) to (1+ <i>numverts</i> + <i>numtriangles</i>)	<i>vert1 vert2 vert3 bound_id</i>	Surface triangle vertex identification numbers followed by the boundary ID of that surface triangle.

In Row 1, *numverts_tot* defines the maximum vertex ID (*vert_id*) in the surface file. CONVERGE allocates an array the size of the *numverts_tot* value to store the vertex information. As CONVERGE reads each of the following rows, it uses the vertex ID as an array index for storing the vertex information. The *numverts* is simply the number of vertices defined in the file, *i.e.*, the number of rows that are used to define vertices in the surface file. For most cases, *numverts_tot* and *numverts* are identical. Typically the vertices are sequentially listed from 1 through *numverts*, though this ordering is not necessary (*i.e.*, the vertex IDs do not need to be sequential).

Immediately following the rows of vertex identification information are *numtriangles* rows of surface triangle definitions. Each surface triangle must have three vertex IDs to define its three corners and a *bound_id* that corresponds to a boundary identification number in the *boundary.in* file. The order in which the three defining vertices are given is critical: the order of the vertices defines which side of the surface triangle points to the inside of the geometry by means of the right-hand rule.

Sealed Surface Geometry File

If you invoke the [sealing](#) process, the Seal tool in CONVERGE Studio will create a *surface_new.dat* file. This file contains information about the seals in addition to the regular surface geometry file information.

Table 25.26: Summary of the sealed surface geometry file format.

Row number	Row format	Description
1	<i>numverts_tot</i> <i>numverts</i> <i>numtriangles</i> <i>numseals</i>	<i>numverts_tot</i> : Maximum vertex id. <i>numverts</i> : Number of vertices defined below. <i>numtriangles</i> : Number of surface triangles. <i>numseals</i> : Number of sealed edges.

Chapter 25: Input and Data Files

Simulation Parameters Surface Geometry: *surface.dat*

Row number	Row format	Description
2 to (1+numverts)	<i>vert_id vx vy</i> <i>vz</i>	Unique vertex identification number followed by the x, y, and z coordinates of the vertex.
(2+numverts) to (1+numverts+numt riangles)	<i>vert1 vert2</i> <i>vert3</i> <i>bound_id</i>	Surface triangle vertex identification numbers followed by the boundary ID of that surface triangle.
(2+numverts+numt riangles) to (1+numverts+numt riangles+numseals)	<i>vert1 vert2</i> <i>direction</i> <i>bound_id</i> <i>moving</i>	<i>vert1</i> : Vertex identifier for the seal edge. <i>vert2</i> : Vertex identifier for the seal edge. <i>direction</i> : Flag for direction of sealing . The direction is defined with respect to the two triangles that share <i>vert1</i> and <i>vert2</i> . The order of <i>vert1</i> and <i>vert2</i> while traveling clockwise around the triangle's normal vector determines which triangle is <i>right</i> and which triangle is <i>left</i> . In the right triangle, <i>vert1</i> comes before <i>vert2</i> . In the left triangle, <i>vert2</i> comes before <i>vert1</i> . 0 = Average of the normal vectors of the right and left triangles, 1 = Parallel to the normal vector of the right triangle (left triangle is removed), -1 = Parallel to the normal vector of the left triangle (right triangle is removed), 2 = Orthogonal to the normal vector of the right triangle and the seal edge (right triangle is removed), -2 = Orthogonal to the normal vector of the left triangle and the seal edge (left triangle is removed). <i>bound_id</i> : Boundary ID of the seal-to boundary. <i>moving</i> : Motion flag for seal triangles. 0 = The seal edges do not move; the seal-to boundary is deformed/stretched to create the seal, 1 = The seal edges move as needed toward the seal-to boundary to create the seal.

One difference between the new sealed surface geometry file (*surface_new.dat*) and the non-sealed geometry file (see Table 25.25) is that the sealed geometry file has the additional column *numseals* in the first row. This file will also have extra rows (for the seals) appended to the end of the surface geometry file.

Surface List: *surface_list.in*

When [*inputs.in*](#) > *surface_filename* = *surface_list.in*, CONVERGE uses a list of [surface geometry files](#) to calculate the motion of any boundaries with [*boundary.in*](#) > *boundary_conditions* > *boundary* > *motion* = SURFACE_LIST.

All of the files specified in *surface_list.in* must be topologically identical, with the same number of vertices and the same set of vertex IDs. Each file contains the locations of the vertices at a specified time. To calculate the vertex locations at other times, CONVERGE interpolates between the locations at the two nearest times in *surface_list.in*.

Chapter 25: Input and Data Files

Simulation Parameters Surface List: surface_list.in

```
version: 3.1
---

time_unit:           SECOND
temporal_type:       CYCLIC
period:              2.0
adjust_inflow_outflow: True
base_surface_file:   surfacet1.dat
surface_files:
  - file:
    name:      surfacet1.dat
    time:     0.0
  - file:
    name:      surfacet2.dat
    time:     0.5
  - file:
    name:      surfacet3.dat
    time:     1.0
  - file:
    name:      surfacet2.dat
    time:     1.5
  - file:
    name:      surfacet1.dat
    time:     2.0
```

Figure 25.59: An example *surface_list.in* file.

Table 25.27: Format of *surface_list.in*.

Parameter	Description
<i>time_unit</i>	SECOND = Times are specified in <i>seconds</i> , CRANK = Times are specified in <i>crank angle degrees</i> .
<i>temporal_type</i>	CYCLIC = Surface motion is cyclic, SEQUENTIAL = Surface motion is sequential.
<i>period</i>	Period of cyclic motion. Used only when <i>temporal_type</i> = CYCLIC.
<i>adjust_inflow_outflow</i>	True = Project INFLOW and OUTFLOW boundaries to a plane at the point of furthest displacement to prevent motion in the normal direction, False = Do not project INFLOW and OUTFLOW boundaries to a plane.
<i>base_surface_file</i>	Surface geometry file used for boundaries for which <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> ≠ SURFACE_LIST.
<i>surface_files</i>	List of surface geometry files that represent the geometry at the specified times. Files and times must be listed in order. When <i>temporal_type</i> = CYCLIC, the first file corresponds to the beginning of the period. Specifying a file for the end of the period is optional. If specified, the file name must match the first file name.
-file	
<i>name</i>	Surface geometry file name (e.g., <i>surfacet1.dat</i>).
<i>time</i>	Time corresponding to this surface geometry file (in <i>seconds</i> or <i>crank angle degrees</i>).

Stream Setup

This section describes input files used to set up streams for your CONVERGE simulation.

Stream Settings: `stream_settings.in`

The `stream_settings.in` file, which is required in the Case Directory for all simulations, specifies the number of streams and their properties.

```
version: 3.1
---

stream_match_species_settings: 0
post_file_option: COMBINED
stream_list:
  - stream:
      id: 0
      name: stream0
      solid_flag: 0
      region_list:
        - region:
            id: 0
            name: Fluid_Region
  - stream:
      id: 1
      name: stream1
      solid_flag: 1
      stream_overwrite: stream1
      region_list:
        - region:
            id: 1
            name: HS_Region
  - stream:
      id: 2
      name: stream2
      solid_flag: 1
      region_list:
        - region:
            id: 2
            name: CS_Region_A
        - region:
            id: 3
            name: CS_Region_B
```

Figure 25.60: An example `stream_settings.in` file.

Chapter 25: Input and Data Files

Simulation Parameters Stream Setup

Table 25.28: Format of *stream_settings.in*.

Parameter	Description
<i>stream_match_species_settings</i>	Used only when there are different species lists for different streams. 0 = CONVERGE matches species by name and renormalizes mass fractions so they add up to 1, 1 = CONVERGE uses the species matching behavior from <i>stream_match_species.in</i> .
<i>post_file_option</i>	<i>COMBINED</i> = CONVERGE writes post files combining data from all streams to the <i>outputs_*/output</i> directory, <i>STREAM_BASED</i> = CONVERGE writes a separate set of post files for each stream to the <i>outputs_*/stream<ID>/output</i> directories, <i>SEPARATE_STEADY_TRANSIENT</i> = CONVERGE writes separate sets of post files for the steady-state and transient streams to the <i>outputs_*/output_steady</i> and <i>outputs_*/output_transient</i> subdirectories, respectively.
	When multiple streams are combined into a post file, the data are structured according to the stream ID.
<i>stream_list</i>	Settings block for the list of streams.
- <i>stream</i>	Sub-block for a stream. Repeat for each stream in the simulation.
<i>id</i>	Stream ID. Each stream must have a unique ID of between 0 and 31, inclusive.
<i>name</i>	Stream name.
<i>solid_flag</i>	0 = The stream is in a fluid phase, 1 = The stream is in the solid phase.
<i>stream_overwrite*</i>	Name of the subdirectory containing override settings for the stream.
<i>region_list</i>	Sub-block for the list of regions in the stream.
- <i>region</i>	Sub-block for a region. Repeat for each region in the stream.
<i>id</i>	Region ID.
<i>name</i>	Region name.

*Sub-blocks or parameters indicated with an asterisk are optional.

Species Matching: *stream_match_species.in*

The *stream_match_species.in* file is required in the Case Directory when [*stream_settings.in*](#) > *stream_match_species_settings* = 1. You can use this file to configure the [species matching](#) behavior for a multi-stream simulation.

Chapter 25: Input and Data Files

Simulation Parameters Stream Setup

```
version: 3.1
---

streams_match_species:
  - stream_match_species:
    connected_streams: [0,1]
    non_conserved_fix: 0
    buffer_gas_species: []
    gas_species_match_list:
      - [IC8H18, C8H18]
      - [NC7H16, C7H16]
      - [H2O, H2O]
      - [CO2, CO2]
    buffer_liquid_species: []
    liquid_species_match_list:
      - []
    passive_match_list:
      - [passive1, passive2]
    passive_value:
      []

  - stream_match_species:
    connected_streams: [0,2]
    non_conserved_fix: 2
    buffer_gas_species: [N2, N2]
    gas_species_match_list:
      - []
    buffer_liquid_species: [N2, N2]
    liquid_species_match_list:
      - [H2O, H2O]
    passive_match_list:
      - []
    passive_value:
      passive3: -1.0
      passive4: NEUMANN
```

Figure 25.61: An example *stream_match_species.in* file.

Chapter 25: Input and Data Files

Simulation Parameters Stream Setup

Table 25.29: Format of *stream_match_species.in*.

Parameter	Description
<i>streams_match_species</i>	Settings block for the list of stream pairs.
- <i>stream_match_species</i>	Sub-block for a pair of connected streams. Repeat for each pair of connected streams.
<i>connected_streams</i>	Streams IDs of connected streams in the format [#,#].
<i>non_conserved_fix</i>	Method for addressing mass conservation errors. 0 = Do not fix, 1 = Renormalize species mass fractions so they add up to 1 (default), 2 = Specify a buffer species to absorb errors.
<i>buffer_gas_species</i> *	Specify the buffer gas species for each stream (e.g., [N2, N2]) in the same order that the streams are listed in the <i>connected_streams</i> parameter. Used only when <i>non_conserved_fix</i> = 2.
<i>gas_species_match_list</i>	Specify pairs of gas species to match across streams (e.g., [IC8H18, C8H18]) in the same order that the streams are listed in the <i>connected_streams</i> parameter. Any gas species not listed here will be matched by name.
<i>buffer_liquid_species</i> *	Specify the buffer liquid species for each stream (e.g., [N2, N2]) in the same order that the streams are listed in the <i>connected_streams</i> parameter. Used only when <i>non_conserved_fix</i> = 2.
<i>liquid_species_match_list</i>	Specify pairs of liquid species to match across streams (e.g., [IC8H18, C8H18]) in the same order that the streams are listed in the <i>connected_streams</i> parameter. Any liquid species not listed here will be matched by name.
<i>passive_match_list</i>	Specify pairs of passives to match across streams (e.g., [passive1, passive2]) in the same order that the streams are listed in the <i>connected_streams</i> parameter. Any passives not listed here will be matched by name.
<i>passive_value</i> *	Sub-block with passive boundary conditions at the stream-stream interface.
< <i>passive name</i> >	Specified value (for Dirichlet boundary condition) or NEUMANN (for Neumann boundary condition).

*Sub-blocks or parameters indicated with an asterisk are optional.

Simulation Control

This section describes input files that contain high-level control parameters for CONVERGE.

Input Parameters: *inputs.in*

The *inputs.in* file, which is required for all simulations, includes parameters that activate models and features, control the grid, and set up output-related options. Some parameters can be either a fixed value or a [temporally varying quantity](#).

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

```
version: 3.1
---

surface_filename: surface.dat
mechanism_filename: mech.dat
thermodynamic_filename: therm.dat

simulation_control:
    crank_flag: 0
    start_time: -500.0
    end_time: 99
    restart_flag: 0
    restart_number: 0
    map_flag: OFF
    check_grid_motion_flag: 0
    load_cyc: AUTO
    imbalance_factor: 125.0
    reread_input: 0
    random_seed: 0

output_control:
    log_level: 0
    twrite_post: 99999
    twrite_transfer: 10.0
    twrite_files: 0.02001
    twrite_restart: 999999
    twrite_restart_max_wct: 999999
    num_restart_files: 1
    write_map_flag: 0
    wall_output_flag: 0
    transfer_flag: 0
    mixing_output_flag: 1
    species_output_flag: 1
    region_flow_flag: 0
    dynamic_flag: 0
    points_flag: 0
    numerics_output_flag: 1
    custom_time_flag: 1
    custom_time_unit: MS
    custom_time_conv: 0.001
    post_slice_output_flag: 0
    uniformity_index_output_flag: 1
    smd_pdf_flag: 1
    walltime_output_flag: 0
    paraview_catalyst_flag: 0
    output_multiplier: AUTO

grid_control:
    base_grid: [0.004, 0.004, 0.004]
    grid_scale: 9
    domain:
        xmin: -0.1
        xmax: 0.1
        ymin: -0.1
        ymax: 0.1
        zmin: -0.1
        zmax: 0.1
    amr_flag: 1
    events_flag: 1
    embedded_flag: 1
    inlaid_mesh_flag: 1
    min_vol_ratio: 0.3
    seal_tol: 0.0005
    auto_reloft_flag: 0

temporal_control:
    time_flag: 1
    dt_start: 5e-07
    dt_min: 1e-08
    dt_max: 5.0e-05
    dt_fixed: 1e-06
```

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

```
mult_dt.Parcel:          1.5
mult_dt.evap:           9999.0
mult_dt.chem:            0.5
mult_dt.coll_mesh:      1e28
mult_dt.move:            2
max_cfl_u:              1.0
max_cfl_nu:             0.20
max_cfl_mach:           50.0

solver_control:
  momentum_solver:       1
  energy_solver:          INTERNAL
  species_solver:         1
  turbulence_solver:     1
  steady_solver:          0
  monitor_steady_state:  0

property_control:
  gas_compressible_flag:  1
  liquid_compressible_flag: 0
  lhv_flag:                0
  individual_properties_flag: 0
  tabular_fluid_prop_flag:  0
  eos_flag:                 IDEAL_GAS
  real_gas_prop_flag:      0
  max_reduced_pres:        6.0
  crit_temp:               133.0
  crit_pres:                3770000.0
  acentric_factor:          0.035
  species_diffusion_model: 0
  prandtl_turb:             0.9
  schmidt_turb:              0.78
  min_temp:                  10.0
  max_temp:                  60000.0
  max_visc:                  10.0
  gravity:                   [gx.dat, gy.dat, gz.dat]

feature_control:
  parcel_control:
    liquid_parcel:        1
    solid_parcel:          0
    gas_parcel:            0
  urea_flag:                1
  combustion_flag:          0
  source_flag:              0
  aniso_cond_flag:          0
  composite_flag:            0
  wallvalue_flag:            0
  udf_flag:                  0
  cht_supercycle_flag:      0
  vof_flag:                  0
  fixed_flow_flag:          0
  fsi_flag:                  0
  radiation_flag:            0
  nucleate_boiling_flag:   0
  skip_species_flag:        0
  mrf_flag:                  0
  mixing_plane_flag:        0
  prox_check_flag:          0
  cosimulation_flag:        1
  motion_sets_flag:          0
  initial_perturbation_flag: 0
  post_emission_flag:       0
  acoustics_flag:            0
  statistics_flag:           0
  erosion_flag:              0
  electric_potential_flag:  0

ga_control:
  ga_flag:                  0
  ga_individual:             0
```

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

```
ga_generation:          0
mpi_control:
    shared_memory_flag:   1
    io_hints_filename:    mpi_io_hints.in
accelerator_control:
    use_gpu_flag:         1
    amgx_config_filename: amgx.json
    enable_pin_mem:       0
```

Figure 25.62: An example *inputs.in* file.

Table 25.30: Format of *inputs.in*.

Parameter	Description	Typical value
<i>surface_filename</i>	The name of the surface geometry data file, e.g., <i>surface.dat</i> . To use a list of surface geometry files, set <i>surface_filename</i> = <i>surface_list.in</i> and include a <i>surface_list.in</i> file in your case setup.	
<i>mechanism_filename</i>	The name of the mechanism data file, e.g., <i>mech.dat</i> .	
<i>thermodynamic_filename</i>	The name of the thermodynamic data file, e.g., <i>therm.dat</i> .	
<i>simulation_control</i>		
<i>crank_flag</i>	0 = Time-based simulation (<i>seconds</i>), 1 = Crank angle-based simulation for engine applications (requires engine.in). 2 = Crank angle-based simulation for non-engine applications (requires rpm.in).	
<i>start_time</i>	Simulation start time (in <i>seconds</i> if <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).	
<i>end_time\$</i>	Simulation end time (in <i>seconds</i> if <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero). If running a CONVERGE + GT-SUITE simulation , you can set <i>end_time</i> = <i>GT</i> to direct the CONVERGE simulation to end in conjunction with the GT-SUITE simulation.	
<i>restart_flag</i>	0 = Do not start from a restart file, 1 = Start from a restart file (<i>restart.rst</i> , or the highest-numbered <i>restart****.rst</i> file if <i>restart.rst</i> does not exist).	
<i>restart_number</i>	Number that is appended to the names of the <i>.out</i> files to prevent overwriting the existing <i>*.out</i> files (e.g., if <i>restart_number</i> = 2, CONVERGE will write <i>thermo2.out</i> instead of <i>thermo.out</i>).	
<i>map_flag</i>	OFF = No mapping,	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>MAP</i>	<i>MAP</i> = Mapping (requires <i>map.in</i> and a map or restart data file). Note that <i>restart_flag</i> = 1 will override <i>map_flag</i> = 1.	
<i>check_grid_motion_flag</i>	0 = Solve the parcel, combustion, and transport equations, 1 = Do not solve the parcel, combustion, and transport equations. Set to check surface motion and grid creation, 2 = Same as 1, but also check for triangle-to-triangle intersections at each time-step, 3 = Check for triangle-to-triangle intersections at each time-step, but do not check grid creation.	0
<i>load_cyc\$</i>	The maximum number of cycles between parallel load balance checks. Enter a positive integer to force regular load balances. If <i>AUTO</i> , CONVERGE will perform a load balance only when it detects that the imbalance is greater than <i>imbalance_factor</i> .	<i>AUTO</i>
<i>imbalance_factor\$</i>	Percentage imbalance (percentage of maximum per-rank cell count to average per-rank cell count) that will trigger a parallel load balance at each load balance check. For cases with very few cells per core (<i>e.g.</i> , fewer than 1000), consider increasing this value to 150-200. CONVERGE calculates a similar imbalance factor based on the number of parcels on each rank. The functional form is complex, but behavior follows similar trends to the cell count imbalance factor (<i>i.e.</i> , a larger <i>imbalance_factor</i> setting will generally require a larger imbalance in parcels per rank before a load balance is triggered).	125
<i>reread_input</i>	0 = Do not reread parameters at each time-step. Parameter reread will be supported in a future minor release.	0
<i>random_seed</i>	Seed for random number generator. Must be a non-negative integer.	
<i>output_control</i>		
<i>log_level\$</i>	The detail level of the screen output . Must be an integer between 0 and 3. Enter a higher value for more detail.	0
<i>twrite_post\$</i>	Time interval (in seconds if <i>inputs.in > simulation_control > crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) for writing post	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>twrite_post§</i>	(<i>post*.h5</i>) files. Specify a file name (e.g., <i>twrite_post.in</i>) to temporally vary this parameter and/or to direct CONVERGE to write post files at specified times .	
<i>twrite_transfer§</i>	Time interval (in seconds if <i>inputs.in > simulation_control > crank_flag = 0</i> or in crank angle degrees if <i>crank_flag</i> is non-zero) for writing heat transfer output. Specify a file name (e.g., <i>twrite_transfer.in</i>) to temporally vary this parameter.	
<i>twrite_files§</i>	Time interval (in seconds if <i>inputs.in > simulation_control > crank_flag = 0</i> or in crank angle degrees if <i>crank_flag</i> is non-zero) for writing *.out files. Specify a file name (e.g., <i>twrite_files.in</i>) to temporally vary this parameter.	
<i>twrite_restart§</i>	Time interval (in seconds if <i>inputs.in > simulation_control > crank_flag = 0</i> or in crank angle degrees if <i>crank_flag</i> is non-zero) for writing restart (<i>restart<restart number>.rst</i>) files. Specify a file name (e.g., <i>twrite_restart.in</i>) to temporally vary this parameter and/or to direct CONVERGE to write restart files at specified times .	
<i>twrite_restart_max_wct§</i>	Maximum wall-clock time (in hours) between <i>restart<restart number>.rst</i> file writes.	
<i>num_restart_files§</i>	Maximum number of restart files (<i>restart<restart number>.rst</i>) to be stored. When this number of restart files is saved to the output directory, CONVERGE will remove the lowest-numbered restart file before writing a new one. This number does not include restart files that are written at specified simulation times .	
<i>write_map_flag</i>	0 = Do not write <i>map_<time>.h5</i> or <i>map_parcel_<time>.h5</i> during the simulation, 1 = Write <i>map_<time>.h5</i> (and <i>map_parcel_<time>.h5</i> , if <i>inputs.in > feature_control > parcel_mode > liquid_parcel, solid_parcel, or gas_parcel = 1</i>) at specified times during the simulation (requires <i>write_map.in</i>).	
<i>post_slice_output_flag</i>	Input is deprecated. Feature replaced by ParaView Catalyst.	0
<i>uniformity_index_output_flag</i>	0 = No uniformity index plane output, 1 = Write uniformity index through a plane to <i>planes_flow.out</i> (requires <i>uniformity_index.in</i>).	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>wall_output_flag</i>	0 = Do not write wall output, 1 = Write wall output (stress, force, and pressure on wall boundaries) to <i>wall_stress<number> <time>.out</i> and average near-wall quantities and heat transfer data to <i>bound<boundary ID>-wall.out</i> . Alternatively, specify a file name (e.g., <i>wall_output.in</i>) to direct CONVERGE to write wall output for specific boundaries.	
<i>transfer_flag</i>	0 = Do not write finite element analysis (FEA) heat transfer output, 1 = Write FEA heat transfer output to <i>transfer.out</i> . Specify a file name (e.g., <i>transfer.in</i>) to write FEA heat transfer output for specific boundaries.	
<i>mixing_output_flag</i>	0 = Do not write mixing output, 1 = Write mixing output to <i>mixing.out</i> .	
<i>species_output_flag</i>	0 = Do not write species mass information; 1 = Write total mass for all species to <i>species_mass.out</i> , 2 = Write total mass and mass fractions for all species to <i>species_mass.out</i> and <i>species_mass_frac.out</i> , respectively; 3 = Write total mass, mass fractions, and standard deviations of mass fractions for all species to <i>species_mass.out</i> , <i>species_mass_frac.out</i> , and <i>species_std_masfrac.out</i> , respectively; 4 = Write total mass, mass fractions, standard deviations of mass fractions, and mole fractions for all species to <i>species_mass.out</i> , <i>species_mass_frac.out</i> , <i>species_std_masfrac.out</i> , and <i>species_mole_frac.out</i> , respectively. 5 = Write total mass, mass fractions, standard deviations of mass fractions, mole fractions, and volume-integrated source terms for all species to <i>species_mass.out</i> , <i>species_mass_frac.out</i> , <i>species_std_masfrac.out</i> , <i>species_mole_frac.out</i> , and <i>species_mass_src.out</i> , respectively. To control the type of output for individual species, specify <i>species_output.in</i> instead of an integer for this parameter.	
<i>region_flow_flag</i>	0 = Do not write inter-region flow data, 1 = Write inter-region flow data without species- specific data to <i>area_avg_regions_flow.out</i> and <i>mass_avg_regions_flow.out</i> ,	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
	2 = Write inter-region flow data, including species-specific data, to <i>area_avg_regions_flow.out</i> and <i>mass_avg_regions_flow.out</i> (requires <i>regions_flow.in</i>).	
	If 1 or 2 and if there is only one region, CONVERGE writes flow data between adjacent boundaries.	
<i>dynamic_flag</i>	0 = CONVERGE assumes all cylinders are aligned with the z axis (<i>i.e.</i> , if cylinders are not aligned with the z axis, the swirl, tumble, and angular momentum flux data in <i>dynamic.out</i> will be incorrect), 1 = CONVERGE uses the information in <i>dynamic.in</i> to calculate swirl, tumble, and angular momentum flux for all cylinders. Requires <i>dynamic.in</i> .	
<i>monitor_points_flag</i>	0 = No monitor points via <i>monitor_points.in</i> , 1 = Monitor points enabled (requires <i>monitor_points.in</i>).	
<i>numerics_output_flag</i>	0 = No numerics output, 1 = Write <i>numerics.out</i> .	1
<i>custom_time_flag</i> *	0 = Do not specify a custom time unit, 1 = Specify a custom time unit.	
<i>custom_time_unit</i> *	Name of the custom time unit (<i>e.g.</i> , ms).	
<i>custom_time_conv</i> *	Conversion factor from the custom time unit to seconds (<i>e.g.</i> , 0.001).	
<i>smd_pdf_flag</i>	0 = No SMD and drop distribution output, 1 = Output SMD and drop distribution through a plane (requires <i>smd_pdf.in</i>), 2 = Output Phase Doppler Particle Analyzer (PDPA) data (requires <i>pdpa.in</i>), 3 = Output parcel radius distribution in a region or box (requires <i>radius_pdf.in</i>).	
<i>walltime_output_flag</i> *	0 = Do not generate wall time information file , 1 = Generate <i>walltime.out</i> .	
<i>paraview_catalyst_flag</i>	0 = Do not perform in situ post-processing with ParaView Catalyst, 1 = Perform in situ post-processing (requires <i>paraview_catalyst.in</i>).	
<i>output_multiplier</i>	Output multiplier. Extrinsic quantities (<i>e.g.</i> , total system mass) are scaled by this value before these quantities are written to *.out files. Quantities scaled by <i>output_multiplier</i> are marked in <i>Chapter 26 - Output Files</i> .	1 for a case with no PERIODIC boundaries, AUTO for a

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
	<p>You can use this parameter to scale quantities for both periodic and non-periodic cases. For example, for a sector case for which you want the output quantities to be scaled, set this parameter to $360/\text{angle of rotation}$. For a non-sector case, set this parameter to the desired scaling factor (for example, you might simulate a single cylinder of a multi-cylinder engine but want the results to reflect the multi-cylinder case).</p> <p>You can set this parameter to <i>AUTO</i> for many cases. If you set this parameter to <i>AUTO</i>:</p> <ul style="list-style-type: none">• For a case without PERIODIC boundaries, CONVERGE will set <i>output_multiplier</i> = 1.• For a case with PERIODIC boundaries, CONVERGE will read calculate the value of <i>output_multiplier</i> as 360 divided by the angle of rotation that is specified in boundary.in > boundaries > boundary > match > angle. <p>Do not use <i>AUTO</i> for cases with multiple rotational PERIODIC boundaries with different angles.</p> <p>If you convert the input files of a PERIODIC case from CONVERGE 2.4 or earlier to CONVERGE 3.1, CONVERGE will calculate the value of <i>output_multiplier</i> as 360 divided by the angle of rotation that is specified in the original boundary.in file.</p>	case with PERIODIC boundaries but only one angle of rotation.
<i>grid_control</i>		
<i>base_grid</i>	Base grid size along the x, y, and z axes (<i>m</i>).	
<i>grid_scale§</i>	Grid scaling factor applied to the x, y, and z <i>base_grid</i> sizes. For each axis, scaled grid = $\text{base_grid}/2^{\text{grid_scale}}$. A negative integer will coarsen the grid, while a positive integer will refine the grid. Specify a file name (e.g., gridscale.in) to set up temporally varying grid scaling values. This parameter must be an integer or a file name.	
<i>domain</i>	Optional domain bounds.	
<i>xmin*</i>	Minimum domain position in the x direction (<i>m</i>).	
<i>xmax*</i>	Maximum domain position in the x direction (<i>m</i>).	
<i>ymin*</i>	Minimum domain position in the y direction (<i>m</i>).	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>y_{max}</i> *	Maximum domain position in the y direction (<i>m</i>).	
<i>z_{min}</i> *	Minimum domain position in the z direction (<i>m</i>).	
<i>z_{max}</i> *	Maximum domain position in the z direction (<i>m</i>).	
<i>amr_flag</i>	0 = No Adaptive Mesh Refinement (AMR), 1 = Enable AMR (requires amr.in).	
<i>events_flag</i>	0 = No events , 1 = Enable events (requires events.in).	
<i>embedded_flag</i>	0 = No fixed embedding , 1 = Enable fixed embedding (requires embedded.in).	
<i>inlaid_mesh_flag</i>	0 = No inlaid mesh , 1 = Enable inlaid mesh (requires inlaid_mesh.in).	
<i>min_vol_ratio</i>	Minimum volume ratio to activate cell pairing. Applies only to the Cartesian grid.	0.3
<i>seal_tol</i>	Sealing tolerance (<i>m</i>).	0.0005
<i>auto_reloft_flag</i>	0 = Do not perform automatic relofting of WALL boundaries, 1 = Automatically reloft WALL boundaries.	
<i>temporal_control</i>		
<i>time_flag</i>	0 = Constant time-step , 1 = Variable time-step.	1
<i>dt_start</i>	Initial time-step (transient) or pseudo time-step (steady-state) size (always in <i>seconds</i> , even when <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> is non-zero). Used only when <i>time_flag</i> = 1.	
<i>dt_min\$</i>	Minimum time-step (transient) or pseudo time-step (steady-state) size (always in <i>seconds</i> , even when <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> is non-zero). Specify a file name (e.g., <i>dt_min.in</i>) for a temporally varying minimum time-step size. Used only when <i>time_flag</i> = 1.	
<i>dt_max\$</i>	Maximum time-step (transient) or pseudo time-step (steady-state) size (always in <i>seconds</i> , even when <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> is non-zero). Specify a file name (e.g., <i>dt_max.in</i>) for a temporally varying maximum time-step size. Used only when <i>time_flag</i> = 1.1.	
<i>dt_fixed\$</i>	Time-step (transient) or pseudo time-step (steady-state) size (always in <i>seconds</i> , even when <i>inputs.in</i> >	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
	<i>simulation_control > crank_flag</i> is non-zero). Used only when <i>time_flag</i> = 0.	
<i>mult_dt.Parcel\$</i>	Time-step limiter based on parcel penetration. This parameter limits the number of cells a parcel can travel in a single time-step. For example, <i>mult_dt.Parcel</i> = 1.5 means a parcel can travel through a maximum of 1.5 cells in a single time-step. Used only when <i>time_flag</i> = 1.	0.1 - 1.5
<i>mult_dt.evap\$</i>	Time-step limiter based on evaporation. Evaporation time-step control is a deprecated feature, and the recommended value essentially disables this parameter. Used only when <i>time_flag</i> = 1.	9999.0
<i>mult_dt.chem\$</i>	Time-step limiter based on cell temperature. This parameter limits the temperature rise of a cell after combustion has occurred in the present time-step. For example, <i>mult_dt.chem</i> = 0.5 will limit the temperature rise to 50% of the cell temperature in a single time-step. Used only when <i>time_flag</i> = 1.	2.0 for accuracy and stability. Higher values are supported, but they may cause CONVERGE to struggle.
<i>mult_dt.coll_mesh\$</i>	Collision mesh multiplier used in calculating the next time-step.	0.1 - 1.0
<i>mult_dt.move\$</i>	Time-step limiter based on surface motion. This multiplicative parameter prevents a moving surface from moving too far through a cell. For example, <i>mult_dt.move</i> = 1.0 will limit surface motion to $1.0 \times 30\% = 30\%$ of the limiting cell's length in a single time-step. Used only when <i>time_flag</i> = 1.	2.0
<i>max_cfl_u\$</i>	Maximum convective CFL number (cfl_u) allowed by CONVERGE. Specify a file name (e.g., <i>max_cfl_u.in</i>) to vary this parameter temporally and/or on a region-by-region basis . Used only when <i>time_flag</i> = 1.	0.5 - 3.0. (0.03 - 0.10 for a VOF case).
<i>max_cfl_nu\$</i>	Maximum diffusive CFL number (cfl_v) allowed by CONVERGE. Specify a file name (e.g., <i>max_cfl_nu.in</i>) to vary this parameter temporally and/or on a region-by-region basis . Used only when <i>time_flag</i> = 1.	0.5 - 2.5

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>max_cfl_mach\$</i>	Maximum Mach CFL number (cfl_{mach}) allowed by CONVERGE. Specify a file name (e.g., <i>max_cfl_mach.in</i>) to vary this parameter temporally and/or on a region-by-region basis . Used only when <i>time_flag</i> = 1.	0.5 - 100.0
<i>solver_control</i>		
<i>momentum_solver</i>	0 = Do not solve momentum equation, 1 = Solve momentum equation .	
<i>energy_solver</i>	<i>OFF</i> = Do not solve energy equation, <i>INTERNAL</i> = Solve energy equation , <i>TOTAL</i> = Solve total energy equation .	
<i>species_solver</i>	0 = Do not solve species equation, 1 = Solve species equation for species mass fractions, 2 = Solve species equation for species densities (available only when <i>feature_control > vof_flag</i> = 1).	
<i>turbulence_solver</i>	0 = No turbulence modeling, 1 = Include turbulence modeling (requires <i>turbulence.in</i>).	
<i>steady_solver</i>	0 = Transient solver , 1 = Steady-state solver .	
<i>monitor_steady_state</i>	0 = Do not activate the steady-state monitor, 1 = Activate the steady-state monitor (requires <i>monitor_steady_state.in</i>).	
<i>property_control</i>		
<i>gas_compressible_flag</i>	0 = Incompressible gas flow, 1 = Compressible gas flow .	
<i>liquid_compressible_flag</i>	0 = Incompressible liquid flow, 1 = Compressible liquid flow .	
<i>lhw_flag</i>	0 = CONVERGE calculates LHV from species data in <i>therm.dat</i> , 1 = Manually specify LHV (requires <i>lhw.in</i>).	
<i>individual_properties_flag</i>	0 = CONVERGE uses the same gas.dat file for all gas species, 1 = CONVERGE uses the gas property files listed in <i>species_transport.in</i> for the corresponding species listed in <i>species_transport.in</i> and uses gas.dat for all other gas species.	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>tabular_fluid_prop_flag</i>	0 = CONVERGE calculates fluid properties via the equation of state, 1 = CONVERGE reads fluid properties from fluid_properties.dat .	
<i>eos_flag</i>	<i>IDEAL_GAS</i> = Ideal_gas_law , <i>REDLICH_KWONG</i> = Redlich-Kwong , <i>REDLICH_KWONG_SOAVE</i> = Redlich-Kwong-Soave , <i>PENG_ROBINSON</i> = Peng-Robinson . Not used when <i>tabular_fluid_prop_flag</i> = 1.	
<i>real_gas_prop_flag</i>	0 = Thermodynamic quantities are functions of temperature, 1 = Thermodynamic quantities are functions of temperature and pressure (<i>i.e.</i> , real gas properties are calculated based on a modified equation of state). Used only when the equation of state is not the ideal gas law (<i>i.e.</i> , when <i>eos_flag</i> is not <i>IDEAL</i>).	
<i>max_reduced_pres</i>	The maximum reduced pressure for the departure function tables (used only when <i>real_gas_prop_flag</i> = 1).	6.0 is a typical value in an engine simulation
<i>crit_temp</i>	Critical temperature (K) for real gas equations of state. Used only when <i>eos_flag</i> is non-zero. For a simulation with multiple gas species, specify a file name (<i>e.g.</i> , <i>crit_cond.dat</i>) to include species-specific critical temperatures .	133K for air
<i>crit_pres</i>	Critical pressure (Pa) for real gas equations of state. Used only when <i>eos_flag</i> is non-zero. For a simulation with multiple gas species, specify a file name (<i>e.g.</i> , <i>crit_cond.dat</i>) to include species-specific critical pressures .	3.77e06 Pa for air
<i>acentric_factor</i>	Acentric factor. For a simulation with multiple gas species, specify a file name (<i>e.g.</i> , <i>crit_cond.dat</i>) to include species-specific acentric factors . Used only when <i>eos_flag</i> = <i>REDLICH_KWONG_SOAVE</i> or <i>PENG_ROBINSON</i> .	
<i>species_diffusion_model</i>	0 = Diffusion calculation is independent of species, 1 = CONVERGE calculates mixture-averaged diffusion coefficients (requires transport.dat), 2 = CONVERGE calculates each species' diffusivity based on schmidt_species.dat (also requires gas.dat),	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
	3 = CONVERGE calculates species diffusion based on schmidt_species.dat and conductivity through a user-defined calculation in <i>hidden.in</i> , 4 = CONVERGE calculates species diffusion based on schmidt_species.dat and conductivity based on transport.dat .	
<i>prandtl_turb\$</i>	Turbulent Prandtl number. Specify a file name (e.g., prandtl_turb.in) to set up region-specific and/or temporally varying values.	0.5 - 1.0
<i>schmidt_turb\$</i>	If <i>inputs.in</i> > <i>solver_control</i> > <i>turbulence_solver</i> = 0, molecular Schmidt number. If <i>turbulence_solver</i> = 1, turbulent Schmidt number.	0.5 - 1.0
	Specify a file name (e.g., schmidt_turb.in) to set up region-specific and/or temporally varying Schmidt number values. The file name option is available only when <i>turbulence_solver</i> = ON.	
<i>min_temp\$</i>	Minimum temperature (K) allowed during a simulation.	10.0 K or greater
<i>max_temp\$</i>	Maximum temperature (K) allowed during a simulation.	Up to 100,000 K
<i>max_visc\$</i>	Maximum turbulence viscosity ($N\cdot s/m^2$) allowed during a simulation.	10.0
<i>gravity\$</i>	Gravitational acceleration (m/s^2) in the x, y, and z directions. Specify a file name (e.g., <i>gravity_x.in</i>) to temporally vary this parameter. Note that CONVERGE applies the gravitational acceleration to the mixture, not to individual species.	
<i>feature_control</i>		
<i>parcel_mode</i>		
<i>liquid_parcel</i>	0 = No Lagrangian liquid parcel modeling, 1 = Lagrangian liquid parcel modeling enabled (requires parcels.in , parcel_introduction.in , parcel_parcel_interaction.in , parcel_wall_interaction.in , and liquid.dat). If combined with <i>vof_flag</i> = 1, ELSA model enabled (requires elsa.in).	
<i>solid_parcel</i>	0 = No Lagrangian solid parcel modeling, 1 = Lagrangian solid parcel modeling enabled (requires parcels.in , parcel_introduction.in , parcel_parcel_interaction.in , parcel_wall_interaction.in , and solid.dat).	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>gas_parcel</i>	0 = No Lagrangian gas parcel modeling. This parameter is reserved for future use.	
<i>urea_flag</i>	0 = No urea injection modeling, 1 = Urea injection modeling (requires urea.in).	
<i>combustion_flag</i>	0 = No combustion, 1 = Combustion modeling enabled (requires combust.in).	
<i>source_flag</i>	0 = No source modeling, 1 = Source modeling enabled (requires source.in).	
<i>aniso_cond_flag</i> *	0 = Thermal and electrical conductivity are isotropic throughout the domain (default), 1 = Thermal and/or electrical conductivity are anisotropic in parts of the domain (requires aniso_cond.in).	
<i>composite_flag</i>	0 = No composite species, 1 = Composite species modeling enabled (requires composite.in).	
<i>wallvalue_flag</i>	0 = No user-defined wall boundary quantities, 1 = User-defined wall boundary quantities (requires wall_value.in).	
<i>udf_flag</i>	0 = No user-defined functions (UDFs), 1 = Enable UDFs (requires udf.in).	
<i>cht_supercycle_flag</i>	0 = No super-cycling in conjugate heat transfer calculations, 1 = Enable super-cycling in conjugate heat transfer calculations (requires supercycle.in).	
<i>vof_flag</i>	0 = No volume of fluid calculations for cavitation, 1 = Volume of fluid calculations enabled (requires vof.in). If combined with <i>parcel_mode > liquid_parcel</i> , <i>solid_parcel</i> , or <i>gas_parcel = 1</i> , ELSA model enabled (requires elsa.in).	
<i>fixed_flow_flag</i>	0 = No fixed flow method, 1 = Fixed flow method enabled (requires fixed_flow.in).	
<i>fsi_flag</i>	0 = No fluid-structure interaction, 1 = Fluid-structure interaction enabled (requires fsi.in).	
<i>radiation_flag</i>	0 = No radiation, 1 = Radiation modeling enabled (requires radiation.in).	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>nucleate_boiling_flag</i>	0 = No nucleate boiling, 1 = Enable nucleate boiling model (requires nucleate_boiling.in).	
<i>skip_species_flag</i>	0 = CONVERGE will not remove species during chemistry computations, 1 = CONVERGE will reduce the number of species for chemistry computations (requires skip_species.in).	
<i>mrf_flag</i>	0 = No multiple reference frame approach, 1 = Multiple reference frame approach, relative formulation (mrf.in required), 2 = Multiple reference frame approach, absolute formulation (mrf.in required).	
<i>mixing_plane_flag</i>	0 = No mixing planes, 1 = Enable one or more mixing planes (requires mixing_plane.in).	
<i>prox_check_flag</i>	0 = No proximity checking, 1 = Perform proximity checking and flag cells within proximity (requires proximity_boundaries.in), 2 = Perform proximity checking and compute the minimum distance for each cell (requires proximity_boundaries.in)	
<i>cosimulation_flag</i>	0 = No co-simulation with third-party solvers, 1 = Co-simulation with third-party solvers (requires cosimulation.in).	
<i>motion_sets_flag</i>	0 = Do not specify sets of moving objects, 1 = Specify sets of moving objects (requires motion_sets.in).	
<i>initial_perturbation_flag</i>	0 = No initialization perturbation, 1 = Perturb the initialization conditions (requires initial_perturbation_region.in).	
<i>post_emission_flag</i>	0 = Do not use emissions post-processing, 1 = Use emissions post-processing to solve emissions and species with fixed major fluid quantities from a restart file (requires <i>inputs.in > simulation_control > restart_flag = 1, inputs.in > feature_control > combustion_flag = 1, inputs.in > solver_control > monitor_steady_state = 1, and combust.in > emissions_flag = 1</i>).	
<i>acoustics_flag</i>	0 = No aeroacoustics modeling, 1 = Aeroacoustics modeling (requires acoustics.in).	
<i>statistics_flag</i>	0. Reserved for future use.	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
<i>erosion_flag</i>	0 = Do not use erosion modeling, 1 = Activate erosion modeling (requires <i>inputs.in</i> > <i>feature_control</i> > <i>parcel_mode</i> > <i>solid_parcel</i> = 1 and <i>erosion.in</i>).	
<i>electric_potential_flag</i>	0 = Do not solve electric potential, 1 = Solve electric potential in solid streams.	
<i>ga_control</i>		
<i>ga_flag</i>	0 = This simulation is not part of a genetic algorithm or design of experiments, 1 = This simulation is part of a CONGO genetic algorithm (GA) or design of experiments (DoE) .	
<i>ga_individual</i>	For GA or DoE cases, in the CONVERGE Seed Case Directory , set to <i>GA_INDIVIDUAL</i> . CONGO will automatically set this value in the individual run directories for a CONGO case.	
<i>ga_generation</i>	For GA or DoE cases, in the CONVERGE Seed Case Directory , set to <i>GA_GENERATION</i> . CONGO will automatically set this value in the individual run directories for a CONGO case.	
<i>mpi_control</i>		
<i>shared_memory_flag</i> *	0 = Store a copy of the surface geometry for each rank, 1 = Store a single copy of the surface geometry on each node. Node-based storage reduces the memory requirement without affecting computational performance. Previous versions of CONVERGE have only rank-based geometry storage.	1
<i>io_hints_filename</i> *	File name for a hint file specific to the installed MPI library. This file consists of <key> <value> pairs that are fed directly to the MPI I/O library. The list of supported keys is dictated by your MPI library. These hints will be applied to the I/O operation for the map, restart, and post files. You cannot set up this file in CONVERGE Studio.	
<i>accelerator_control</i> *		
<i>use_gpu_flag</i>	0 = Do not use GPU acceleration (default), 1 = Use GPU acceleration.	
<i>amgx_config_filename</i>	Name of the configuration (*.json) file for the AmgX linear solver library.	
<i>enable_pin_mem</i>	Used only for the AmgX linear solver library.	

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

Parameter	Description	Typical value
	0 = Do not use pinned memory, 1 = Enable pinned memory to increase the bandwidth for CPU-GPU data transfers.	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

§ CONVERGE reads this parameter at each time-step when [inputs.in](#) > simulation_control > reread_input

Prandtl Number: prandtl_turb.in

To set up region-by-region and/or temporally varying turbulent Prandtl number values, specify a file name (e.g., *prandtl_turb.in*) for [inputs.in](#) > property_control > *prandtl_turb* and include that file in your case setup.

Use the first row of the file to specify a temporally varying type (TEMPORAL). The next row specifies the temporal type (SEQUENTIAL or CYCLIC). If you specify CYCLIC, you must also include the cyclic period (in seconds if [inputs.in](#) > simulation_control > crank_flag = 0 or in CAD if crank_flag is non-zero). The data in the CYCLIC input file will cycle according to the period you provide. The third row contains the headings for the two columns of data. The first heading must be *second* (if [inputs.in](#) > simulation_control > crank_flag = 0) or *crank* (if crank_flag is non-zero). The heading of the second column must be *prandtl_turb* to indicate that the turbulent Prandtl number varies.

As shown in the example below, the turbulent Prandtl number specified below the headings row (*crank* in this example) applies to all regions that are not specifically listed in the *region_id* parameters. After listing these default values for all of the other regions, list the region-specific turbulent Prandtl number under rows containing the *num_regions* and *region_id* parameters and values for these parameters as shown in the example below. After the default turbulent Prandtl numbers, this example has three other sets of turbulent Prandtl numbers: one set for region 1, one set for region 2, and one set for regions 3 and 4. Your file can contain as many data sets for different regions as are needed.

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

```
TEMPORAL
CYCLIC    720
crank     prandtl_turb
0.0      0.9
720      0.9

# For region 1
1      num_regions
1      region_id
0.0    0.2
720    0.2

# For region 2
1      num_regions
2      region_id
0.0    0.78
720    0.78

# For regions 3 and 4
2      num_regions
3      region_id
4      region_id
0.0    0.3
720    0.3
```

Figure 25.63: An example *prandtl_turb.in* file.

Schmidt Number: *schmidt_turb.in*

To set up region-by-region and/or temporally varying turbulent Schmidt number values, specify a file name (e.g., *schmidt_turb.in*) for [inputs.in](#) > *property_control* > *schmidt_turb* and include that file in your case setup. Note that turbulence modeling must be active (i.e., [inputs.in](#) > *solver_control* > *turbulence_solver* = 1) to invoke region-by-region and/or temporally varying Schmidt number values. The region-by-region option is available only when the [SAGE detailed chemical kinetics solver](#) is active (i.e., [combust.in](#) > *sage_model* > *active* is non-zero).

Use the first row of the file to specify a temporally varying type (*TEMPORAL*). The next row specifies the temporal type (*SEQUENTIAL* or *CYCLIC*). If you specify *CYCLIC*, you must also include the cyclic period (in seconds if [inputs.in](#) > *simulation_control* > *crank_flag* = 0 or in CAD if *crank_flag* is non-zero). The data in the *CYCLIC* input file will cycle according to the period you provide. The third row contains the headings for the two columns of data. The first heading must be *second* (if [inputs.in](#) > *simulation_control* > *crank_flag* = 0) or *crank* (if *crank_flag* is non-zero). The heading of the second column must be *schmidt_turb* to indicate that the turbulent Schmidt number varies.

As shown in the example below, the turbulent Schmidt number specified below the headings row (*crank* in this example) applies to all regions that are not specifically listed in the *region_id* parameters. After listing these default values for all of the other regions, list the region-specific turbulent Schmidt number under rows containing the parameters *num_regions* and *region_id* and values for these parameters as shown in the example below. After the default turbulent Schmidt numbers, this example has two other sets of turbulent Schmidt numbers: one set for regions 1 and 2 and one set for regions 3, 4, and 5. Your file can contain as many data sets for different regions as are needed.

Chapter 25: Input and Data Files

Simulation Parameters Simulation Control

```
TEMPORAL
CYCLIC    720
crank     schmidt_turb
0.0       0.9
720       0.9

# For regions 1 and 2
2        num_regions
1        region_id
2        region_id
0.0      0.78
720      0.78

# For regions 3, 4, and 5
3        num_regions
3        region_id
4        region_id
5        region_id
0.0      0.3
720      0.3
```

Figure 25.64: An example *schmidt_turb.in* file.

Restart File Specification: *twrite_restart.in*

The *inputs.in* > *output_control* > *twrite_restart* parameter controls the frequency with which CONVERGE writes *restart* files. You can specify a file name for this parameter (e.g., *twrite_restart.in*), and you can use the *twrite_restart.in* file to temporally vary this parameter and/or to direct CONVERGE to write additional restart files at specified times. To direct CONVERGE to write a restart file at a specified simulation time, include the optional *save* column. The following figure includes an example of a *twrite_restart.in* file that contains the *save* column. Set *save* to 0 if you do not want CONVERGE to write a restart file at that simulation time or to 1 if you do want CONVERGE to write a restart file at that simulation time.

For restart files written according to the value of *twrite_restart* (whether in *inputs.in* or *twrite_restart.in*), the parameter *inputs.in* > *output_control* > *num_restart_files* determines how many of these restart files CONVERGE will save (the oldest restart file will be the first to be overwritten). These restart files are named *restart<restart number>.rst* (e.g., *restart0001.rst*).

For restart files written according to the value of *save*, there is no limit on how many of these restart files CONVERGE will save. These restart files are named *restart_<simulation time or CAD>.rst*. These files do not count toward the *num_restart_files* to be saved.

```
TEMPORAL
CYCLIC    720.0
crank     twrite_restart   save
-147      1.0            1
-130      2.0            0
0          1.0            1
10         5.0            1
100        10.0           0
```

Figure 25.65: An example *twrite_restart.in*. CONVERGE will write and overwrite restart files (*restart<restart number>.rst*) according to the frequency specified in the *twrite_restart* column.

CONVERGE will write and save restart file at -147 CAD (*restart_-1.470000e+02.rst*), 0 CAD (*restart_0.000000e+00.rst*), and 10 CAD (*restart_1.000000e+01.rst*).

Post File Specification: `twrite_post.in`

The `inputs.in` > `output_control` > `twrite_post` parameter controls the frequency with which CONVERGE writes `post` files. You can specify a file name for this parameter (e.g., `twrite_post.in`), and you can use the `twrite_post.in` file to temporally vary this parameter and/or to direct CONVERGE to write additional post files at specified times. To direct CONVERGE to write a post file at a specified simulation time, include the optional `save` column. The following figure includes an example of a `twrite_post.in` file that contains the `save` column. Set `save` to 0 if you do not want CONVERGE to write a post file at that simulation time or to 1 if you do want CONVERGE to write a post file at that simulation time.

```
TEMPORAL
CYCLIC    720
crank      twrite_post   save
-360.0     10.0        1
-20.0      2.0         1
+150.0     20.0        0
+360.0     10.0        1
```

Figure 25.66: A temporally varying `twrite_post.in` file with the specification to write additional post files at particular times. In this example, CONVERGE will write a post file at -360.0 CAD (because `save` = 1 for that simulation time), then CONVERGE will write a post file every 10 CAD from -360 to -20.1 CAD, then CONVERGE will write a post file at -20.0 CAD, and so on.

Region-Dependent CFL Number: `max_cfl.in`

You can have a simulation with temporally varying maximum CFL numbers that vary on a region-by-region basis. For example, in regions with high velocity, the time-step calculated based on the maximum CFL number may be very small, resulting in significantly longer runtimes. If the flow in one region is less important than the flow in another, you can increase the maximum CFL number for the less-important region to reduce runtimes.

In the example shown in Figure 25.67 below, there are three groups of regions. Each group has different maximum CFL numbers.

Use the first row of the file to specify a temporally varying boundary condition (`TEMPORAL`). The next row specifies the type of temporally varying boundary condition (`SEQUENTIAL` or `CYCLIC`). If you specify `CYCLIC`, you must also include the cyclic period (in seconds if `inputs.in` > `simulation_control` > `crank_flag` = 0 or in CAD if `crank_flag` is non-zero). The data in the `CYCLIC` input file will cycle according to the period you provide. The third row contains the headings for the two columns of data. The first heading must be `second` (if `inputs.in` > `simulation_control` > `crank_flag` = 0) or `crank` (if `crank_flag` is non-zero). The heading of the second column must be the parameter associated with the CFL number. Note that all three of the CFL parameters (`inputs.in` > `temporal_control` > `max_cfl_u`, `temporal_control` > `max_cfl_nu`, and `temporal_control` > `max_cfl_mach`) can be specified in a region-dependent manner.

As shown in the example below, you must specify the maximum CFL numbers for all of the regions that are not specifically listed in the `region_id` parameters. After listing these default

Chapter 25: Input and Data Files

Simulation Parameters Region-Dependent CFL Number: max_cfl.in

values for all of the other regions, list the region-specific maximum CFL numbers under rows containing the *num_regions* and *region_id* parameters and values for these parameters as shown in the example below. After the default temporally varying maximum CFL numbers, this example has two other sets of temporally varying maximum CFL numbers: one set for regions 2 and 5 and one set for regions 8 and 11. Your file can contain as many data sets for different regions as are needed.

```
TEMPORAL
CYCLIC    720
crank      MAX_CFL_U
# For regions 0, 1, 3, 4, 6, 7, 9, and 10
-20     1.1
50      1.3
350     1.7
700     1.1

# For regions 2 and 5
2      num_regions
2      region_id
5      region_id
-20    2.0
50     2.2
100    2.8
700    2.0

# For regions 8 and 11
2      num_regions
8      region_id
11     region_id
-10    1.8
30     1.5
150    1.3
520    1.8
```

Figure 25.67: An example *max_cfl.in* file with region-by-region temporally varying maximum CFL numbers. The # rows are comments, which are not read by CONVERGE.

Solver Parameters: solver.in

The *solver.in* file, which is required for all simulations, contains parameters to control the numerical schemes and convergence criteria for each governing equation in the simulation.

For each equation (*e.g.*, the momentum transport equation denoted by the prefix *mom*), you choose the solver type, the convergence tolerance, a minimum and maximum number of iterations for CONVERGE to perform, and an under-relaxation factor and a preconditioner to apply to the equation. You can also specify parameters to control the flux schemes and flux limiters for the various governing equations.

We recommend setting up the *solver.in* parameters in CONVERGE Studio (*Case Setup > Simulation Parameters > Solver parameters*), which can provide recommendations for a wide variety of cases.

```
version: 3.1
---
NS_solver_scheme:          PISO
NS_solver_type:            DENSITY_BASED
```

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

```
PISO:  
    tol_scale:          20  
    piso_tol:           0.001  
    piso_itmin:         2  
    piso_itmax:         21  
    omega_presrat:     0.7  
  
SIMPLE:  
    tol_scale:          20  
    simple_tol:          0.001  
    simple_itmin:        2  
    simple_itmax:        21  
    omega_presrat:      0.7  
    omega_simple:        0.5  
    couple_turb_flag:    0  
  
Steady_state_solver:  
    steady_auto_flag:   0  
    steady_residual_output_flag: 1  
    steady_switch_solver_flag: 1  
    steady_tol_update_freq: 10  
    steady_min_num_amr: 20  
    steady_piso_tol_init: 0.01  
    steady_tol_scale_init: 10  
    steady_max_cfl_u_final: NOT_USED  
  
Numerical_Schemes:  
    flux_scheme:  
        mom:                  FLUX_BLENDING  
        global:                FLUX_BLENDING  
        turb:                  FLUX_BLENDING  
    fv_upwind_factor:  
        mom:                  0.5  
        global:                0.5  
        turb:                  1  
    monotone_tolerance: 1.0e-05  
    upwind_all_dir_flag: 0  
    muscl_blend_factor:  
        mom:                  1  
        global:                1  
        turb:                  1  
    flux_limiter:  
        mom:                  STEP  
        global:                STEP  
        turb:                  STEP  
    implicit_fraction: 1  
    conserve:              0.5  
    strict_conserve_level: 2  
    rc_flag:                LEGACY_RC  
    omega_turb:              1.0  
  
Transport:  
    mom:  
        solver_type: CONVERGE_BICGSTAB  
        tol:            1.0e-6  
        itmin:          0  
        itmax:          200  
        omega:           1  
        preconditioner: NONE  
    pres:  
        solver_type: HYPRE_BICGSTAB  
        tol:            1e-08  
        itmin:          2  
        itmax:          200000  
        omega:           1.3  
        preconditioner: NONE  
        reference_pressures:  
            - reference_pressure:  
                stream_id: 0  
                reference_location: [0.0, 0.0, 0.0]  
                reference_value: 101325.0
```

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

```
density:
    solver_type:          SOR
    tol:                  1.0-05
    itmin:                0
    itmax:                2
    omega:                1
    preconditioner:      NONE
energy:
    solver_type:          SOR
    tol:                  1.0e-05
    itmin:                0
    itmax:                2
    omega:                1
    preconditioner:      NONE
species:
    solver_type:          SOR
    tol:                  1.0e-05
    itmin:                0
    itmax:                2
    omega:                1
    preconditioner:      NONE
passive:
    solver_type:          SOR
    tol:                  1.0e-05
    itmin:                0
    itmax:                200
    omega:                1
    preconditioner:      NONE
scalar:
    solver_type:          SOR
    tol:                  1.0e-05
    itmin:                0
    itmax:                50
    omega:                1
    preconditioner:      NONE
radiation:
    solver_type:          SOR
    tol:                  0.0001
    itmin:                0
    itmax:                2500
    omega:                1
    preconditioner:      NONE
wall_dist:
    solver_type:          CONVERGE_BICGSTAB
    tol:                  1.0e-06
    itmin:                0
    itmax:                500
    omega:                1.1
    preconditioner:      AMG
tke:
    solver_type:          SOR
    tol:                  0.0001
    itmin:                2
    itmax:                200
    omega:                0.5
    preconditioner:      NONE
eps:
    solver_type:          SOR
    tol:                  0.0001
    itmin:                2
    itmax:                200
    omega:                0.5
    preconditioner:      NONE
omega:
    solver_type:          SOR
    tol:                  0.001
    itmin:                2
    itmax:                200
    omega:                0.7
    preconditioner:      NONE
v2:
```

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

```
solver_type:          SOR
tol:                  0.001
itmin:                2
itmax:                50
omega:                0.7
preconditioner:       NONE
f:
  solver_type:        SOR
  tol:                0.001
  itmin:              2
  itmax:              100
  omega:              1.1
  preconditioner:     NONE
zeta:
  solver_type:        SOR
  tol:                0.001
  itmin:              2
  itmax:              50
  omega:              0.7
  preconditioner:     NONE
electric_potential:
  solver_type:        CONVERGE_BICGSTAB
  tol:                1e-6
  itmin:              2
  itmax:              5000
  omega:              1.1
  preconditioner:     NONE
```

Figure 25.68: An example *solver.in* file.

Table 25.31: Format of *solver.in*.

Parameter	Description	Recommended value for a transient, compressible gas case
<i>NS_solver_scheme\$</i>	Iterative algorithm for the Navier-Stokes equations. <i>PISO</i> = PISO algorithm , <i>SIMPLE</i> = SIMPLE algorithm .	
<i>NS_solver_type\$</i>	Flow variable which is used as the iterative algorithm convergence criterion. <i>DENSITY_BASED</i> = Density is used as the convergence criterion, <i>PRESSURE_BASED</i> = Pressure is used as the convergence criterion.	<i>density_based</i>
<i>PISO</i>		
<i>tol_scale\$</i>	Multiplication factor for the PISO algorithm. This parameter must be at least 2.	20
<i>piso_tol\$</i>	Convergence criterion for the PISO algorithm.	1e-3
<i>piso_itmin\$</i>	Minimum number of iterations for the PISO algorithm.	2
<i>piso_itmax\$</i>	Maximum number of iterations for the PISO algorithm.	9

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
<i>omega_presrat\$</i>	Under-relaxation factor for density (pressure ratio) in PISO algorithm. Under-relax if density gradients are large. You can use a file (e.g., <i>omega_presrat.in</i>) for temporally varying under-relaxation values.	0.7
<hr/> <i>SIMPLE</i> <hr/>		
<i>tol_scale\$</i>	Multiplication factor for the SIMPLE algorithm. This parameter must be at least 2.	20
<i>simple_tol\$</i>	Convergence criterion for the SIMPLE algorithm.	1e-3
<i>simple_itmin\$</i>	Minimum number of iterations for the SIMPLE algorithm.	2
<i>simple_itmax\$</i>	Maximum number of iterations for the SIMPLE algorithm.	9
<i>omega_presrat\$</i>	Under-relaxation factor for density (pressure ratio) in SIMPLE algorithm. Under-relax if density gradients are large. You can use a file (e.g., <i>omega_presrat.in</i>) for temporally varying under-relaxation values.	0.7
<i>omega_simple\$</i>	Under-relaxation factor for momentum update inside the SIMPLE algorithm.	0.5
<i>couple_turb_flag\$</i>	0 = Do not solve turbulence transport equations within the SIMPLE loop, 1 = Solve turbulence transport equations within the SIMPLE loop.	0
<hr/> <i>Steady_state_solver</i> <hr/>		
<i>steady_auto_flag</i>	0 = Run a steady-state simulation without automatic monitoring of solver settings such as tolerances (recommended), 1 = Allow automatic monitoring and control of the steady-state solver settings via the parameters below. You must set inputs.in > solver_control > monitor_steady_state = 1 and supply a monitor_steady_state.in file that lists quantities to monitor and the monitoring configuration.	0
<i>steady_residual_output_flag</i>	0 = Do not write relative residuals to residuals.out , 1 = Write relative residuals to residuals.out .	

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
<code>steady_switch_solver_flag\$</code>	0 = Use the specified pressure solver for the duration of the simulation (for cases with combustion), 1 = Switch between BiCGSTAB and SOR pressure solvers to determine the optimal solver at the final grid scaling value (only for cases without combustion).	
<code>steady_tol_update_freq\$</code>	The frequency (in cycles) with which CONVERGE updates settings (e.g., doubles the maximum CFL number) and monitors the solution variables to determine if they have reached a local steady state.	20
<code>steady_min_num_amr\$</code>	The minimum number of times that CONVERGE applies AMR in a steady-state simulation. If grid scaling is specified, AMR is applied after the end of grid scaling.	10
<code>steady_piso_tol_init\$</code>	The PISO/SIMPLE convergence criterion for the steady-state solver (used instead of <code>PISO > piso_tol</code> or <code>SIMPLE > simple_tol</code>).	
<code>steady_tol_scale_init\$</code>	The PISO/SIMPLE multiplication factor for the steady-state solver (used instead of <code>PISO > tol_scale</code> or <code>SIMPLE > tol_scale</code>). Must be at least 2.	
<code>steady_max_cfl_u_final\$</code>	The maximum convection CFL number used during the final tolerance tightening stage. Enter <code>NOT_USED</code> (recommended) or specify a value less than inputs.in > temporal_control > max_cfl_u .	
<i>Numerical_schemes</i>		
<code>flux_scheme</code>	Flux scheme control. <code>FLUX_BLENDING</code> = Lower-order flux blending scheme, <code>MUSCL_1</code> = MUSCL with 1D flux limiter, <code>MUSCL_2</code> = MUSCL with step limiter and additional limiting factor, <code>MUSCL_CVG</code> = MUSCL with 3D gradient-based slope limiter (recommended MUSCL scheme).	
<code>mom</code>	Flux scheme for the momentum equation.	<code>FLUX_BLENDING</code>
<code>global</code>	Flux scheme for all transport equations except momentum and turbulence.	<code>FLUX_BLENDING</code>

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
<i>turb</i>	Flux scheme for the turbulence equations.	<i>FLUX_BLENDING</i>
<i>fv_upwind_factor</i>	Upwinding control (0.5 is central, 1.0 is first-order upwind).	
<i>mom</i>	Upwinding in the momentum finite volume solver.	0.5
<i>global</i>	Upwinding in the finite volume solver for all transport equations except momentum and turbulence.	0.5
<i>turb</i>	Upwinding in the turbulence finite volume solver.	1.0
<i>monotone_tolerance\$</i>	Tolerance for the <i>step</i> flux limiter above which CONVERGE switches to a lower-order spatial discretization to preserve stability.	1e-5
<i>upwind_all_dir_flag\$</i>	0 = Switch to lower-order spatial discretization only for the direction (e.g., <i>x</i> , <i>y</i> , or <i>z</i>) in which non-monotonicity is detected. 1 = Switch to lower-order spatial discretization for all directions (e.g., <i>x</i> , <i>y</i> , and <i>z</i>) when non-monotonicity is detected in just one direction (e.g., <i>x</i>).	0
<i>muscl_blend_factor</i>	Blending factor used for the MUSCL scheme (1.0 is completely reconstructed central difference, 0.0 is completely second-order upwind).	
<i>mom</i>	Factor for the momentum equation.	1.0
<i>global</i>	Factor for all transport equations except momentum and turbulence.	1.0
<i>turb</i>	Factor for the turbulence equations.	1.0
<i>flux_limiter</i>	Flux limiter control. Refer to the Flux Limiters section to see all available options.	
<i>mom</i>	Flux limiter to apply to the momentum equation.	<i>step</i>
<i>global</i>	Flux limiter to apply to all transport equations except momentum and turbulence.	<i>step</i>
<i>turb</i>	Flux limiter to apply to the turbulence equations.	<i>step</i>
<i>implicit_fraction\$</i>	Fraction of implicitness used in solving the governing equations.	1.0

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
<i>conserve\$</i>	Fraction of momentum that CONVERGE will solve in conservative form.	1.0
<i>strict_conserve_level\$</i>	0 = No strict conservation (no Jacobi iteration), 1 = Strict conservation (Jacobi iteration) for sie, enthalpy, density, scalars , and species, 2 = 1 + Strict conservation (Jacobi iteration) for passives . A value of 2 is not allowed if inputs.in > solver_control > steady_solver = 1 .	1 if inputs.in > temporal control > max_cfl_u is greater than 2.5, 2 if max_cfl_u is less than or equal to 2.5.
<i>rc_flag\$</i>	OFF = No Rhie-Chow scheme, LEGACY_RC = CONVERGE uses the legacy Rhie-Chow scheme, GENERALIZED_RC = CONVERGE uses the generalized Rhie-Chow scheme.	LEGACY_RC or GENERALIZE_D_RC
<i>omega_turb\$</i>	Over-relaxation factor for the turbulence equations.	1.0
<i>Transport</i>		
<i>mom</i>	Momentum solution control.	
<i>solver_type\$</i>	Specify the type of solver for momentum. SOR = SOR, CONVERGE_BICGSTAB = CONVERGE implementation of BiCGSTAB, HYPRE_BICGSTAB = HYPRE implementation of BiCGSTAB.	SOR
<i>tol\$</i>	Convergence criterion for momentum. Specify a file name (e.g., <i>mom_tol.in</i>) to temporally vary this parameter.	1e-5
<i>itmin\$</i>	Minimum number of iterations for momentum.	0
<i>itmax\$</i>	Maximum number of iterations for momentum.	50
<i>omega\$</i>	Under-relaxation factor for momentum.	1.0
<i>preconditioner\$</i>	Specify a preconditioner for momentum. Used only when <i>solver_type</i> = HYPRE_BICGSTAB. None = No preconditioner, ILU = Euclid/ILU preconditioner, AMG = Multigrid preconditioner.	None
<i>pres</i>	Pressure solution control.	
<i>solver_type\$</i>	Specify the type of solver for pressure.	SOR

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
	<i>SOR</i> = SOR, <i>CONVERGE_BICGSTAB</i> = CONVERGE implementation of BiCGSTAB, <i>HYPRE_BICGSTAB</i> = HYPRE implementation of BiCGSTAB, <i>AMGX</i> = AmgX linear solver library (requires inputs.in > <i>accelerator_control</i> > <i>use_gpu_flag</i> = 1 and an AmgX configuration file). <i>CONVERGE_BICGSTAB_GPU</i> = GPU-enabled version of CONVERGE_BICGSTAB (requires inputs.in > <i>accelerator_control</i> > <i>use_gpu_flag</i> = 1). <i>CONVERGE_JACOBI_GPU</i> = GPU-enabled version of a Jacobi solver (requires inputs.in > <i>accelerator_control</i> > <i>use_gpu_flag</i> = 1), <i>CONVERGE_RSOR_GPU</i> = GPU-enabled version of a Reduced SOR solver (requires inputs.in > <i>accelerator_control</i> > <i>use_gpu_flag</i> = 1).	
<i>tol\$</i>	Convergence criterion for pressure. Specify a file name (e.g., <i>pres_tol.in</i>) to temporally vary this parameter.	1e-8
<i>itmin\$</i>	Minimum number of iterations for pressure.	2
<i>itmax\$</i>	Maximum number of iterations for pressure.	500
<i>omega\$</i>	Under-relaxation factor for pressure.	1.1
<i>preconditioner\$</i>	Specify a preconditioner for pressure. <i>None</i> = No preconditioner, <i>ILU</i> = Euclid/ILU preconditioner (available only for CONVERGE_BICGSTAB or HYPRE_BICGSTAB), <i>AMG</i> = Multigrid preconditioner (available only for CONVERGE_BICGSTAB or HYPRE_BICGSTAB), <i>SOR</i> = SOR preconditioner (available only for CONVERGE_BICGSTAB), <i>SSOR</i> = Symmetric SOR preconditioner (available only for CONVERGE_BICGSTAB), <i>RSOR_GPU</i> = GPU-enabled reduced SOR preconditioner (available only for CONVERGE_BICGSTAB_GPU).	<i>None</i>
<i>reference_pressures</i>	Reference points for fixing the pressure for all Neumann pressure boundary conditions in incompressible flow cases.	

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
<i>- reference_pressure</i>	For multiple reference pressures, repeat the <i>reference_pressure</i> block through <i>reference_value</i> . CONVERGE allows a maximum of one location per fluid stream.	
<i>stream_id</i>	Stream ID for the reference location.	
<i>reference_location</i>	Coordinates of the reference pressure location (<i>m</i>).	
<i>reference_value</i>	Reference pressure magnitude (Pa).	
<i>density</i>	Density solution control.	
<i>solver_type\$</i>	Specify the type of solver for density. <i>SOR</i> = SOR, <i>CONVERGE_BICGSTAB</i> = CONVERGE implementation of BiCGSTAB, <i>HYPRE_BICGSTAB</i> = HYPRE implementation of BiCGSTAB.	<i>SOR</i>
	Neither BiCGSTAB option is available when the geometry includes moving boundaries.	
<i>tol\$</i>	Convergence criterion for density. Specify a file name (e.g., <i>density_tol.in</i>) to temporally vary this parameter.	1e-4
<i>itmin\$</i>	Minimum number of iterations for density.	0
<i>itmax\$</i>	Maximum number of iterations for density.	2
<i>omega\$</i>	Under-relaxation factor for density.	1.0
<i>preconditioner\$</i>	Specify a preconditioner for density. Used only when <i>solver_type</i> = <i>CONVERGE_BICGSTAB</i> or <i>HYPRE_BICGSTAB</i> . <i>None</i> = No preconditioner, <i>ILU</i> = Euclid/ILU preconditioner, <i>AMG</i> = Multigrid preconditioner (available only for <i>HYPRE_BICGSTAB</i>), <i>SOR</i> = SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>), <i>SSOR</i> = Symmetric SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>).	<i>None</i>
<i>energy</i>	Energy solution control.	
<i>solver_type\$</i>	Specify the type of solver for specific internal energy. <i>SOR</i> = SOR,	<i>SOR</i>

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
	CONVERGE_BICGSTAB = CONVERGE implementation of BiCGSTAB, HYPRE_BICGSTAB = HYPRE implementation of BiCGSTAB.	
	Neither BiCGSTAB option is available when inputs.in > solver_control > steady_solver = 1 or when the geometry includes moving boundaries.	
tol\$	Convergence criterion for specific internal energy. Specify a file name (e.g., <i>energy_tol.in</i>) to temporally vary this parameter.	1e-4
itmin\$	Minimum number of iterations for specific internal energy.	0
itmax\$	Maximum number of iterations for specific internal energy.	2
omega\$	Under-relaxation factor for specific internal energy.	1.0
preconditioner\$	Specify a preconditioner for specific internal energy. <i>None</i> Used only when solver_type = CONVERGE_BICGSTAB or HYPRE_BICGSTAB. <i>None</i> = No preconditioner, <i>ILU</i> = Euclid/ILU preconditioner, <i>AMG</i> = Multigrid preconditioner (available only for HYPRE_BICGSTAB), <i>SOR</i> = SOR preconditioner (available only for CONVERGE_BICGSTAB), <i>SSOR</i> = Symmetric SOR preconditioner (available only for CONVERGE_BICGSTAB).	
species	Species solution control.	
solver_type\$	Specify the type of solver for species. <i>SOR</i> = SOR, CONVERGE_BICGSTAB = CONVERGE implementation of BiCGSTAB, HYPRE_BICGSTAB = HYPRE implementation of BiCGSTAB.	SOR
	Neither BiCGSTAB option is available when inputs.in > solver_control > steady_solver = 1 or when the geometry includes moving boundaries.	
tol\$	Convergence criterion for species. Specify a file name (e.g., <i>species_tol.in</i>) to temporally vary this	1e-4

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
	parameter.	
<i>itmin\$</i>	Minimum number of iterations for species. For VOF simulations with HRIC , CONVERGE internally sets this to 25 if the specified <i>itmin</i> is less than 25.	0
<i>itmax\$</i>	Maximum number of iterations for species	2
<i>omega\$</i>	Under-relaxation factor for species.	1.0
<i>preconditioner\$</i>	Specify a preconditioner for species. Used only when <i>solver_type</i> = <i>CONVERGE_BICGSTAB</i> or <i>HYPRE_BICGSTAB</i> . <i>None</i> = No preconditioner, <i>ILU</i> = Euclid/ILU preconditioner, <i>AMG</i> = Multigrid preconditioner (available only for <i>HYPRE_BICGSTAB</i>), <i>SOR</i> = SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>), <i>SSOR</i> = Symmetric SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>).	<i>None</i>
<i>passive</i>	Passive solution control.	
<i>solver_type\$</i>	Specify the type of solver for passives. <i>SOR</i> = SOR, <i>CONVERGE_BICGSTAB</i> = CONVERGE implementation of BiCGSTAB, <i>HYPRE_BICGSTAB</i> = HYPRE implementation of BiCGSTAB.	<i>SOR</i>
	Neither BiCGSTAB option is available when inputs.in > <i>solver_control</i> > <i>steady_solver</i> = 1 or when the geometry includes moving boundaries.	
<i>tol\$</i>	Convergence criterion for passives. Specify a file name (e.g., <i>passive_tol.in</i>) to temporally vary this parameter.	1e-5
<i>itmin\$</i>	Minimum number of iterations for passives.	0
<i>itmax\$</i>	Maximum number of iterations for passives.	50
<i>omega\$</i>	Under-relaxation factor for passives.	1.0
<i>preconditioner\$</i>	Specify a preconditioner for passives. Used only when <i>solver_type</i> = <i>CONVERGE_BICGSTAB</i> or <i>HYPRE_BICGSTAB</i> . <i>None</i> = No preconditioner,	<i>None</i>

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
	<i>ILU</i> = Euclid/ILU preconditioner, <i>AMG</i> = Multigrid preconditioner (available only for <i>HYPRE_BICGSTAB</i>), <i>SOR</i> = SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>), <i>SSOR</i> = Symmetric SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>).	
<i>scalar</i>	Scalar solution control.	
<i>solver_type\$</i>	Specify the type of solver for scalars. <i>SOR</i> = SOR, <i>CONVERGE_BICGSTAB</i> = CONVERGE implementation of BiCGSTAB, <i>HYPRE_BICGSTAB</i> = HYPRE implementation of BiCGSTAB.	<i>SOR</i>
	Neither BiCGSTAB option is available when <i>inputs.in</i> > <i>solver_control</i> > <i>steady_solver</i> = 1.	
<i>tol\$</i>	Convergence criterion for scalars. Specify a file name (e.g., <i>scalar_tol.in</i>) to temporally vary this parameter.	1e-5
<i>itmin\$</i>	Minimum number of iterations for scalars.	0
<i>itmax\$</i>	Maximum number of iterations for scalars.	50
<i>omega\$</i>	Under-relaxation factor for scalars.	1.0
<i>preconditioner\$</i>	Specify a preconditioner for scalars. Used only when <i>solver_type</i> = <i>CONVERGE_BICGSTAB</i> or <i>HYPRE_BICGSTAB</i> . <i>None</i> = No preconditioner, <i>ILU</i> = Euclid/ILU preconditioner, <i>AMG</i> = Multigrid preconditioner (available only for <i>HYPRE_BICGSTAB</i>), <i>SOR</i> = SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>), <i>SSOR</i> = Symmetric SOR preconditioner (available only for <i>CONVERGE_BICGSTAB</i>).	None
<i>radiation</i>	Radiation solution control.	
<i>solver_type</i>	Specify the type of solver for radiation. <i>SOR</i> = SOR,	<i>SOR</i>

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
	<code>CONVERGE_BICGSTAB</code> = CONVERGE implementation of BiCGSTAB (only available for the P1 radiation model).	
	Neither BiCGSTAB option is available when inputs.in > <code>solver_control</code> > <code>steady_solver</code> = 1.	
<code>tol\$</code>	Convergence criterion for radiation. Specify a file name (e.g., <code>rad_tol.in</code>) to temporally vary this parameter.	1e-8
<code>itmin\$</code>	Minimum number of iterations for radiation.	0
<code>itmax\$</code>	Maximum number of iterations for radiation.	2500
<code>omega\$</code>	Under-relaxation factor for radiation.	1.0
<code>preconditioner</code>	Reserved for future use.	0
<code>wall_dist</code>	Wall distance solution control.	
<code>solver_type\$</code>	Specify the type of solver for wall distance. <i>SOR</i> = SOR, <code>CONVERGE_BICGSTAB</code> = CONVERGE implementation of BiCGSTAB, <code>HYBRID</code> = CONVERGE BiCGSTAB for the first time-step, SOR for all subsequent time-steps.	<code>HYBRID</code>
<code>tol\$</code>	Convergence criterion for wall distance. Specify a file name (e.g., <code>wall_dist_tol.in</code>) to temporally vary this parameter.	1e-6
<code>itmin\$</code>	Minimum number of iterations for wall distance.	0
<code>itmax\$</code>	Maximum number of iterations for wall distance.	500
<code>omega\$</code>	Under-relaxation factor for wall distance.	1.1
<code>preconditioner\$</code>	Specify a preconditioner for species. Used only when <code>solver_type</code> = <code>CONVERGE_BICGSTAB</code> or <code>HYBRID</code> . <i>None</i> = No preconditioner, <i>ILU</i> = Euclid/ILU preconditioner, <i>AMG</i> = Multigrid preconditioner, <i>SOR</i> = SOR preconditioner (available only for <code>CONVERGE_BICGSTAB</code>), <i>SSOR</i> = Symmetric SOR preconditioner (available only for <code>CONVERGE_BICGSTAB</code>).	<i>None</i>

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
<i>tke</i> or <i>sgs_ke</i>	Turbulent kinetic energy (RANS) or sub-grid scale kinetic energy (LES) solution control.	
<i>solver_type</i>	Specify the type of solver for turbulent kinetic energy. <i>SOR</i> = SOR.	<i>SOR</i>
<i>tol\$</i>	Convergence criterion for turbulent kinetic energy. Specify a file name (e.g., <i>tke_tol.in</i>) to temporally vary this parameter.	1e-3
<i>itmin\$</i>	Minimum number of iterations for turbulent kinetic energy.	2
<i>itmax\$</i>	Maximum number of iterations for turbulent kinetic energy.	50
<i>omega\$</i>	Under-relaxation factor for turbulent kinetic energy.	0.7
<i>preconditioner</i>	Reserved for future use.	0
<i>eps</i> or <i>sgs_eps</i>	Turbulent dissipation (RANS) or sub-grid scale dissipation (LES) solution control.	
<i>solver_type</i>	Specify the type of solver for turbulent dissipation. <i>SOR</i> = SOR.	<i>SOR</i>
<i>tol\$</i>	Convergence criterion for turbulent dissipation. Specify a file name (e.g., <i>eps_tol.in</i>) to temporally vary this parameter.	1e-3
<i>itmin\$</i>	Minimum number of iterations for turbulent dissipation.	2
<i>itmax\$</i>	Maximum number of iterations for turbulent dissipation.	50
<i>omega\$</i>	Under-relaxation factor for turbulent dissipation.	0.7
<i>preconditioner</i>	Reserved for future use.	0
<i>omega</i>	Specific dissipation rate solution control.	
<i>solver_type</i>	Specify the type of solver for specific dissipation rate. <i>SOR</i> = SOR.	<i>SOR</i>
<i>tol\$</i>	Convergence criterion for specific dissipation rate. Specify a file name (e.g., <i>omega_tol.in</i>) to temporally vary this parameter.	1e-3

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
<i>itmin\$</i>	Minimum number of iterations for specific dissipation rate.	2
<i>itmax\$</i>	Maximum number of iterations for specific dissipation rate.	50
<i>omega\$</i>	Under-relaxation factor for specific dissipation rate.	0.7
<i>preconditioner</i>	Reserved for future use.	
<i>v2</i>	Velocity variance normal to the streamline solution control.	
<i>solver_type</i>	Specify the solver type for velocity variance. <i>SOR</i> = SOR.	<i>SOR</i>
<i>tol\$</i>	Convergence criterion for velocity variance. Specify a file name (e.g., <i>v2_tol.in</i>) to temporally vary this parameter.	
<i>itmin\$</i>	Minimum number of iterations for velocity variance.	
<i>itmax\$</i>	Maximum number of iterations for velocity variance.	
<i>omega\$</i>	Under-relaxation factor for velocity variance.	
<i>preconditioner</i>	Reserved for future use.	
<i>f</i>	Elliptic relaxation function solution control.	
<i>solver_type</i>	Specify the solver type for elliptic relaxation function. <i>SOR</i> = SOR, <i>CONVERGE_BICGSTAB</i> = CONVERGE implementation of BiCGSTAB.	<i>SOR</i>
<i>tol\$</i>	Convergence criterion for elliptic relaxation function. Specify a file name (e.g., <i>f_tol.in</i>) to temporally vary this parameter.	
<i>itmin\$</i>	Minimum number of iterations for elliptic relaxation function.	
<i>itmax\$</i>	Maximum number of iterations for elliptic relaxation function.	
<i>omega\$</i>	Under-relaxation factor for elliptic relaxation function.	
<i>preconditioner</i>	<i>None</i> = No preconditioner,	

Chapter 25: Input and Data Files

Simulation Parameters Solver Parameters: solver.in

Parameter	Description	Recommended value for a transient, compressible gas case
	<i>SOR</i> = SOR preconditioner (available only for CONVERGE_BICGSTAB).	
<i>zeta</i>	Velocity scales ratio solution control.	
<i>solver_type</i>	Specify the solver type for velocity scales ratio. <i>SOR</i> = SOR.	<i>SOR</i>
<i>tol\$</i>	Convergence criterion for velocity scales ratio. Specify a file name (e.g., <i>zeta_tol.in</i>) to temporally vary this parameter.	
<i>itmin\$</i>	Minimum number of iterations for velocity scales ratio.	
<i>itmax\$</i>	Maximum number of iterations for velocity scales ratio.	
<i>omega\$</i>	Under-relaxation factor for velocity scales ratio.	
<i>preconditioner</i>	Reserved for future use.	
<i>electric_potential</i>	Electric potential solution control.	
<i>solver_type</i>	Specify the solver type for electric potential. CONVERGE_BICGSTAB = CONVERGE implementation of BiCGSTAB, NONLINEAR_KRYLOV = Nonlinear Krylov acceleration (NKA) method.	
<i>tol\$</i>	Convergence criterion for electric potential. Specify a file name (e.g., <i>phi_tol.in</i>) to temporally vary this parameter.	
<i>itmin\$</i>	Minimum number of iterations for electric potential.	
<i>itmax\$</i>	Maximum number of iterations for electric potential.	
<i>omega\$</i>	Under-relaxation factor for electric potential.	
<i>preconditioner</i>	Reserved for future use.	

§ CONVERGE reads this parameter at each time-step when [*inputs.in*](#) > *simulation_control* > *reread_input* = 1.

Steady-State Monitor: monitor_steady_state.in

The steady-state monitor can track solution progress for [steady-state](#) or [transient](#) simulations to determine when user-specified variables have reached a steady state. For steady-state simulations ([*inputs.in*](#) > *solver_control* > *steady_solver* = 1) with automatic solution monitoring

Chapter 25: Input and Data Files

Simulation Parameters Steady-State Monitor: monitor_steady_state.in

([solver.in](#) > *Steady_state_solver* > *steady_auto_flag* = 1), the steady-state monitor is required. For other simulations, this feature is optional.

To activate the steady-state monitor, set [inputs.in](#) > *solver_control* > *monitor_steady_state* = 1 and include a *monitor_steady_state.in* file in your case setup. In *monitor_steady_state.in*, specify any number of variables from the *.out files for CONVERGE to monitor. At each time-step, CONVERGE will read the specified variables and determine whether each quantity is at a steady state according to the user-specified tolerance type (*variables* > *variable* > *tol_type*) and tolerance value (*variables* > *variable* > *tol_avg*). When the quantity reaches a steady state, CONVERGE will either stop the simulation or begin grid scaling (the latter applies only to steady-state simulations with automated grid scaling). If you use *monitor_steady_state.in* to monitor multiple quantities, CONVERGE will stop the simulation or begin grid scaling when all quantities reach a steady state.

When you run a simulation that includes the steady-state monitor, CONVERGE will write the monitored variables and grid scaling information (if applicable) to the restart files. You can restart from these files with the steady-state monitor activated, and CONVERGE will automatically retain the monitored variable information and adopt the previous grid scaling.

When monitoring most variables for steady state, CONVERGE evaluates the difference between two successive samples sets of data (with a sample size of *monitor_steady_state.in* > *variables* > *variable* > *sample_size*). If *monitor_steady_state.in* > *variables* > *variable* > *tol_type* = *RELATIVE*, the difference is evaluated according to

$$diff = \frac{|M_2 - M_1|}{\left(\frac{M_1 + M_2}{2} \right)}, \quad (25.83)$$

If you choose *absolute* for the tolerance type, CONVERGE calculates the difference between two successive sample sets via

$$diff = |M_2 - M_1| \quad (25.84)$$

When monitoring *MASS_FLOW_RATE_NET*, CONVERGE operates on a single sample set. If you choose a relative tolerance, CONVERGE calculates the difference as

$$diff = \frac{|M|}{\left| \sum_{inflows} \dot{m} \right| + \left| \sum_{outflows} \dot{m} \right|}. \quad (25.85)$$

When you choose an absolute tolerance when monitoring *MASS_FLOW_RATE_NET*, CONVERGE evaluates the difference as

Chapter 25: Input and Data Files

Simulation Parameters Steady-State Monitor: monitor_steady_state.in

$$diff = |M|. \quad (25.84)$$

If the difference is no greater than *monitor_steady_state.in > variables > variable > tol_avg* and if the standard deviation of the second sample set is no greater than *variables > variable > max_std*, then CONVERGE considers that particular quantity to be at a steady state.

When there is a chance that a solution may be periodic, set *monitor_steady_state.in > variables > variable > check_periodicity_flag = 1*. CONVERGE will compare the standard deviations σ_1 and σ_2 of two successive sample sets using either Equation 25.84 or Equation 25.85 above, depending on whether you specified *relative* or *absolute* for *tol_type*. CONVERGE will use the same formula, but with σ_1 instead of M_1 and σ_2 instead of M_2 . If the difference between the standard deviations falls within the value you specify for *monitor_steady_state.in > variables > variable > tol_std*, CONVERGE will end the simulation.

Note that when the steady-state monitor is enabled, CONVERGE writes text output to the *.out files at each time-step regardless of the interval specified for *inputs.in > output_control > twrite_files*.

```
version: 3.1
---

min_cycles_steady: 1000
monitor_units: SECONDS

variables:
  - variable:
      variable_name: MASS_FLOW_RATE@BOUND_ID_10
      filename: mass_avg_flow.out
      stream_id: [0,1]
      column_num: NOT_USED
      monitor_delay: 0.1
      sample_size: 100
      duration_size: 0.0001
      tol_type: RELATIVE
      tol_avg: 0.01
      max_std: 0.01
      check_periodicity_flag: 1
      tol_std: 0.001

  - variable:
      variable_name: MASS_FLOW_RATE_NET
      filename: mass_balance.out
      stream_id: ALL
      column_num: NOT_USED
      monitor_delay: 0.1
      sample_size: 100
      duration_size: 0.0001
      tol_type: RELATIVE
      tol_avg: 0.01
      max_std: 0.01
      check_periodicity_flag: 1
      tol_std: 0.001
```

Figure 25.69: An example *monitor_steady_state.in* file.

Chapter 25: Input and Data Files

Simulation Parameters Steady-State Monitor: monitor_steady_state.in

Table 25.32: Format of *monitor_steady_state.in*.

Parameter	Description
<i>min_cycles_steady</i> *\$	Minimum number of cycles for steady-state calculations.
<i>monitor_units</i>	The units to use for the steady-state monitor. <i>SECONDS</i> = simulation time is in seconds, <i>CYCLES</i> = simulation time is in cycles.
<i>variables</i>	
- <i>variable</i>	This settings sub-block specifies parameters for the monitor variable. Repeat this sub-block for each monitor variable.
<i>variable_name</i>	Name of the variable to monitor. <i>MASS_FLOW_RATE_NET</i> = Net mass flow rate through the domain (i.e., the sum of mass flow rates through all INFLOW and OUTFLOW boundaries, nozzle injectors, and boundary injectors), <i><VARIABLE>@BOUND_ID_<boundary_id></i> = Quantity calculated on a boundary, <i><VARIABLE>@REGION_ID_<region_id></i> = Quantity calculated over a region, <i><VARIABLE>@REGIONS_<region_id_from>_TO_<region_id_to></i> = Flow of a quantity between two regions, <i><VARIABLE>@POINT_ID_<monitor_point_num></i> = Value at a monitor point. Standard values of <i><VARIABLE></i> are listed in the <i>Steady-state monitor</i> window in CONVERGE Studio. We recommend choosing from these standard variables. CONVERGE Studio will automatically format <i>variable_name</i> and populate <i>filename</i> based on your selections. Note that the <i>@POINT_ID</i> option is available only when <i><VARIABLE></i> is set to a species name or passive name.
	To use a custom variable, specify <i>variable_name</i> , <i>filename</i> , and <i>column_num</i> . The value of <i>variable_name</i> does not have to match the name in the *.out file.
<i>filename</i>	Name of the *.out file to which CONVERGE writes the variable.
<i>stream_id</i> *	ID(s) of the stream(s) in which to monitor the variable. Specify <i>ALL</i> to monitor the variable in all streams where the variable exists.
<i>column_num</i>	Column number of the *.out file in which CONVERGE writes the variable. Used only for custom variables. Set to <i>NOT_USED</i> for standard variables.
<i>monitor_delay</i>	Start-up delay after which CONVERGE will begin monitoring the quantities of interest.
<i>sample_size</i>	The size of the sample that CONVERGE uses to determine if a quantity has reached steady state.

Chapter 25: Input and Data Files

Simulation Parameters Steady-State Monitor: monitor_steady_state.in

Parameter	Description
<i>duration_size\$</i>	The minimum time (or pseudo-time) over which to determine if the specified variable has reached steady state.
<i>tol_type</i>	Type of tolerance (either ABSOLUTE or RELATIVE).
<i>tol_avg\$</i>	Magnitude of the tolerance.
<i>max_std\$</i>	Maximum standard deviation for the quantity.
<i>check_periodicity_flag\$</i>	If 1, CONVERGE compares the difference in the standard deviation of the mean for two adjacent sample windows. If this difference is within <i>tol_std</i> , the variable is periodic. 0 = Do not check for periodicity, 1 = Check for periodicity.
<i>tol_std\$</i>	Tolerance for the difference between standard deviations when CONVERGE checks for periodicity.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

§ CONVERGE reads this parameter at each time-step when [inputs.in](#) > simulation_control > reread_input = 1.

Multiple Reference Frame Approach: mrf.in

To use the [multiple reference frame](#) (MRF) approach, set [inputs.in](#) > feature_control > mrf_flag = 1 or 2 (relative or absolute formulation, respectively). For each moving reference frame, supply the region ID and the details of the rotation with respect to the local reference frame.

```
version: 3.1
---

- mrf_region:
  region_id: 0
  specification:
    method: COPY_FROM_BOUNDARY
    boundary_id: 34
    rotation_axis: [1.0, 2.0, 3.0]
    rotation_point: [1.5, 2.5, 3.5]
    rotation_speed: 200000.0

- mrf_region:
  region_id: 1
  specification:
    method: MANUAL_INPUT
    boundary_id: -1
    rotation_axis: [1.0, 2.0, 3.0]
    rotation_point: [1.5, 2.5, 3.5]
    rotation_speed: 200000.0
```

Figure 25.70: An example *mrf.in* file.

Chapter 25: Input and Data Files

Simulation Parameters Multiple Reference Frame Approach: mrf.in

Table 25.33: Format of *mrf.in*.

Parameter	Description
- <i>mrf_region</i>	This settings block specifies the MRF region details. Repeat this block through <i>rotation_speed</i> for each MRF region.
<i>region_id</i>	Region ID (from initialize.in) for the local reference frame.
<i>specification</i>	
<i>method</i>	<i>MANUAL_INPUT</i> = Do not copy boundary rotation information, <i>COPY_FROM_BOUNDARY</i> = Copy rotation information from another boundary. On the following line, enter the boundary ID for <i>boundary_id</i> . CONVERGE ignores any rotation information specified below <i>boundary_id</i> . If you wish to supply a varying rotational speed for the local reference frame, choose a <i>boundary_id</i> corresponding to a boundary configured with the desired rotation properties. CONVERGE copies these properties and applies them to the local reference frame.
<i>boundary_id</i>	Boundary ID from which to copy rotation information (if <i>method</i> = <i>COPY_FROM_BOUNDARY</i>). For INTERFACE boundaries, use a positive value to specify the forward side or a negative value to specify the reverse side (e.g., if the boundary ID of the INTERFACE is 5, specify 5 for the forward side or -5 for the reverse side).
<i>rotation_axis</i>	The x, y, and z components of the vector that describes the rotation of the local reference frame.
<i>rotation_point</i>	The x, y, and z coordinates (in meters) of the point about which the local reference frame rotates.
<i>rotation_speed</i>	Rotation speed (in degrees per second) of the local reference frame with respect to the inertial reference frame.

Fixed Flow Method: *fixed_flow.in*

To activate fixed flow modeling, set [inputs.in](#) > *feature_control* > *fixed_flow_flag* = 1 and include a *fixed_flow.in* file in your case setup.

Chapter 25: Input and Data Files

Simulation Parameters Fixed Flow Method: `fixed_flow.in`

```
version: 3.1
---

fixed_stream_ids: [0,1]
fixed_spray_film_flag: TRACK_PARCELS
solve_fixed_species_flag: ON
solver_control:
    piso_tol: 1e-2
    piso_itmax: 3
    fully_solve_trace_species: ON
    solve_density_transport: ON
    use_cfl_limits: OFF
fixed_time_control:
    reference_time_stream_id: SELF
    start_time: 0.01
    end_time: 99.9
    temporal_type: CYCLIC
    cyclic_period: 0.50
    fixed_intervals:
        - [0.05, 0.20]
        - [0.25, 0.50]
```

Figure 25.71: An example `fixed_flow.in` file.

Table 25.34: Format of `fixed_flow.in`.

Parameter	Description
<code>fixed_stream_ids</code>	List of <code>stream_ids</code> where flow will be fixed. All the regions within a specified stream will be fixed.
<code>fixed_spray_film_flag</code>	Flag to control the behavior of spray/film parcels during <code>fixed_intervals</code> (see below). <code>TRACK_PARCELS</code> = Allow parcels to move unhindered, <code>FIX_PARCELS</code> = Hold parcels in place, <code>REMOVE_PARCELS</code> = Remove parcels from domain, <code>NOT_ACTIVE</code> = Perform a fixed flow simulation without parcel modeling.
<code>solve_fixed_species_flag</code>	<code>OFF</code> = Do not solve species transport during fixed flow intervals, <code>ON</code> = Solve species transport during fixed flow intervals.
<code>solver_control*</code>	Solver settings to be used during fixed flow intervals. Used only when <code>solve_fixed_species_flag = ON</code> .
<code>piso_tol</code>	PISO/SIMPLE convergence tolerance for fixed flow intervals.
<code>piso_itmax</code>	Maximum number of PISO/SIMPLE iterations for fixed flow intervals. The solver always proceeds to the next time-step after reaching this number of iterations (<i>i.e.</i> , the solver does not attempt to recover).
<code>fully_solve_trace_species</code>	<code>OFF</code> = During fixed flow intervals, CONVERGE solves only the species for which the change in mass fraction from the previous PISO/SIMPLE iteration exceeds the species tolerance (solver.in > <code>Transport</code> > <code>species tol</code>), <code>ON</code> = CONVERGE solves all species during fixed flow intervals.

Chapter 25: Input and Data Files

Simulation Parameters Fixed Flow Method: fixed_flow.in

Parameter	Description
<code>solve_density_transport</code>	<i>OFF</i> = CONVERGE does not solve for density transport during fixed flow intervals, <i>ON</i> = CONVERGE solves for density transport during fixed flow intervals.
<code>use_cfl_limits</code>	<i>OFF</i> = Do not apply time-step limiters based on CFL number (e.g., dt_cfl) during fixed flow intervals, <i>ON</i> = Apply time-step limiters based on CFL number during fixed flow intervals.
<i>fixed_time_control</i>	
<code>reference_time_stream_id</code>	Stream to which the start and end times refer. <i>SELF</i> = CONVERGE compares the start and end times to the simulation time of the current stream, <i><Stream ID></i> = CONVERGE compares the start and end times to the simulation time of the specified stream.
<code>start_time</code>	Time (in seconds if inputs.in > <code>simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero) at which fixed flow can first be activated. Note that you can have several <code>fixed_intervals</code> while this feature is active.
<code>end_time</code>	Time (in seconds if inputs.in > <code>simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero) at which fixed flow is deactivated.
<code>temporal_type</code>	Temporal type for <code>fixed_intervals</code> . <i>SEQUENTIAL</i> = Each <code>fixed_intervals</code> is fixed once. <i>CYCLIC</i> = Listed <code>fixed_intervals</code> repeat every <code>cyclic_period</code> until <code>end_time</code> .
<code>cyclic_period</code>	Cyclic period (in seconds if inputs.in > <code>simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero) for <code>fixed_intervals</code> . Used only when <code>temporal_type</code> = CYCLIC.
<code>fixed_intervals</code>	List of time periods during which flow will be fixed.
- [<i><x></i> , <i><y></i>]	Specify each fixed flow interval as a pair of start time <i><x></i> and end times <i><y></i> in a new line beginning with a hyphen.

* Sub-blocks or parameters indicated with an asterisk are optional.

Skip Species: `skip_species.in`

The [skip_species](#) feature allows CONVERGE to avoid transporting species that have insignificant mass, which may save computational time. To activate this option, set [inputs.in](#) > `feature_control > skip_species_flag` = 1 and include a `skip_species.in` file in your case setup.

Chapter 25: Input and Data Files

Simulation Parameters Skip Species: skip_species.in

```
version: 3.1
---

temporal:
  type: CYCLIC
  period: 720.0
skip_session:
  start_time: -572.5
  end_time: -39.5
  percent_mass_not_skipped: 99.9
species_to_keep:
  - CH4
  - OH
  - CO
  - CH2O
hc_species:
  CH4: 1.0
non_hc_species:
  N2: 1.0
```

Figure 25.72: Example *skip_species.in* file.

You can have multiple skip species sessions in a simulation. Figure 25.73 shows an example file that includes multiple skip species sessions (note that the *skip_session* settings block is repeated for each skip session).

```
version: 3.1
---

temporal:
  type: CYCLIC
  period: 720.0
skip_session:
  start_time: -572.5
  end_time: -39.5
  percent_mass_not_skipped: 99.9
skip_session:
  start_time: 50
  end_time: 75
  percent_mass_not_skipped: 99.9
species_to_keep:
  - CH4
  - OH
  - CO
  - CH2O
hc_species:
  CH4: 1.0
non_hc_species:
  N2: 1.0
```

Figure 25.73: Example *skip_species.in* file that includes two skip species sessions.

Table 25.35: Format of *skip_species.in*.

Parameter	Description
<i>temporal</i>	
<i>type</i>	Specify whether skip species is SEQUENTIAL or CYCLIC.
<i>period</i>	If <i>type</i> is CYCLIC, specify the period (in seconds if <i>inputs.in</i> > <i>simulation_control > crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).

Chapter 25: Input and Data Files

Simulation Parameters Skip Species: skip_species.in

Parameter	Description
<i>skip_session</i>	This settings block specifies the skip session parameters. Repeat this block for each skip session.
<i>start_time</i>	Skip species start time.
<i>end_time</i>	Skip species end time.
<i>percent_mass_not_skipped</i>	The percentage of total mass that will not be converted into mass of other species (see below).
<i>species_to_keep</i>	
<species name>	This species list allows you to force CONVERGE to keep certain species. In addition to the species listed here, skip species automatically keeps species in boundary conditions (specified in boundary.in) and region initialization (initialize.in). For an engine simulation, we recommend keeping the fuel, oxidizer, and some intermediate species.
<i>hc_species</i>	
<species name>	This species list specifies the hydrocarbon species to which skipped hydrocarbon species will be converted. CONVERGE will normalize the mass fractions if they do not sum to 1. If you do not list any hydrocarbon species, CONVERGE will convert the skipped hydrocarbon species to all of the non-skipped hydrocarbon species based on the mass fractions of the non-skipped species.
<i>non_hc_species</i>	
<species name>	This species list specifies the non-hydrocarbon species to which skipped non-hydrocarbon species will be converted. CONVERGE will normalize the mass fractions if they do not sum to 1. If you do not list any non-hydrocarbon species, CONVERGE will convert the skipped non-hydrocarbon species to all of the non-skipped non-hydrocarbon species based on the mass fractions of the non-skipped species.

Mixing Plane: [mixing_plane.in](#)

To enable a [mixing_plane](#), set [inputs.in](#) > *feature_control* > *mixing_plane_flag* = 1 and include a *mixing_plane.in* file in your case setup.

Chapter 25: Input and Data Files

Simulation Parameters Mixing Plane: mixing_plane.in

```
version: 3.1
---

mixing_planes:
  - mixing_plane:
      id: 0
      averaging:
        type: AREA_AVG
      velocity:
        type: MASSFLOW
        value: 4.87e-05
      direction:
        side_a: [0.5, 0, -0.866]
        side_b: [0.5, 0, -0.866]
      direction_transformation:
        type: CYLINDRICAL
        axial_origin: [0, 0, 0]
        axial_direction: [0, 0, 1]
```

Figure 25.74: Example *mixing_plane.in* file.

Chapter 25: Input and Data Files

Simulation Parameters Mixing Plane: mixing_plane.in

Table 25.36: Format of *mixing_plane.in*.

Parameter	Description
<i>mixing_planes</i>	
- <i>mixing_plane</i>	Repeat this sub-block for each mixing plane.
<i>id</i>	Mixing plane ID.
<i>averaging</i>	
<i>type</i>	Type of averaging. <i>AREA_AVG</i> = CONVERGE calculates area-weighted averages of flow variables at this mixing plane. <i>MASS_AVG</i> = CONVERGE calculates mass-weighted averages of flow variables at this mixing plane.
<i>velocity*</i>	Sub-block for velocity boundary condition.
<i>type</i>	Velocity boundary condition type. <i>MASSFLOW</i> = CONVERGE calculates the velocity at the mixing plane from a target mass flow rate.
<i>value</i>	Target mass flow rate (<i>kg/s</i>).
<i>direction</i>	
<i>side_a</i>	Direction of the velocity on side A of the mixing plane. Specify a vector in Cartesian or cylindrical coordinates, <i>NORM</i> , or <i>NEGATIVE_NORM</i> . <i>NORM</i> = Normal vector of the cell face on the mixing plane, <i>NEGATIVE_NORM</i> = $(-1)^{*}NORM$.
<i>side_b</i>	Direction of the velocity on side B of the mixing plane. Specify a vector in Cartesian or cylindrical coordinates, <i>NORM</i> , or <i>NEGATIVE_NORM</i> . <i>NORM</i> = Normal vector of the cell face on the mixing plane, <i>NEGATIVE_NORM</i> = $(-1)^{*}NORM$.
<i>direction_transformation*</i>	Direction transformation sub-block.
<i>type</i>	Coordinate system used to specify the velocity. <i>CARTESIAN</i> = Cartesian coordinates (default), <i>CYLINDRICAL</i> = Cylindrical coordinates.
<i>axial_origin</i>	For <i>type</i> = <i>CYLINDRICAL</i> , the x, y, and z coordinates of the cylindrical system origin.
<i>axial_direction</i>	For <i>type</i> = <i>CYLINDRICAL</i> , the x, y, and z coordinates of a vector that specifies the axis of the cylindrical system.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

25.5 Boundary Conditions

This section describes the input files related to boundaries and boundary conditions.

Boundary Conditions: boundary.in

The *boundary.in* file, which is required for all simulations, contains information about conditions for each boundary.

```
version: 3.1
---

rotation_axis: [1.0, 2.0, 3.0]

boundary_conditions:
  - boundary:
      id: 1
      type: WALL
      name: wall_bdry_example
      region: 0
      motion: TRANSLATING
      geometry_motion: MOVING
      roughness:
        height: 0
        constant: 0
      velocity:
        type: LAW_OF_WALL
        value: AUTO_GENERATE
        echo_profile: 0
      pressure:
        type: NEUMANN
      temperature:
        type: LAW_OF_WALL
        value: 450
        heat_model: 0
      species:
        type: NEUMANN
      passive:
        type: NEUMANN
      turbulence:
        tke:
          type: NEUMANN
          value: 0
        eps:
          type: LAW_OF_WALL
        near_wall_treatment: SCALABLE
      electric:
        type: DIRICHLET
        value: 1.0
  - boundary:
      id: 2
      type: OUTFLOW
      name: outflow_bdry_example
      region: 2
      velocity:
        type: NEUMANN
        value: [0, 0, 0]
      pressure:
        type: DIRICHLET
        value: 101325
        presdist: 0
      temperature:
        type: NEUMANN
        value: 0
      species:
        type: NEUMANN
      passive:
        type: NEUMANN
      turbulence:
        tke:
          type: NEUMANN
          value: 0
        eps:
```

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

```
        type:          NEUMANN
        value:          0
    backflow:
        temperature:
            type:          DIRICHLET
            value:          800
        species:
            type:          DIRICHLET
            value:
                CO2:          0.19235
                H2O:          0.08852500000000001
                N2:           0.71913
        passive:
            type:          DIRICHLET
            value:
                EXHAUST:      1
    turbulence:
        tke:
            type:          INTENSITY
            value:          0.02
        eps:
            type:          LENGTH_SCALE
            value:          0.003
- boundary:
    id:              3
    type:          INFLOW
    name:          inflow_bdry_example
    region:          1
    velocity:
        type:          NEUMANN
        value:          [0, 0, 0]
    pressure:
        type:          DIRICHLET_TOTAL
        value:          101325
    temperature:
        type:          DIRICHLET
        value:          363
    species:
        type:          DIRICHLET
        value:
            N2:           0.77
            O2:           0.23
    passive:
        type:          DIRICHLET
        value:
            INTAKE:      1
    turbulence:
        tke:
            type:          INTENSITY
            value:          0.01
        eps:
            type:          LENGTH_SCALE
            value:          0.005
- boundary:
    id:              4
    type:          TWO_D
    name:          two_d_bdry_example
    region:          0
- boundary:
    id:              5
    type:          PERIODIC
    name:          Front_face
    region:          0
    match:
        boundary_id:      3
        type:          ROTATING
        axis:          [0, 0, 1]
        angle:          60
- boundary:
    id:              6
    type:          INTERFACE
```

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

```
name: interface_bdry_example
disconnect: 0
forward:
    id: 7
    type: FLOW_THROUGH
    name: region_forward_example
    region: 1
reverse:
    id: 8
    type: FLOW_THROUGH
    name: region_reverse_example
    region: 0
- boundary:
    id: 9
    type: SYMMETRY
    name: symmetry_bdry_example
    region: 0
- boundary:
    id: 10
    type: GT-SUITE
    name: gt-suite_bdry_example
    region: 2
    gt_id: 7
    passive:
        type: NEUMANN
    turbulence:
        tke:
            type: INTENSITY
            value: 0.02
        eps:
            type: LENGTH_SCALE
            value: 0.003
    - boundary:
        id: 11
        type: WALL
        name: linked_bdry_example
        region: 0
        motion: LINKED
        roughness:
            height: 0
            constant: 0.5
        velocity:
            linked_id: 1
        pressure:
            type: NEUMANN
        temperature:
            type: FLUX
            value: 0
        species:
            type: NEUMANN
        passive:
            type: NEUMANN
        turbulence:
            tke:
                type: NEUMANN
                value: 0
            eps:
                type: LAW_OF_WALL
```

Figure 25.75: An example *boundary.in* file.

The information for each boundary consists of a boundary ID followed by boundary conditions for velocity, pressure, temperature, species mass fractions, [passive](#) values, turbulent kinetic energy, and turbulent dissipation, as well as a region identification number. For a description of the various boundary conditions included in CONVERGE, refer to [Chapter 9 - Boundaries and Boundary Conditions](#).

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Table 25.37: Format of the *rotation_axis* settings block. This block is optional.

Parameter	Description
<i>rotation_axis</i> *	The vector for the direction of the crank shaft. Used only in engine cases in which there are multiple cylinders with different axes of rotation.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Boundary information is contained within a single *boundary_conditions* block, with a separate *boundary* sub-block for each boundary. These sub-blocks need not be in any specific order. For clarity, the parameters for each type of boundary are tabulated separately below.

Table 25.38: Format of a *boundary* settings sub-block for an INFLOW boundary.

Parameter	Description	Typical value
<i>- boundary</i>		
<i>id</i>	The integer ID for this boundary.	
<i>type</i>	INFLOW = Inflow boundary type.	
<i>name</i> *	Name for this boundary.	
<i>region</i>	Region identification number to which this boundary is assigned.	
<i>fluctuation</i>	Optional sub-block for fluctuating inflow settings.	
<i>method</i> *	<i>OFF</i> = No fluctuations in the inflow, <i>DIGITAL_FILTER</i> = Digital filter method, <i>FOURIER</i> = Fourier method.	
<i>intensity</i>	Intensity of the inflow fluctuations as a fraction of the freestream velocity.	0 - 1
<i>lengthscale</i>	Length scale (<i>m</i>) of the inflow fluctuations.	
<i>direction</i>	<i>NORMAL</i> = Fluctuations are normal to the INFLOW boundary, <i>ALL</i> = Fluctuations are isotropic.	
<i>nscbc</i>	Optional sub-block for activating the Navier-Stokes characteristic boundary condition .	
<i>method</i>	<i>OFF</i> = No NSCBC scheme, <i>POINSOT_LELE</i> = Activate the Poinsot-Lele NSCBC inflow, <i>CORRECTION_BASED</i> = Activate the correction-based NSCBC inflow.	
<i>sigma</i>	Dimensionless NSCBC tuning parameter (σ). Used only when <i>nscbc</i> > <i>method</i> = <i>CORRECTION-BASED</i> .	0.25
<i>lengthscale</i>	NSCBC length scale (<i>L</i>). Used only when <i>nscbc</i> > <i>method</i> = <i>AUTO</i> = <i>CORRECTION-BASED</i> .	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	<i>AUTO</i> = Automatically calculate length scale based on size of domain, Any positive real number = Specified flow length scale (<i>m</i>).	
<i>reverse_flow_treatment</i> *	0 = No reverse flow treatment.	0
<i>velocity</i>		
<i>type</i>	NEUMANN = Specified zero gradient, DIRICHLET = Specified value, MASS_FLOW = Calculate velocity based on specified mass flow, NORMAL_NEUMANN = Specified normal gradient, AVERAGE_VELOCITY = Calculate velocity based on pressure, PUMP = Calculate velocity from a pump file (file name specified via <i>value</i> below).	
<i>value</i>	NEUMANN = Velocity gradient vector (must be 0, 0, 0), DIRICHLET = Velocity vector (m/s), MASS_FLOW = Boundary mass flow rate (kg/s), PUMP = File name (e.g., <i>pump_massflow.in</i>), AVERAGE_VELOCITY = Average velocity (m/s).	
	Not used for NORMAL_NEUMANN. For DIRICHLET and MASS_FLOW, you can set <i>value</i> = <i>user_<UDF name></i> to use the velocity or mass flow rate calculated by a user-defined function (UDF) based on the CONVERGE_PROFILE macro. Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>enforced_mass_flow</i> *	Boundary mass flow rate (kg/s). Used only when <i>value</i> is set to the file name (e.g., <i>velocity.in</i>) (above) of a spatially varying boundary condition. Not available for PUMP boundary conditions.	
	If <i>enforced_mass_flow</i> is set to a number, CONVERGE scales the velocities in the input file to achieve the specified mass flow rate. If <i>enforced_mass_flow</i> is set to <i>auto</i> , CONVERGE calculates the mass flow rate based on the velocities in the input file.	
<i>direction</i> *	Direction of the inflow for <i>type</i> = MASS_FLOW. For <i>direction_transformation</i> = CARTESIAN, x, y, and z components. For <i>direction_transformation</i> = CYLINDRICAL, radial, tangential, and axial components	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
<i>direction_transformation</i> *	Coordinate system for <i>type</i> = MASS_FLOW. If not specified, CONVERGE will use Cartesian coordinates.	
<i>type</i>	CARTESIAN (default), CYLINDRICAL.	
<i>axial_origin</i>	For <i>type</i> = CYLINDRICAL, the Cartesian x, y, and z coordinates of the origin of the cylindrical coordinate system.	
<i>axial_direction</i>	For <i>type</i> = CYLINDRICAL, the direction of the z axis of the cylindrical coordinate system expressed in Cartesian x, y, and z coordinates.	
<i>pressure</i>		
<i>type</i>	NEUMANN = Zero pressure gradient, DIRICHLET = Specified value of the static pressure, DIRICHLET_TOTAL = Specified value of the total pressure.	
<i>value</i>	0 (for <i>type</i> = NEUMANN), Static pressure (N/m^2) (for <i>type</i> = DIRICHLET), Total pressure (N/m^2) (for <i>type</i> = DIRICHLET_TOTAL), <i>user_<UDF name></i> = Pressure calculated by a user-defined function (UDF) based on the CONVERGE_PROFILE macro (for <i>type</i> = DIRICHLET or DIRICHLET_TOTAL). Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>reference_center</i>	Reference location (x, y, and z, in m) for the gravity reference location. Required if and only if the pressure condition is Dirichlet and gravity is specified.	
<i>temperature</i>		
<i>type</i>	NEUMANN = Zero temperature gradient, DIRICHLET = Specified temperature value.	
<i>value</i>	0 (for <i>type</i> = NEUMANN), Temperature (K) (for <i>type</i> = DIRICHLET), GT_SUITE = Use temperature from GT-Suite CHT co-simulation (for <i>type</i> = DIRICHLET), <i>user_<UDF name></i> = Use temperature calculated by a user-defined function (UDF) based on the CONVERGE_PROFILE macro (for <i>type</i> = DIRICHLET). Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>species</i>		
<i>type</i>	NEUMANN = Zero species gradient,	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	<i>DIRICHLET</i> = Specified species value.	
<i>value</i>	Used only for <i>type</i> = <i>DIRICHLET</i> . List < <i>species name</i> > below for each species.	
	To use species mass fractions calculated by a user-defined function (UDF) based on the <i>CONVERGE_PROFILE</i> macro, set <i>value</i> = <i>user_<UDF name></i> and do not list < <i>species name</i> > below. Refer to the CONVERGE 3.1 UDF Manual for more information.	
< <i>species name</i> >	Mass fraction of this species. Repeat for each species.	
<i>passive</i>		
<i>type</i>	<i>NEUMANN</i> = Zero passive gradient, <i>DIRICHLET</i> = Specified passive value.	
<i>value</i>	Used only for <i>type</i> = <i>DIRICHLET</i> . List < <i>passive name</i> > below for each species.	
	To use passive values calculated by a user-defined function (UDF) based on the <i>CONVERGE_PROFILE</i> macro, set <i>value</i> = <i>user_<UDF name></i> and do not list < <i>passive name</i> > below. Refer to the CONVERGE 3.1 UDF Manual for more information.	
< <i>passive name</i> >	Value of this passive. Repeat for each passive.	
<i>turbulence</i>		
<i>tke</i>	Used only for RANS turbulence models.	
<i>type</i>	<i>INTENSITY</i> = Specified turbulence intensity, <i>DIRICHLET</i> = Specified turbulent kinetic energy value.	
<i>value</i>	Turbulence intensity (for <i>type</i> = <i>INTENSITY</i>), Turbulent kinetic energy (m^2/s^2) (for <i>type</i> = <i>DIRICHLET</i>), <i>user_<UDF name></i> = Turbulent kinetic energy calculated by a user-defined function (UDF) based on the <i>CONVERGE_PROFILE</i> macro (for <i>type</i> = <i>DIRICHLET</i>). Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>eps</i>	Used only for RANS $k-\epsilon$ turbulence models.	
<i>type</i>	<i>LENGTH_SCALE</i> = Specified length scale for turbulent kinetic energy dissipation, <i>DIRICHLET</i> = Specified turbulent kinetic energy dissipation rate value.	
<i>value</i>	Length scale (m) (for <i>type</i> = <i>LENGTH_SCALE</i>),	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	Turbulent kinetic energy dissipation rate (m^2/s^3) (for <i>type</i> = <i>DIRICHLET</i>), <i>user_<UDF name></i> = Turbulent kinetic energy dissipation rate calculated by a user-defined function (UDF) based on the <i>CONVERGE_PROFILE</i> macro (for <i>type</i> = <i>DIRICHLET</i>). Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>omega</i>	Used only for RANS $k-\omega$ turbulence models.	
<i>type</i>	<i>LENGTH_SCALE</i> = Specified length scale for specific turbulent kinetic energy dissipation, <i>DIRICHLET</i> = Specified value for specific turbulent kinetic energy dissipation rate.	
<i>value</i>	Length scale (m) (for <i>type</i> = <i>LENGTH_SCALE</i>), Specific turbulent kinetic energy dissipation rate (s^{-1}) (for <i>type</i> = <i>DIRICHLET</i>), <i>user_<UDF name></i> = Specific turbulent kinetic energy dissipation rate calculated by a user-defined function (UDF) based on the <i>CONVERGE_PROFILE</i> macro (for <i>type</i> = <i>DIRICHLET</i>). Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>sgs_ke</i>	Used only for one- and two-equation LES turbulence models.	
<i>type</i>	<i>INTENSITY</i> = Specified turbulence intensity, <i>DIRICHLET</i> = Specified sub-grid kinetic energy value.	
<i>value</i>	Turbulence intensity (for <i>type</i> = <i>INTENSITY</i>), Sub-grid kinetic energy (m^2/s^2) (for <i>type</i> = <i>DIRICHLET</i>), <i>user_<UDF name></i> = Sub-grid kinetic energy calculated by a user-defined function (UDF) based on the <i>CONVERGE_PROFILE</i> macro (for <i>type</i> = <i>DIRICHLET</i>). Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>sgs_eps</i>	Used only for two-equation LES turbulence models.	
<i>type</i>	<i>LENGTH_SCALE</i> = Specified length scale for sub-grid kinetic energy dissipation, <i>DIRICHLET</i> = Specified sub-grid kinetic energy dissipation value.	
<i>value</i>	Length scale (m) (for <i>type</i> = <i>LENGTH_SCALE</i>), Sub-grid kinetic energy dissipation (m^2/s^3) (for <i>type</i> = <i>DIRICHLET</i>), <i>user_<UDF name></i> = Sub-grid kinetic energy dissipation calculated by a user-defined function (UDF) based on the <i>CONVERGE_PROFILE</i> macro (for <i>type</i> =	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
<i>DIRICHLET</i>). Refer to the CONVERGE 3.1 UDF Manual for more information.		

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Table 25.39: Format of a *boundary settings* sub-block for an OUTFLOW boundary.

Parameter	Description	Typical value
<i>- boundary</i>		
<i>id</i>	The integer ID for this boundary.	
<i>type</i>	<i>OUTFLOW</i> = Outflow boundary type.	
<i>name*</i>	Name for this boundary.	
<i>region</i>	Region identification number to which this boundary is assigned.	
<i>sponge</i>	Optional sub-block for sponge layer acoustic fluctuation damping.	
<i>center*</i>	Coordinates of the center of the sponge layer (<i>m</i>). Must be in the same plane as the OUTFLOW boundary	
<i>direction</i>	Direction vector for sponge layer growth. Must be orthogonal to the OUTFLOW plane and point into the domain from the <i>center</i> .	
<i>distance</i>	Length of the sponge layer (<i>m</i>).	
<i>nscbc</i>	Optional sub-block to activate the Navier-Stokes characteristic boundary condition .	
<i>method</i>	<i>OFF</i> = No NSCBC scheme, <i>POINSOT_LELE</i> = Activate the Poinsot-Lele NSCBC outflow, <i>CORRECTION_BASED</i> = Activate the correction-based NSCBC outflow.	
<i>sigma</i>	Dimensionless NSCBC tuning parameter (σ).	0.25
<i>lengthscale</i>	NSCBC length scale (<i>L</i>). <i>AUTO</i> = Automatically calculate length scale based on size of domain, Any positive real number = Specified flow length scale (<i>m</i>).	<i>AUTO</i>
<i>reverse_flow_treatment*</i>	0 = No reverse flow treatment (recommended when there is no reverse flow), 1 = Reverse flow target velocity is set to zero on this boundary (recommended when there is reverse flow),	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	2 = Reverse flow target velocity is set to the velocity of the cells adjacent to this boundary.	
<i>velocity</i>		
<i>type</i>	NEUMANN = Specified gradient, DIRICHLET = Specified value, MASS_FLOW = Calculate velocity based on specified mass flow, AVERAGE_VELOCITY = Calculate based on pressure,	
<i>value</i>	Velocity gradient vector (s^{-1}) (for <i>type</i> = NEUMANN), Velocity vector (m/s) (for <i>type</i> = DIRICHLET), Boundary mass flow rate (kg/s) (for <i>type</i> = MASS_FLOW), Average velocity (m/s) (for <i>type</i> = AVERAGE_VELOCITY). For <i>type</i> = DIRICHLET or MASS_FLOW, you can set <i>value</i> = user_<UDF name> to use the velocity or mass flow rate calculated by a user-defined function (UDF) based on the CONVERGE_PROFILE macro. Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>pressure</i>		
<i>type</i>	NEUMANN = Zero pressure gradient, DIRICHLET = Specified value of the static pressure, DIRICHLET_TOTAL = Specified value of the total pressure, TRANSONIC = Transonic pressure boundary condition. CONVERGE will adjust the outflow pressure to fall within prescribed Mach number limits.	
<i>value</i>	0 (for <i>type</i> = NEUMANN), Static pressure (N/m^2) (for <i>type</i> = DIRICHLET), Total pressure (N/m^2) (for <i>type</i> = DIRICHLET_TOTAL), Initial static pressure (N/m^2) (for <i>type</i> = TRANSONIC). For <i>type</i> = DIRICHLET or DIRICHLET_TOTAL, you can set <i>value</i> = user_<UDF name> to use the pressure calculated by a user-defined function (UDF) based on the CONVERGE_PROFILE macro. Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>reference_center</i>	Reference location (x, y, and z, in m) for the gravity reference location. Required if and only if the pressure condition is Dirichlet and gravity is specified.	
<i>presdist*</i>	Optional positive weighting value to dampen reflected acoustic waves (not available for <i>type</i> = NEUMANN).	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	Not recommended for general use.	
	For TRANSONIC, this value gives the distance to ambient pressure (in m).	
<i>min_mach</i> *	Minimum Mach number at the outflow. Used only when <i>type</i> = TRANSONIC.	
<i>max_mach</i> *	Maximum Mach number at the outflow. Used only when <i>type</i> = TRANSONIC.	
<i>temperature</i>		
<i>type</i>	NEUMANN = Zero temperature gradient.	
<i>value</i>	0	
<i>species</i>		
<i>type</i>	NEUMANN = Zero species gradient.	
<i>value</i>	0	
<i>passive</i>		
<i>type</i>	NEUMANN = Zero passive gradient.	
<i>value</i>	0	
<i>turbulence</i>		
<i>tke</i>	Used only for RANS turbulence models.	
<i>type</i>	NEUMANN = Zero turbulent kinetic energy gradient.	
<i>value</i>	0	
<i>eps</i>	Used only for RANS $k-\epsilon$ turbulence models.	
<i>type</i>	NEUMANN = Zero turbulent kinetic energy dissipation rate gradient.	
<i>value</i>	0	
<i>omega</i>	Used only for RANS $k-\omega$ turbulence models.	
<i>type</i>	NEUMANN = Zero specific turbulent kinetic energy dissipation rate gradient.	
<i>value</i>	0	
<i>sgs_ke</i>	Used only for one- and two-equation LES turbulence models.	
<i>type</i>	NEUMANN = Zero sub-grid kinetic energy gradient.	
<i>value</i>	0	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
<i>sgs_eps</i>	Used only for two-equation LES turbulence models.	
<i>type</i>	<i>NEUMANN</i> = Zero sub-grid kinetic energy dissipation gradient.	
<i>value</i>	0	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Table 25.40: Format of a *boundary settings* sub-block for a WALL boundary.

Parameter	Description	Typical value
<i>- boundary</i>		
<i>id</i>	The integer ID for this boundary.	
<i>type</i>	<i>WALL</i> = Wall boundary type.	
<i>name*</i>	Name for this boundary.	
<i>region</i>	Region identification number to which this boundary is assigned.	
<i>motion</i>	STATIONARY = No boundary motion, TRANSLATING = Translational motion, ROTATING = Rotational motion, ROTATING_AND_TRANSLATING = Rotational and translational motion, ARBITRARY = Arbitrary motion, DEPENDENT = Dependent motion, FSI = Motion based on fluid-structure interaction, LINKED = Motion linked to that of another boundary, ABAQUS = Motion linked to the Abaqus finite element solver (requires cosimulation.in), <i>SURFACE_LIST</i> = Motion calculated from a list of surface geometry files (requires surface_list.in), DEFORM = Motion based on a deforming inlaid mesh , <i>user_<UDF name></i> = Motion calculated by a user-defined function (UDF) based on the CONVERGE_MOTION macro. Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>geometry_motion</i>	If <i>motion</i> is STATIONARY, set <i>geometry_motion</i> to FIXED or RELOFT . If TRANSLATING: FIXED or MOVING, If ROTATING: FIXED or MOVING, If ROTATING_AND_TRANSLATING: MOVING, If ARBITRARY: MOVING, If DEPENDENT: MOVING or RELOFT , If FSI: MOVING,	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	If <i>USER: MOVING</i> , If <i>LINKED: MOVING</i> , If <i>ABAQUS: MOVING</i> , If <i>SURFACE_LIST: MOVING</i> , If <i>DEFORM: MOVING</i> , If <i>user_<UDF name></i> : <i>MOVING</i> .	
<i>roughness</i>		
<i>height</i>	Absolute surface roughness height (<i>m</i>). Specify a constant value (e.g., 0.0009) or a the name of a file containing a spatially or temporally varying profile (e.g., <i>roughness_height.in</i>).	
<i>constant</i>	Roughness constant.	0.5
<i>velocity</i>		
<i>type</i>	<i>LAW_OF_WALL</i> = Law-of-the-wall velocity, <i>SLIP</i> = Slip-wall velocity, <i>DIRICHLET</i> = Specified velocity, <i>TANGENTIAL</i> = Specified tangential velocity.	
<i>value</i>	Velocity vector (<i>m/s</i>), File name for a motion profile (e.g., <i>velocity.in</i>), <i>AUTO_GENERATE</i> = Generate a piston motion table using a crank-slider motion mechanism and data from engine.in (available only if <i>inputs.in > simulation_control > crank_flag = 1</i> and <i>boundary.in > boundary_conditions > boundary > motion = TRANSLATING</i>), <i>MOTION_CONFIG</i> = Generate motion based on a motion set configuration (requires motion_sets.in).	
<i>motion_name</i>	Specify the name of the motion set. This name must match motion_sets.in > motion_object > motion_name . Used only if <i>value = MOTION_CONFIG</i> .	
<i>echo_profile*</i>	Output the piston motion profile to piston_profile<boundary ID>.out .	
<i>phase_lag_angle*</i>	Phase lag angle for multi-cylinder cases.	
<i>inclination_angle*</i>	Inclination angle for multi-cylinder cases.	
<i>translate_axis</i>	Axis of translational motion (only for WALL <i>boundary.in > boundary_conditions > boundary > motion = TRANSLATING</i> or <i>ROTATING_AND_TRANSLATING</i>).	
<i>translate_speed</i>	Speed of translational motion (<i>m/s</i>) (only for WALL <i>boundary.in > boundary_conditions > boundary > motion = TRANSLATING</i> or <i>ROTATING_AND_TRANSLATING</i>).	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
<i>origin</i>	Origin of rotational motion (only for WALL <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = ROTATING or ROTATING_AND_TRANSLATING).	
<i>rotation_axis</i>	Axis of rotational motion (only for WALL <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = ROTATING or ROTATING_AND_TRANSLATING).	
<i>rotation_speed</i>	Speed of rotational motion (deg/s) (only for WALL <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = ROTATING or ROTATING_AND_TRANSLATING).	
<i>linked_id</i>	ID of the boundary from which this boundary inherits motion (only for WALL <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = LINKED).	
<i>pressure</i>		
<i>type</i>	NEUMANN = Zero pressure gradient.	
<i>temperature</i>		
<i>type</i>	LAW_OF_WALL = Law-of-the-wall temperature, DIRICHLET = Specified temperature value, NEUMANN = Zero temperature gradient, FLUX = Specified heat flux, CONVECTION = Convection, RADIATION_CONVECTION = Radiation and convection.	
<i>value</i>	Temperature (K) (for <i>type</i> = LAW_OF_WALL or DIRICHLET), 0 (for <i>type</i> = NEUMANN), Wall heat flux (W/m ²) (for <i>type</i> = FLUX). Not used for <i>type</i> = CONVECTION or RADIATION_CONVECTION. For <i>type</i> = LAW_OF_WALL or DIRICHLET, set this parameter to CHT1D to perform 1D CHT modeling (requires cht1d.in), or GT-SUITE to read temperature data from GT-SUITE (requires cosimulation.in).	
	For <i>type</i> = LAW_OF_WALL or DIRICHLET, set <i>value</i> = <i>user_<UDF name></i> to use the temperature calculated by a user-defined function (UDF) based on the CONVERGE_PROFILE macro. Refer to the CONVERGE 3.1 UDF Manual for more information.	
<i>heat_model*</i>	For <i>type</i> = LAW_OF_WALL, 0 = O'Rourke and Amsden, 1 = Han and Reitz, 2 = Angelberger,	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	3 = GruMo-UniMORE, 4 = Jayatilleke.	
<i>reference_temperature</i>	For <i>type</i> = CONVECTION or RADIATION_CONVECTION, the reference far-field temperature (K).	
<i>htc</i>	For <i>type</i> = CONVECTION or RADIATION_CONVECTION, the boundary heat transfer coefficient (W/m ² -K).	
<i>radiation_temperature</i>	For <i>type</i> = RADIATION_CONVECTION, the radiation source temperature (K).	
<i>emissivity</i>	For <i>type</i> = RADIATION_CONVECTION, the surface emissivity.	0 - 1
<i>contact_resistance</i> *	Thermal contact resistance (K-m ² /W).	
<i>species</i>		
<i>type</i>	NEUMANN = Zero species gradient.	
<i>passive</i>		
<i>type</i>	NEUMANN = Zero passive gradient.	
<i>turbulence</i>		
<i>tke</i>	Used only for RANS turbulence models.	
<i>type</i>	NEUMANN = Zero turbulent kinetic energy gradient, DIRICHLET = Specified turbulent kinetic energy value, LAW_OF_WALL = Use a wall function to calculate turbulent kinetic energy (not recommended when turbulence.in > Wall_modeling > near_wall_treatment = STANDARD; recommended to use turbulence.in > turbulence_model = RANS_K_EPS_*, RANS_K_OMEGA_*, DDES_K_OMEGA_SST, or IDDES_K_OMEGA_SST; requires parameters to be set in turbulence.in > Wall_modeling).	
<i>value</i>	0 (for <i>type</i> = NEUMANN), Turbulent kinetic energy (m ² /s ²) (for <i>type</i> = DIRICHLET).	
<i>eps</i>	Used only for RANS $k-\epsilon$ turbulence models.	
<i>type</i>	LAW_OF_WALL = Use a wall function to calculate kinetic energy dissipation rate, NEUMANN = Zero turbulent kinetic energy dissipation rate gradient.	
<i>value</i>	0 (for <i>type</i> = NEUMANN).	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	Not used for <i>type</i> = <i>LAW_OF_WALL</i> .	
<i>omega</i>	Used only for RANS $k-\omega$ turbulence models.	
<i>type</i>	<i>LAW_OF_WALL</i> = Use a wall function to calculate specific turbulent kinetic energy dissipation rate, <i>NEUMANN</i> = Zero specific turbulent kinetic energy dissipation rate gradient.	
<i>value</i>	0 (for <i>type</i> = <i>NEUMANN</i>).	
	Not used for <i>type</i> = <i>LAW_OF_WALL</i> .	
<i>sgs_ke</i>	Used only for one- and two-equation LES turbulence models.	
<i>type</i>	<i>NEUMANN</i> = Zero sub-grid kinetic energy gradient, <i>LAW_OF_WALL</i> = Use a wall function to calculate sub-grid kinetic energy.	
<i>value</i>	0 (for <i>type</i> = <i>NEUMANN</i>), Sub-grid kinetic energy (m^2/s^2) (for <i>type</i> = <i>DIRICHLET</i>).	
<i>sgs_eps</i>	Used only for two-equation LES turbulence models.	
<i>type</i>	<i>LAW_OF_WALL</i> = Use a wall function to calculate sub-grid kinetic energy dissipation rate, <i>NEUMANN</i> = Zero sub-grid kinetic energy dissipation rate gradient.	
<i>value</i>	0 (for <i>type</i> = <i>NEUMANN</i>).	
	Not used for <i>type</i> = <i>LAW_OF_WALL</i> .	
<i>near_wall_treatment</i> *	For <i>type</i> = <i>LAW_OF_WALL</i> , <i>STANDARD</i> = Standard wall treatment (not recommended if <i>boundary.in</i> > <i>turbulence</i> > <i>tke</i> > <i>type</i> = <i>LAW_OF_WALL</i>), <i>SCALABLE</i> = Scalable wall treatment, <i>NON_EQUILIBRIUM</i> = Non-equilibrium wall treatment, <i>ENHANCED</i> = Enhanced wall treatment, <i>ANALYTIC</i> = Analytic wall treatment.	
<i>electric</i>		
<i>type</i>	<i>DIRICHLET</i> = Specified voltage, <i>FLUX</i> = Specified current per unit area, <i>NEUMANN</i> = Zero voltage gradient. Note: <i>type</i> = <i>CURRENT</i> cannot currently be configured in CONVERGE Studio. You must configure it manually.	
<i>value</i>	Voltage (V) (for <i>type</i> = <i>DIRICHLET</i>), Current per unit area (A/m^2) (for <i>type</i> = <i>FLUX</i>),	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description	Typical value
	0 (for <i>type</i> = NEUMANN), COUPLED (for a coupled solid-solid INTERFACE boundary with <i>type</i> = DIRICHLET).	
<i>contact_resistance</i> *	Electrical contact resistance ($\Omega \cdot m^2$). Used only when <i>value</i> = COUPLED.	
<i>cht_resolution_level</i>	Resolution of conjugate heat transfer (CHT) interpolation points. Required only if 1D or 3D CHT is enabled. Set to AUTO or specify an integer value.	AUTO
<i>torque</i> *		
<i>center</i>	Origin used in torque calculations.	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Table 25.41: Format of a *boundary* settings sub-block for an INTERFACE boundary.

Parameter	Description
- <i>boundary</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	INTERFACE = Interface boundary type.
<i>name</i> *	Name for this boundary.
<i>region</i>	Region identification number to which this boundary is assigned.
<i>motion</i>	STATIONARY = No boundary motion, TRANSLATING = Translational motion, ROTATING = Rotational motion, ROTATING_AND_TRANSLATING = Rotational and translational motion, ARBITRARY = Arbitrary motion, DEPENDENT = Dependent motion, FSI = Motion based on fluid-structure interaction, LINKED = Motion linked to that of another boundary, ABAQUS = Motion linked to the Abaqus finite element solver (requires cosimulation.in), SURFACE_LIST = Motion calculated from a list of surface geometry files (requires surface.list.in), DEFORM = Motion based on a deforming inlaid mesh .
<i>geometry_motion</i>	If <i>motion</i> is STATIONARY, set <i>geometry_motion</i> to FIXED or RELOFT . If TRANSLATING: FIXED or MOVING, If ROTATING: FIXED or MOVING, If ROTATING_AND_TRANSLATING: MOVING, If ARBITRARY: MOVING, If DEPENDENT: MOVING or RELOFT , If FSI: MOVING, If USER: MOVING,

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description
	If LINKED: MOVING, If ABAQUS: MOVING, If SURFACE_LIST: MOVING, If DEFORM: MOVING.
<i>velocity</i>	
<i>value</i>	Velocity vector (m/s), File name for a motion profile (e.g., <i>velocity.in</i>), AUTO_GENERATE = Generate a piston motion table using a crank-slider motion mechanism and data from engine.in (available only if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 1 and <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = TRANSLATING), MOTION_CONFIG = Generate motion based on a motion set configuration (requires motion_sets.in).
<i>motion_name</i>	Specify the name of the motion set. This name must match motion_sets.in > <i>motion_object</i> > <i>motion_name</i> . Used only if <i>value</i> = MOTION_CONFIG above.
<i>translate_axis</i>	Axis of translational motion (only for INTERFACE <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = TRANSLATING or ROTATING_AND_TRANSLATING).
<i>translate_speed</i>	Speed of translational motion (m/s) (only for INTERFACE <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = TRANSLATING or ROTATING_AND_TRANSLATING).
<i>origin</i>	Origin of rotational motion (only for INTERFACE <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = ROTATING or ROTATING_AND_TRANSLATING).
<i>rotation_axis</i>	Axis of rotational motion (only for INTERFACE <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = ROTATING or ROTATING_AND_TRANSLATING).
<i>rotation_speed</i>	Speed of rotational motion (deg/s) (only for INTERFACE <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = ROTATING or ROTATING_AND_TRANSLATING).
<i>linked_id</i>	ID of the boundary from which this boundary inherits motion (only for INTERFACE <i>boundary.in</i> > <i>boundary_conditions</i> > <i>boundary</i> > <i>motion</i> = LINKED).
<i>forward</i>	In the <i>forward</i> sub-block, enter information including the <i>id</i> , <i>type</i> , <i>name</i> , and <i>region</i> of the forward boundary.
<i>reverse</i>	In the <i>reverse</i> sub-block, enter information including the <i>id</i> , <i>type</i> , <i>name</i> , and <i>region</i> of the reverse boundary.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

A flow-through INTERFACE boundary is a special type of an INTERFACE boundary.

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Table 25.42: Format of a *boundary settings* sub-block for a flow-through INTERFACE boundary.

Parameter	Description
<i>- boundary</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	INTERFACE = Interface boundary type.
<i>name*</i>	Name for this boundary.
<i>region</i>	Region identification number to which this boundary is assigned.
<i>motion</i>	STATIONARY = No boundary motion, TRANSLATING = Translational motion, ROTATING = Rotational motion, ROTATING_AND_TRANSLATING = Rotational and translational motion, ARBITRARY = Arbitrary motion, DEPENDENT = Dependent motion, FSI = Motion based on fluid-structure interaction, LINKED = Motion linked to that of another boundary, ABAQUS = Motion linked to the Abaqus finite element solver (requires cosimulation.in), SURFACE_LIST = Motion calculated from a list of surface geometry files (requires surface_list.in), DEFORM = Motion based on a deforming inlaid mesh .
<i>geometry_motion</i>	If <i>motion</i> is STATIONARY, set <i>geometry_motion</i> to FIXED. If TRANSLATING: FIXED or MOVING, If ROTATING: FIXED or MOVING, If ROTATING_AND_TRANSLATING: MOVING, If ARBITRARY: MOVING, If DEPENDENT: MOVING, If FSI: MOVING, If USER: MOVING, If LINKED: MOVING, If ABAQUS: MOVING. If SURFACE_LIST: MOVING, If DEFORM: MOVING.
<i>disconnect</i>	0 = The INTERFACE triangles cannot be associated with events in events.in , 1 = The INTERFACE triangles can be associated with events in events.in to control the flow through the interface.
<i>forward</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	FLOW_THROUGH = Flow through boundary option.
<i>name</i>	Name for this boundary.
<i>region</i>	Region identification number on the forward side of the FLOW_THROUGH boundary.
<i>reverse</i>	

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description
<i>id</i>	The integer ID for this boundary.
<i>type</i>	<i>FLOW_THROUGH</i> = Flow through boundary option.
<i>name</i>	Name for this boundary.
<i>region</i>	Region identification number on the reverse side of the <i>FLOW_THROUGH</i> boundary.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

A seal INTERFACE boundary is a special type of an INTERFACE boundary.

Table 25.42: Format of a *boundary settings* sub-block for a seal INTERFACE boundary.

Parameter	Description
- <i>boundary</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	INTERFACE = Interface boundary type.
<i>name*</i>	Name for this boundary.
<i>seal_to_boundary</i>	ANY = Seal to any surface, [List of boundaries] = List of boundaries to seal to.
<i>partial_seal</i>	NO = Remove seals that have open edges after their intersections have been resolved, YES = Keep seals that have open edges after their intersections have been resolved.
<i>keep_flag</i>	BOTH = Keep both sides of the seal, FORWARD = Only keep the forward side of the seal, REVERSE = Only keep the reverse side of the seal, NEITHER = Keep neither side.
- <i>forward</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	SEAL = Sealing boundary option.
<i>region</i>	Region identification number on the forward side of the SEAL boundary.
<i>name*</i>	Name for this boundary.
<i>reference_wall</i>	The integer ID of the WALL boundary from which boundary conditions and motion are taken for the SEAL boundary (<i>i.e.</i> , the seal origin).
- <i>reverse</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	SEAL = Sealing boundary option.
<i>region</i>	Region identification number on the reverse side of the SEAL boundary.

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Parameter	Description
<i>name</i> *	Name for this boundary.
<i>reference_wall</i>	The integer ID of the WALL boundary from which boundary conditions and motion are taken for the SEAL boundary (<i>i.e.</i> , the seal origin).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Table 25.43: Format of a *boundary* settings sub-block for a SYMMETRY boundary.

Parameter	Description
- <i>boundary</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	SYMMETRY = Symmetry boundary type.
<i>name</i>	Name for this boundary.
<i>region</i>	Region identification number to which this boundary is assigned.

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Table 25.44: Format of a *boundary settings* sub-block for a GT-SUITE boundary.

Parameter	Description
- <i>boundary</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	GT-SUITE = GT-SUITE boundary type.
<i>name*</i>	Name for this boundary.
<i>region</i>	Region identification number to which this boundary is assigned.
<i>gt_id</i>	Boundary ID for the GT-SUITE component that corresponds to this CONVERGE boundary.
<i>GT_velocity_profile*</i>	File name for a spatially-varying velocity profile.
<i>GT_pressure_profile*</i>	File name for a spatially-varying pressure profile.
<i>passive</i>	
<i>type</i>	NEUMANN = Zero species gradient, DIRICHLET = Specified species value.
<i>value</i>	Repeat for each passive (only for <i>type</i> = DIRICHLET).
< <i>passive name</i> >	Value of this passive.
<i>turbulence</i>	
<i>tke</i>	
<i>type</i>	INTENSITY = Specify turbulent kinetic energy as an intensity fraction, DIRICHLET = Specify turbulent kinetic energy directly.
<i>value</i>	Intensity fraction (for <i>type</i> = INTENSITY), Turbulent kinetic energy (m^2/s^2) (for <i>type</i> = DIRICHLET).
<i>eps</i>	
<i>type</i>	LENGTH_SCALE = Specify the length scale of the turbulence, DIRICHLET = Specify the turbulent kinetic energy dissipation rate directly.
<i>value</i>	Length scale of the turbulence (m) (for <i>type</i> = LENGTH_SCALE), Turbulent kinetic energy dissipation rate (m^2/s^3) (for <i>type</i> = DIRICHLET).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Table 25.45: Format of a *boundary settings* sub-block for a PERIODIC boundary. Note that each PERIODIC boundary must have one unique matching PERIODIC boundary in the domain.

Parameter	Description
<i>- boundary</i>	
<i>id</i>	The integer ID for this boundary.
<i>type*</i>	PERIODIC = Periodic boundary type.
<i>name</i>	Name for this boundary.
<i>region</i>	Region identification number to which this boundary is assigned.
<i>match</i>	
<i>boundary_id</i>	Boundary identification number for the periodic matching boundary.
<i>type*</i>	ROTATING = Rotational periodicity. TRANSLATING = Translational periodicity.
<i>axis</i>	For ROTATING periodic boundary, the axis of rotation. CONVERGE determines the axis vector from the right-hand rule.
<i>origin</i>	For ROTATING periodic boundary, a point on the axis of rotation to define the location of the axis.
<i>angle</i>	For ROTATING periodic boundary, the angle (<i>deg</i>) of periodicity. CONVERGE determines the sign of the angle of rotation from the right-hand rule.
<i>direction</i>	For TRANSLATING periodic boundary, the direction and distance vector of periodicity (e.g., if the distance between the two faces in the x direction is 5 mm, the vector should be [5e-3, 0, 0]).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Table 25.46: Format of a *boundary settings* sub-block for a TWO-D boundary.

Parameter	Description
<i>- boundary</i>	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	TWO_D = Two-dimensional boundary type.
<i>name</i>	Name for this boundary.
<i>region</i>	Region identification number to which this boundary is assigned.
<i>geometry_motion*</i>	RELOFT (applies only to reloft boundaries).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Chapter 25: Input and Data Files

Boundary Conditions Boundary Conditions: boundary.in

Table 25.47: Format of a *boundary settings* sub-block for a MIXING_PLANE boundary.

Parameter	Description
- boundary	
<i>id</i>	The integer ID for this boundary.
<i>type</i>	MIXING_PLANE = Mixing_plane boundary type.
<i>name</i>	Name for this boundary.
<i>region</i>	Region ID to which this boundary is assigned.
<i>mixing_plane</i>	
<i>id</i>	Mixing plane ID assigned to this boundary.
<i>side</i>	Mixing plane side assigned to this boundary (<i>Side_A</i> or <i>Side_B</i>).

Arbitrary Motion Profile

To set up a WALL boundary with arbitrary motion, set [boundary.in](#) > *boundary_conditions* > *boundary* > *motion* = ARBITRARY and *boundary_conditions* > *boundary* > *geometry_motion* = MOVING. Specify a file name for the arbitrary motion profile via [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value* (e.g., *arbitrary_motion.in*).

For the arbitrary motion profile, the first row must consist of the word TEMPORAL. The second row should consist of SEQUENTIAL if the data are non-repeating or CYCLIC followed by the period in seconds (if [inputs.in](#) > *simulation_control* > *crank_flag* = 0) or crank angle degrees (if *crank_flag* is non-zero). If the data are sequential, CONVERGE uses the first or last value when the simulation time is outside the range of specified data.

The third row should consist of the word *second* (if [inputs.in](#) > *simulation_control* > *crank_flag* = 0) or *crank* (if *crank_flag* is non-zero), followed by *x*, *y*, *z*, *v1x*, *v1y*, *v1z*, *v2x*, *v2y* and *v2z*. The fourth row should consist of the word *supplied_position* and then specify xyz coordinates that give the temporally varying origin of the local coordinate system. CONVERGE will use the coordinate system origin as a reference to move the boundary. After the origin coordinates, specify two sets of xyz vectors. The first vector (*v1x*, *v1y*, *v1z*) define the initial direction of the x axis for the local coordinate system. The second vector (*v2x*, *v2y*, *v2z*) defines the initial direction of the y axis for the local coordinate system. The orientation vectors must be orthogonal to each other. The location and vectors you specify in the *supplied_position* row correspond to the position and orientation of the moving WALL with arbitrary motion in the [surface geometry file](#).

The fifth and subsequent rows should each contain a value of time (in seconds if [inputs.in](#) > *simulation_control* > *crank_flag* = 0 or in crank angle degrees if *crank_flag* is non-zero), followed by the location of the origin of the local coordinate system and two orthogonal vectors to define the orientation of the boundary at that time.

Chapter 25: Input and Data Files

Boundary Conditions Arbitrary Motion Profile

TEMPORAL	720								
CYCLIC	x	y	z	v1x	v1y	v1z	v2x	v2y	v2z
crank	0.25	1.40	1.21	1.0	0.0	0.0	0.0	0.7	-0.7
supplied_position	0.0e-05	0.25	1.40	1.20	1.0	0.0	0.0	0.7	-0.7
	2.3e-05	-0.99	0.87	0.00	0.99	-2.9	0.0	2.9	0.0
	4.6e-05	-0.98	1.73	0.00	0.99	-5.81	0.0	5.81	0.0
	6.9e-05	-0.96	2.58	0.00	0.99	-8.71	0.0	8.71	0.0

Figure 25.76: Excerpt of an arbitrary motion profile (e.g., *arbitrary_motion.in*) for a moving WALL boundary.

Valve Motion Profile

You can specify valve motion via a valve lift profile. CONVERGE can automatically create OPEN and CLOSE [events](#) between regions separated by valves based on the valve lift profile you provide. To invoke a valve motion profile, specify the valve motion file name (e.g., *intake_lift.in*) for [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value*.

To set up a valve lift file, type *TEMPORAL* in the first row and *valve* in the second row.

In the third row, input the motion type. CONVERGE offers four valve motion types, one rotational and three translational. For rotational motion, type *rotate* followed by a vector and a rotation angle to define the angle of inclination of the valve motion relative to the cylinder axis. Note that the right-hand rule is used when determining the direction of the rotation about the axis. For translational motion, type *direction* and one of the following. First, you can directly specify a vector along the direction of motion of the valve as measured when the cylinder is aligned to the z axis. Second, you can specify *BOUND_ID* followed by a boundary ID. The valve motion will be along the average normal of that boundary ID (e.g., the bottom of a valve). Third, you can specify *AUTO*. CONVERGE will search for the interface between the moving boundary and an adjacent stationary boundary, then use the normal of a plane defined by three points on this interface.

In the fourth row, enter *min_lift* followed by the minimum lift (in m).

In the fifth row, enter *SEQUENTIAL* if the data are non-repeating or *CYCLIC* followed by the period in seconds (if [inputs.in](#) > *simulation_control* > *crank_flag* = 0) or *crank angle degrees* (if *crank_flag* is non-zero). In the sixth row, enter *second* (if [inputs.in](#) > *simulation_control* > *crank_flag* = 0) or *crank* (if *crank_flag* is non-zero), followed by *lift*. The first column specifies the simulation time and the second column specifies the valve lift distance in the direction specified in the third row. Note that the lift values specified in the valve lift file are relative to the fully seated (*i.e.*, zero-lift) position, not relative to the minimum lift position. CONVERGE internally reduces the distance it moves the valve by the minimum lift distance to account for the fact that the valve is already at 0.0002 m lift in the surface geometry. Do not correct the lift data to account for the specified minimum lift value.

If the data are sequential, CONVERGE uses the first or last value when the simulation time is outside the range of specified data. For both sequential and cyclic data, CONVERGE interpolates the data you specify to obtain values of the variables at times between entries.

Chapter 25: Input and Data Files

Boundary Conditions Valve Motion Profile

```
TEMPORAL
valve
direction -3.08949E-1 0.0 -9.510785E-1
min_lift 0.0002
CYCLIC 720
crank lift
0 0
350 0
353 0.000036
.
.
605 0.00003
608 0.00002
611 0.00002
614 0
617 0
720 0
```

Figure 25.77: Excerpt of a valve motion file (e.g., *intake_lift.in*).

Note that when you specify the valve motion direction in a V engine or other simulation in which the cylinder is inclined relative to the z axis, CONVERGE rotates the valve motion direction vector (or angle) specified for that piston's velocity boundary condition in [boundary.in](#). If you have already measured the valve motion direction with the cylinder aligned with the z axis, you do not need to calculate the new direction vector (or angle) of the valve in the V engine configuration.

You can use the valve lift profile to run a set of flowbench cases for variable valve lifts with the steady-state solver, altering your case setup only through the valve lift profile. CONVERGE will move the valve to the appropriate lift before beginning iterations. Specify four points in the valve lift profile as shown in the example file below. Ensure that the first time is negative, the second time is 0, and the third and fourth times are very large. Figure 25.78 below gives an excerpt of a sample motion file for a steady-state flowbench case.

```
TEMPORAL
valve
direction 0 0 1
min_lift 2e-4
SEQUENTIAL
second lift
-1.0 0.0
0.0 0.006
9e4 0.006
1e5 0.0
```

Figure 25.78: An example *valve_lift.in* for a steady-state flowbench case.

To change the surface geometry from a straight engine (piston moves along the z axis) to a V engine (pistons are inclined relative to the z axis), specify a *rotation_axis* in [boundary.in](#).

Boundary Motion Profile

A translating or rotating WALL boundary can be governed by a motion profile. To specify a motion profile, specify a file name (e.g., *velocity.in*) for [boundary.in](#) > *boundary_conditions* > *boundary* > *velocity* > *value*.

Chapter 25: Input and Data Files

Boundary Conditions Boundary Motion Profile

To create a motion file, type TEMPORAL in the first row. In the second row, enter SEQUENTIAL if the data are non-repeating or CYCLIC followed by the period in seconds (if [inputs.in](#) > simulation_control > crank_flag = 0) or crank angle degrees (if crank_flag is non-zero).

In the third row, enter second (if [inputs.in](#) > simulation_control > crank_flag = 0) or crank (if crank_flag is non-zero). For a translating boundary, follow this with x, y, and z. The first column specifies the simulation time and the next three columns specify the boundary displacement (in m) in the x, y, and z directions. For a rotating boundary, follow this with angle. This specifies the rotation angle in degrees.

If the data are sequential, CONVERGE uses the first or last value when the simulation time is outside the range of specified data. For both sequential and cyclic data, CONVERGE interpolates the data you specify to obtain values of the variables at times between entries.

Figure 25.79 below gives an excerpt of a sample motion file. In this example, the motion is for a piston boundary in a single-cylinder engine model. Therefore, the surface file has the piston in its BDC location and the piston motion is along the z axis. The z column values are all positive, indicating upward displacement of the piston towards top dead center (TDC) position. The values in the x and y columns are zeros because the piston is moving along the z axis.

```
TEMPORAL
CYCLIC    720
crank      x      y      z
-360       0      0     0.106040316
-359.5     0      0     0.10602925
-359       0      0     0.106014713
-358.5     0      0     0.105996867
-358       0      0     0.105975426
```

Figure 25.79: Excerpt of a sample motion file specifying translating piston motion along the z axis.

CONVERGE supports translating and rotating boundary motion for steady-state simulations. CONVERGE will translate or rotate the boundary to the specified position or angle before the beginning of the simulation. This allows you to alter some aspects of your case setup (e.g., rotating a throttle body plate to a different position) without altering *surface.dat*. Figure 25.80 below gives an excerpt of a sample rotation file for a steady-state case.

```
TEMPORAL
SEQUENTIAL
second      Angle
0.0         10.0
```

Figure 25.79: Excerpt of a sample motion file specifying a rotation for a steady-state case.

Pump Mass Flow: `pump_massflow.in`

To set up a pump velocity boundary condition, set [boundary.in](#) > boundary_conditions > boundary > velocity > type = PUMP and specify a file name for boundary_conditions > boundary > velocity > value (e.g., *pump_massflow.in*). This file specifies a series of pressures (in Pa) and corresponding mass flow rates (in kg/s).

Chapter 25: Input and Data Files

Boundary Conditions Pump Mass Flow: *pump_massflow.in*

```
TEMPORAL
SEQUENTIAL
pressure    pump_massflow
92220.0     0.11986
92726.0     0.11317
93480.0     0.103326
94830.0     0.08555
96490.0     0.06357
98771.0     0.033416
100403.0    0.011857
101300.0    0.0
```

Figure 25.80: Example of a pump velocity boundary condition file (e.g., *pump_massflow.in*).

Table 25.48: Format of *pump_massflow.in*.

Column	Parameter	Units
1	Pressure	Pa
2	Mass flow rate	kg/s

Wall Values: *wall_value.in*

The *wall_value.in* file defines custom quantities related to wall boundaries. You can specify spatially or temporally varying data for wall boundary values. CONVERGE does not use wall boundary data for any models available by default, but it is available for use in UDFs. In the sample shown below in Figure 25.81, a file name is specified for the thickness for Boundary 1 (*bound_id* = 1). This *wall_value.in* file, which contains wall thickness data versus time or space, must be saved in your case setup.

```
version: 3.1
---

wall_value_names:
  - thickness
  - radius

walls:
  - wall:
      boundary_id: 2
      use_file: 0
      values:
        thickness: 0.2
        radius: 3.0e-4

  - wall:
      boundary_id: 1
      use_file: 1
      values_file: wall_thick.in
```

Figure 25.81: An example *wall_value.in* file.

The *wall_value_names* setting block lists the names of the labels for wall boundary data. CONVERGE will expect these labels for each wall boundary specified in the *walls > wall > values* setting sub-block or in the *walls > wall > values_file* file. Typical labels include *thickness* and *radius*.

Chapter 25: Input and Data Files

Boundary Conditions Wall Values: wall_value.in

The *walls* setting block is used to specify these custom values for each wall.

Table 25.49: Format of the *walls* settings block.

Parameter	Description
<i>walls</i>	
- <i>wall</i>	This settings sub-block specifies custom quantities for a wall boundary. Repeat this sub-block through <i>values</i> (if <i>use_file</i> = 0) or <i>values_file</i> (if <i>use_file</i> = 1) for each wall for which you want to specify custom values.
<i>boundary_id</i>	The boundary ID of the WALL boundary for which values are specified.
<i>use_file</i>	0 = Specify values in <i>wall_values.in</i> with labels listed in <i>wall_value_names</i> setting block. 1 = Specify the name of the file containing wall values in <i>values_file</i> .
<i>values</i>	Specify values for each label listed in the <i>wall_values_name</i> setting block. Required only if <i>use_file</i> = 0 for this <i>boundary_id</i> .
<i>values_file</i>	Specify the name of the file containing wall values. Required only if <i>use_file</i> = 1 for this <i>boundary_id</i> .

Co-simulation: cosimulation.in

To perform a co-simulation that couples CONVERGE with another solver (GT-SUITE or Abaqus), set *inputs.in* > *feature_control* > *cosimulation_flag* = 1 and include a *cosimulation.in* file in your case setup. You must also specify the appropriate option(s) in *boundary.in*.

Chapter 25: Input and Data Files

Boundary Conditions Co-simulation: cosimulation.in

```
version: 3.1
---

GT-SUITE:
  code_name: GTPOWER
  version: v2016
  scheme: GT LEAD
  averaging_method: REGION_AUTO
  model_file: EXAMPLE_MODEL_FILENAME.gtm

ABAQUS:
  reference_pressure: 101325.0
  time_shift: 0.0
  transfer_pressure_flag: 0
  transfer_stress_flag: 0
  interpolation_method: CENTROID_VALUE
  reverse_normals_flag: 1
  host_name: localhost:6660
  cosimulation_objects:
    - cosimulation_object:
        object_name: Cyl1
        object_type: SURFACE
        boundary_id: [1]
        incoming_field: DISPLACEMENT
        outgoing_field: TRACTION_VECTOR
        events:
          - event:
              event_flag: LOCATION
              region_id_a: 0
              region_id_b: 1
              deflection_open: 2e-6
              deflection_close: 1e-6
              location: [7.6e-2, 8.43e-1, 2.93e-1]
              direction: [0, 0, 1]
    - cosimulation_object:
        object_name: Cyl2
        object_type: SURFACE
        boundary_id: [2]
        incoming_field: DISPLACEMENT
        outgoing_field: TRACTION_VECTOR
```

Figure 25.82: An example *cosimulation.in* file.

Chapter 25: Input and Data Files

Boundary Conditions Co-simulation: cosimulation.in

Table 25.50: Format of the GT-SUITE settings block (required only for CONVERGE + GT-SUITE co-simulation).

Parameter	Description	Typical value
GT-SUITE		
<i>code_name</i>	Name of the GT-SUITE executable (e.g., <i>GTPOWER</i> , <i>GTSUITE</i> , or <i>GTSUITEmp</i>) that couples with CONVERGE.	
<i>version</i>	Version number for GT-SUITE that enables CONVERGE to find the server files of GT-SUITE (GTLINK). CONVERGE 3.1 is compatible with GT-SUITE versions 7.3, 7.4, 7.5, 2016, and newer.	
<i>scheme</i>	Scheme type. <i>CONVERGE_LEAD</i> = At each time-step, CONVERGE and then GT-SUITE advances in time (deprecated), <i>GT_LEAD</i> = At each time-step, GT-SUITE and then CONVERGE advances in time (recommended).	<i>GT_LEAD</i>
<i>averaging_method</i>	Averaging method of calculating boundary information (used only for 1D flow coupling). Because CONVERGE and GT-SUITE have different spatial representations of the flow, one of these methods is needed to transport flow from one domain to the other. <i>EXTRAPOLATION</i> = Performs a linear extrapolation from the GT-SUITE cell-centered value and the first CONVERGE cell-centered value. This option is particularly sensitive to grid spacing mismatch and generally not recommended. <i>BOUND</i> = Takes the average value across the CONVERGE boundary and passes it to GT-SUITE. This option cannot account for any boundary-normal gradient. <i>REGION_AUTO</i> = Calculates an average across an automatically generated virtual region of the CONVERGE domain. This virtual region is generated based on local cell spacing. <i>REGION_MANUAL</i> = Calculates an average across a user-specified region of the CONVERGE domain. To set up this option, you must use CONVERGE Studio to create a new region based on the normal distance to the coupling boundary in the GT-SUITE model. Next, assign the GT-SUITE coupling boundary to this new region. Refer to the CONVERGE Studio 3.1 Manual for more information.	<i>REGION_AUTO</i> <i>TO</i>
<i>model_file</i>	Name of a GT-SUITE file that contains the GT-SUITE model parameters.	

Chapter 25: Input and Data Files

Boundary Conditions Co-simulation: cosimulation.in

Table 25.51: Format of the ABAQUS settings block (required only for CONVERGE + Abaqus co-simulation).

Parameter	Description	Typical value
ABAQUS		
<i>reference_pressure</i>	Pressure to subtract from the CONVERGE simulation to the Abaqus simulation. Specify a real value in <i>Pa</i> or enter <i>AVERAGE</i> to use the average pressure over the object.	0.0
<i>time_shift</i>	Time difference from CONVERGE to Abaqus. This time is added to the CONVERGE time to yield the Abaqus time.	0.0
<i>transfer_pressure_flag</i>	0 = Do not transfer pressure forces, 1 = Transfer pressure forces.	1
<i>transfer_stress_flag</i>	0 = Do not transfer viscous forces, 1 = Transfer viscous forces.	1
<i>interpolation_method</i>	Interpolation method to calculate the force on a face centroid. <i>CENTROID_VALUE</i> = Use the centroid value, <i>NODE_BASED</i> = Interpolate from the nodes.	
<i>reverse_normals_flag</i>	0 = Do not reverse normals from the CONVERGE mesh to the Abaqus mesh, 1 = Reverse the normals from the CONVERGE mesh to the Abaqus mesh for consistency.	1
<i>host_name</i>	Host name and port where Abaqus is running (<i>e.g.</i> , "localhost:6600").	
cosimulation_objects		
- <i>cosimulation_object</i>	This settings sub-block specifies the Abaqus co-simulation object parameters. Repeat this sub-block through <i>outgoing_field</i> for each Abaqus co-simulation object.	
<i>object_name</i>	Name of this Abaqus co-simulation object.	
<i>object_type</i>	Type of this object. <i>SURFACE</i> = Surface object.	
<i>boundary_id</i>	List of boundary ID numbers making up this object.	
<i>incoming_field</i>	<i>DISPLACEMENT</i> = Abaqus specifies the displacement field to CONVERGE.	
<i>outgoing_field</i>	<i>TRACTION_VECTOR</i> = CONVERGE specifies the force field to Abaqus.	
<i>events*</i>		
- <i>event</i>	This settings sub-block specifies the event parameters. Repeat this sub-block for each event for this object.	

Chapter 25: Input and Data Files

Boundary Conditions Co-simulation: cosimulation.in

Parameter	Description	Typical value
<i>event_flag</i>	<i>LOCATION</i> = Event triggers based on distance from specified location, <i>MAX</i> = Event triggers based on maximum displacement relative to initial position.	
<i>region_id_a</i>	Region ID of the first region.	
<i>region_id_b</i>	Region ID for the second region.	
<i>deflection_open</i>	Object displacement (<i>m</i>) at which to OPEN the regions.	
<i>deflection_close</i>	Object displacement (<i>m</i>) at which to CLOSE the regions.	
<i>location</i>	Location at which to measure the displacement.	
<i>direction</i>	Unit vector along which to calculate motion.	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

25.6 Initial Conditions and Events

This section describes the input and data files related to initial conditions for your CONVERGE simulation.

Domain Initialization: initialize.in

Use the *initialize.in* file to [uniformly initialize the domain](#) at the beginning of a CONVERGE simulation. This file is required for all simulations.

```
version: 3.1
---

- region:
    region_id: 0
    vel_init: [0.0, 0.0, 0.0]
    temp_init: 300.0
    pres_init: 101325.0
    turbulence_init:
        approach: VALUE
        value:
            tke: 1.3
            eps: 10.0
            omega: 100.0
        scale:
            visc_ratio: 1.0
            length_scale: 1.0
    species_init:
        O2: 0.2323
        N2: 0.7733
    passive_init:
        P1: 1

- region:
    region_id: 2
    vel_init: copy_from_boundary_11
    temp_init: 300.0
    pres_init: 101325.0
    turbulence_init:
        approach: VALUE
        value:
            tke: 1.2
            eps: 10.0
            omega: 1e-3
        scale:
            visc_ratio: 1.0
            length_scale: 1.0
    species_init: copy_from_boundary_11
    passive_init:
        P1: 1.0
```

Figure 25.83: An example *initialize.in* file for a case with two regions.

Table 25.52: Format of *initialize.in*.

Parameter	Description
- <i>region</i>	This settings block specifies the initialization parameters for a region. Repeat this block for each region in the domain.
<i>region_id</i>	A unique integer to identify the region. All of the boundaries in this region must have the same <i>region_id</i> , and each <i>region_id</i> in <i>initialize.in</i> must match the corresponding <i>region_id</i> in <i>boundary.in</i> . CONVERGE will write region-specific output files so that you can visualize results on a region-by-region basis.

Chapter 25: Input and Data Files

Initial Conditions and Events Domain Initialization: initialize.in

Parameter	Description
<i>vel_init</i>	Initial velocity (m/s). To copy the value from a velocity boundary condition, enter <i>copy_from_boundary_<ID></i> , where <i><ID></i> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the velocity boundary condition type is Dirichlet.
<i>temp_init</i>	Initial temperature (K). To copy the value from a temperature boundary condition, enter <i>copy_from_boundary_<ID></i> , where <i><ID></i> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the temperature boundary condition type is Dirichlet.
<i>pres_init</i>	Initial pressure (Pa). To copy the value from a pressure boundary condition, enter <i>copy_from_boundary_<ID></i> , where <i><ID></i> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the pressure boundary condition type is Dirichlet.
<i>turbulence_init</i>	
<i>approach</i>	Turbulence initialization methodology. <i>VALUE</i> = Specify turbulence parameter values directly, <i>SCALE</i> = Specify turbulence through viscosity ratio and length scale.
<i>value</i>	Used only if <i>approach</i> = <i>VALUE</i> .
<i>tke</i>	Initial turbulent kinetic energy (m^2/s^2). To copy the value from a turbulent kinetic energy (tke) boundary condition, enter <i>copy_from_boundary_<ID></i> , where <i><ID></i> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the tke boundary condition type is Dirichlet.
<i>eps</i>	Initial turbulent dissipation (m^2/s^3). To copy the value from a turbulent dissipation (eps) boundary condition, enter <i>copy_from_boundary_<ID></i> , where <i><ID></i> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the eps boundary condition type is Dirichlet.
<i>omega</i>	Initial specific dissipation (s^{-1}). To copy the value from a specific dissipation (omega) boundary condition, enter <i>copy_from_boundary_<ID></i> , where <i><ID></i> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the omega boundary condition type is Dirichlet.
<i>scale</i>	Used only if <i>approach</i> = <i>SCALE</i> .

Chapter 25: Input and Data Files

Initial Conditions and Events Domain Initialization: initialize.in

Parameter	Description
<i>visc_ratio</i>	Viscosity ratio used to initialize the turbulence quantities.
<i>length_scale</i>	Length scale used to initialize the turbulence quantities.
<i>species_init*</i>	List the species names and initial mass fractions in this sub-block. To copy the species names and mass fractions from a species boundary condition, enter <i>copy_from_boundary_<ID></i> , where <ID> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the species boundary condition type is Dirichlet.
< <i>species name</i> >	Initialization mass fraction of this species.
<i>passive_init*</i>	List the passive names and initial values in this sub-block. To copy the passive names and values from a passive boundary condition, enter <i>copy_from_boundary_<ID></i> , where <ID> is the boundary ID (e.g., <i>copy_from_boundary_4</i>). This option is available only if the boundary type is GT-SUITE or if the passive boundary condition type is Dirichlet.
< <i>passive name</i> >	Initialization value of this passive.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

When specifying region-specific initial values, you do not have to list the region-by-region blocks of information in any specific order. For example, region 6 can be defined before region 4, and region 5 does not need to exist.

If you initialize [species](#) or [passives](#) in *initialize.in*, the names in *initialize.in* must also appear in the [chemical reaction mechanism file](#) or in [species.in](#). CONVERGE will normalize the species mass fractions to ensure that the mass fractions add up to one.

Events: events.in

The *events.in* file controls several types of events. To activate events in your simulation, set [*inputs.in* > grid_control > events_flag = 1](#) and include an *events.in* file in your case setup.

CONVERGE controls the flow between regions by activating or deactivating [disconnect triangles](#). The *events.in* file describes OPEN events (to connect regions and thus allow flow) and CLOSE events (to disconnect regions and thus prohibit flow). If no *events.in* file is included in your case setup, CONVERGE will consider all regions to be disconnected at all times (*i.e.*, there will be no flow between regions). When disconnect triangles are activated (*i.e.*, regions are not connected), they act as a SYMMETRY boundary condition.

For engine cases that include valves, you can set up [VALVE events](#), which direct CONVERGE to automatically create OPEN and CLOSE events based on the specified [valve lift profiles](#).

For [contact resistance modeling](#), you can have OPEN, CLOSE, and VALVE events.

For simulations that include [fluid-structure interaction modeling](#), you can set up [FSI events](#) in *events.in*. There are two types of FSI events: FSI with rigid-body motion and FSI with a beam bending model.

Note that the region IDs can be specified in either order, and the events do not need to be specified in chronological order.

Chapter 25: Input and Data Files

Initial Conditions and Events Events: events.in

```
version: 3.1
---

events:
  - event:
      type: VALVE
      contact_resistance_flag: 0
      regions: [0,2]
      temporal_type: CYCLIC
      period: 720
      time: 309
  - event:
      type: RIGID_FSI
      fsi_event_name: fsi_event_1
      fsi_object_name: suction_valve
      regions: [0,1]
      displacement_open: 1e-06
      displacement_close: 1e-06
      direction: [1.0, 0.0, 0.0]
  - event:
      type: RIGID_FSI
      fsi_event_name: fsi_event_2
      fsi_object_name: suction_valve
      regions: [0,1]
      displacement_type: 1
      displacement_open: 1e-03
      displacement_close: 1e-03
      axis: [0.0, 1.0, 0.0]
  - event:
      type: BEAM
      fsi_event_name: fsi_event_3
      fsi_object_name: discharge_reed
      regions: [0,2]
      displacement_open: 1e-06
      displacement_close: 1e-06
      location: 1.0e-2
  - event:
      type: OPEN
      regions: [3, 1]
      contact_resistance_flag: 0
      temporal_type: CYCLIC
      period: 360
      time: 0
  - event:
      type: CLOSE
      regions: [3, 1]
      contact_resistance_flag: 0
      temporal_type: CYCLIC
      period: 360
      time: 360
  - event:
      type: OPEN
      contact_resistance_flag: 0
      temporal_type: SEQUENTIAL
      boundary_id: [3, 4]
      time: 115
  - event:
      type: OPEN
      contact_resistance_flag: 0
      temporal_type: GRIDSCALE
      starting_gridscale: 0
      delay_time: 600
      boundary_id: [11, 12]

no_dis_triangles:
  regions: [0,1,3,4]

disconnect_concentric_tol: 1e-4
disconnect_concentric_dist: 1e-3
```

Figure 25.84: Sample *events.in* file.

Chapter 25: Input and Data Files

Initial Conditions and Events Events: events.in

Table 25.53: Format of the *events* settings block.

Parameter	Description
<i>events</i>	
- <i>event</i>	This settings sub-block specifies the parameters for an open, close, or valve event. Repeat this sub-block through <i>time</i> for each event of type <i>OPEN</i> , <i>CLOSE</i> , or <i>VALVE</i> (with or without contact resistance).
<i>type</i>	Type of event (see below for FSI event types). <i>OPEN</i> = Event which opens the connection between two regions, <i>CLOSE</i> = Event which closes the connection between two regions, <i>VALVE</i> = CONVERGE automatically performs region connection and disconnection based on the valve lift profile specified in boundary.in .
<i>contact_resistance_flag</i>	Flag to activate contact resistance. 0 = No contact resistance, 1 = Contact resistance.
<i>contact_resistance_value</i> *	Value of contact resistance ($K \cdot m^2/W$).
<i>regions</i>	Region IDs of the first and second region.
<i>temporal_type</i>	Temporal type (<i>CYCLIC</i> , <i>SEQUENTIAL</i> , <i>PERMANENT</i> , or <i>GRIDSCALE</i>) for this event. <i>GRIDSCALE</i> requires a grid scaling file (e.g., gridscale.in) and is available only for inlaid mesh events.
<i>period</i>	The <i>CYCLIC</i> period for this event (in <i>seconds</i> if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).
<i>starting_gridscale</i>	Grid scaling value at which the state of the inlaid mesh boundaries (specified in <i>boundary_id</i>) changes to <i>CLOSE</i> (if <i>type</i> = <i>OPEN</i>) or <i>OPEN</i> (if <i>type</i> = <i>CLOSE</i>). Applies only when <i>temporal_type</i> = <i>GRIDSCALE</i> .
<i>delay_time</i>	Delay time (in <i>cycles</i> for steady-state cases, in <i>seconds</i> for transient cases with inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0, or in <i>crank angle degrees</i> for transient cases when <i>crank_flag</i> is non-zero) for triggering a grid scaling event, measured from when the simulation reaches the <i>starting_gridscale</i> value. Must be greater than or equal to zero. Applies only when <i>temporal_type</i> = <i>GRIDSCALE</i> .
<i>boundary_id</i>	Boundary IDs (specified only for <i>OPEN</i> or <i>CLOSE</i> events that are used to deactivate or activate an inlaid mesh). Associated interior and exterior inlaid mesh boundaries must be activated simultaneously.
<i>time</i>	Event start time (in <i>seconds</i> if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degree</i> if <i>crank_flag</i> is non-zero). Not used when <i>temporal_type</i> = <i>GRIDSCALE</i> .
- <i>event</i>	This settings sub-block specifies the parameters for a rigid FSI or FSI beam. Repeat this sub-block through <i>location</i> (for <i>type</i> = <i>BEAM</i>), <i>direction</i> (for <i>type</i> = <i>RIGID_FSI</i> and <i>displacement_type</i> = 0), or <i>axis</i> (for

Chapter 25: Input and Data Files

Initial Conditions and Events Events: events.in

Parameter	Description
	<i>type</i> = RIGID_FSI and <i>displacement_type</i> = 1) for each event of type RIGID_FSI or BEAM.
<i>type</i>	Type of FSI event. RIGID_FSI = FSI with rigid-body motion, BEAM = FSI with a beam bending model.
<i>fsi_event_name</i>	Name of the FSI event.
<i>fsi_object_name</i>	Name of the corresponding FSI object in fsi.in .
<i>regions</i>	Region IDs of the first and second region.
<i>displacement_type</i> *	Specify whether to use translation or rotation when <i>type</i> = RIGID_FSI. 0 = Translation FSI event (default), 1 = Rotation FSI event.
<i>displacement_open</i>	FSI object displacement (in <i>meters</i> if <i>displacement_type</i> = 0 or in <i>degrees</i> if <i>displacement_type</i> = 1) at which to open the regions.
<i>displacement_close</i>	FSI object displacement (in <i>meters</i> if <i>displacement_type</i> = 0 or in <i>degrees</i> if <i>displacement_type</i> = 1) at which to close the regions.
<i>initial_state</i> *	Event state at the initial displacement. OPEN = Event is in OPEN state, CLOSED = Event is in CLOSED state.
<i>location</i>	Location at which to measure the displacement (for <i>type</i> = BEAM).
<i>direction</i>	Unit vector along which motion takes place during the event. Used only when <i>type</i> = RIGID_FSI and <i>displacement_type</i> = 0.
<i>axis</i>	Unit vector about which the angular displacement takes place during the event. Specify FROM_OBJECT to use <ul style="list-style-type: none"> • fsi.in > rigid_fsi_objects > object > axis for a case with only 1D rotation, • fsi.in > rigid_fsi_objects > object > general_constrained_motion > rotation > axis for a general rotation constrained case, • fsi.in > rigid_fsi_objects > object > rot_vel_init_axis for all other cases. This parameter is used only when <i>type</i> = RIGID_FSI and <i>displacement_type</i> = 1.
- <i>event</i>	This settings sub-block specifies the parameters for a user-defined FSI event. Repeat this sub-block through <i>regions</i> for each user-defined FSI event.
<i>type</i>	Type of FSI event. <i>user_<UDF name></i> = Event controlled by a user-defined function (UDF) based on the CONVERGE_EVENT macro. Refer to the CONVERGE 3.1 UDF Manual for more information about UDFs.
<i>regions</i>	Region IDs of the first and second region.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Chapter 25: Input and Data Files

Initial Conditions and Events Events: events.in

Table 25.54: Format of the *no_dis_triangles* settings block.

Parameter	Description
<i>no_dis_triangles</i>	
<i>regions</i>	List of regions that are always disconnected.

CONVERGE evaluates the proximity between two concentric circles in terms of the difference in their radii and the distance between the two centers. Tables 25.55 and 25.56 below describe the two parameters that are used during events that contain two concentric circles.

Table 25.55: Format of the *disconnect_concentric_tol* settings block.

Parameter	Description
<i>disconnect_concentric_tol</i>	Difference between the radii of the two concentric circles. Default value: 0.0.

Table 25.56: Format of the *disconnect_concentric_dist* settings block.

Parameter	Description
<i>disconnect_concentric_dist</i>	Distance between the centers of the two concentric circles. Default value: 1e-3.

Mapping Variables: **map.in**, **map.h5**, **map_parcel.h5**

The mapping capability in CONVERGE allows you to initialize the domain with spatially varying parameters. (CONVERGE will initialize all other parameters via the information in [initialize.in](#).) To activate mapping, set [*inputs.in*](#) > *simulation_control* > *map_flag* = MAP. The *map.in* file defines the parameters to be mapped, the name of the file containing the spatially varying data, a data rotation angle, data scaling factors, and data translation distances.

Any number of species can be supplied in the mapping data. If at least one species is mapped, CONVERGE assumes that all non-zero species are included in the mapping (*i.e.*, all species values not specified in *map.in* will be set to zero). CONVERGE normalizes the species mass fractions if they do not add up to 1.0.

Chapter 25: Input and Data Files

Initial Conditions and Events Mapping Variables: map.in, map.h5, map_parcel.h5

```
version: 3.1
---

cells_map_all_regions: 0
map_phase_flag: 0
cells:
  - map_region:
    filename: map_1_3.h5
    regions: [1, 3]
    datasets: pressure
    rotation:
      axis: X
      angle: 2.2
    manipulations:
      temp:
        translate: 3.1
        scale: 1.1
      radius:
        translate: 1.0
      H2O:
        scale: 2.0

  - map_region:
    filename: map_2_4.h5
    regions: [2, 4]
    datasets: pressure
    rotation:
      axis: X
      angle: 2.2
    manipulations:
      temp:
        translate: 3.1
        scale: 1.1
      radius:
        translate: 1.0
      H2O:
        scale: 2.0

parcels_map_all_regions: 0
match_regions: 1
match_boundaries: 1
force_parcels_into_domain: 1
parcels:
  - map_region:
    filename: map.Parcel_1_3.h5
    regions: [1, 3]
    datasets: [temp, radius]
    rotation:
      axis: Y
      angle: 1.2
    manipulations:
      temp:
        translate: 1.1
        scale: 2.1
      radius:
        translate: 0.02
      H2O:
        translate: 0.02
  fsi_map:
    - map_region:
      filename: map_1_3.h5
      source_object: obj1
      target_object: obj2
      datasets: [POSITION, VELOCITY]
```

Figure 25.85: An example *map.in* file. In this case, there is one mapping procedure for regions 1 and 3 and a different mapping procedure for regions 2 and 4.

Chapter 25: Input and Data Files

Initial Conditions and Events Mapping Variables: map.in, map.h5, map_parcel.h5

Table 25.57: Format of *map.in*.

Parameter	Description
<i>cells_map_all_regions</i>	0 = Do not map cells in all regions, 1 = Map cells in all regions.
<i>map_phase_flag</i>	0 = Do not allow mapping across streams, 1 = Allow mapping across streams with the same phase (fluid or solid).
<i>cells*</i>	
- <i>map_region</i>	This settings sub-block specifies the mapping configuration for a region. Repeat this sub-block through <i>filename</i> or optionally through <i>scale</i> for each region with different mapping configurations.
<i>filename</i>	Name of the data file containing the mapping data, e.g., <i>map_1_3.h5</i> or <i>restart0001.rst</i> . For region-by-region mapping, you can supply different file names in different <i>map_region</i> blocks.
<i>regions*</i>	The ID number(s) of the region(s) to be mapped by the mapping strategy described below.
<i>datasets*</i>	Datasets to map with the strategy described below. Specify multiple datasets as a square bracket-enclosed series. If you do not specify any datasets, CONVERGE will map all available datasets.
<i>rotation*</i>	
<i>axis*</i>	Axis about which the coordinates and velocity will be rotated. The rotation is applied after scaling and translating.
<i>angle*</i>	Rotation angle about the specified axis. The right hand rule is used to determine the direction of rotation. Also, note that the rotation for velocity is applied before the scaling and offset on velocity.
<i>manipulations*</i>	Use the < <i>dataset name</i> > sub-blocks to manipulate individual datasets.
< <i>dataset name</i> >*	This settings sub-block specifies the translation and/or scale factors for a dataset. Repeat this sub-block for each < <i>dataset name</i> >.
<i>translate*</i>	Translation factor for the mapped dataset.
<i>scale*</i>	Scaling factor for the mapped dataset.
<i>parcels_map_all_regions</i>	Map all parcels in the geometry. CONVERGE will only map parcels whose names match the rest of the Case Setup.
<i>match_regions*</i>	0 = Do not map the parcels into the same region, 1 = Map the parcels into the same region.
<i>match_boundaries*</i>	0 = Do not map the parcels on the boundary, 1 = Map the parcels on the boundary.
<i>force_parcels_into_domain*</i>	0 = Do not move the parcels into the domain if the parcels are outside the domain, 1 = Move the parcels into the domain if the parcels are outside the domain.

Chapter 25: Input and Data Files

Initial Conditions and Events Mapping Variables: map.in, map.h5, map_parcel.h5

Parameter	Description
<i>parcels</i> *	
- <i>map_region</i>	This settings sub-block specifies the mapping configuration for a region. Repeat this sub-block through <i>filename</i> or optionally through <i>scale</i> for each region with different mapping configurations.
<i>filename</i>	Name of the data file containing the parcel mapping data, e.g., <i>map_parcel_1_3.h5</i> or <i>restart0001.rst</i> . For region-by-region mapping, you can supply different file names in different <i>map_region</i> blocks.
<i>regions</i> *	The ID number(s) of the region(s) to be mapped by the mapping strategy described below.
<i>datasets</i> *	Datasets to map with the strategy described below. If you do not specify any datasets, CONVERGE will map all available datasets.
<i>rotation</i> *	
<i>axis</i> *	Axis about which the coordinates and velocity will be rotated. The rotation is applied after scaling and translating.
<i>angle</i> *	Rotation angle about the specified axis. The right hand rule is used to determine the direction of rotation. Also, note that the rotation for velocity is applied before the scaling and offset on velocity.
<i>manipulations</i> *	Use the < <i>dataset name</i> > sub-block to manipulate individual datasets.
< <i>dataset name</i> >*	This settings sub-block specifies the translation and/or scale factors for a dataset. Repeat this sub-block for each < <i>dataset name</i> >.
<i>translate</i> *	Translation factor for the mapped variable.
<i>scale</i> *	Scaling factor for the mapped variable.
<i>fsi_map</i> *	
- <i>map_region</i>	This sub-block specifies the mapping configuration for an FSI object. Repeat for each FSI object to be mapped.
<i>filename</i>	Name of the data file containing the mapping data, e.g., <i>map_1_3.h5</i> or <i>restart0001.rst</i> . FSI mapping data is stored in the same file as cell mapping data.
<i>source_object</i>	Name of the FSI object from which data will be mapped.
<i>target_object</i>	Name of the FSI object in the new simulation to which data will be mapped.
<i>datasets</i> *	Datasets to map. POSITION and VELOCITY are the only available datasets for FSI objects. If you do not specify any datasets, CONVERGE maps only the FSI object name.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

To initialize parameters via mapping, you must include a mapping data file in your case setup. You can manually create an HDF5-formatted data file, use a map file from a previous

Chapter 25: Input and Data Files

Initial Conditions and Events Mapping Variables: map.in, map.h5, map.Parcel.h5

CONVERGE simulation (e.g., [*map <time>.h5*](#)), or use a restart file from a previous CONVERGE simulation (e.g., [*restart<restart number>.rst*](#)). Specify the file via *map.in > cells > map_region > filename* and, if applicable, *map.in > parcels > map_region > filename*.

If you specify an HDF5-formatted data file (referred to as *map.h5* hereafter), it must contain spatially varying data organized by each variable into datasets. To view or analyze an HDF5-formatted file, use a utility provided by the HDF group, such as HDFView. Within the *map.h5* file, cell variables are organized by index. For example, the fifth entry in each of the *X*, *Y*, and *Z* datasets provides the *x*, *y*, and *z* location of the quantities listed in the fifth entry of each of the other datasets (such as *pressure*, *temperature*, etc.). Within a *map.h5* file, you must supply *X*, *Y*, and *Z* datasets. Each entry in these datasets must be the appropriate coordinate value, indexed by the location in the dataset. Remember that CONVERGE will translate and/or rotate the *x*, *y*, and *z* coordinates based on the *map.in > manipulations > dataset name > translate*, *manipulations > dataset name > translate*, and *map.in > cells > rotation > angle* parameters. Other datasets must be listed with the names given below in Table 25.58. The datasets may be in any order.

Table 25.58: Cell variables that can be mapped via *map.h5*.

Quantity to be mapped	Dataset name	Units
Coordinates	<i>x, y, z</i>	<i>m</i>
X component of velocity	<i>u_vel</i>	<i>m/s</i>
Y component of velocity	<i>v_vel</i>	<i>m/s</i>
Z component of velocity	<i>w_vel</i>	<i>m/s</i>
Temperature	<i>temperature</i>	<i>K</i>
Pressure	<i>pressure</i>	<i>Pa</i>
Species mass fraction	species name (e.g., <i>h2o</i>)	N/A
Passive	passive name (e.g., <i>soot</i>)	N/A
Turbulent kinetic energy	<i>tke</i>	<i>m²/s²</i>
Turbulent dissipation	<i>eps</i>	<i>m²/s³</i>
Specific dissipation rate value in the cell.	<i>Omega</i>	(<i>s⁻¹</i>)

The HDF5-formatted parcel mapping data file (referred to as *map.Parcel.h5* hereafter) has the same format as *map.h5* (i.e., you must include the *X*, *Y*, and *Z* datasets, and each subsequent column must correspond to one of the parcel variables to be mapped). Note that the locations of the velocity values will be translated by the rotation angle, which is given by *map.in > parcels > rotation > angle*.

Chapter 25: Input and Data Files

Initial Conditions and Events Mapping Variables: map.in, map.h5, map.Parcel.h5

Table 25.59: Parcel variables that can be mapped via *map.Parcel.h5*.

Quantity to be mapped	Dataset name	Units
Coordinates.	x, y, z	m
x component of velocity.	u_{vel}	m/s
y component of velocity.	v_{vel}	m/s
z component of velocity.	w_{vel}	m/s
Temperature.	$temp$	K
Number of drops.	num	N/A
Radius of drops.	$radius$	m
Region ID of the parcel.	$region_id$	N/A
Boundary ID of the parcel.	$bound_id$	N/A
Species mass fraction.	species name (e.g., IC8H18)	N/A
Film flag.	$film$	0 = The parcel is not in a film, 1 = The parcel is in a film.
Injector from which the parcel originated.	$from_injector$	N/A
Nozzle from which the parcel originated.	$from_nozzle$	N/A

After reading the mapped data, CONVERGE interpolates from the nearest data point to initialize each cell. Figure 25.86 below shows an excerpt of an example *map.h5* file viewed in HDFView.

Chapter 25: Input and Data Files

Initial Conditions and Events Mapping Variables: map.in, map.h5, map_parcel.h5

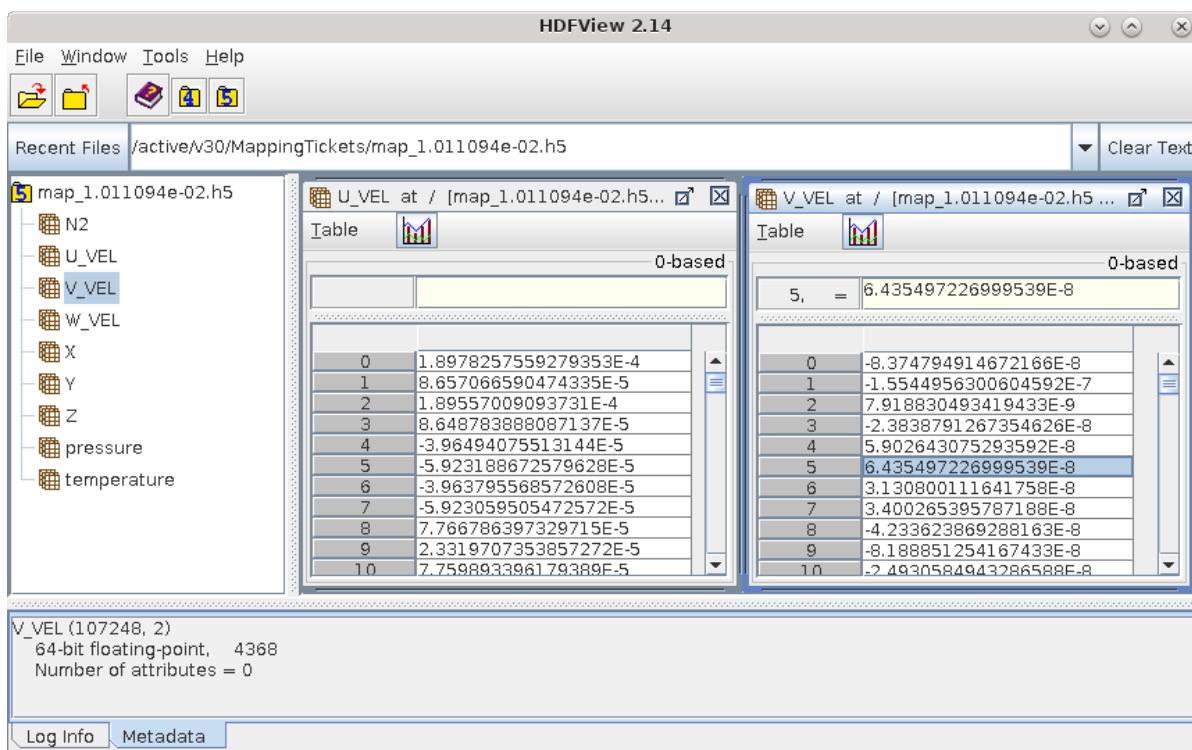


Figure 25.86: An excerpt of an example mapping data file (e.g., *map.h5*) shown in HDFView.

Initialization Perturbation Modeling: *initial_perturbation_region.in*

To specify perturbations that are applied to the initial conditions at the beginning of a CONVERGE simulation, set *inputs.in* > *feature_control* > *initial_perturbation_flag* = 1 and include the *initial_perturbation_region.in* file in the Case Directory. These perturbations are superimposed on the field you specify in *initialize.in*.

To generate the perturbations, CONVERGE uses a pseudorandom number generator that is initialized from a user-specified seed value (*inputs.in* > *simulation_control* > *random_seed*). If you run multiple simulations on the same computer hardware with the same case setup and the same value of *random_seed*, CONVERGE will generate identical perturbations for those simulations. If you want to apply different perturbations to each simulation, you must use different values of *random_seed* for each simulation.

You can apply coherent isotropic perturbations or white noise Gaussian perturbations. Isotropic perturbations have the same functional form as [turbulent inflow fluctuations generated by the Fourier method](#), where the perturbations are represented by a sum of random Fourier modes. Gaussian perturbations take the form

$$u_i' = \frac{1}{\sqrt{3}} u_{mag} R_G \quad (25.86)$$

for velocity and

Chapter 25: Input and Data Files

Initial Conditions and Events Initialization Perturbation Modeling: initial_perturbation_region.in

$$\phi' = \phi_{mag} R_G \quad (25.87)$$

for temperature and turbulent kinetic energy (tke), where R_G is a random number generated from a standard Gaussian distribution and the magnitudes u_{mag} and ϕ_{mag} are user-specified.

```
version: 3.1
---

perturbation_regions:
  - perturbation_region:
      region_id: 0
      velocity:
        magnitude: 5
        type: ISOTROPIC
        length_scale: 0.0025
      temperature:
        magnitude: 2
        type: ISOTROPIC
        length_scale: 0.0025
      tke:
        magnitude: 3
        type: GAUSSIAN
  - perturbation_region:
      region_id: [1,2]
      velocity:
        magnitude: 10
        type: ISOTROPIC
        length_scale: 0.001
      temperature:
        magnitude: 20
        type: ISOTROPIC
        length_scale: 0.001
      tke:
        magnitude: 5
        type: GAUSSIAN
```

Figure 25.87: An example *initial_perturbation_region.in* file.

Table 25.60: Format of *initial_perturbation_region.in*.

Parameter	Description
<i>perturbation_regions</i>	
- <i>perturbation_region</i>	This settings sub-block specifies the parameters for a perturbation region. Repeat this sub-block through <i>tke > length_scale</i> for each perturbation region.
<i>region_id</i>	Region ID(s) for this perturbation region.
<i>velocity</i>	
<i>magnitude</i>	Magnitude (m/s) of the velocity perturbation.
<i>type</i>	<i>ISOTROPIC</i> or <i>GAUSSIAN</i> .

Chapter 25: Input and Data Files

Initial Conditions and Events Initialization Perturbation Modeling: initial_perturbation_region.in

Parameter	Description
<i>length_scale</i>	Length scale (m) of the velocity perturbation (for <i>type</i> = ISOTROPIC). This corresponds to the shortest allowed wavelength for the Fourier modes, which is typically several times larger than the cell size. We do not recommend a length scale smaller than twice the cell size, as CONVERGE is unable to resolve fluctuations at those scales. This parameter is required for <i>type</i> = ISOTROPIC and ignored for <i>type</i> = GAUSSIAN.
<i>temperature</i>	
<i>magnitude</i>	Magnitude (K) of the temperature perturbation.
<i>type</i>	ISOTROPIC or GAUSSIAN.
<i>length_scale</i>	Length scale (m) of the temperature perturbation (for <i>type</i> = ISOTROPIC). This corresponds to the shortest allowed wavelength for the Fourier modes, which is typically several times larger than the cell size. We do not recommend a length scale smaller than twice the cell size, as CONVERGE is unable to resolve fluctuations at those scales. This parameter is required for <i>type</i> = ISOTROPIC and ignored for <i>type</i> = GAUSSIAN.
<i>tke</i>	
<i>magnitude</i>	Magnitude (m^2/s^2) of the turbulent kinetic energy perturbation.
<i>type</i>	ISOTROPIC or GAUSSIAN.
<i>length_scale</i>	Length scale (m) of the turbulent kinetic energy perturbation (for <i>type</i> = ISOTROPIC). This corresponds to the shortest allowed wavelength for the Fourier modes, which is typically several times larger than the cell size. We do not recommend a length scale smaller than twice the cell size, as CONVERGE is unable to resolve fluctuations at those scales. This parameter is required for <i>type</i> = ISOTROPIC and ignored for <i>type</i> = GAUSSIAN.

25.7 Physical Models

This section describes the input files that contain information about the physical models for your CONVERGE simulation.

Discrete Phase Modeling

This section describes input files that contain options for discrete phase modeling in CONVERGE.

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parcel Settings: parcels.in

To include parcels for discrete phase modeling, set at least one of *inputs.in > feature_control > parcel_mode > [phase]_parcel = 1* and include a *parcels.in* file in your case setup.

```
version: 3.1
---

collision_file_flag: 0
collision_mesh_embed_level: 2
liquid_parcels:
    combine_parcels:
        active: 0
        start_time: -999999
        end_time: -999999
        max_parcels_per_node: 50
    sub_cycle:
        active: 0
        dt: 1e+32
    collision_control:
        collision_mesh_flag: 0
        collision_model: NTC
        outcome_model: POST
    wall_interaction_model: REBOUND_SLIDE
    parcel_list:
        - parcel:
            parcel_name: LiqParcel_1
            parcel_species:
                DIESEL2: 1
            turb_dispersion_control:
                dispersion_model: OROURKE
            porous_media_interaction_control:
                porous_trap_parcel_model: NONE
                porous_break_parcel_radius_ratio: 0.1
            remove_control:
                tiny_parcel_mass_ratio: 0.001
                tiny_parcel_radius: 1e-20
            evap_control:
                evap_model: FROSSLING
                evap_source_flag: 0
                evap_species: c7h16
                boiling:
                    evap_flag_flash_boiling: 0
                    size_flag_flash_boiling: 0
                    evap_scale_factor_flash_boiling: 1
                    size_scale_factor_flash_boiling: 1
                d0_diffuse: 4.16e-06
                n_diffuse: 1.6
                scaling:
                    heat_trans_coeff: 1
                    mass_trans_coeff: 1
            temp_discretization:
                active: 0
                radius: 1000
                layers_per_drop: 15
                recirculation_flag: 0
            urea_model: NONE
        drag_control:
            drag_model: DYNAMIC
            cfocbck: 0.08333
            csubd: 10
            csubk: 8
        breakup_control:
            breakup_model: MOD-KH-RT
            kh:
                new_parcel_flag: 1
                new_parcel_cutoff: 0.05
                no_enlarge_flag: 0
                shed_factor: 1
                vel_const: 0.188
```

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

```
        size_const:          0.6
        time_const:         7
    khact:
        turb_kc:           0.45
        turb_ke:           0.27
        turb_s:            0.01
        c_tcav:            0.1
    rt:
        distribution:      UNIFORM
        time_const:        1
        size_const:         0.1
        length_const:      99999.9
    tab:
        distribution:      UNIFORM
    lisa:
        distribution:      UNIFORM
        length_const:      12
        size_const:         0.5
solid_parcels:
    collision_control:
        collision_mesh_flag: 0
        collision_model:     NONE
        collision_outcome_control:
            outcome_model:   ELASTIC
            coefficient_of_restitution: 0.4
            stick_percentage: 10
    wall_interaction_control:
        wall_interaction_model: REBOUND
        coefficient_of_restitution:
            type:             CONSTANT
            normal_direction: 0.8
            tangential_direction: 0.4
            stick_percentage: 10
    parcel_list:
        - parcel:
            parcel_name:      solid.Parcel_sand
            parcel_species:   SAND
            turb_dispersion_control:
                dispersion_model: DEFAULT
            porous_media_interaction_control:
                porous_trap_parcel_model: NONE
            remove_control:
                tiny_parcel_mass_ratio: 1.0e-3
                tiny_parcel_radius:     1.0e-20
        force_control:
            drag_control:
                drag_model:       CLIFT_GRACE_WEBER
                sphericity:       1.0
            added_mass_control:
                active:           0
                added_mass_coefficient: 0.5
            lift_control:
                lift_model:       NONE
                undisturbed_fluid_force_flag: 0
        thermal_exchange_control:
            convection_model: NONE
            turbulence_modification_model: NONE
gas_parcels:
    active:          0
```

Figure 25.88: An example *parcels.in* file.

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Table 25.61: Format of *parcels.in*.

Parameter	Description	Typical value
<i>collision_file_flag</i>	Flag to specify parcel collision control. 0 = Use default parcel collision behavior, 1 = Specify parcel collision control in <i>parcel.ParcelInteraction.in</i> .	
<i>collision_mesh_embed_level</i>	The embed level used to create the collision mesh. For example, a value of 2 will yield a collision mesh size that is 1/4 of the base grid size.	
<i>liquid_parcels</i>		
<i>combine_parcels</i>		
<i>active</i>	0 = Do not combine parcels, 1 = Combine parcels.	
<i>start_time</i>	CONVERGE will start combining parcels at this time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>end_time</i>	CONVERGE will stop combining parcels at this time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>max_parcels_per_node</i>	Maximum number of parcels per fluid cell. CONVERGE will combine parcels in the cell if this number is exceeded.	
<i>sub_cycle</i>		
<i>active</i>	0 = Do not activate spray sub-cycling, 1 = Activate spray sub-cycling.	
<i>dt</i>	Spray sub-cycling time-step (s or CAD).	
<i>collision_control</i>		
<i>collision_mesh_flag</i>	Collision mesh model flag. 0 = No collision mesh, 1 = Collision mesh.	
<i>collision_model</i>	Collision model flag. <i>NONE</i> = No collision model, <i>OROURKE</i> = O'Rourke collision model, <i>NTC</i> = NTC collision model.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>outcome_model</i>	Collision outcomes flag. <i>OROURKE</i> = O'Rourke outcomes, <i>POST</i> = Post outcomes.	
<i>wall_interaction_model</i>	Parcel-wall interaction model flag. <i>REBOUND_SLIDE</i> = Rebound/slide model, <i>VANISH</i> = Drops vanish when they impinge on a solid boundary, <i>FILE</i> = Include <i>parcel_wall_interaction.in</i> in your case setup.	
<i>parcel_list</i>		
- <i>parcel</i>	Repeat this sub-block for each parcel.	
<i>parcel_name</i>	Name of this parcel.	
<i>parcel_species</i>		
< <i>species name</i> >	Mass fraction of this species.	
	Repeat this row for each species. Mass fractions must sum to 1.	
<i>turb_dispersion_control</i>		
<i>dispersion_model</i>	Turbulent dispersion flag. <i>NONE</i> = No turbulent dispersion, <i>OROURKE</i> = O'Rourke turbulent dispersion, <i>TKE_PRESERVING</i> = tke-preserving turbulent dispersion.	
<i>porous_media_interaction_control</i>		
<i>porous_trap.Parcel_model</i>	Porous media drop interaction control flag. <i>OFF</i> = No interaction between drop and porous media, <i>TRAP_ONLY</i> = Drops are trapped by porous media, <i>TRAP_AND_BREAKUP</i> = Drops are trapped and broken up by porous media.	
<i>porous_break_parcel_radius_ratio</i>	Ratio of post-breakup to pre-breakup drop radii. Used only for <i>porous_trap.Parcel_mode</i> = <i>TRAP_AND_BREAKUP</i> .	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>remove_control</i>		
<i>tiny.Parcel.mass_ratio</i>	Minimum parcel mass ratio. Parcels with a mass smaller than this value times the mass per parcel at its introduction will be removed.	1e-3
	Mass per parcel at introduction is specified by <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>injection_control</i> > <i>injection_velocity_control</i> > <i>mass_per_parcel</i> , <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>injection_control</i> > <i>vof_oneway_mapping</i> > <i>mass_per_parcel</i> , or <i>parcel_introduction.in</i> > <i>boundary_injections</i> > <i>boundary_injection</i> > <i>injection_control</i> > <i>mass_per_parcel</i> .	
<i>tiny.Parcel.radius</i>	Minimum parcel radius (m). Parcels with a radius smaller than this value will be removed.	
<i>evap_control</i>		
<i>evap_model</i>	Drop evaporation model flag. <i>NONE</i> = No drop evaporation, <i>FROSSLING</i> = Frossling drop evaporation, <i>CHIANG</i> = Chiang drop evaporation	
<i>evap_source_flag</i>	Evaporation source flag. 0 = All species will evaporate to <i>evap_species</i> , 1 = Single-component species will evaporate to their respective species and multi-component liquid species will evaporate to a composite species (if composites are used), 2 = Single-component species will evaporate to their respective species and multi-component liquid species will evaporate to their base species.	
<i>evap_species</i>	Name of the species that will be sourced when evaporation occurs. Used only when <i>evap_source_flag</i> = 0.	
<i>boiling</i>	0 = No drop boiling, 1 = Use drop boiling model. Not available if <i>evap_model</i> = <i>NONE</i> .	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>evap_flag_flash_boiling</i>	0 = No drop flash boiling, 1 = Use drop flash boiling model. Not available if <i>evap_model</i> = <i>NONE</i> .	
<i>size_flag_flash_boiling</i>	0 = Do not reduce drop size due to flash boiling, 1 = Reduce drop size due to flash boiling.	
<i>evap_scale_factor_flash_boiling</i>	Scale factor for flash boiling.	
<i>size_scale_factor_flash_boiling</i>	Scale factor for diameter reduction due to flash boiling.	
<i>d0_diffuse</i>	Liquid species mass diffusivity variable. The parameter <i>d0_diffuse</i> equals D_0 in the following equation: $\rho_{\text{gas}} D = 1.293 D_0 (T_{\text{gas}} / 273)^{n_0 - 1}.$	
<i>n_diffuse</i>	Liquid species mass diffusivity variable. The parameter <i>n_diffuse</i> equals n_0 in the following equation: $\rho_{\text{gas}} D = 1.293 D_0 (T_{\text{gas}} / 273)^{n_0 - 1}.$	
<i>scaling</i>		
<i>heat_trans_coeff</i>	Scaling applied to the drop heat transfer coefficient.	
<i>mass_trans_coeff</i>	Scaling applied to the drop mass transfer coefficient.	
<i>temp_discretization</i>		
<i>active</i>	Drop temperature discretization flag. 0 = Use the Uniform Temperature Model to compute drop thermal transfer. 1 = Use the Uniform Temperature Model to compute drop thermal transfer for parcels smaller than <i>radius</i> and use the Discretized Temperature Model for larger parcels.	
<i>radius</i>	Drop radius (<i>m</i>) at which CONVERGE switches between drop temperature models. The default setting for this parameter ensures that the Spherically Symmetric Heat Equation is disabled.	1e-5 to 1e-4

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>layers_per_drop</i>	Number of equal-volume parts into which the drop domain will be discretized when using the spherically symmetric heat equation.	10 - 20
<i>recirculation_flag</i>	Flag to activate the effective thermal conductivity model. 0 = CONVERGE does not use the effective thermal conductivity model, 1 = CONVERGE uses the effective thermal conductivity model, which simulates the effect of recirculation in the spherically symmetric heat equation.	1
<i>urea_model</i>	<i>NONE</i> = No urea-water solution, <i>MOLTEN_SOLID</i> = Urea-water solution depletion calculation with the molten solid approach <i>DETAIL_DECOMPOSITION</i> = Urea-water solution depletion calculation with the detailed decomposition approach.	
<i>drag_control</i>		
<i>drag_model</i>	Drop drag flag. <i>NONE</i> = No drop drag, <i>SPHERE</i> = Spherical drop drag, <i>DYNAMIC</i> = Dynamic drop drag.	
<i>cfocbck</i>	TAB/dynamic drop drag constant.	0.08333
<i>csubd</i>	TAB/dynamic drop drag constant.	10
<i>csubk</i>	TAB/dynamic drop drag constant.	8
<i>breakup_control</i>		
<i>breakup_model</i>	Parcel breakup model flag. <i>KH</i> = Kelvin-Helmholtz breakup model, <i>KH-ACT</i> = Kelvin-Helmholtz breakup model with aerodynamics, cavitation, and turbulence, <i>RT</i> = Rayleigh-Taylor breakup model, <i>KH-RT</i> = Kelvin-Helmholtz and Rayleigh-Taylor breakup models acting concurrently, <i>MOD-KH-RT</i> = Modified KH-RT breakup model, <i>TAB</i> = Taylor analogy breakup model,	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	<i>LISA</i> = Linearized instability sheet atomization breakup model.	
<i>kh</i>		
<i>new.Parcel_flag</i>	New child parcel flag. 0 = No child parcels, 1 = Child parcels.	
<i>new.Parcel_cutoff</i>	Fraction of injected mass per parcel used for determining when to create a new parcel. Used only when <i>new.Parcel_flag</i> = 1.	0.03 - 0.10
<i>no_enlarge_flag</i>	KH enlargement flag. 0 = Activate drop enlargement for the KH model, 1 = Deactivate drop enlargement for the KH model.	0
<i>shed_factor</i>	Fraction of parent parcel mass that goes into child parcels.	0.1 - 1.0
<i>vel_const</i>	KH model velocity constant.	0.188
<i>size_const</i>	KH model size constant.	0.61
<i>time_const</i>	KH model time constant.	5 - 100
<i>khact</i>		
<i>turb_kc</i>	KH-ACT model constant for turbulence-induced breakup.	0.45
<i>turb_ke</i>	KH-ACT model constant for turbulence-induced breakup.	0.27
<i>turb_s</i>	KH-ACT model constant for turbulence-induced breakup.	0.01
<i>c_tcav</i>	KH-ACT model constant for turbulence- or cavitation-induced breakup.	0.1 - 1.0
<i>rt</i>		
<i>distribution</i>	RT drop size distribution flag. <i>UNIFORM</i> = No drop size distribution with RT breakup, <i>CHI-SQUARED</i> = χ^2 distribution with RT breakup, <i>ROSIN-RAMMLER</i> = Rosin-Rammler distribution with RT breakup. If <i>ROSIN-RAMMLER</i> , CONVERGE uses the Rosin-Rammler distribution	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	parameter (parcel_introduction.in > injections > injection > injection_control > $q_{_rr}$).	
<i>time_const</i>	RT model time constant.	1.0
<i>size_const</i>	RT model size constant.	0.1 - 1.0
<i>length_const</i>	RT model breakup length constant.	0 - 50
<i>tab</i>		
<i>distribution</i>	TAB drop size distribution flag. <i>UNIFORM</i> = No drop size distribution with TAB breakup, <i>CHI-SQUARED</i> = Chi-squared distribution with TAB breakup, <i>ROSIN-RAMMLER</i> = Rosin-Rammler distribution with TAB breakup. If <i>ROSIN-RAMMLER</i> , CONVERGE uses the Rosin-Rammler distribution parameter (parcel_introduction.in > injections > injection > injection_control > $q_{_rr}$).	
<i>lisa</i>		
<i>distribution</i>	LISA drop size distribution flag. <i>UNIFORM</i> = No drop size distribution with LISA breakup, <i>CHI-SQUARED</i> = Chi-squared distribution with LISA breakup, <i>ROSIN-RAMMLER</i> = Rosin-Rammler distribution with LISA breakup. If <i>ROSIN-RAMMLER</i> , CONVERGE uses the Rosin-Rammler distribution parameter (parcel_introduction.in > injections > injection > injection_control > $q_{_rr}$).	
<i>length_const</i>	LISA model breakup length constant.	12.0
<i>size_const</i>	LISA model breakup size constant.	0.5
<i>solid_parcels</i>		
<i>collision_control</i>		
<i>collision_mesh_flag</i>	Flag to specify parcel collision control. 0 = Use default parcel collision behavior, 1 = Specify parcel collision control in <i>parcel.Parcel_interactions.in</i> .	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>collision_model</i>	Collision model flag. <i>NONE</i> = No collision model, <i>OROURKE</i> = O'Rourke collision model, <i>NTC</i> = NTC collision model.	
<i>collision_outcome_control</i>		
<i>outcome_model</i>	Collision outcome flag. <i>ELASTIC</i> = All parcels rebound off of each other, <i>STICK</i> = After collision, <i>stick_percentage</i> of parcels stick together. The remainder of colliding parcels rebound.	
<i>coefficient_of_restitution</i>	Coefficient of restitution for ELASTIC parcels.	
<i>stick_percentage</i>	Percentage of parcels that will stick together when <i>outcome_model</i> = STICK.	
<i>wall_interaction_control</i>		
<i>wall_interaction_model</i>	Parcel-wall interaction model flag. <i>REBOUND</i> = Rebound model, <i>SLIDE</i> = Slide model, <i>VANISH</i> = Parcels vanish when they impinge on a solid boundary, <i>STICK</i> = After impinging on a boundary, <i>stick_percentage</i> of parcels stick to the boundary and the remainder of impinging parcels rebound, <i>FILE</i> = Use settings from <i>parcel_wall_interaction.in</i> .	
<i>coefficient_of_restitution</i>		
<i>type</i>	<i>CONSTANT</i> = Constant coefficient of restitution. This parameter is reserved for future use.	
<i>normal_direction</i>	Wall-normal multiplicative factor for the coefficient of restitution.	0 - 1
<i>tangential_direction</i>	Wall-tangential multiplicative factor for the coefficient of restitution.	0 - 1
<i>stick_percentage</i>	Percentage of parcels that will stick to the wall when <i>wall_interaction_model</i> = STICK.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>parcel_list</i>		
- <i>parcel</i>	Repeat this sub-block for each parcel.	
<i>parcel_name</i>	Name of this parcel.	
<i>parcel_species</i>		
< <i>species name</i> >	Mass fraction of this species. Repeat for each species. Mass fractions must sum to 1.	
<i>turb dispersion control</i>		
<i>dispersion_model</i>	Turbulent dispersion flag. <i>DEFAULT</i> = No turbulent dispersion, <i>OROURKE</i> = O'Rourke turbulent dispersion, <i>TKE_PRESERVING</i> = tke-preserving turbulent dispersion.	
<i>porous media interaction control</i>		
<i>porous_trap(parcel_model)</i>	<i>NONE</i> = No interaction with porous media, <i>TRAP_ONLY</i> = All parcels are trapped when they encounter porous media.	
<i>remove_control</i>		
<i>tiny_parcel_mass_ratio</i>	Minimum parcel mass ratio. Parcels with a mass smaller than this value times the mass per parcel at its introduction will be removed.	1e-3
	Mass per parcel at introduction is specified by <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>injection_control</i> > <i>injection_velocity_control</i> > <i>mass_per_parcel</i> or <i>parcel_introduction.in</i> > <i>boundary_injections</i> > <i>boundary_injection</i> > <i>injection_control</i> > <i>mass_per_parcel</i> .	
<i>tiny_parcel_radius</i>	Minimum parcel radius (<i>m</i>). Parcels with a radius smaller than this value will be removed.	
<i>force_control</i>		
<i>drag_control</i>		
<i>drag_model</i>	Drag model for this parcel.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	<p><i>NONE</i> = No drag, <i>SPHERE</i> = Basic correlation for spherical particles, <i>TRACER</i> = Particles move with flow, <i>CLIFT_GRACE_WEBER</i> = Clift-Grace-Weber drag correlation (default recommendation), <i>STOKES</i> = Stokes' Law drag model for low-Reynolds-number flows, <i>STOKES_CUNNINGHAM</i> = Stokes' Law with the Cunningham correction for low-Reynolds-number flows with non-continuum effects, <i>HAIDER_LEVENSPIEL</i> = Haider-Levenspiel model for non-spherical parcels.</p>	
<i>sphericity</i>	Sphericity of the solid parcel, defined as the ratio of the surface area of a sphere of the parcel's volume divided by the surface area of the parcel. Used only when <i>drag_model</i> = <i>HAIDER_LEVENSPIEL</i> . If <i>sphericity</i> = 1, CONVERGE uses the <i>CLIFT_GRACE_WEBER</i> model instead of the <i>HAIDER_LEVENSPIEL</i> model.	
<i>added_mass_control</i>		
<i>active</i>	0 = No added mass force, 1 = Added mass force is active.	
<i>added_mass_coefficient</i>	Added mass coefficient C_a .	0 - 1
<i>lift_control</i>		
<i>lift_model</i>	<i>NONE</i> = No lift model, <i>SAFFMAN_MEI</i> = Saffman-Mei lift model.	
<i>undisturbed_fluid_force_flag</i>	0 = No undisturbed fluid force, 1 = Undisturbed fluid force is active.	
<i>thermal_exchange_control</i>		
<i>convection_model</i>	<i>NONE</i> = No convection model, <i>RANZ_MARSHALL</i> = Use a Ranz-Marshall type of correlation to model convection heat transfer.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>turbulence_modification_model</i>	<i>NONE</i> = No turbulence modification. This parameter is reserved for future use.	
<i>gas_parcels</i>		
<i>active</i>	0 = No gas-phase parcels. Reserved for future use.	

Parcel Introduction: *parcel_introduction.in*

If you specify parcel modeling in your simulation, you must specify *parcel_introduction.in* to instruct CONVERGE how to introduce parcels to the domain.

```
version: 3.1
---

injections:
  - injection:
      injection_name: Injector_0
      stream_id: 0
      injected_parcel:
        parcel_type: liquid_parcel
        parcel_name: liquid(parcel)_1
      penet_control:
        liquid_frac: 0.99
        vapor_frac: 0.001
        vapor_bin_size: 0.004
      injection_control:
        cone_distribution: CLUSTER
        dynamic_spray_cone_angle_flag: 0
        azimuth_angle_start: 0
        azimuth_angle_end: 360
        spray_bc_model: CONVENTIONAL
      injection_velocity_control:
        injection_velocity_mode: RATE_SHAPE
        injection_velocity: [-1.0, 1.0, 2.0]
        velocity_randomness_fraction: 0.01
        mass_per_parcel: 1e-12
        injection_ratio:
          type : mass
          value: 0.1
      position:
        angle_xy_inj: [0, 0, 0]
        angle_xz_inj: 0
        cone_flag: 1
        temporal_type: SEQUENTIAL
      period: 0
      start_time: 100
      duration: 10
      tot_mass: 0.0033
      temp: 300
      tke: 3000
      eps: 300
      inject_distribution: ROSIN-RAMMLER
      q_rr: 3.27
      swirl_fraction: 0
      discharge_coeff_flag: 1
      discharge_coeff_model:
        cd: 1
        cv: 1
      rate_shape: imp_rate_shape1.in
      lisa_model:
```

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

```
active: 0
injection_pres: 101300.0
khact_model:
    active: 0
    nozzle_flow_data: converge
vof_oneway_mapping:
    filename: vof_spray.dat
    transformation: NONE
    temp_map:
        active: 0
        scale: 1
        offset: 0
    tke_map:
        active: 0
        scale: 1
        offset: 0
    eps_map:
        active: 0
        scale: 1
        offset: 0
    mass_per_parcel: 4.5e-12
    liq_frac_threshold: 0.4
    init_cell_turb: 0
nozzle_control:
    num_parcels_per_nozzle: 3000000
    coord_sys: POLAR-COPY
    polar_cpy_num: 6
nozzles:
    - nozzle:
        nozzle_name: Nozzle_0
        geometry:
            diameter: 7.75e-05
            length: 2e-05
            position: [0, 0, 0]
            orientation: [0, 0, -1]
            radial_dist: 0.0028
            axial_dist: 0
            theta: 0
            angle_xy: 0
            angle_xz: 27
        spray_cone:
            inj_radius: 1.75e-05
            angle: 26
            thickness: 5
            smd: 3.36e-05
            amp_distort: 0
boundary_injections:
    - boundary_injection:
        injection_control:
            boundary_id: 1
            injected_parcel:
                parcel_type: liquid_parcel
                parcel_name: liquid_parcel_2
            temporal_type: SEQUENTIAL
            period: 0.0
            start_time: 0.005
            duration: 0.1
            temp: 300
            inject_distribution: uniform
            smd: 1e-04
            q_rr: 3.5
            injection_velocity_control:
                method: NORMAL_DIRECTION
                velocity_direction: [0, 0, -1]
                velocity_magnitude: 10.0
                velocity_randomness_fraction: 0.01
                injection_ratio:
                    type: AREA
                    value: 0.1
            mass_per_parcel: 1.0e-12
```

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

```
predefined_Shape_injection:
```

Figure 25.89: An example *parcel_introduction.in* file.

Table 25.62: Format of *parcel_introduction.in*.

Parameter	Description	Typical value
<i>injections</i>		
- <i>injection</i>	This settings sub-block specifies the configuration of an injector.	
<i>injection_name</i>	Name of this injector.	
<i>stream_id</i>	Stream ID for this injector.	
<i>injected.Parcel</i>	Parcel type sub-block.	
<i>parcel_type</i>	LIQUID_PARCEL = Liquid-phase parcel, SOLID_PARCEL = Solid-phase parcel.	
<i>parcel_name</i>	Name of this parcel.	
<i>penet_control</i>	Penetration control sub-block.	
<i>liquid_frac</i>	Liquid fuel mass fraction for calculating 0.95 spray penetration during post-processing.	
<i>vapor_frac</i>	Fuel vapor mass fraction for calculating 1e-3 vapor penetration.	
<i>vapor_bin_size</i>	Penetration bin size (m).	1e-3
<i>injection_control</i>		
<i>cone_distribution</i>	CLUSTER = Cluster injected parcels near the center of the cone, UNIFORM = Do not cluster parcels.	
<i>dynamic_spray_cone_angle_flag</i>	0 = Do not use a dynamic spray cone angle, 1 = Use a dynamic spray cone angle (requires dynamic_spray_cone_angle.in).	
<i>azimuth_angle_start</i>	Starting angle for partial-azimuth injection (deg).	0
<i>azimuth_angle_end</i>	Ending angle for partial-azimuth injection (deg).	360
<i>spray_bc_model</i>	Spray boundary condition type. CONVENTIONAL = Regular spray injection, ONEWAY = Map liquid parcels for injectors using data from a VOF simulation (vof_spray.dat required), ELSA = Eulerian-Lagrangian Spray Atomization model.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>injection_velocity_control</i>		
<i>injection_velocity_mode</i>	RATE_SHAPE = Specify a velocity rate-shape file, PROFILE = Specify a velocity profile file, CONSTANT = Constant velocity. For RATE_SHAPE, CONVERGE scales the parcel velocity to match the specified total mass. For PROFILE, CONVERGE uses the exact velocity profile specified.	
<i>injection_velocity</i>	Parcel velocity (specify u, v, and w). Not used for <i>injection_velocity_mode</i> = RATE_SHAPE or PROFILE.	
<i>velocity_randomness_fraction</i>	Random multiplicative factor applied to parcel injection velocity.	
<i>mass_per_parcel</i>	Mass (kg) per parcel. Not used for <i>injection_velocity_mode</i> = RATE_SHAPE.	
<i>injection_ratio</i>		
<i>type</i>	Reserved for future use.	
<i>value</i>	Reserved for future use.	
<i>position</i>	Position of the injector center along the x, y, and z axes (m).	
<i>angle_xy_inj</i>	Injector rotation angle (degrees) in the xy plane. Used only when <i>spray_bc_model</i> = 0 or 2. For illustrations, refer to Chapter 14 - Discrete Phase Modeling .	
<i>angle_xz_inj</i>	Injector rotation angle (degrees) in the xz plane. Used only when <i>spray_bc_model</i> = 0 or 2. For illustrations, refer to Chapter 14 - Discrete Phase Modeling .	
<i>cone_flag</i>	Spray cone type. 0 = Hollow cone spray, 1 = Solid cone spray.	
<i>temporal_type</i>	Specify whether the injection is SEQUENTIAL or CYCLIC.	
<i>period</i>	The period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero). Used only when <i>temporal_type</i> = CYCLIC. The period must be longer than the injection duration (see below). If the simulation starts	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	during a cyclic injection period, CONVERGE will ignore the amount of fuel that would have been injected prior to the start time.	
<i>start_time</i>	Start of injection (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero). This parameter can be a tabular profile (<i>inject_start_time.in</i>) for variable RPM cases. See Chapter 11 - Discrete Phase Modeling .	
<i>duration</i>	Duration of injection (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero). This parameter can be a tabular profile (<i>inject_duration.in</i>) for variable RPM cases. See Chapter 11 - Discrete Phase Modeling .	
<i>tot_mass</i>	Total spray mass (kg) injected in the computational domain for the current injector. This parameter can be a tabular profile (<i>inject_mass.in</i>) for variable RPM cases. See Chapter 11 - Discrete Phase Modeling . For sector simulations, <i>tot_mass</i> is for only the simulated nozzles (i.e., <i>tot_mass</i> does not represent the injected mass in the entire cylinder). If this injector has multiple nozzles, CONVERGE will split this mass equally between each nozzle.	
	When the steady-state solver is active (inputs.in > solver_control > steady_solver = 1), this parameter corresponds to mass injected per unit time (kg/s).	
<i>temp</i>	Liquid spray temperature (K) at the time of injection.	
<i>tke</i>	Turbulent kinetic energy (m^2/s^2) at the time of injection.	
<i>eps</i>	Turbulent dissipation (m^2/s^3) at the time of injection.	
<i>inject_distribution</i>	Injection distribution.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	<p><i>BLOB</i> = No distribution, <i>CHI-SQUARED</i> = Chi-squared distribution, <i>ROSIN-RAMMLER</i> = Rosin-Rammler distribution, <i>UNIFORM</i> = Injected parcel radius is set to $0.5 * \text{smd_dist}$. This option allows for a constant injected radius that is independent of the nozzle diameter (<i>parcel_introduction.in</i> > <i>injections</i> > <i>injection nozzles</i> > <i>nozzle</i> > <i>geometry</i> > <i>diameter</i>). Enter a file name (e.g., injdist.in) instead of a number to specify an injection profile in a input file.</p>	
<i>q_rr</i>	Rosin-Rammler distribution parameter 3.5 (used for any model in which you have activated the Rosin-Rammler size distribution).	3.5
<i>swirl_fraction</i>	The fraction of the spray that is in the azimuthal direction in a cylindrical coordinate system. Must be between -1 and 1, inclusive. Valid only for a hollow cone (<i>cone_flag</i> = 0). Refer to the Injector Inputs section in Chapter 14 - Discrete Phase Modeling for a more detailed definition of <i>swirl_fraction</i> .	
<i>discharge_coeff_flag</i>	Nozzle discharge coefficient model flag. 0 = No discharge coefficient model, 1 = Discharge coefficient model with internally calculated nozzle velocity coefficient (C_v), 2 = Discharge coefficient model with user-specified nozzle velocity coefficient (C_v).	
<i>discharge_coeff_model</i>		
<i>cd</i>	Nozzle discharge coefficient (C_d). Specify a constant value or, for a variable discharge coefficient, a file name (e.g., discharge_coeff_inj<num>.in).	
<i>cv</i>	Nozzle velocity coefficient (C_v). Specify a constant value or, for a variable velocity coefficient, a file name (e.g., velocity_coeff_inj<num>.in). Used only when <i>discharge_coeff_flag</i> = 2.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>rate_shape</i>	Rate-shape profile. Specify CONSTANT for a constant rate-shape. Specify a file name (e.g., <i>rateshape_inj<num>.in</i>) for a variable rate-shape profile.	
<i>lisa_model</i>		
<i>active</i>	0 = Inactive, 1 = Active.	
<i>injection_pres</i>	Injection pressure (Pa) for initialization.	
<i>khact_model</i>		
<i>active</i>	0 = Inactive, 1 = Active.	
<i>nozzle_flow_data</i>	CONVERGE = Tke, eps, and nozzle contraction coefficients are calculated by CONVERGE, $<\text{filename}>$ = Tke, eps, and nozzle contraction coefficients are provided in a separate input file (provide file name).	
<i>vof_oneway_mapping</i>		
<i>filename</i>	VOF spray file name (must be <i>vof_spray.dat</i>).	
<i>transformation</i>	Map liquid parcels for injectors using data from a VOF simulation and perform the following transformation. NONE = No transformation, NOZZLE-DIAMETER = Use the nozzle diameter specified by <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>nozzles</i> > <i>nozzle</i> > <i>geometry</i> > <i>diameter</i> (use when the nozzle is not a full circle), TRANSLATION = Translate the position data in <i>vof_spray.dat</i> according to the difference in position between the Lagrangian simulation injector (given by <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>injection_control</i> > <i>position</i>) and the VOF simulation injector (given by <i>Injector_Center</i> in <i>vof_spray.dat</i>), ROTATION = Rotate the data in <i>vof_spray.dat</i> according to the difference in orientation between the Lagrangian simulation injector (given by <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> >	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	<p><i>injection_control > angle_xy_inj</i> and <i>angle_xz_inj</i>) and the VOF simulation injector (given by <i>Injector_Axi_Vec</i> in <i>vof_spray.dat</i>), <i>TRANSLATION-ROTATION</i> = Translate and then rotate the data in <i>vof_spray.dat</i> as described above, <i>INJECTOR-COPY</i> = Copy data from the base injector and then translate and rotate the data as described above (requires one base injector for which <i>transformation ≠ INJECTOR-COPY</i>), <i>NOZZLE-COPY</i> = Copy the attributes of the specified nozzle to the remaining nozzles (only one nozzle per injector is allowed with this option). This option requires you to provide <i>vof_spray_copy.in</i>.</p>	
<i>temp_map</i>		
<i>active</i>	0 = Use constant temperature (<i>parcel_introduction.in > injections > injection > injection_control > temp</i>), 1 = Read temperature from <i>vof_spray.dat</i> with scaling and offset.	
<i>scale</i>	Scale factor to apply to temperature read from <i>vof_spray.dat</i> .	1.0
<i>offset</i>	Offset (K) to apply to temperature read from <i>vof_spray.dat</i> . Offset is applied after scaling.	0.0
<i>tke_map</i>		
<i>active</i>	0 = Use constant turbulent kinetic energy (<i>parcel_introduction.in > injections > injection > injection_control > tke</i>), 1 = Read turbulent kinetic energy from <i>vof_spray.dat</i> with scaling and offset.	
<i>scale</i>	Scale factor to apply to turbulent kinetic energy read from <i>vof_spray.dat</i> .	1.0
<i>offset</i>	Offset (m^2/s^2) to apply to turbulent kinetic energy read from <i>vof_spray.dat</i> . Offset is applied after scaling.	0.0
<i>eps_map</i>		
<i>active</i>	0 = Use constant turbulent dissipation (<i>parcel_introduction.in > injections > injection > injection_control > eps</i>),	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	1 = Read turbulent dissipation from <i>vof_spray.dat</i> with scaling and offset.	
<i>scale</i>	Scale factor to apply to turbulent dissipation read from <i>vof_spray.dat</i> .	1.0
<i>offset</i>	Offset (m^2/s^3) to apply to turbulent dissipation read from <i>vof_spray.dat</i> . Offset is applied after scaling.	0.0
<i>mass_per_parcel</i>	Maximum mass (in kg) that CONVERGE assigns to each liquid parcel.	4.5e-12
<i>liq_frac_threshold</i>	If a cell has a liquid volume fraction equal to or greater than the specified threshold, CONVERGE injects liquid parcels according to the data from <i>vof_spray.dat</i> . If the volume fraction is less than the threshold, CONVERGE does not inject liquid parcels.	
<i>init_cell_turb</i>	0 = Do not use <i>vof_spray.dat</i> turbulence quantities to initialize cells, 1 = Use <i>vof_spray.dat</i> turbulence quantities to initialize cells.	
<i>nozzle_control</i>		
<i>num_parcels_per_nozzle</i>	Total number of computational parcels to be injected for each nozzle of the injector.	
<i>coord_sys</i>	Coordinate system for this nozzle. <i>POLAR</i> = Set up nozzle using a polar coordinate system, <i>CARTESIAN</i> = Set up nozzle using a Cartesian coordinate system, <i>POLAR-COPY</i> = Set up one nozzle using a polar coordinate system, and copy that nozzle azimuthally. Requires <i>polar_cpy_num</i> .	
<i>polar_cpy_num</i>	Total number of copied polar nozzles.	
<i>nozzles</i>		
<i>- nozzle</i>		
<i>nozzle_name</i>	Name of this nozzle.	
<i>geometry</i>		
<i>diameter</i>	Nozzle diameter (m). Must be a constant value or a file name (e.g., <i>noz_diameter_inj<num>.in</i>).	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>length</i>	Nozzle length (<i>m</i>). Used only when the KH-ACT model is active.	
<i>position</i>	Position of the nozzle along the x, y, and z axes (<i>m</i>).	
<i>orientation</i>	Spray orientation unit vector (used only when <i>parcel_introduction.in > injections > injection > nozzle_control > coord_sys = CARTESIAN</i>).	
<i>radial_dist</i>	Nozzle radial position (<i>m</i>). Used only when <i>parcel_introduction.in > injections > injection > nozzle_control > coord_sys = POLAR</i> or <i>POLAR-COPY</i> . For illustrations, refer to Chapter 14 - Discrete Phase Modeling .	
<i>axial_dist</i>	Nozzle axial position (<i>m</i>). Used only when <i>parcel_introduction.in > injections > injection > nozzle_control > coord_sys = POLAR</i> or <i>POLAR-COPY</i> . For illustrations, refer to Chapter 14 - Discrete Phase Modeling .	
<i>theta</i>	Nozzle azimuthal position (<i>degrees</i>). Used only when <i>parcel_introduction.in > injections > injection > nozzle_control > coord_sys = POLAR</i> or <i>POLAR-COPY</i> . For illustrations, refer to Chapter 14 - Discrete Phase Modeling .	
<i>angle_xy</i>	Nozzle rotation angle (<i>degrees</i>) in the xy plane. Used only when <i>parcel_introduction.in > injections > injection > nozzle_control > coord_sys = POLAR</i> or <i>POLAR-COPY</i> . For illustrations, refer to Chapter 14 - Discrete Phase Modeling .	
<i>angle_xz</i>	Nozzle rotation angle (<i>degrees</i>) in the xz plane. Used only when <i>parcel_introduction.in > injections > injection > nozzle_control > coord_sys = POLAR</i> or <i>POLAR-COPY</i> . For illustrations, refer to Chapter 14 - Discrete Phase Modeling .	
<i>spray_cone</i>		

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>inj_radius</i>	Circular injection radius (<i>m</i>). For a hollow cone spray, parcels are injected on the circumference of the circle. For a solid cone spray, parcels are injected inside of the circle (<i>i.e.</i> , an injected parcel is randomly initialized at a point inside the circle so that, if enough parcels are injected, the circle would appear to be filled in). Specify a file name (<i>e.g.</i> , <i>inj_radius inj<num>.in</i>) to set up a cone angle profile.	
<i>angle</i>	Spray cone angle (<i>degrees</i>). Specify a file name (<i>e.g.</i> , <i>cone_angle inj<num>.in</i>) to set up a cone angle profile.	
<i>thickness</i>	Spray thickness (<i>degrees</i>). Used only for hollow cone sprays (<i>parcel_introduction.in > injections > injection > injection_control > cone_flag = 0</i>).	
<i>smd</i>	Sauter mean diameter (<i>m</i>) when a distribution is used (<i>i.e.</i> , <i>parcel_introduction.in > injections > injection > injection_control > inject_distribution = CHISQUARED, ROSIN-RAMMLER, or UNIFORM</i>).	
<i>amp_distort</i>	Initial drop distortion amplitude for TAB breakup model.	
<i>boundary_injections*</i>		
<i>- boundary_injection</i>	Sub-block for a boundary injection . Repeat this sub-block for each boundary injection.	
<i>injection_control</i>		
<i>boundary_id</i>	Boundary ID of the boundary from which parcels are introduced. Must be a WALL or INFLOW boundary.	
<i>injected_parcel</i>		
<i>parcel_type</i>	<i>LIQUID_PARCEL</i> or <i>SOLID_PARCEL</i> .	
<i>parcel_name</i>	Parcel name.	
<i>temporal_type</i>	<i>SEQUENTIAL</i> or <i>CYCLIC</i> .	
<i>period</i>	Period (in seconds if <i>inputs.in > simulation_control > crank_flag = 0</i> or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) of	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	parcel injection. Used only when <i>temporal_type</i> = CYCLIC.	
<i>start_time</i>	Injection start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>duration</i>	Injection duration (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>temp</i>	Temperature (K) of parcels injected from this boundary.	
<i>inject_distribution</i>	Injection size distribution. <i>CHI-SQUARED</i> = Chi-squared distribution, <i>ROSIN-RAMMLER</i> = Rosin-Rammler distribution, <i>UNIFORM</i> = Injected parcel radius is set to 0.5*smd.	
	Enter a file name (e.g., <i>injdist.in</i>) to specify an injection profile in an input file.	
<i>smd</i>	Sauter mean diameter (<i>m</i>) of the injection distribution.	
<i>q_rr</i>	Rosin-Rammler distribution parameter 3.5 (used only when <i>inject_distribution</i> = <i>ROSIN-RAMMLER</i>).	
<i>injection_velocity_control</i>		
<i>method</i>	<i>NORMAL_DIRECTION</i> = Velocities of injected parcels are normal to the boundary, <i>PRESCRIBED_DIRECTION</i> = Velocities of injected parcels are in a user-specified direction, <i>INJECT_WITH_FLOW</i> = Injected parcels have the same velocity as the local flow (available only for INFLOW boundaries).	
<i>velocity_direction</i>	Direction of injected parcel velocities. Specify a vector with a magnitude of 1. Used only when <i>method</i> = <i>PRESCRIBED_DIRECTION</i> .	
<i>velocity_magnitude</i>	Magnitude of injected parcel velocities (<i>m/s</i>). Not used when <i>method</i> = <i>INJECT_WITH_FLOW</i> .	
<i>velocity_randomness_fraction</i>	Reserved for future use.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>injection_ratio</i>		
<i>type</i>	MASS = Injection ratio is the parcel mass fraction (available only if <i>method</i> = <i>INJECT_WITH_FLOW</i>), VOLUME = Injection ratio is the parcel volume fraction (available only if <i>method</i> = <i>INJECT_WITH_FLOW</i>), AREA = Injection ratio is a fraction of the boundary surface area.	
<i>value</i>	Value of the injection ratio. Must be equal to or greater than 0 and less than 1.	
<i>mass_per.Parcel</i>	Mass per injected parcel (kg).	
<i>predefined_Shape_injection</i>		

*Blocks or parameters indicated with an asterisk are optional.

Parcel-Parcel Interaction: *parcel.Parcel_interaction.in*

If you specify parcel modeling in your simulation, you must specify *parcel.Parcel_interaction.in* to instruct CONVERGE how parcels interact with each other.

```
version: 3.1
---

parcel.Parcel_interactions:
  - collision_control:
    collision_mesh_flag: 0
    collision_pair:
      - parcel:
        parcel_type: liquid.Parcel
        parcel_name: liquid.Parcel_1
      - parcel:
        parcel_type: liquid.Parcel
        parcel_name: liquid.Parcel_2
    collision_model: NONE
    outcome_model: POST
    outcome.Parcel:
      parcel_type: liquid.Parcel
      parcel_name: liquid.Parcel_3
  - collision_control:
    collision_mesh_flag: 0
    collision_pair:
      - parcel:
        parcel_type: liquid.Parcel
        parcel_name: liquid.Parcel_1
      - parcel:
        parcel_type: solid.Parcel
        parcel_name: solid.Parcel_1
    collision_model: NONE
    outcome_model: POST
    outcome.Parcel:
      parcel_type: solid.Parcel
      parcel_name: solid.Parcel_2
  - collision_control:
    collision_mesh_flag: 0
    collision_pair:
      - parcel:
```

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

```
parcel_type:          liquid.Parcel
parcel_name:         liquid.Parcel_1
- parcel:
  parcel_type:      liquid.Parcel
  parcel_name:     liquid.Parcel_1
collision_model:    NONE
outcome_model:      POST
outcome_parcel:
  parcel_type:      liquid.Parcel
  parcel_name:     liquid.Parcel_1
```

Figure 25.90: An example *parcel.Parcel_interaction.in* file.

Table 25.63: Format of *parcel.Parcel_interaction.in*.

Parameter	Description
<i>parcel.Parcel_interaction</i>	
- <i>collision_control</i>	Repeat this settings block for each set of collision partners.
<i>collision_mesh_flag</i>	0 = Do not use a collision mesh, 1 = Use a collision mesh.
<i>collision_pair</i>	
- <i>parcel</i>	Provide two copies of this settings block.
<i>parcel_type</i>	<i>LIQUID_PARCEL</i> = This collision partner is liquid phase, <i>SOLID_PARCEL</i> = This collision partner is solid phase.
<i>parcel_name</i>	Name of this parcel.
<i>collision_model</i>	Collision model flag. <i>NONE</i> = No collision model, <i>OROURKE</i> = O'Rourke collision model, <i>NTC</i> = NTC collision model.
<i>outcome_model</i>	Collision outcomes flag. <i>OROURKE</i> = O'Rourke outcomes, <i>POST</i> = Post outcomes.
<i>outcome_parcel</i>	Settings block to describe the parcel created by a collision.
<i>parcel_type</i>	<i>LIQUID_PARCEL</i> = The outcome parcel is liquid phase, <i>SOLID_PARCEL</i> = The outcome parcel is solid phase.
<i>parcel_name</i>	Name of the outcome parcel.

Parcel-Wall Interaction: *parcel.Wall_interaction.in*

If you specify *parcels.in > [type] > wall_interaction_model = FILE*, you must provide *parcel.Wall_interaction.in* to instruct CONVERGE how parcels interact with solid boundaries.

```
version: 3.1
---
liquid.wall_interaction_control:
  combine_parcels:
    active: 0
```

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

```
start_time: -999999
end_time: -999999
max_film_parcels_per_boundary_node: 50
default_wall_control:
    wall_model: FILM
    init_wall_film: 0
    film_control:
        film_mesh:
            active: 0
            embedding_scale: 0
        rebound:
            weber_no: 5
        splash:
            model: OROURKE
            criterion_flag: 0
            critical_value: 3330
            fraction: 0.7
            tstar_critical: 1.1
            temp_boil: USE_RAOUULTS_LAW
            wruck_model_flag: 0
        strip:
            active: 0
            time_const: 12
            size_const: 0.5
    separation:
        const: 3
    evap_control:
        active: 1
        evap_source_flag: 2
        evap_species: H2O
        scaling:
            heat_trans_coeff: 1
            mass_trans_coeff: 1
    temp_discretization:
        active: 0
        height: 1000
        layers_per_film: 10
        urea_model: NONE
custom_wall_control:
    - wall_interaction:
        parcel_name: liquid_parcel_1
        boundary_id: ALL
        wall_model: FILM
        init_wall_film: 0
        film_control:
            rebound:
                weber_no: 5
            splash:
                model: OROURKE
                criterion_flag: 0
                critical_value: 3330
                fraction: 0.7
                tstar_critical: 1.1
                temp_boil: USE_RAOUULTS_LAW
                wruck_model_flag: 0
            strip:
                active: 0
                time_const: 12
                size_const: 0.5
            separation:
                const: 3
        evap_control:
            active: 1
            evap_source_flag: 2
            evap_species: H2O
            scaling:
                heat_trans_coeff: 1
                mass_trans_coeff: 1
        temp_discretization:
            active: 0
            height: 1000
```

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

```
    layers_per_film:          10
    urea_model:                NONE
solid_wall_interaction_control:
    default_wall_control:
        wall_interaction_model:
            wall_model:                  REBOUND
            coefficient_of_restitution:
                type:                      CONSTANT
                normal_direction:          0.8
                tangential_direction:     0.4
                stick_percentage:          10
custom_wall_control:
    - wall_interaction:
        parcel_name:               solid.Parcel_1
        boundary_id:                [10, 11]
        wall_interaction_model:
            wall_model:                  REBOUND
            coefficient_of_restitution:
                type:                      CONSTANT
                normal_direction:          0.8
                tangential_direction:     0.4
                stick_percentage:          10
```

Figure 25.91: An example *parcel_wall_interaction.in* file.

Table 25.64: Format of *parcel_wall_interaction.in*.

Parameter	Description	Typical value
<i>liquid_wall_interaction_control</i>		
<i>combine_parcels</i>		
<i>active</i>	0 = Do not combine film parcels, 1 = Combine film parcels.	
<i>start_time</i>	CONVERGE will start combining film parcels at this time (in seconds if <i>simulation_control > crank_flag = 0</i> in <i>inputs.in</i> or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>end_time</i>	CONVERGE will stop combining film parcels at this time (in seconds if <i>simulation_control > crank_flag = 0</i> in <i>inputs.in</i> or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>max_film_parcels_per_boundary_node</i>	Maximum number of film parcels per fluid cell. CONVERGE will combine film parcels in the cell if this number is exceeded.	
<i>default_wall_control</i>	This block controls the default liquid-parcel-wall interaction settings. You can optionally specify custom interaction settings for specific parcel names.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>wall_model</i>	Spray-wall interaction model flag. <i>REBOUND_SLIDE</i> = Rebound/slide model, <i>FILM</i> = Wall film model, with the updated evaporation model, <i>VANISH</i> = Drops vanish when they impinge on a solid boundary, <i>FILM_LEGACY</i> = Legacy approach for film momentum calculation.	
<i>init_wall_film</i>	Initialization of wall film. 0 = No initialization, 1 = Initialization (requires <i>film_init.in</i>).	
<i>film_control</i>		
<i>film_mesh</i>		
<i>active</i>	0 = No adaptive film mesh, 1 = Adaptive film mesh.	
<i>embedding_scale</i>	The embed level used to create the adaptive film mesh. For example, a value of 2 will yield a film mesh size that is 1/4 of the base grid size.	
<i>rebound</i>		
<i>weber_no</i>	Weber rebound number. When the parcel's Weber number is lower than this value, the parcel will rebound. This parameter is used by the splash model (<i>splash > model</i>).	5.0
<i>splash</i>		
<i>model</i>	<i>OROURKE</i> = O'Rourke splash model, <i>KUHNKE</i> = Kuhnke splash model (dependent on wall temperature and wetness), <i>BAI-GOSMAN</i> = Bai-Gosman splash model (dependent on wall temperature and Weber number).	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>criterion_flag</i>	Wall film model splash criterion flag. 0 = Determine splashing based on a parameter defined by Weber number, film thickness and liquid viscosity, 1 = Determine splashing based on Weber number only. This parameter is used only for the O'Rourke splash model (<i>i.e.</i> , when <i>splash > model</i> = OROURKE).	
<i>critical_value</i>	Critical value for splashing. The meaning of this value depends on the value of E_{crit}^2 (<i>splash_crit_flag</i> = 0), case dependent for We_{splash} (<i>splash_crit_flag</i> = 1)	3330.0 for
<i>fraction</i>	Fraction of drop mass splashed. This parameter is used only for the O'Rourke splash model (<i>i.e.</i> , when <i>splash > model</i> = OROURKE).	0.5 - 1.0
<i>tstar_critical</i>	Critical value of the non-dimensionalized wall temperature T^* used in the Kuhnke film splash model.	1.1
<i>temp_boil</i>	Parcel boiling temperature used to evaluate splash outcomes (K). This optional parameter applies only when <i>splash > model</i> = KUHNKE or BAI-GOSMAN and only when <i>feature_control > urea_flag > 0</i> in inputs.in . If this parameter is specified, the value must be between 200 K and 600 K. If this parameter is not specified, CONVERGE determines the parcel boiling temperature from liquid.dat using Raoult's Law.	
<i>wruck_model_flag</i>	Wruck spray-wall heat transfer model flag. 0 = Wruck model deactivated. 1 = Wruck model activated.	
<i>strip</i>		
<i>active</i>	Film stripping flag. 0 = No film stripping, 1 = Film stripping.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>time_const</i>	Film stripping time constant. Used only when <i>strip</i> > <i>active</i> = 1.	
<i>size_const</i>	Film stripping size constant. Used only when <i>strip</i> > <i>active</i> = 1.	
<i>separation</i>		
<i>const</i>	Separation constant. This parameter is used only for the O'Rourke splash model (<i>i.e.</i> , when <i>splash</i> > <i>model</i> = OROURKE).	3.0
<i>evap_control</i>		
<i>active</i>	Film evaporation model flag. 0 = No film evaporation, 1 = Film evaporation is enabled.	
<i>evap_source_flag</i>	Evaporation source flag. 0 = All species will evaporate to <i>evap_species</i> , 1 = Single-component species will evaporate to their respective species and multi-component liquid species will evaporate to a composite species (if composites are used), 2 = Single-component species will evaporate to their respective species and multi-component liquid species will evaporate to their base species.	
<i>evap_species</i>	Name of the species that will be sourced when evaporation occurs. Used only when <i>evap_source_flag</i> = 0.	
<i>scaling</i>		
<i>heat_trans_coeff</i>	Scaling applied to the drop heat transfer coefficient.	
<i>mass_trans_coeff</i>	Scaling applied to the drop mass transfer coefficient.	
<i>temp_discretization</i>		
<i>active</i>	Film temperature discretization flag. 0 = Use the Uniform Temperature Model to compute film thermal transfer. 1 = Use the Uniform Temperature Model to compute film thermal transfer for films thinner than <i>film_height</i> and use the Discretized Temperature Model for thicker films.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>height</i>	If the film height (<i>m</i>) is greater than this threshold, CONVERGE will use the Discretized Temperature Model.	1000
<i>layers_per_film</i>	Number of finite volume cells into which the film will be discretized when using the Discretized Temperature Model.	
<i>urea_model</i>	1 = Urea-water solution depletion calculation with the multi-component model, 2 = Urea-water solution depletion calculation with the molten solid approach 3 = Urea-water solution depletion calculation with the detailed decomposition approach.	
<i>custom_wall_control</i> *	Specify custom interaction settings for specific liquid parcel names.	
- <i>wall_interaction</i>	Repeat this sub-block for each custom wall interaction.	
<i>parcel_name</i>	Parcel name to which this custom wall control applies.	
<i>boundary_id</i>	List of boundary IDs on which this custom wall control applies. You can alternately specify all boundaries with the keyword ALL.	
<i>wall_model</i>	Spray-wall interaction model flag. <i>REBOUND_SLIDE</i> = Rebound/slide model, <i>FILM</i> = Wall film model, with the updated evaporation model, <i>VANISH</i> = Drops vanish when they impinge on a solid boundary, <i>FILM_LEGACY</i> = Legacy approach for film momentum calculation.	
<i>init_wall_film</i>	Initialization of wall film. 0 = No initialization, 1 = Initialization (requires <i>film_init.in</i>)	
<i>film_control</i>		
<i>rebound</i>		

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>weber_no</i>	Weber rebound number. When the parcel's Weber number is lower than this value, the parcel will rebound. This parameter is used by the splash model (<i>splash > model</i>).	5.0
<i>splash</i>		
<i>model</i>	OROURKE = O'Rourke splash model, KUHNKE = Kuhnke splash model (dependent on wall temperature and wetness), BAI-GOSMAN = Bai-Gosman splash model (dependent on wall temperature and Weber number).	
<i>criterion_flag</i>	Wall film model splash criterion flag. 0 = Determine splashing based on a parameter defined by Weber number, film thickness and liquid viscosity, 1 = Determine splashing based on Weber number only. This parameter is used only for the O'Rourke splash model (<i>i.e.</i> , when <i>splash > model</i> = OROURKE).	
<i>critical_value</i>	Critical value for splashing. The meaning of this value depends on the value of <i>splash_crit_flag</i> (see right). This parameter is used only for the O'Rourke splash model (<i>i.e.</i> , when <i>splash > model</i> = OROURKE). E_{crit}^2 (<i>splash_crit_flag</i> = 0), case dependent for We_{splash} (<i>splash_crit_flag</i> = 1)	3330.0 for E_{crit}^2 (<i>splash_crit_flag</i> = 0), case dependent for We_{splash} (<i>splash_crit_flag</i> = 1)
<i>fraction</i>	Fraction of drop mass splashed. This parameter is used only for the O'Rourke splash model (<i>i.e.</i> , when <i>splash > model</i> = OROURKE).	0.5 - 1.0
<i>tstar_critical</i>	Critical value of the non-dimensionalized wall temperature T^* used in the Kuhnke film splash model.	1.1
<i>temp_boil</i>	Parcel boiling temperature used to evaluate splash outcomes (K). This optional parameter applies only when <i>splash > model</i> = KUHNKE or BAI-GOSMAN and only when <i>feature_control > urea_flag > 0</i> in <u>inputs.in</u> . If this parameter	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	is specified, the value must be between 200 K and 600 K. If this parameter is not specified, CONVERGE determines the parcel boiling temperature from liquid.dat using Raoult's Law.	
<i>wruck_model_flag</i>	Wruck spray-wall heat transfer model flag. 0 = Wruck model deactivated. 1 = Wruck model activated.	
<i>strip</i>		
<i>active</i>	Film stripping flag. 0 = No film stripping, 1 = Film stripping.	
<i>time_const</i>	Film stripping time constant. Used only when <i>strip > active = 1</i> .	
<i>size_const</i>	Film stripping size constant. Used only when <i>strip > active = 1</i> .	
<i>separation</i>		
<i>const</i>	Separation constant. This parameter is used only for the O'Rourke splash model (<i>i.e.</i> , when <i>splash > model = OROURKE</i>).	3.0
<i>evap_control</i>		
<i>active</i>	Film evaporation model flag. 0 = No film evaporation, 1 = Film evaporation is enabled.	
<i>evap_source_flag</i>	Evaporation source flag. 0 = All species will evaporate to <i>evap_species</i> , 1 = Single-component species will evaporate to their respective species and multi-component liquid species will evaporate to a composite species (if composites are used), 2 = Single-component species will evaporate to their respective species and multi-component liquid species will evaporate to their base species.	
<i>evap_species</i>	Name of the species that will be sourced when evaporation occurs. Used only when <i>evap_source_flag = 0</i> .	
<i>scaling</i>		

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
<i>heat_trans_coeff</i>	Scaling applied to the drop heat transfer coefficient.	
<i>mass_trans_coeff</i>	Scaling applied to the drop mass transfer coefficient.	
<i>temp_discretization</i>		
<i>active</i>	Film temperature discretization flag. 0 = Use the Uniform Temperature Model to compute film thermal transfer. 1 = Use the Uniform Temperature Model to compute film thermal transfer for films thinner than <i>film_height</i> and use the Discretized Temperature Model for thicker films.	
<i>height</i>	If the film height (<i>m</i>) is greater than this 1000 threshold, CONVERGE will use the Discretized Temperature Model.	
<i>layers_per_film</i>	Number of finite volume cells into which the film will be discretized when using the Discretized Temperature Model.	
<i>urea_model</i>	1 = Urea-water solution depletion calculation with the multi-component model, 2 = Urea-water solution depletion calculation with the molten solid approach 3 = Urea-water solution depletion calculation with the detailed decomposition approach.	
<i>solid_wall_interaction_control</i>	This block controls the default solid-parcel-wall interaction settings. You can optionally specify custom interaction settings for specific parcel names.	
<i>default_wall_control</i>	This block controls the default solid-parcel-wall interaction settings. You can optionally specify custom interaction settings for specific parcel names.	
<i>wall_interaction_model</i>		
<i>wall_model</i>	Parcel-wall interaction model flag. REBOUND = Rebound model, SLIDE = Slide model, VANISH = Parcels vanish when they impinge on a solid boundary,	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	STICK = Parcels stick to a solid boundary when they impinge upon it.	
<i>coefficient_of_restitution</i>		
<i>type</i>	CONSTANT = Constant coefficient of restitution. This parameter is reserved for future use.	
<i>normal_direction</i>	Wall-normal multiplicative factor for the coefficient of restitution.	0 - 1
<i>tangential_direction</i>	Wall-tangential multiplicative factor for the coefficient of restitution.	0 - 1
<i>stick_percentage</i>	Percentage of parcels that will stick to the wall when <i>wall_interaction_model</i> = STICK.	
<i>custom_wall_control</i> *	Specify custom interaction settings for specific solid parcel names.	
- <i>wall_interaction</i>	Repeat this sub-block for each parcel.	
<i>parcel_name</i>	Name of this parcel.	
<i>boundary_id</i>	List of boundary IDs on which this custom wall control applies. You may alternately specify all boundaries with the keyword ALL.	
<i>wall_interaction_model</i>		
<i>wall_model</i>	Parcel-wall interaction model flag. REBOUND = Rebound model, SLIDE = Slide model, VANISH = Parcels vanish when they impinge on a solid boundary, STICK = Parcels stick to a solid boundary when they impinge upon it.	
<i>coefficient_of_restitution</i>		
<i>type</i>	CONSTANT = Constant coefficient of restitution. This parameter is reserved for future use.	
<i>normal_direction</i>	Wall-normal multiplicative factor for the coefficient of restitution.	0 - 1
<i>tangential_direction</i>	Wall-tangential multiplicative factor for the coefficient of restitution.	0 - 1
<i>stick_percentage</i>	Percentage of parcels that will stick to the wall when <i>wall_interaction_model</i> = STICK.	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Initialization of Wall Film: `film_init.in`

To initialize the wall film, set [`parcels.in`](#) > `default_wall_control` > `init_wall_file` = 1 and include a `film_init.in` file in your case setup. A wall film can be initialized on an entire boundary or in a circular or rectangular shape.

```
version: 3.1
---

wall_films:
  - wall_film:
      film_type: BOUNDARY
      parcel_type: LIQUID_PARCEL
      parcel_name: liquid_parcel_1
      boundary_id: 1
      property:
        film_mass_type: 0
        film_mass: 1e-05
        thickness: 1e-05
        film_temp: 300
        num_parcels: 10000
        drop_size: 1e-06
  - wall_film:
      film_type: CIRCLE
      parcel_type: LIQUID_PARCEL
      parcel_name: liquid_parcel_2
      stream_id: 0
      center: [0.082, 0.0, 0.07]
      radius: 1e-03
      normal_vector: [1.0, 0.0, 0.0]
      property:
        film_mass_type: 1
        film_mass: 1e-05
        thickness: 1e-05
        film_temp: 300
        num_parcels: 10000
        drop_size: 1e-06
  - wall_film:
      film_type: RECTANGLE
      parcel_type: LIQUID_PARCEL
      parcel_name: liquid_parcel_3
      stream_id: 0
      size: [2e-3, 1e-3]
      normal_vector: [1.0, 0.0, 0.0]
      tangential_vector: [0.0, 1.0, 0.0]
      property:
        film_mass_type: 1
        film_mass: 1e-05
        thickness: 1e-05
        film_temp: 300
        num_parcels: 10000
        drop_size: 1e-06
```

Figure 25.92: An example `film_init.in` file.

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Table 25.65: Format of the *wall_films* settings block.

Parameter	Description
<i>wall_films</i>	
- <i>wall_film</i>	Sub-block for an initial film. Repeat for each initial film.
<i>film_type</i>	<i>CIRCLE</i> = Circular initial film, <i>RECTANGLE</i> = Rectangular initial film, <i>BOUNDARY</i> = Initial film along a boundary.
<i>parcel_type</i>	<i>LIQUID_PARCEL</i> .
<i>parcel_name</i>	Parcel name.
<i>boundary_id</i>	The ID of the boundary on which to initialize the film. Specify only for <i>BOUNDARY</i> films.
<i>stream_id</i>	The ID of the stream in which to initialize the film. Specify only for <i>CIRCLE</i> and <i>RECTANGLE</i> films.
<i>center</i>	The x, y, and z coordinates (<i>m</i>) of the film center. Specify only for <i>CIRCLE</i> films.
<i>radius</i>	The radius (<i>m</i>) of the film. Specify only for <i>CIRCLE</i> films.
<i>size</i>	The width and length (<i>m</i>) of the film. Specify only for <i>RECTANGLE</i> films.
<i>normal_vector</i>	The x, y, and z coordinates of the initial film normal vector.
<i>tangential_vector</i>	The x, y, and z coordinates of the length axis of the film rectangle. Specify only for <i>RECTANGLE</i> films.
<i>property</i>	
<i>film_mass_type</i>	0 = Specify film mass, 1 = Specify film thickness.
<i>film_mass</i>	Total film mass (<i>kg</i>). Required only for <i>film_mass_type</i> = 0.
<i>thickness</i>	Total film thickness (<i>m</i>). Required only for <i>film_mass_type</i> = 1.
<i>film_temp</i>	Film temperature (<i>K</i>).
<i>num_parcels</i>	Total number of parcels in the wall film.
<i>drop_size</i>	Film parcel radius (<i>m</i>).

Dynamic Spray Cone Angle: *dynamic_spray_cone_angle.in*

To use a dynamic spray cone angle, set *parcels.in* > *injections* > *injection* > *injection_control* > *dynamic_spray_cone_angle_flag* = 1 and include a *dynamic_spray_cone_angle.in* file in your case setup.

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

```
version: 3.1
---
fuel_type:          1
density_ambient:   22.8
chamber_region_id: 0
pressure_injection: [2.0e5, 5.0e5, 3.0e5]
```

Figure 25.93: An example *dynamic_spray_cone_angle.in* file.

Table 25.66: Format of *dynamic_spray_cone_angle.in*.

Parameter	Description
<i>fuel_type</i>	0 = Naphtha, 1 = ULSD (ultra-low-sulfur diesel).
<i>density_ambient</i>	Ambient gas density (kg/m^3). If the value of this parameter is less than zero, CONVERGE calculates the ambient gas density from the region ID of the ambient chamber (see next row).
<i>chamber_region_id</i>	Region ID of the ambient chamber.
<i>pressure_injection</i>	List that specifies the injection pressure (in Pa) for each injector.

Injection Distribution: *injdist.in*

To specify an injection drop distribution profile, specify a file name (e.g., *injdist.in*) for [parcel introduction.in](#) > *injections* > *injection* > *injector_control* > *inject_distribution* and include that file in your case setup.

```
TABULAR
SEQUENTIAL
value    vol_frac      diameter
1        0.0            0.0e-06
2        1.08681E-05    10.0e-06
3        0.000141233    20.0e-06
.
.
49       0.999999986   480.0e-06
50       0.999999997   490.0e-06
51       1.0            500.0e-06
```

Figure 25.94: An example *injdist.in* file.

Table 25.67: Format of *injdist.in*.

Column	Header	Description
1	<i>value</i>	1, 2, 3, etc.
2	<i>vol_frac</i>	Cumulative volume fraction. Must be a cumulative distribution function (i.e., the first value must be for a volume fraction of 0.0, and the last value must be for a volume fraction of 1.0).
3	<i>diameter</i>	Diameter (m).

Injection Parameter Profiles

For each injector or nozzle, the mass flow rate, discharge and velocity coefficients, nozzle diameter, injection radius, and spray cone angle can vary in time. This variation is controlled in each case by an injection parameter profile file (e.g., a rate-shape file for mass flow). If the simulation time falls between two profile entries, CONVERGE interpolates to the current time.

A rate-shape file does not specify injection timing in an absolute sense. Rather, the total injection duration is specified by [*parcel introduction.in*](#) > *injections* > *injection* > *injection_control* > *duration*. CONVERGE scales the rate-shape to match the specified absolute duration. A rate-shape file with three relative temporal entries of [0, 1, 2] behaves identically to a rate-shape file with relative temporal entries of [0, 2, 4].

For any of these quantities, if you want to invoke a profile file, you must specify a file name for the relevant [*parcel introduction.in*](#) parameter and then include the corresponding profile file in your case setup.

Table 25.68: Injector rate-shape options.

Quantity	Parameter	Suggested file name	Required column header in rate-shape file
Mass flow	<i>parcels introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>injection_control</i> > <i>rate_shape</i>	<i>rateshape_inj<num>.in</i>	<i>rate_shape</i>
Discharge coefficient	<i>parcels introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>injection_control</i> > <i>discharge_coeff_model</i> > <i>Cd</i>	<i>discharge_coeff_i<num>.in</i>	<i>discharge_coeff</i>
Velocity coefficient	<i>parcels introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>injection_control</i> > <i>discharge_coeff_model</i> > <i>Cv</i>	<i>velocity_coeff_inj<num>.in</i>	<i>velocity_coeff</i>
Nozzle diameter	<i>parcels introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>nozzles</i> > <i>nozzle</i> > <i>geometry</i> > <i>diameter</i>	<i>noz_diameter_inj<num>.in</i>	<i>noz_diameter</i>
Injection radius	<i>parcels introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>nozzles</i> > <i>nozzle</i> > <i>spray_cone</i> > <i>inj_radius</i>	<i>inj_radius_inj<num>.in</i>	<i>inj_radius</i>
Spray cone angle	<i>parcels introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>nozzles</i> > <i>nozzle</i> > <i>spray_cone</i> > <i>angle</i>	<i>cone_angle_inj<num>.in</i>	<i>cone_angle</i>

All of these profile files have the same format. The first line of the file must be *TEMPORAL*. The second line indicates the temporal type: *CYCLIC* (must be followed by the period) or *SEQUENTIAL*.

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Table 25.69: Format of the injection parameter profile files.

Column	Header	Description
1	<i>crank</i> or <i>time</i>	Relative time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degree if crank_flag is non-zero. This time is relative to the start of injection time specified via parcel_introduction.in > injections > injection > injection_control > start_time.
2	<i>rate_shape</i> , <i>discharge_coeff</i> <i>f</i> , <i>velocity_coeff</i> , <i>noz_diameter</i> , <i>inj_radius</i> , or <i>cone_angle</i>	Injection parameter value at this time. For mass flow rate, CONVERGE normalizes the entire rate-shape to match the specified total injection mass.

```
TEMPORAL
SEQUENTIAL
CRANK    rate_shape
1        1057.2
2        1141.0
3        3713.2
4        12518.0
5        22231.0
6        30167.0
7        36601.0
8        39283.0
9        41372.0
10       39233.0
11       37451.0
12       38671.0
13       39180.0
14       42454.0
15       42291.0
16       41575.0
17       41254.0
18       39946.0
19       40217.0
20       40845.0
21       41472.0
22       40716.0
23       40475.0
24       40944.0
25       40662.0
```

Figure 25.95: An example *rateshape_inj<num>.in* file. The format of the other injection parameter profile files is identical except for the *rate_shape* column header.

Variable RPM

For variable RPM cases with spray injection, if the rate-shape is constant ([parcel_introduction.in](#) > injections > injection > injection_control > rate_shape = CONSTANT), then CONVERGE automatically generates a *rateshape_inj<num>.in* file for each injector. For variable RPM cases, you can include tabular profiles for injection start time, injection duration, and total mass injected. To activate this option, specify a file name for *injections* > *injection* > *injection_control* > *start_time*, *injectors* > *injector* > *injector_control* > *duration*, and/or *injectors* > *injector* > *injector_control* > *tot_mass*.

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

If you specify a profile for *tot_mass* and/or *duration*, you must also specify a profile for *start_time*. The number of entries in the *tot_mass* and/or *duration* profiles must match the number of entries in the *start_time* profile.

The table below lists the relevant [*parcel_introduction.in*](#) parameter for each quantity as well as the required column header in the corresponding rate-shape file.

Table 25.70: Injector rate-shape options.

Quantity	<i>parcel_introduction.in</i> parameter	Suggested file name	Required column header in rate-shape file
Start of injection time	<i>injections > injection > injection_control > start_time</i>	<i>start_time_inj<num>.in</i>	<i>inject_start_time</i>
Duration of injection	<i>injections > injection > injection_control > duration</i>	<i>duration_inj<num>.in</i>	<i>inject_duration</i>
Total mass injected	<i>injections > injection > injection_control > tot_mass</i>	<i>tot_mass_inj<num>.in</i>	<i>inject_mass</i>

The figure below shows an excerpt of [*parcel_introduction.in*](#) that specifies profiles for these parameters for a variable RPM engine case.

```
injections:
  - injection:
    injection_control:
      spray_bc_model: CONVENTIONAL
      position: [0.082, 0.0, 0.07]
      angle_xy_inj: 0.0
      angle_xz_inj: 0.0
      parcel_species:
        IC9H18: 1.0
      cone_flag: 1
      temporal_type: SEQUENTIAL
      start_time: start_time_inj<num>.in
      duration: duration_inj<num>.in
      tot_mass: tot_mass_inj<num>.in
```

Figure 25.96: An excerpt of *parcel_introduction.in* for a variable RPM case.

The figure below shows a sample *start_time_inj<num>.in* file. The first line of the file must be the keyword *TABULAR*, followed by the keyword *SEQUENTIAL* in the next row. The heading of the first column is the keyword *value*, which represents an index of the injection.

```
TABULAR
SEQUENTIAL
value   inject_start_time
1      -240
2      480
3      1200
4      1920
5      2640
6      3360
```

Figure 25.97: A sample *start_time_inj<num>.in* file that gives injection start times for a variable RPM engine simulation.

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Eulerian-Lagrangian Spray Atomization: elsa.in

To activate [ELSA](#), activate spray modeling and VOF modeling (set *inputs.in* > *feature_control* > *parcel_mode* > *liquid_parcel* = 1 and *feature_control* > *vof_flag* = 1) and include an *elsa.in* file in your case setup.

```
version: 3.1
---

elsa:
  - stream:
    stream_id: 0
    elsa_transition_criteria:
      elsa_transition_flag: 1
      elsa_transition_alpha: 0.9
      elsa_transition_area_ratio: 2
      elsa_min_diameter: 2e-8
      elsa_mass_per_parcel: 3e-8
      elsa_evap_flag: 0
      elsa_alpha_min: 0.0
      elsa_diameter_max: 0.01
      elsa_scale: 1.0
      elsa_smd_cutoff: 0.0
      elsa_liquid_penet_factor: 3.0
      elsa_liquid_penet_mass_percent: NINETY_SEVEN
    dpm_to_vof_transition_criteria:
      dpm_to_vof_transition_alpha: 0
    elsa_injectors:
      injectors:
        - injector:
          elsa_inj_name: injector_aa
          elsa_sigma_const_1: 4.0
          elsa_sigma_const_2: 1.0
        nozzles:
          - nozzle:
            elsa_noz_name: nozzle_aa
            elsa_region_id_1: 1
            elsa_region_id_2: 2
            elsa_transition_region_ids: [0, 1]
            elsa_passive_liquid: injector_aa_nozzle_aa_elsa
          - nozzle:
            elsa_noz_name: nozzle_bb
            elsa_region_id_1: 1
            elsa_region_id_2: 2
            elsa_transition_region_ids: [0]
            elsa_passive_liquid: injector_aa_nozzle_bb_elsa
```

Figure 25.98: An example *elsa.in* file.

Table 25.71: Format of *elsa.in*.

Parameter	Description	Typical value
<i>elsa</i>		
- <i>stream</i>	Sub-block for a stream. Repeat for each stream in which an ELSA model is active.	
<i>stream_id</i>	Stream ID.	
<i>elsa_transition_criteria</i>		
<i>elsa_transition_flag</i>	Flag to turn on ELSA transition. 0 = No ELSA parcel transition.	

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	1 = ELSA parcel transition activated.	
<i>elsa_transition_alpha</i>	Maximum gas volume fraction in a cell. The first ELSA criterion is satisfied if the gas volume fraction in a cell exceeds the specified value.	0.9
<i>elsa_transition_area_ratio</i>	Ratio between the fluid surface area density and the minimum fluid surface area density in a cell. The second ELSA criterion is satisfied if the computed ratio in a cell exceeds the specified ratio.	2.0
<i>elsa_min_diameter</i>	Minimum diameter of parcels generated during Lagrangian conversion (<i>m</i>).	2e-8
<i>elsa_mass_per_parcel</i>	Maximum mass per parcel generated during Lagrangian conversion (<i>kg</i>).	3e-8
<i>elsa_evap_flag</i>	Flag to turn on Eulerian fluid evaporation. 0 = No Eulerian fluid evaporation, 1 = Activate Eulerian fluid evaporation.	
<i>elsa_alpha_min</i>	Void fraction cutoff from vapor side (<i>i.e.</i> , $\alpha < (1 - \text{elsa_alpha_min})$) to trigger transition.	
<i>elsa_diameter_max</i>	Upper clipping bound for max diameter of Lagrangian particle.	0.01
<i>elsa_scale</i>	Scaling factor for the Lagrangian particle diameter.	1.0
<i>elsa_smd_cutoff</i>	Void fraction cutoff from vapor side (<i>i.e.</i> , $\alpha < (1 - \text{elsa_smd_cutoff})$) used in SMD calculation.	
<i>elsa_liquid penet_factor</i>	ELSA transition occurs when the liquid penetration exceeds the nozzle diameter by this factor.	
<i>elsa_liquid penet_percent</i>	Percentage mass to consider in liquid penetration to trigger ELSA transition. NINETY: 90%, NINETY_FIVE: 95%, NINETY_SEVEN: 97%, NINETY_NINE: 99%.	
<i>dpm_to_vof_transition_criteria*</i>	Sub-block for modeling conversion of Lagrangian parcels to Eulerian VOF representation.	
<i>dpm_to_vof_transition_alpha</i>	Void fraction at which Lagrangian parcels will be converted to a Eulerian VOF	0

Chapter 25: Input and Data Files

Physical Models Discrete Phase Modeling

Parameter	Description	Typical value
	representation. If this is set to zero, Lagrangian parcels will never be converted.	
<i>elsa_injectors</i>		
<i>injectors</i>		
- <i>injector</i>	Sub-block for an ELSA injector. Repeat this sub-block through <i>elsa_passive_liquid</i> for each injector.	
<i>elsa_inj_name</i>	Injector name (must match the injector name in parcel_introduction.in).	
<i>elsa_sigma_const_1</i>	First fluid surface area density equation tuning constant.	4.0
<i>elsa_sigma_const_2</i>	Second fluid surface area density equation tuning constant.	1.0
<i>nozzles</i>		
- <i>nozzle</i>	Sub-block for an ELSA nozzle. Repeat this sub-block for each nozzle.	
<i>elsa_noz_name</i>	Nozzle name (must match the nozzle name in parcel_introduction.in).	
<i>elsa_region_id_1</i>	First region for this nozzle.	
<i>elsa_region_id_2</i>	Second region for this nozzle.	
<i>elsa_transition_region_ids</i>	Number of regions in which this nozzle can transition.	
<i>elsa_passive_liquid</i>	Passive to use for this nozzle. Naming convention is <injector name>_<nozzle name>_elsa, where <injector name> and <nozzle name> match the injector and nozzle names in parcel_introduction.in .	<injector name>_<nozzle name>_elsa

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Combustion Modeling

This section describes input files that contain options for combustion modeling in CONVERGE.

Combustion: *combust.in*

To model *combustion*, set *inputs.in* > *feature_control* > *combustion_flag* = 1 and include a *combust.in* file in your case setup.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

```
version: 3.1
---

general_control:
  region_flag: 0
  temporal_type: SEQUENTIAL
  start_time: 0
  end_time: 10
  temp_cutoff: 600
  hc_minimum: 1e-08
  emissions_flag: 0

sage_model:
  active: 1
  option: CONSTANT_VOLUME
  ode_solver:
    type: DENSE
    analyt_jac: 1
    rel_tol: 0.0001
    abs_tol: 1e-14
    reaction_multiplier: 1
  solve_temp: 0
  delta_temp: 2
  dmr_flag: 0
  cvode_error_output_flag: 0
  surface_chemistry_flag: 0

thickened_flame_model:
  active: 1
  region_flag: 0
  start_time: 0.0
  thickening_method:
    option: 0
    max_thickening_factor: 10.0
    num_grid_across_flame: 7
    target_flame_thickness: 0.001
  thickening_amr_couple:
    active: 0
    amr_strength_level: 0
  flame_sensor_model:
    option: 2
    sensor_slope: 1.0
    sensor_species: [C3H8]
    ref_reaction_rate: 0.001
    reaction_sensor_model:
      gamma: 0.4
      beta: 5.0
      reaction_rate_ratio: 0.5
  Jaravel_sensor_model:
    ref_psi: 20.0
    alpha_hot: 0.005
    alpha_cold: 0.05
    time_scale_constant: 20.0
  efficiency_factor_model:
    option: 1
    constant_efficiency_factor: 1.0
    uprime_model: 0
    uprime_multiplier: 1.0
    charlette_model:
      beta: 0.5
    colin_model:
      beta: 0.3
  wall_treatment_flag: 0

ceq_model:
  active: 0
  subset_species: []
  cm2: 0.1

eddy_dissipation_model:
  active: 0
  factor_a: 4.0
  factor_b: 0.5
  max_mixing_rate: 2500.0
  hybrid_flag: 0
```

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

```
adaptive_zone_model:
  active:                                1
  bin_options:
    c7h16:                               0.001
    react_ratio:                          0.05
    temp:                                 5
    tf_reac_mult:                         0.01
  nox_flag:                             0
  output_flag:                           0
  hr_map_flag:                           0
  stiffness_load_balance_flag:          0
three_points_pdf_model:
  active:                                0
  fluctuation_variable:                 PHI
  transport_variance_flag:              0
  variance_cmu:                          0.1
  mixing_relaxation_flag:               0
  relaxation_constant:                  3.0
fuel_name:                            C3H8
ctc_model:
  active:                                0
  init_time:                            -9999999
  mult_scale_flag:                      1
  tau_fraction:                         0.2
  cm2:                                   0.1
  denomc:                               7680000000
  temp_cutoff:                          1000
shell_model:
  active:                                0
  af04:                                  125000
fgm_model:
  active:                                0
  fgm_table_filename:                   "./fgm_table.h5"
  cfd_c_chi:                            2
ecfm_model:
  active:                                0
  stretch_alpha:                         1.6
  destruct_beta:                         1
  itnfs_model:                           1
  spark:
    active:                                0
  auto_ignition:
    active:                                0
    tki_table_filename:                   tki_table.h5
  ai_fsd_factor:                         0
  post_flame_model:                     1
  reinit_flag:                           0
g_eqn_model:
  active:                                0
  burned_region:                        CEQ
  on_flame:                             CEQ
  unburned_region:                      NONE
  init_value:                            -0.1
  grad_g_flag:                           1
  temp_cutoff:                          3000
  spark:
    active:                                0
ecfm3z_model:
  active:                                0
  mix_betam:                            2
  stretch_alpha:                         1.6
  destruct_beta:                         1
  itnfs_model:                           1
  auto_ignition:
    active:                                0
    tki_table_filename:                   tki_table.h5
  ai_fsd_factor:                         0
  post_flame_model:                     1
  reinit_flag:                           0
rif_model:
  active:                                0
```

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

```
nproc_flamelet:          1
init_zmin:                1e-05
unburned_temp_offset:    0
flamelet_c_chi:          2
cfd_c_chi:                2
chi_clip:                 1000
num_flamelets:           1
grid_type:                4
num_zgrids:               100
pdf_flag:                  0
transport_all_species:
  active:                   0
  num_transport_species: [O2, N2, CO2, H2O]
boundary_condition:
  rif_bc0:
    O2:                      0
  rif_bc1:
    {}
  bc1_temp:                 700
sl_model:
  active:                   0
  sl_option:
    constant_laminar_flamespeed: 0.16
    constant_laminar_flame_thickness: 5e-4
    tlf_table_filename:          tlf_table.h5
    laminar_flamespeed_correlations:
      general_control:
        temp_ref:              298
        pres_ref:              101325
        temp_a:                 1.2
        temp_m:                 -0.8
        pres_a:                 -0.26
        pres_m:                 0.22
        dilution:                0
      metghalchi:
        bm:                     0.2632
        b2:                     -0.8472
        equiv_ratio:            1.13
      gulder:
        omega:                 0.4658
        eta:                    -0.326
        xi:                     4.48
  st_model:
    active:                   0
    st_option:
      peters_correlation:
        a4:                     0.78
        b1:                      2
        b3:                      1
        g_prime_cs:             2
      zimont_correlation:
        a:                       0.52
      pitsch_correlation:
        b1:                      2
        b3:                      1
mix_frac_output:
  mix_frac:
    active:                   0
  mix_frac_var:
    active:                   0
  c_chi:                      2
borghi_diagram_output_flag: 0
```

Figure 25.99: An example *combust.in* file.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Table 25.72: Format of *combust.in*.

Parameter	Description	Typical value
<i>general_control</i>		
<i>region_flag</i>	0 = Combustion is not region-dependent, 1 = Combustion is region-dependent (requires combust_region.in).	
<i>temporal_type</i>	Temporal type (CYCLIC, SEQUENTIAL, or PERMANENT) of combustion modeling.	
<i>period</i>	The CYCLIC period (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero). Used only when <i>temporal_type</i> = CYCLIC.	
<i>start_time</i>	Combustion start time in seconds (if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0) or crank angle degrees (if <i>crank_flag</i> is non-zero).	
<i>end_time</i>	Combustion end time in seconds (if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0) or crank angle degrees (if <i>crank_flag</i> is non-zero).	
<i>temp_cutoff</i>	Combustion temperature cutoff (K) below which chemistry calculations are not solved.	600
<i>hc_minimum</i>	Minimum cell HC+CO species mole fraction for combustion modeling.	1.0e-14 - 1.0e-18
<i>emissions_flag</i>	0 = No emissions modeling, 1 = Emissions modeling (requires emissions.in).	
<i>sage_model</i>		
<i>active</i>	0 = No SAGE detailed chemical kinetics solver, 1 = SAGE detailed chemical kinetics solver, 11 = SAGE detailed chemical kinetics solver within the G-Equation model (<i>i.e.</i> , select this option when <i>combust.in</i> > <i>g_eqn_model</i> > <i>burned_region</i> = SAGE, <i>g_eqn_model</i> > <i>on_flame</i> = SAGE, and/or <i>g_eqn_model</i> > <i>unburned_region</i> = SAGE).	
<i>option</i>	CONSTANT_VOLUME = Perform constant-volume combustion simulation, CONSTANT_PRESSURE = Perform constant-pressure combustion simulation.	
<i>ode_solver</i>		

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>type</i>	<i>DENSE</i> = CVODE dense solver (recommended when the total number of species is no greater than 100), <i>ITERATIVE</i> = CVODE with preconditioned iterative solver, <i>ITERATIVE_SUPERLU</i> = CVODE with preconditioned iterative solver with SuperLU (SuperLU, 2011), <i>ITERATIVE_CONVERGE</i> = CVODE with optimized preconditioned iterative solver (recommended when the total number of species is greater than 100), <i>NONSTIFF_DENSE</i> = CVODE dense solver for non-stiff systems (recommended for single-step mechanisms).	
<i>analyt_jac</i>	0 = Solve Jacobian matrix numerically, 1 = Solve Jacobian matrix analytically.	1
<i>rel_tol</i>	Relative iteration error for each species.	1e-4
<i>abs_tol</i>	Absolute iteration error for each species.	1e-14
<i>reaction_multiplier</i>	Scaling factor of reaction rates. Specify a file name for a case with a variable reaction multiplier.	1.0
<i>solve_temp</i>	SAGE temperature solution flag. 0 = Do not re-solve temperature in this cell unless the temperature change exceeds <i>delta_temp</i> , 1 = Always re-solve temperature in every cell.	
<i>delta_temp</i>	Magnitude of the temperature change (K) above which the temperature will be re-solved.	
<i>dmr_flag</i>	0 = No dynamic mechanism reduction, 1 = Dynamic mechanism reduction enabled (requires sage_dmr.in).	
<i>cvode_error_output_flag</i>	0 = Do not write <i>zero_d_cases_rank<number>.out</i> or <i>zero_d_solver.out</i> , 1 = Write <i>zero_d_cases_rank<number>.out</i> for any cells that have CVODE errors and a temperature of less than 5000 K, and write <i>zero_d_solver.out</i> .	
<i>surface_chemistry_flag</i>	0 = No surface chemistry, 1 = Surface chemistry enabled (requires <i>surface_chemistry.in</i>).	
<i>thickened_flame_model</i>		

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>active</i>	0 = No thickened flame model (TFM), 1 = TFM (used only if <i>combust.in</i> > <i>sage_model</i> > <i>active</i> = 1 above and turbulence.in > <i>turbulence_model</i> = <i>LES_*</i> , and requires <i>combust.in</i> > <i>sl_model</i> > <i>active</i> = 5 below).	
<i>region_flag</i>	0 = TFM parameters do not vary by region, 1 = TFM parameters (<i>start_time</i> and <i>thickening_method</i> > <i>max_thickening_factor</i>) vary by region (requires tfm_region.in).	
<i>start_time</i>	TFM start time in seconds (if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0) or crank angle degrees (if <i>crank_flag</i> is non-zero). The TFM end time will be <i>combust.in</i> > <i>general_control</i> > <i>end_time</i> .	
<i>thickening_method</i>		
<i>option</i>	0 = Specify maximum thickening factor (<i>max_thickening_factor</i> required), 1 = Specify number of grid points across the flame (<i>num_grid_across_flame</i> required), 2 = Specify target flame thickness (<i>target_flame_thickness</i> required).	
<i>max_thickening_factor</i>	Maximum thickening factor. Used only when <i>option</i> = 0.	
<i>num_grid_across_flame</i>	Number of grid points across the flame. Used only when <i>option</i> = 1 or when <i>thickening_amr_couple</i> > <i>active</i> = 1.	
<i>target_flame_thickness</i>	Target flame thickness (<i>m</i>). Used only when <i>option</i> = 2.	1e-3
<i>thickening_amr_couple</i>		
<i>active</i>	0 = No coupling between TFM and Adaptive Mesh Refinement (AMR), 1 = Coupling between TFM and AMR (<i>num_grid_across_flame</i> required).	
<i>amr_strength_level</i>	Method of calculating integer-valued AMR level from real-valued target AMR level. 0 = Use the floor value, 1 = Round to the nearest integer, 2 = Use the ceiling value.	
<i>flame_sensor_model</i>		
<i>option</i>	0 = No flame sensor (uniform thickening),	

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
	1 = Standard reaction rate model, 2 = Jaravel's sensor model (solves a transported passive).	
<i>sensor_slope</i>	Multiplier (β) for the sensor species reaction rate. Not used when <i>option</i> = 0.	5.0
<i>sensor_species</i>	Name of species that is used as a reaction rate sensor. Not used when <i>option</i> = 0.	
<i>ref_reaction_rate</i>	Maximum sensor species reaction rate based on the 1D planar laminar flame ($kg/m^3\text{-s}$). Not used when <i>option</i> = 0.	
<i>reaction_sensor_model</i>		
<i>gamma</i>	No longer used.	0.4
<i>beta</i>	No longer used.	5.0
<i>reaction_rate_ratio</i>	No longer used.	0.5
<i>Jaravel_sensor_model</i>	Used only when <i>flame_sensor_model</i> > <i>option</i> = 2.	
<i>ref_psi</i>	Maximum value of the flame sensor.	20.0
<i>alpha_hot</i>	Time scale of the flame sensor sink on the hot side (s).	0.005
<i>alpha_cold</i>	Time scale of the flame sensor sink on the cold side (s).	0.05
<i>time_scale_constant</i>	Time scale of the flame sensor source (s).	20.0
<i>efficiency_factor_model</i>		
<i>option</i>	0 = Constant efficiency factor (<i>constant_efficiency_factor</i> required), 1 = Charlette's model (<i>charlette_model</i> block required), 2 = Colin's model (<i>colin_model</i> block required).	
<i>constant_efficiency_factor</i>	Value of the constant efficiency factor.	
<i>uprime_model</i>	0 = Sub-scale velocity based on the curl of the resolved velocity, 1 = Sub-scale velocity based on the local sub-grid tke.	
<i>uprime_multiplier</i>	Sub-scale velocity multiplier.	
<i>charlette_model</i>		
<i>beta</i>	Charlette's model parameter.	0.5
<i>colin_model</i>		

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>beta</i>	Colin's model parameter.	0.3
<i>wall_treatment_flag</i>	0 = Turn off wall treatment, 1 = Turn on wall treatment.	
<i>ceq_model</i>		
<i>active</i>	0 = No CEQ model, 1 = CEQ model, 2 = CEQ model with a turbulent time-scale.	
<i>subset_species</i>	List of species names to solve with CEQ.	
<i>cm2</i>	Turbulent time-scale constant. Used only when <i>ceq_model</i> > <i>active</i> = 2.	0.1 - 1.0
<i>eddy_dissipation_model</i>		
<i>active</i>	0 = No Eddy Dissipation Model, 1 = Eddy Dissipation Model (EDM) .	
<i>factor_a</i>	EDM A factor.	
<i>factor_b</i>	EDM B factor.	
<i>max_mixing_rate</i>	Maximum mixing rate as determined by the inverse of turbulent time scale.	
<i>hybrid_flag</i>	0 = Off (requires only one reaction in the reaction mechanism file), 1 = On (requires no more than two reactions in the reaction mechanism file).	
<i>adaptive_zone_model</i>		
<i>active</i>	0 = No adaptive zoning, 1 = Adaptive zoning (used only if either <i>combust.in</i> > <i>sage_model</i> > <i>active</i> is non-zero, <i>combust.in</i> > <i>ceq_model</i> > <i>active</i> is non-zero, <i>combust.in</i> > <i>ecfm_model</i> > <i>post_flame_model</i> = 2, <i>combust.in</i> > <i>g_eqn_model</i> > <i>active</i> = 1, or if <i>combust.in</i> > <i>ecfm3z_model</i> > <i>post_flame_model</i> = 2).	
<i>bin_options</i>		
< <i>species name</i> >	Adaptive zoning bin size for this species (cube root of species mass fraction) or file name (e.g., <i>adaptive_zoning_bin_C7H16.in</i>).	
<i>react_ratio</i>	Adaptive zoning bin size for the progress equivalence ratio or file name (e.g., <i>adaptive_zoning_bin_react_ratio.in</i>).	0.001 - 0.01
<i>temp</i>	Adaptive zoning bin size for temperature (K) or file name (e.g., <i>adaptive_zoning_bin_temp.in</i>).	0.1 - 10.0

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>nox_flag</i>	0 = Do not conserve NOx during species re-mapping, 1 = Conserve NOx during species re-mapping.	1 for emissions prediction
<i>output_flag</i>	0 = No additional output for adaptive zoning, 1 = Generate additional output for adaptive zoning (<i>az_info <dump number> <dump time>.out</i>) and make <i>az_id</i> available in <i>post.in</i> .	
<i>hr_map_flag</i>	0 = Do not use heat release mapping, 1 = Use heat release mapping.	
<i>stiffness_load_balance_flag</i>	0 = Do not load balance chemistry according to the stiffness of the ODE systems, 1 = Load balance chemistry according to the stiffness of the ODE systems.	
<i>three_points_pdf_model</i>	CONVERGE reads this parameter only if <i>adaptive_zone_model > active = 1</i> . Stiffness-based load balancing is always active when <i>adaptive_zone_model > active = 0</i> .	
<i>active</i>	0 = No three-point PDF method, 1 = Three-point PDF method (requires <i>combust.in > sage_model > active = 1</i>).	
<i>fluctuation_variable</i>	<i>PHI</i> = Use equivalence ratio as the fluctuating variable, <i>TEMP</i> = Use temperature as the fluctuating variable.	
<i>transport_variance_flag</i>	0 = Calculate fluctuating variable variance explicitly, 1 = Solve fluctuating variable variance transport equation.	0
<i>variance_cmu</i>	Model constant for explicit variance calculation.	0.1
<i>mixing_relaxation_flag</i>	0 = Do not use mixing relaxation, 1 = Use mixing relaxation (<i>relaxation_constant</i> required).	
<i>relaxation_constant</i>	Relaxation constant for mixing relaxation.	1.0 - 5.0
<i>fuel_name</i>	Species name of the fuel. It must be included in the reaction mechanism and thermodynamic data files as well. Used by the CTC/Shell, NOx, and soot models. Not used in cases with multi-component fuels.	
<i>ctc_model</i>		

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>active</i>	0 = No Characteristic Time Combustion (CTC) model, 1 = CTC model .	
<i>init_time</i>	Time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or crank angle degrees if <i>crank_flag</i> is non-zero) at which the CTC model will be reinitialized. You can specify a file name (e.g., ctc_init_time.in) for reinitializing different regions at different times.	
<i>mult_scale_flag</i>	0 = Single scale CTC model, 1 = Multi-scale CTC model.	
<i>tau_fraction</i>	Time scale function for the multi-scale CTC model.	0.05–1.0
<i>cm2</i>	Turbulent time scale constant for the CTC model.	0.1–1.0
<i>denomc</i>	Chemical time scale constant for the CTC model.	7.68e9
<i>temp_cutoff</i>	Shell/CTC transition temperature (K).	1000–1300
<i>shell_model</i>		
<i>active</i>	0 = No Shell ignition model, 1 = Modified Shell ignition model , 2 = Original Shell model (recommended).	
<i>af04</i>	Ignition delay parameter for the Shell model.	1.0e5 – 5.0e5
<i>fgm_model</i>		
<i>active</i>	0 = No Flamelet Generated Manifold (FGM) model, 1 = FGM model .	
<i>fgm_table_filename</i>	File name for the FGM lookup table .	
<i>cfd_c_chi</i>	Scalar dissipation constant.	
<i>ecfm_model</i>		
<i>active</i>	0 = No Extended Coherent Flame Model (ECFM), 1 = ECFM enabled.	
<i>stretch_alpha</i>	Constant for the turbulent stretch introduced by the surface density production term.	
<i>destruct_beta</i>	Constant for the surface density destruction term.	

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>itnfs_model</i>	Turbulent flame stretch model. 0 = Meneveau and Poinsot (1991) model using the fitted form (recommended if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>RANS_*</i>), 1 = Meneveau and Poinsot (1991) model using the integrated form (recommended if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>RANS_*</i>), 2 = Meneveau and Poinsot (1991) model using the integrated form in another space (recommended if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>RANS_*</i>), 3 = CCDVP efficiency function modified as used in AVBP (recommended if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>LES_*</i>) (Charlette et al., 2002), 4 = Bougrine et al., 2014 efficiency function (recommended if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>RANS_*</i> or <i>LES_*</i>), 5 = Suillaud efficiency function (recommended if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>RANS_*</i> or <i>LES_*</i>), 6 = Cant efficiency function (recommended if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>RANS_*</i>) (Cant and Bray, 1989).	1
<i>spark</i>		
<i>active</i>	Laminar ignition (LI) spark model. 0 = No spark model, 1 = ISSIM spark model (issim.in required).	
<i>auto_ignition</i>		
<i>active</i>	0 = No autoignition, 1 = TKI enabled (tki_table_filename required).	
<i>tki_table_filename</i>	File name (e.g., <i>tki_table.h5</i>) for the TKI table .	
<i>ai_fsd_factor</i>	Flame surface density deposition factor due to autoignition.	0
<i>post_flame_model</i>	Used only if <i>ecfm_model</i> > <i>active</i> = 1 above. 0 = The burned zone is solved by the method of Colin et al. (2003) , 1 = The burned zone is solved by the CEQ method. 2 = The burned zone is solved with SAGE (requires <i>combust.in</i> > <i>sage_model</i> > <i>active</i> = 0).	
<i>reinit_flag</i>	0 = CONVERGE will not reinitialize the combustion domain (appropriate for a single-	

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
	cycle engine case), 1 = CONVERGE will reinitialize the combustion domain (requires ecfm3z_reinit.in). 2 = CONVERGE will automatically reinitialize the combustion domain one <i>crank angle degree</i> before the combustion start time (does not require ecfm3z_reinit.in).	
<i>g_eqn_model</i>		
<i>active</i>	0 = No G-Equation model, 1 = G-Equation model .	
<i>burned_region</i>	<i>CEQ</i> = Use the CEQ equilibrium model in the burned region, <i>SAGE</i> = Use the SAGE solver in the burned region. The <i>burned_region</i> , <i>on_flame</i> , and <i>unburned_region</i> parameters are not independent. Refer to Chapter 16 or CONVERGE Studio for more information on valid combinations.	
<i>on_flame</i>	<i>CEQ</i> = Use the CEQ equilibrium model on the flame, <i>SAGE</i> = Use the SAGE solver on the flame. The <i>burned_region</i> , <i>on_flame</i> , and <i>unburned_region</i> parameters are not independent. Refer to Chapter 16 or CONVERGE Studio for more information on valid combinations.	
<i>unburned_region</i>	<i>NONE</i> = Do not use a combustion model in the unburned region, <i>SAGE</i> = Use the SAGE solver in the unburned region, <i>tki_table.h5</i> = Use the TKI table in the unburned region. The <i>burned_region</i> , <i>on_flame</i> , and <i>unburned_region</i> parameters are not independent. Refer to Chapter 16 or CONVERGE Studio for more information on valid combinations.	
<i>init_value</i>	Initialization of <i>G</i> . Specify a numerical value or a file name (e.g., g_eqn_init.in).	-0.1
<i>grad_g_flag</i>	Method to reinitialize the gradient of <i>G</i> . 0 = Explicit method, 1 = Sussman (implicit) method, which is slower but more accurate.	1

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>temp_cutoff</i>	Temperature (K) above which G is initialized. The G equation is always solved at the flame front regardless of temperature. Use this feature when transitioning from one model (e.g., SAGE) to the G-Equation model. Deactivate this feature by setting it to a very large value (e.g., 5000 K).	
<i>spark</i>		
<i>active</i>	0 = Initialize G in source.in (recommended), 1 = Use kernel model for ignition (requires g_eqn_spark.in).	0
<i>ecfm3z_model</i>		
<i>active</i>	0 = No 3-Zone Extended Coherent Flame Model (ECFM3Z), 1 = ECFM3Z enabled.	
<i>mix_betam</i>	Mixing constant.	
<i>stretch_alpha</i>	Constant for the turbulent stretch introduced by the surface density production term.	
<i>destruct_beta</i>	Constant for the surface density destruction term.	
<i>itnfs_model</i>	0 = No turbulent flame stretch model, 1 = Turbulent flame stretch model enabled.	1
<i>auto_ignition</i>		
<i>active</i>	0 = No autoignition, 1 = TKI enabled (<i>tki_table_filename</i> required).	
<i>tki_table_filename</i>	File name (e.g., <i>tki_table.h5</i>) for the TKI table .	
<i>ai_fsd_factor</i>	Flame surface density deposition factor due to auto-ignition.	0
<i>post_flame_model</i>	Used only if <i>ecfm3z_model > active = 1</i> above. 0 = The burned zone is solved by the method of Colin et al. (2003) , 1 = The burned zone is solved by the CEO method, 2 = The burned zone is solved with SAGE (requires <i>combust.in > sage_model > active = 0</i>).	
<i>reinit_flag</i>	0 = CONVERGE will not reinitialize the combustion domain (appropriate for a single-cycle engine case), 1 = CONVERGE will reinitialize the combustion domain (requires ecfm3z_reinit.in),	

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
	2 = CONVERGE will automatically reinitialize the combustion domain one <i>crank angle degree</i> before the combustion start time (does not require ecfn3z_reinit.in).	
<i>rif_model</i>		
<i>active</i>	0 = No Representative Interactive Flamelet (RIF) model, 1 = RIF model .	
<i>nproc_flamelet</i>	Number of processors for each flamelet. Enter 1 to solve RIF in serial or an integer greater than 1 to solve RIF in parallel.	1
<i>init_zmin</i>	Minimum value of Z (mixture fraction) to initialize flamelet.	1e-5 for a diesel case
<i>unburned_temp_offset</i>	Unburned temperature offset (K).	0.0
<i>flamelet_c_chi</i>	Flamelet scalar dissipation constant.	2.0
<i>cfд_c_chi</i>	CFD scalar dissipation constant.	2.0
<i>chi_clip</i>	Maximum value of the scalar dissipation rate.	1000.0
<i>num_flamelets</i>	Number of flamelets in the RIF model.	1
<i>grid_type</i>	The nature of the grid in the z coordinate. 1 = Equidistant grid, 2 = Equidistant grid with refinement from <i>Z_min</i> (minimum fuel mass fraction) to $2 \times Z_{st}$ (Z_{st} = fuel mass fraction at equivalence ratio = 1), 3 = User-specified grid (this option is currently not available), 4 = Hyperbolic grid (recommended).	4
<i>num_zgrids</i>	Number of grid points on the z coordinate.	101
<i>pdf_flag</i>	0 = Integration based on β probability density function (PDF), 1 = Integration based on clipped Gaussian PDF.	0
<i>transport_all_species</i>		
<i>active</i>	0 = CONVERGE will transport all species, 1 = CONVERGE will transport only the species listed as <i>num_transport_species</i> .	1
<i>num_transport_species</i>	List of species to be transported.	
<i>boundary_condition</i>		
<i>rif_bc0</i>		

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i><species name></i>	Mass fraction of this oxidizer species. Repeat for additional oxidizer species.	
<i>rif_bc1</i>		
<i><species name></i>	Mass fraction of this fuel species. Repeat for additional fuel species.	
<i>bc1_temp</i>	Gaseous phase fuel temperature (K).	700
<i>sl_model</i>		
<i>active</i>	Laminar flamespeed model (used only if <i>combust.in > fgm_model > active = 1</i> and <i>combust.in > st_model > active</i> is non-zero, if <i>combust.in > ecfm_model > active</i> is non-zero, if <i>combust.in > g_eqn_model > active = 1</i> , or if <i>combust.in > ecfm3z_model > active</i> is non-zero).	
<i>sl_option</i>	0 = Constant, 1 = Metghalchi , 2 = Gulder , 3 = Rahim flamespeed correlation for methane , 4 = IFPEN Metghalchi , 5 = User-defined data file , 11 = Metghalchi with different correlation for gasoline (for ECFM/ECFM3Z only), 21 = Gulder with different correlation for gasoline (for ECFM/ECFM3Z only).	
<i>constant_laminar_flamespeed</i>	Laminar flamespeed (m/s). Used when <i>combust.in > sl_model > active = 0</i> .	
<i>constant_laminar_flame_thickness</i>	Laminar flame thickness.	
<i>tlf_table_filename</i>	File name for a laminar flamespeed table (e.g., <i>tlf_table.h5</i>).	
<i>laminar_flamespeed_correlations</i>		
<i>general_control</i>		
<i>temp_ref</i>	Reference temperature (K). Used when <i>combust.in > sl_model > active = 1</i> or 2.	298
<i>pres_ref</i>	Reference pressure (Pa). Used when <i>combust.in > sl_model > active = 1</i> or 2.	101325
<i>temp_a</i>	Constant (a_T) for the temperature exponent equation: $\gamma = a_T + m_T(\phi - 1)$.	See below .

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>temp_m</i>	Slope (m_T) for the temperature exponent equation: $\gamma = a_T + m_T(\phi - 1)$	Metghalchi: -0.8 Gulder: 0
<i>pres_a</i>	Constant (a_p) for the pressure exponent equation: $\beta = a_p + m_p(\phi - 1)$	See below.
<i>pres_m</i>	Slope (m_p) for the pressure exponent equation: $\beta = a_p + m_p(\phi - 1)$	Metghalchi: 0.22 Gulder: 0
<i>dilution</i>	Dilution species mass fraction when <i>combust.in > sl_model > active = 1</i> (e.g., EGR). In the following flamespeed equation, Y_{dil} is the mass fraction of the dilution species: $s_l = s_{l,ref} \left(\frac{T_u}{T_{u,ref}} \right)^\gamma \left(\frac{P}{P_{ref}} \right)^\beta (1 - 2.1Y_{dil})$	
	Dilution species mole fraction when <i>combust.in > sl_model > active = 2</i> . The same equation is used to calculate flamespeed but mole fraction, x_{dil} , is used instead of Y_{dil} .	
	When <i>inputs.in > simulation_control > crank_flag</i> is non-zero, you can set <i>dilution = dilute_init.in</i> to allow a transport passive, <i>egr_fraction</i> , to indicate the spatially varying dilution level.	
<i>metghalchi</i>		
<i>bm</i>	Metghalchi constant (m). Used only when <i>combust.in > sl_model > active = 1</i> .	See below.
<i>b2</i>	Metghalchi constant (m). Used only when <i>combust.in > sl_model > active = 1</i> .	See below.
<i>equiv_ratio</i>	Metghalchi reference equivalence ratio. Used only when <i>combust.in > sl_model > active = 1</i> .	See below.
<i>gulder</i>		
<i>omega</i>	Gulder coefficient for calculating laminar flamespeed. Used only when <i>combust.in > sl_model > active = 2</i> .	See below.
<i>eta</i>	Gulder coefficient for calculating laminar flamespeed. Used only when <i>combust.in ></i>	See below.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
	$sl_model > active = 2.$	
xi	Gulder coefficient for calculating laminar flamespeed. Used only when $combust.in > sl_model > active = 2.$	See below.
st_model		
$active$	Turbulent flamespeed calculation model (used only if $combust.in > fgm_model > active = 1$ or $combust.in > g_eqn_model > active = 1$). 0 = No turbulent flamespeed model, 1 = Peters flamespeed model without G-prime (G-Equation and FGM), 11 = Peters flamespeed model with G-prime (G-Equation), 2 = Zimont (FGM), 3 = Pitsch (G-Equation with LES), 4 = Viscosity ratio correlation.	
st_option		
$peters_correlation$		
$a4$	Turbulent flamespeed correlation constant. Used only when $combust.in > st_model > active = 1$ or 11.	0.78
$b1$	Turbulent flamespeed correlation constant. A larger value increases the turbulent flamespeed while a smaller value decreases the turbulent flamespeed. This parameter has the largest influence on the turbulent flamespeed. Used only when $active = 1$ or 11.	2.0
$b3$	Turbulent flamespeed correlation constant. Used only when $combust.in > st_model > active = 1$ or 11.	1.0
g_prime_cs	G-prime dissipation constant used to solve for the variance of G. Used only when $combust.in > st_model > active = 11.$	2.0
$zimont_correlation$		
a	Turbulent flamespeed constant for the Zimont model. Used only when $combust.in > st_model > active = 2.$	
$pitsch_correlation$		

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>b1</i>	Turbulent flamespeed correlation constant. A larger value increases the turbulent flamespeed while a smaller value decreases the turbulent flamespeed. This parameter has the largest influence on the turbulent flamespeed. Used only when <i>combust.in > st_model > active = 3</i> .	2.0
<i>b3</i>	Turbulent flamespeed correlation constant. Used only when <i>combust.in > st_model > active = 3</i> .	1.0
<i>mix_frac_output</i>		
<i>mix_frac</i>		
<i>active</i>	0 = No mixture fraction calculation, 1 = Mixture fraction calculation.	
	The mixture fraction (<i>Z</i>) is defined as	
	$Z = \sum_{i=1}^{n_s} \frac{N_{C,i} MW_C Y_i}{MW_i} + \sum_{i=1}^{n_s} \frac{N_{H,i} MW_H Y_i}{MW_i},$	
	where <i>i</i> is the species, <i>n_s</i> is the number of species, <i>N_{C,i}</i> , <i>N_{H,i}</i> are the number of carbon and hydrogen atoms respectively in species <i>i</i> , <i>MW_C</i> , <i>MW_H</i> are the molecular weights of carbon and hydrogen respectively, <i>Y_i</i> is the species mass fraction, and <i>MW_i</i> is the molecular weight of species <i>i</i> . The <i>Z</i> values range from 0 in the oxidizer stream to 1 in the fuel stream.	
<i>mix_frac_var</i>		
<i>active</i>	0 = No mixture fraction variance calculation, 1 = Mixture fraction variance calculation.	
<i>c_chi</i>	Constant used in modeling the scalar dissipation in the mixture fraction variance calculation.	
<i>borghi_diagram_output_flag</i>	0 = Do not write <i>borghi.out</i> , 1 = Write <i>borghi.out</i> . Requires <i>combust.in > g_eqn_model > active = 1</i> , <i>combust.in > ecfm_model > active = 1</i> , or <i>combust.in > ecfm3z_model > active = 1</i> .	

Table 25.73 provides recommended values of the Metghalchi constants for several fuels ([Metghalchi and Keck, 1982](#)).

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Table 25.73: Fuel-specific values of the Metghalchi constants.

FUEL	<i>sl_metghalchi_bm</i> (m/s)	<i>sl_metghalchi_b</i> 2	<i>sl_metghalchi_equiv_ratio</i> (m/s)	<i>sl_temp_a</i>	<i>sl_pres_a</i>
Methanol	0.3692	-1.4051	1.11	2.11	-0.13
Propane	0.3422	-1.3865	1.08	2.13	-0.17
Isooctane	0.2632	-0.8472	1.13	2.26	-0.18

Values for some of the G-Equation parameters are functions of the fuel used in the simulation. Table 25.74 provides recommended values of the Gulder constants for various fuels ([Gulder, 1984](#)).

Table 25.74: Fuel-specific values of the Gulder constants.

FUEL	<i>sl_gulder_omega</i>	<i>sl_gulder_eta</i>	<i>sl_gulder_xi</i>	<i>sl_temp_a</i>	<i>sl_pres_a</i>
Methane	0.4220	0.150	5.18	2.00	-0.50
Propane	0.4460	0.120	4.95	1.77	-0.20
Methanol	0.4920	0.250	5.11	1.75	$-0.2\phi^{-0.5}, \phi \leq 1$ $-0.2\phi, \phi > 1$
Ethanol	0.4650	0.250	6.34	1.75	$-0.17\phi^{-0.5}, \phi \leq 1$ $-0.17\phi^{0.5}, \phi > 1$
Isooctane	0.4658	-0.326	4.48	1.56	-0.22
Isooctane/ Methanol*	$0.4658(1-0.53V)$	-0.326	4.48	1.55	-0.22
Isooctane/ Ethanol*	$0.4658(1+0.07V^{0.35})$	-0.326	4.48	$1.56 + 0.23V^{0.46}$	-0.22

*For isoctane/alcohol mixtures, V is the volume fraction of alcohol. These correlations are valid for $0 \leq V \leq 0.2$.

Region-Based Combustion: `combust_region.in`

To activate region-based combustion modeling, set `combust.in` > `general_control` > `region_flag` = 1 and include the `combust_region.in` file in your case setup. Regions that are not listed in `combust_region_control` will have combustion modeling deactivated.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

```
version: 3.1
---

combust_region_control:
  - combust_region:
    region_id:      0
    temporal_type: SEQUENTIAL
    start_time:    0.0
    end_time:      0.01
  - combust_region:
    region_id:      2
    temporal_type: CYCLIC
    period:        720.0
    start_time:    0.0
    end_time:      0.01
```

Figure 25.100: An example *combust_region.in* file.

Table 25.75: Format of *combust_region.in*.

Parameter	Description
<i>combust_region_control</i>	
- <i>combust_region</i>	This settings sub-block specifies a combustion region. Repeat this sub-block for each combustion region.
<i>region_id</i>	The region in which combustion calculations will be performed. The value given here must also be a <i>region_id</i> in <i>initialize.in</i> .
<i>temporal_type</i>	Temporal type: <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>PERMANENT</i> .
<i>period</i>	The <i>CYCLIC</i> period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>start_time</i>	Combustion model start time in seconds (if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0) or crank angle degrees (if <i>crank_flag</i> is non-zero).
<i>end_time</i>	Combustion model end time in seconds (if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0) or crank angle degrees (if <i>crank_flag</i> is non-zero).

Region-Basd CTC Initialization: *ctc_init_time.in*

To direct CONVERGE to reinitialize the Characteristic Time Combustion (CTC) model on a region-by-region basis, specify a file name (e.g., *ctc_init_time.in*) for *combust.in* > *ctc_model* > *init_time* and include that file in your case setup. This option may be useful for multi-cycle or multi-cylinder engine simulations. This option is available only for crank angle-based simulations (i.e., only when *inputs.in* > *simulation_control* > *crank_flag* is non-zero).

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

```
version: 3.1
---

ctc_init_time_control:
  - ctc_region:
    region_id: [0,1]
    init_time: 120.0
    init_period: 720.0
  - ctc_region:
    region_id: [3]
    init_time: 360.0
    init_period: 720.0
```

Figure 25.101: An example *ctc_init_time.in* file.

Table 25.76: Format of *ctc_init_time.in*.

Parameter	Description
<i>ctc_init_time_control</i>	
- <i>ctc_region</i>	This settings sub-block specifies the configuration for a CTC region. Repeat this sub-block for each CTC region.
<i>region_id</i>	The region ID(s) for this CTC region. Use [] to specify multiple regions (e.g., [1,2,3]). These values must be consistent with the <i>region_id</i> values in initialize.in .
<i>init_time</i>	The time (in <i>crank angle degrees</i>) at which the CTC model will be reinitialized.
<i>init_period</i>	The period (in <i>crank angle degrees</i>) after which the CTC model will be reinitialized.

Region-Based ECFM/ECFM3Z Stretch Alpha: *stretch_alpha.in*

To activate temporally-varying or region-based stretch alpha modeling, set *ecfm_model > active* or *ecfm3z_model > active = 1* in [combust.in](#) and specify a file name (e.g., *stretch_alpha.in*) for *ecfm_model > stretch_alpha* or *ecfm3z_model > stretch_alpha*. Provide the appropriate *stretch_alpha.in* file in the Case Directory. Figure 25.102 shows an example *stretch_alpha.in* file for an ECFM simulation. This example specifies *stretch_alpha* throughout the computational domain at $t = 0$ and $t = 1.0$, and it separately specifies *stretch_alpha* in regions 0 and 1 at $t = 0$ and $t = 2.0$. The format of this file is otherwise similar to that of a conventional [temporally varying parameter](#).

```
TEMPORAL
SEQUENTIAL
second      ecfm_stretch_alpha
0.0          1.55
1.0          1.65

2           num_regions
0           region_id
1           region_id
0.0         3.55
2.0         4.0
```

Figure 25.102: An example *stretch_alpha.in* file for an ECFM simulation. Note the keyword *ecfm_stretch_alpha*.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Figure 25.103 shows an example *stretch_alpha.in* file for an ECFM3Z simulation. This example specifies *stretch_alpha* throughout the computational domain at $t = 0$ and $t = 1.0$, and it does not separately specify *stretch_alpha* in any regions.

```
TEMPORAL
SEQUENTIAL
second      ecfm3z_stretch_alpha
0.0         1.55
1.0         1.65
```

Figure 25.103: An example *stretch_alpha.in* file for an ECFM3Z simulation. Note the keyword *ecfm3z_stretch_alpha*.

Region-Based G-Equation Initialization: *g_eqn_init.in*

To activate region-based initialization of *G* in the [G-Equation model](#), specify a file name (e.g., *g_eqn_init.in*) for [*combust.in*](#) > *g_eqn_model* > *init_value* and then include that file in your case setup. In this file you must specify a value of *G_EQN_PASSIVE* for each region, even if combustion does not occur in all regions in the domain. CONVERGE will ignore the information related to regions in which combustion does not occur.

```
version: 3.1
---

g_eqn_init_region_control:
  - g_eqn_region:
    region_id: [0, 1, 2]
    init_value: -0.1
    init_time: 120.0
    init_period: 720.0
  - g_eqn_region:
    region_id: [5]
    init_value: -0.1
    init_time: 120.0
    init_period: 720.0
```

Figure 25.104: An example *g_eqn_init.in* file.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Table 25.77: Format of *g_eqn_init.in*.

Parameter	Description
<i>g_eqn_init_region_control</i>	
- <i>g_eqn_region</i>	This settings sub-block specifies the configuration of the G-Equation region. Repeat this sub-block for each region.
<i>region_id</i>	The region ID(s) for this G-Equation region. Use [] to specify multiple regions (e.g., [1,2,3]). These values must be consistent with the <i>region_id</i> values in initialize.in .
<i>init_value</i>	Initial value of <i>G_EQN_PASSIVE</i> in this region.
<i>init_time</i>	Time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) to initialize <i>G_EQN_PASSIVE</i> .
<i>init_period</i>	The CYCLIC period over which to reinitialize <i>G_EQN_PASSIVE</i> (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).

Region-Based TFM: *tfm_region.in*

Two TFM parameters ([combust.in](#) > *thickened_flame_model* > *start_time* and *thickened_flame_model* > *max_thickening_factor*) can vary by region. To allow these TFM parameters to vary by region, set [combust.in](#) > *thickened_flame_model* > *region_flag* = 1 and include the *tfm_region.in* file in your case setup.

```
version: 3.1
---
tfm_regions:
  - tfm_region:
    region_id: 0
    tfm_start_time: 0.0
    thickening_factor: 1.0
  - tfm_region:
    region_id: 1
    tfm_start_time: 0.0
    thickening_factor: 4.5
  - tfm_region:
    region_id: 2
    tfm_start_time: 0.0
    thickening_factor: 13.5
  - tfm_region:
    region_id: [3,4]
    tfm_start_time: 0.0
    thickening_factor: 4.5
```

Figure 25.105: An example *tfm_region.in* file.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Table 25.78: Format of *tfm_region.in*.

Parameter	Description
<i>tfm_regions</i>	
- <i>tfm_region</i>	This settings sub-block specifies the configuration for a TFM region. Repeat this sub-block for each region or group of regions.
<i>region_id</i>	The region(s) in which the subsequent TFM parameters will be applied. Each value given here must also be a <i>region_id</i> in initialize.in . Use square brackets if specifying more than one region. If you specify a region in which the SAGE detailed chemical kinetics solver is not active, TFM will have no effect on that region.
<i>tfm_start_time</i>	Start time (in <i>seconds</i> if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for TFM in the given region. Because TFM may affect the establishment of the initial flame kernel, for an engine simulation you may wish to set this time to be several <i>crank angle degrees</i> after ignition sourcing. If <i>tfm_start_time</i> preceeds the combustion start time (combust.in > <i>general_control</i> > <i>start_time</i>), TFM will have no effect until after combustion begins.
<i>thickening_factor</i>	Maximum thickening factor for TFM in the given region.

Initial Dilution: *dilute_init.in*

To allow a transport passive, *egr_fraction*, to indicate the dilution amount that varies by region, set [combust.in](#) > *sl_model* > *sl_option* > *laminar_flamespeed_correlations* > *general_control* > *dilution = dilute_init.in* and include a *dilute_init.in* file in your case setup. This option is available only when [inputs.in](#) > *simulation_control* > *crank_flag* is non-zero.

```
version: 3.1
---

dilute_init_region_control:
  - dilute_region:
    region_id: [0]
    init_value: 1.0
    init_time: 120.0
    init_period: 720.0
  - dilute_region:
    region_id: [1,2]
    init_value: 0.0
    init_time: 120.0
    init_period: 720.0
```

Figure 25.106: An example *dilute_init.in* file.

Table 25.79: Format of *dilute_init.in*.

Parameter	Description
<i>dilute_init_region_control</i>	
- <i>dilute_region</i>	This settings block specifies the region in which the dilution is set. Repeat this settings sub-block for each region.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description
<i>region_id</i>	The region in which the dilution value is set. The value given here must also be in initialize.in > <i>region_id</i> .
<i>init_value</i>	Initial percentage of exhaust gas dilution.
<i>init_time</i>	Time in <i>crank angle degrees</i> at which to set the initial value.
<i>init_period</i>	Cycle period in <i>crank angle degrees</i> .

Adaptive Zoning Bin Size: *adaptive_zone_bin_*.in*

[Adaptive zoning](#) can include variable bin size(s). When appropriately configured, a variable bin size saves computational time compared to using a fixed fine bin size throughout the entire range of the variable. For example, you can use a variable bin size to refine the zones only in the temperature range in which combustion is most important.

To set up this option, specify a file name for a [combust.in](#) > *adaptive_zone_model* > *bin_options* parameter (e.g., set *adaptive_zone_model* > *bin_options* > *temp = adaptive_zone_bin_temp.in*). You can use the variable bin size option for as many of the binning variables as desired.

Figure 25.107 below shows an example variable bin size file for temperature. Files for other variables are configured in an analogous manner. This file specifies a 10-K bin size for all of the cells in the temperature range of 0 to 1000 K, 5-K bin for cells with temperatures between 1000 and 2000 K, 20-K bin for cells with temperatures from 2000 to 3000 K, and 50-K bin for cells with temperatures greater than 3000 K.

```
TABULAR
SEQUENTIAL
temperature adaptive_zone_tol_temp
0.0      10.0
1000.0   5.0
2000.0   20.0
3000.0   50.0
```

Figure 25.107: An example *adaptive_zone_bin_temp.in* file.

The variable bin size files must be *TABULAR* and *SEQUENTIAL*, and these two words must be the first and second lines, respectively, of the file. The third line must list the column headings: first the variable (*temperature*, *phi*, *phit*, *pressure*, or *<species name>*) and then the corresponding *adaptive_zone_bin_** parameter. For a species file, the species name (e.g., *c7h16*) must match the species name given in [combust.in](#) > *adaptive_zone_model* > *bin_options*.

Table 25.80: Format of *adaptive_zone_bin_*.in*.

Column	Description	Units
1	Lower temperature bound for this adaptive zoning bin.	K
2	Temperature tolerance for this adaptive zoning bin.	K

FGM Lookup Table Generation: fgm.in

The [Flamelet Generated Manifold \(FGM\) model](#) requires a lookup table. You must generate this lookup table before running an FGM simulation.

This section describes the process of generating the lookup table from the command line. Alternatively, you can use CONVERGE Studio to generate an FGM table. See the CONVERGE Studio 3.1 Manual for more information.

The name of the FGM lookup table file (*e.g.*, *fgm_table.h5*) in your case setup must match [*combust.in*](#) > *fgm_model* > *fgm_table_filename*.

When generating the FGM table, we recommend setting *fgm.in* > *fgm_table_control* > *subset_species_flag* = 1. (If *subset_species_flag* = 0, the FGM table becomes large and increases memory usage.) When *subset_species_flag* = 1, you must list the fuel species, oxidizer species, O₂, N₂, CO₂, and H₂O in *fgm.in* > *fgm_table_control* > *subset_species*.

If your simulation includes [NOx modeling](#) and/or [Hiroyasu-NSC soot modeling](#), additional species are required in *fgm.in* > *fgm_table_control* > *subset_species* to obtain NOx and/or soot information. The following table lists these requirements.

Table 25.81: Minimum required species in *fgm_table_control* > *subset_species* in *fgm.in* based on emission model.

Emission model	Minimum required species in <i>fgm_table_control</i> > <i>subset_species</i> in <i>fgm.in</i>
NOx modeling (<i>i.e.</i> , <i>emissions.in</i> > <i>nox_model</i> > <i>nox_thermal</i> > <i>active</i> = 1 or <i>nox_model</i> > <i>nox_prompt</i> > <i>active</i> = 1)	Fuel, oxidizer, O ₂ , N ₂ , CO ₂ , H ₂ O, CO, H ₂ , H, OH, O
Hiroyasu-NSC soot modeling (<i>i.e.</i> , <i>emissions.in</i> > <i>hiroy_soot_model</i> > <i>active</i> = 1)	Fuel, oxidizer, O ₂ , N ₂ , CO ₂ , H ₂ O, CO, H ₂
Hiroyasu-NSC soot modeling using C ₂ H ₂ as soot formation species (<i>i.e.</i> , <i>emissions.in</i> > <i>hiroy_soot_model</i> > <i>hiroy_form_flag</i> = 1)	Fuel, oxidizer, O ₂ , N ₂ , CO ₂ , H ₂ O, CO, H ₂ , C ₂ H ₂

CONVERGE uses the parameters in *fgm.in*, along with a [reaction mechanism file](#) (*e.g.*, *mech.dat*), [thermodynamic data file](#) (*e.g.*, *therm.dat*), [*gas.dat*](#), and, for 1D premixed cases, [*transport.dat*](#) to generate the lookup table. Once you have saved these four files to the Case Directory, go to a terminal, navigate to the Case Directory, and then type

```
<CONVERGE executable> -u fgm
```

to generate the lookup table, where `<CONVERGE executable>` is the name of any CONVERGE executable (*e.g.*, *converge-mpich*). You can run this command in parallel only for the 0D ignition and 1D premixed flamelet types. You need to generate the *fgm_table.h5* file

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

only once unless there are changes to the flamelet type, chemical mechanism, fuel/oxidizer compositions, equation of state, gas transport properties, pressure, or enthalpy.

```
version: 3.1
---

fgm_flamelet_type: 1D-DIFFUSION
fgm_table_control:
  table_size:
    num_zmean: 100
    num_prog_var: 40
    num_enthalpy: 40
    num_zvar: 10
    num_pres: 5
  low_pressure: 1.0
  high_pressure: 120.0
  pressure: 1.0
  subset_species_flag: 1
  subset_species:
    - CH4
    - O2
    - N2
    - CO2
    - H2O
    - CO
    - H2
    - NO
  prog_var_species:
    CO: 1.0000
    CO2: 1.0000
  fgm_temp_loss: 0.333
  fgm_temp_gain: 2.0
fgm_streams_control:
  oxidizer_stream:
    temperature: 291.0
    species:
      O2: 0.23300
      N2: 0.76700
  fuel_stream:
    temperature: 294.0
    species:
      CH4: 0.15640
      O2: 0.19650
      N2: 0.64710
flamelet_generation_control:
  rel_tol: 1e-08
  abs_tol: 1e-20
```

Figure 25.108: An example *fgm.in* file.

Table 25.82: Format of *fgm.in*.

Parameter	Description	Typical value
<i>fgm_flamelet_type</i>	Flamelet type. <i>0D-IGNITION</i> = 0D ignition, <i>1D-DIFFUSION</i> = 1D diffusion, <i>1D-PREMIXED</i> = 1D premixed.	
<i>fgm_table_control</i>		
<i>table_size</i>		
<i>num_zmean</i>	Number of grid points in mean mixture fraction.	1

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>num_prog_var</i>	Number of grid points in reaction progress variable.	40
<i>num_enthalpy</i>	Number of grid points in enthalpy.	40
<i>num_zvar</i>	Number of grid points in mixture fraction variance for 1D FGM.	10
<i>num_pres</i>	Number of grid points in pressure for 0D FGM.	5
<i>low_pressure</i>	Low pressure for 0D FGM (<i>bar</i>).	1
<i>high_pressure</i>	High pressure for 0D FGM (<i>bar</i>).	120
<i>pressure</i>	Pressure for 1D FGM (<i>bar</i>).	1
<i>subset_species_flag</i>	0 = All species are tabulated, 1 = Only a subset of species are tabulated.	
<i>subset_species</i>	This settings sub-block is used to define the species for which species-specific output data will be written. This settings sub-block is used only when <i>subset_species_flag</i> = 1.	
<i><subset species name></i>	Species name. Repeat this line for each subset species. Each species must be on a separate line.	
<i>prog_var_species</i>	This settings sub-block is used for the reaction progress definition.	
<i><species name></i>	Coefficient for this species. Repeat this line for each species. Each species must be on a separate line.	
<i>fgm_temp_loss</i>	Temperature (Z) multiplier for lower heat loss bound.	0.333
<i>fgm_temp_gain</i>	Temperature (Z) multiplier for upper heat gain bound.	2.0
<i>fgm_streams_control</i>		
<i>oxidizer_stream</i>		
<i>temperature</i>	Temperature of oxidizer for 1D FGM (K).	300
<i>species</i>	This settings sub-block is used for the species in the oxidizer (Z = 0 boundary condition)	
<i><species name></i>	Mass fraction for this species in the oxidizer (Z = 0 boundary condition).	

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
Repeat this line for each oxidizer species. Each species must be on a separate line.		
<i>fuel_stream</i>		
<i>temperature</i>	Temperature of fuel for 1D FGM (K).	300
<i>species</i>	This settings sub-block is used for the species in the fuel ($Z = 1$ boundary condition).	
<i><species name></i>	Mass fraction for this species in the fuel ($Z = 1$ boundary condition).	
Repeat this line for each fuel species. Each species must be on a separate line.		
<i>flamelet_generation_control</i>		
<i>rel_tol</i>	Relative tolerance for the flamelet solver.	1e-8
<i>abs_tol</i>	Absolute tolerance for the flamelet solver.	1e-20

ECFM3Z Reinitialization: *ecfm3z_reinit.in*

To specify when and where CONVERGE will reinitialize the combustion domain in an [ECFM](#) or [ECFM3Z](#) simulation, set [*combust.in* > *ecfm_model* > *reinit_flag* = 1](#) or [*ecfm3z_model* > *reinit_flag* = 1](#) and include the *ecfm3z_reinit.in* file in your case setup. (Note that the automatic reinitialization option [[*combust.in* > *ecfm_model* > *reinit_flag* = 2](#) or [*ecfm3z_model* > *reinit_flag* = 2](#)] does not require *ecfm3z_reinit.in*.)

```
version: 3.1
---

ecfm3z_reinit_region_control:
  - ecfm3z_region:
    region_id: [0, 2]
    init_time: 0.0
  - ecfm3z_region:
    region_id: [1]
    init_time: 0.1
```

Figure 25.109: An example *ecfm3z_reinit.in* file.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Table 25.83: Format of *ecfm3z_reinit.in*.

Parameter	Description
<i>ecfm3z_reinit_region_control</i>	
- <i>ecfm3z_region</i>	This settings sub-block specifies the ECFM3Z region. Repeat this sub-block for each ECFM3Z region.
<i>region_id</i>	The region ID(s) for this ECFM or ECFM3Z region. Use [] to specify multiple regions (e.g., [1,2,3]). These values must be consistent with the <i>region_id</i> values in <i>initialize.in</i> .
<i>init_time</i>	Time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) at which to reinitialize the specified regions.

ISSIM: issim.in

To activate the Imposed Stretch Spark Ignition Model, set [*combust.in*](#) > *ecfm_model* > *spark* > *active* = 1 in and include the *issim.in* file in your case setup.

```
version: 3.1
---

general_control:
    c_ignition_mass:          1.0
    c_flame_wrinkling:        2.0
spark_plugs:
    - spark_plug:
        position:            [0,0,0]
        electrode_distance:   0.001
        electrode_diameter:   0.001
        secondary_resistance: 10000.0
        secondary_inductance: 30.0
ignitions:
    - ignition:
        plug_id:              0
        temporal_type:         SEQUENTIAL
        start_time:            0.0
        initial_sec_energy:    0.0389
    - ignition:
        plug_id:              0
        temporal_type:         SEQUENTIAL
        start_time:            0.0
        initial_sec_energy:    0.0389
```

Figure 25.110: Example *issim.in* file.

Table 25.84: Format of *issim.in*.

Parameter	Description	Typical value
<i>general_control</i>		
<i>c_ignition_mass</i>	Correction factor for the ignition mass. Must be at least 1.0.	1.0.
<i>c_flame_wrinkling</i>	Ignition flame surface wrinkling parameter. Set to 1.0 for laminar spherical ignition. Set to greater than 1.0 for turbulent ignition.	2.0

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

Parameter	Description	Typical value
<i>spark_plugs</i>		
- <i>spark_plug</i>	This settings sub-block specifies the configuration of a spark plug. Repeat this sub-block for each spark plug.	
<i>position</i>	The x, y, and z coordinates (<i>m</i>) of the spark plug.	
<i>electrode_distance</i>	Inter-electrode distance (<i>m</i>).	
<i>electrode_diameter</i>	Electrode diameter (<i>m</i>).	
<i>secondary_resistance</i>	Secondary resistance (<i>ohm</i>).	
<i>secondary_inductance</i>	Secondary inductance (<i>Henry</i>).	
<i>ignitions</i>		
- <i>ignition</i>	This settings sub-block specifies the ignition configuration. Repeat this sub-block for each ignition.	
<i>plug_id</i>	The number of the spark plug associated with this ignition. The first spark plug has an index of 0.	
<i>temporal_type</i>	Temporal type: SEQUENTIAL or CYCLIC.	
<i>period</i>	The CYCLIC ignition period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Ignition start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>initial_sec_energy</i>	Initial secondary circuit energy (J).	

G-Equation Spark: g_eqn_spark.in

To use the kernel model for spark-ignition in the [G-Equation model](#), set *combust.in* > *g_eqn_model* > *spark* > *active* = 1 and include the *g_eqn_spark.in* file in your case setup.

Chapter 25: Input and Data Files

Physical Models Combustion Modeling

```
version: 3.1
---

general_control:
  spark_efficiency:          0.3
  num_kernel_init:           100
  karlovitz_ignition:        80
  g_eqn_c_chi:               2

spark_plugs:
  - spark_plug:
    spark_control:
      position: [0,0,0]
      gap: 0.001
      gap_direction: [1,0,0]
      max_displace: 30.0
    temporal_control:
      type: CYCLIC
      cyclic_period: 720
      start_time: 30.0
      end_time: 30.0
```

Figure 25.111: An example *g_eqn_spark.in* file.

Table 25.85: Format of *g_eqn_spark.in*.

Parameter	Description
<i>general_control</i>	
<i>spark_efficiency</i>	Spark efficiency for the kernel model.
<i>num_kernel_init</i>	Number of spark kernels initialized for the kernel model.
<i>karlovitz_ignition</i>	Karlovitz number at which ignition occurs.
<i>g_eqn_c_chi</i>	Mixture fraction variance dissipation constant.
<i>spark_plugs</i>	
- <i>spark_plug</i>	This settings sub-block specifies the configuration of a spark plug. Repeat this sub-block for each spark plug.
<i>spark_control</i>	
<i>position</i>	The x, y, and z coordinates (<i>m</i>) of the spark plug.
<i>gap</i>	Spark plug gap (<i>m</i>).
<i>gap_direction</i>	The x, y, and z components of the spark plug gap vector.
<i>max_displace</i>	The maximum displacement (<i>m</i>) of the spark kernels.
<i>temporal_control</i>	
<i>type</i>	The temporal type (SEQUENTIAL or CYCLIC) of the spark plug.
<i>cyclic_period</i>	Spark plug period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>start_time</i>	Spark plug start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>end_time</i>	Spark plug end time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

Emissions Modeling

This section describes input files that contain options for emissions modeling in CONVERGE.

Emissions: `emissions.in`

To model `emissions`, activate `combustion modeling` (set `inputs.in > feature_control > combustion_flag = 1` and include a `combust.in` file in your case setup), set `combust.in > emissions_flag = 1`, and include an `emissions.in` file in your case setup.

```
version: 3.1
---

nox_model:
  nox_thermal:
    active: 0
    rate_flag: 0
    nox_radical_model: 0
  nox_prompt:
    active: 0
    equiv_ratio: 0
  nox_scaling_factor: 0
hiroy_soot_model:
  active: 0
  hiroy_asf : 0
  hiroy_esf: 0
  hiroy_diam: 0
  hiroy_oxid_factor: 0
  hiroy_density: 0
  hiroy_form_flag: 1
phenom_soot_model:
  active: 0
  phenom_inception_factor: 0
  phenom_coagulation_factor: 0
  phenom_oh_collision_factor: 0
  phenom_no_oxidation_factor: 0
  phenom_surface_growth_value: 0
detailed_soot_model:
  active: PM
  coupled_solver_flag: 1
  detailed_soot_condensation_flag: 0
  detailed_soot_alpha_corrector_flag: 0
  pm_num_moments: 0
  pm_surface_growth_factor: 0
  pm_alpha_corrector: 0
  psm_num_sections: 0
  psm_num_subsections: 0
  psm_surface_growth_factor1: 0
  psm_surface_growth_factor2: 0
  psm_alpha_corrector1: 0
  psm_alpha_corrector2: 0
  psm_biggestsoot_diameter: 0
custom_soot_precursor:
  active: 0
  precursor_species: [C2H2,A2R5]
```

Figure 25.112: An example `emissions.in` file.

Table 25.86: Format of `emissions.in`.

Parameter	Description	Typical value
<code>nox_model</code>		

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

Parameter	Description	Typical value
<hr/>		
<i>nox_thermal</i>		
<i>active</i>	0 = No Extended Zel'dovich (thermal) NOx model, 1 = Extended Zel'dovich NOx model.	
<i>rate_flag</i>	0 = Default NOx rate coefficient, 1 = User-specified NOx rate coefficient (passive nox rate.dat required).	
<i>nox_radical_model</i>	Assumption made for O/OH model: 0 = Default equilibrium, 1 = Partial equilibrium, 2 = No assumptions, <i>i.e.</i> , calculate O/OH. (Requires O and OH in the reaction mechanism file . Recommended for $T_{max} < 2000$ K).	
<hr/>		
<i>nox_prompt</i>		
<i>active</i>	0 = No prompt NOx model, 1 = Prompt NOx model.	
<i>equiv_ratio</i>	Global equivalence ratio used in the prompt NOx model.	1.2
<i>nox_scaling_factor</i>	Mass scaling factor to convert NO to NOx.	1.533
<hr/>		
<i>hiroy_soot_model</i>		
<i>active</i>	0 = No Hiroyasu-NSC soot model, 1 = Hiroyasu-NSC soot model.	
<i>hiroy_asf</i>	Soot formation pre-exponential factor ($s^{-1}\cdot bar^{0.5}$).	1e2 - 6e2
<i>hiroy_esf</i>	Soot formation activation energy parameter (cal/mol).	1.25e4
<i>hiroy_diam</i>	Soot particle diameter (cm).	2.5e-6
<i>hiroy_oxid_factor</i>	Soot oxidation model factor.	1 - 5
<i>hiroy_density</i>	Soot density (g/cm^3).	2.0
<i>hiroy_form_flag</i>	This flag is used only when a detailed reaction mechanism is available. 0 = Use the sum of the hydrocarbon species as soot formation species, 1 = Use C2H2 as the soot formation species.	
<hr/>		
<i>phenom_soot_model</i>		
<i>active</i>	0 = No phenomenological soot model, <i>GOKUL</i> = Gokul soot model, <i>DALIAN</i> = Dalian soot model,	

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

Parameter	Description	Typical value
	WASEDA = Waseda soot model.	
<i>phenom_inception_factor</i>	Soot inception pre-exponential factor. A larger value results in higher soot level.	Gokul: 2e3, Dalian: 1e11, Waseda: 1e3.
<i>phenom_coagulation_factor</i>	Soot coagulation factor. A larger value results in higher soot level.	Gokul: 9, Dalian: 2, Waseda: 4.299.
<i>phenom_oh_collision_factor</i>	Soot oxidation of OH collision factor. A larger value results in lower soot level.	1.3e-1
<i>phenom_no_oxidation_factor</i>	Soot oxidation of NO factor. A larger value results in lower soot level. Used only for Waseda model (<i>phenom_soot_model > active</i> = WASEDA).	1.82
<i>phenom_surface_growth_value</i>	Soot surface growth factor. A larger value results in higher soot level.	Gokul: 9e4, Dalian: 1.05e4, Waseda: 3e-1.
<i>detailed_soot_model</i>		
<i>active</i>	0 = No detailed soot model, <i>PM</i> = Particulate Mimic (PM) model , <i>PSM</i> = Particulate Size Mimic (PSM) model , <i>SSM</i> = Sectional Soot Model (SSM) (requires <i>ssm_soot.in</i>).	
<i>coupled_solver_flag</i>	0 = Do not couple detailed soot model with detailed chemistry, 1 = Couple detailed soot model with detailed chemistry.	
<i>detailed_soot_condensation_flag</i>	0 = No detailed soot condensation submodel, 1 = Activate detailed soot condensation submodel (required if <i>detailed_soot_model > active</i> is non-zero).	
<i>detailed_soot_alpha_corrector_flag</i>	0 = Use alpha corrector values specified in <i>emissions.in</i> (<i>pm_alpha_corrector</i> for the PM models and <i>psm_alpha_corrector1</i> and <i>psm_alpha_corrector2</i> for the PSM models), 1 = Calculate alpha corrector dynamically based on local information.	

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

Parameter	Description	Typical value
<i>pm_num_moments</i>	Number of moments for PM model. Must be between 2 and 6, inclusive.	
<i>pm_surface_growth_factor</i>	Soot dependence for the PM model surface reactions. -1 = No surface reactions, 0.0 = Function of number density, 2.0 = Function of surface area, 3.0 = Function of soot volume. Intermediate values between 0.0 and 2.0 indicate a weighted average between the number density and surface area approaches, and intermediate values between 2.0 and 3.0 indicate a weighted average between the surface area and soot volume approaches.	2.25 for diesel fuel cases
<i>pm_alpha_corrector</i>	Fraction of sites on soot surface available for surface reactions in the PM model.	3e-1 for diesel fuel cases
<i>psm_num_sections</i>	Number of sections (soot volume fraction) in the PSM model.	
<i>psm_num_subsections</i>	Number of subsections for each section in the PSM model.	
<i>psm_surface_growth_factor1</i>	Describes the soot dependence from precursor size to 40 nm for the PSM model surface reactions.	2.0
<i>psm_surface_growth_factor2</i>	Describes the soot dependence from 40 nm to <i>psm_biggestsoot_diameter</i> for the PSM model surface reactions.	2.25
<i>psm_alpha_corrector1</i>	Fraction of sites on the soot surface available for PSM model surface reactions in the range of precursor size to 40 nm.	9.5e-1
<i>psm_alpha_corrector2</i>	Fraction of sites on the soot surface available for PSM model surface reactions in the range of 40 nm to <i>psm_biggestsoot_diameter</i> .	3e-1
<i>psm_biggestsoot_diameter</i>	Biggest soot size for PSM model (m).	1e-7
<i>custom_soot_precursor</i>		
<i>active</i>	Soot precursor used to initialize the phenomenological, PM or PSM models. 0 = Default soot precursor (PAH), 1 = Use user-specified soot precursors to initialize the selected soot model	
<i>precursor_species</i>	List of species to use as soot precursors.	

Sectional Soot Model: ssm_soot.in

To use the [Sectional Soot Model \(SSM\)](#) detailed soot model, set *detailed_soot_model > active = SSM* in [emissions.in](#) and include an *ssm_soot.in* file in your case setup.

```
version: 3.1
---

ssm_num_sections: 0
ssm_biggestsoot_diameter: 0
ssm_soot_density: 0
ssm_coupled_solver_flag: 1
ssm_soot_diffusion_model: 1
ssm_soot_collision_model:
    active: 0
    efficiency_model: 0
ssm_soot_nucleation_model:
    active: 1
    precursor_species:
        A4: 1.0
        A3: 1.0
ssm_soot_condensation_model:
    active: 1
    condensation_species:
        A4: 1.0
        A3: 1.0
    nu_factor: 500.0
ssm_soot_coagulation_model:
    active: 1
    effective_factor: 1.0
ssm_soot_surface_growth_model:
    active: 1
    alpha_factor: 0.3
    correlation_bkl_factor: 5.54
    steric_cut: 4.0e-9
    steric_width: 4.0e-09
    steric_min: 2.0
    steric_max: 2.0
    haca_reaction_control:
        - haca_reaction:
            id: 1
            reactants:
                Csoot: 1.0
                H: 1.0
            products:
                Csootstar: 1.0
                H2: 1.0
                cf_HACA: -1.0
                size: 0
                activation_e: 0
                a_factor: 1.0
                b_factor: 0
                ps: 0
                activation_e_rev: 0
                a_factor_rev: 0
                b_factor_rev: 0
                ps_rev: 0
        - haca_reaction:
            id: 2
            reactants:
                Csoot: 1.0
                H: 1.0
            products:
                Csootstar: 1.0
                H2: 1.0
                cf_HACA: -1.0
                size: 0
                activation_e: 0
                a_factor: 1.0
```

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

```
b_factor:          0
ps:               0
activation_e_rev: 0
a_factor_rev:     0
b_factor_rev:    0
ps_rev:          0
```

Figure 25.113: An example *ssm_soot.in* file.

Table 25.87: Format of *ssm_soot.in*.

Parameter	Description
<i>ssm_num_sections</i>	Number of sections.
<i>ssm_biggestsoot_diameter</i>	Biggest soot size for the SSM soot model (nm).
<i>ssm_soot_density</i>	Soot density (kg/m ³).
<i>ssm_coupled_solver_flag</i>	0 = Do not consider the species that are consumed by inception, surface growth, and oxidation reactions (one-way coupling), 1 = Consider the species that are consumed by inception, surface growth, and oxidation reactions (two-way coupling).
<i>ssm_soot_diffusion_model</i>	0 = No molecular diffusion for soot, 1 = Diffusion based on benzene coefficient and number of carbon atoms in section as described in Mauss et al. (1994) .
<i>ssm_soot_collision_model</i>	
<i>active</i>	0 = Do not use soot collision model, 1 = Use soot collision model (required).
<i>efficiency_model</i>	Collision efficiency term. 0 = Use collision efficiency as described in Aubagnac-Karkar et al. (2018) , 1 = Use collision efficiency function as described in Raj et al. (2010) that depends on soot diameter.
<i>ssm_soot_nucleation_model</i>	
<i>active</i>	0 = Do not use soot nucleation model, 1 = Use non-reversible soot nucleation model, 2 = Use reversible soot nucleation model, 3 = Use non-reversible nucleation model with a damping factor.
<i>precursor_species</i>	This settings sub-block is used only when <i>ssm_soot.in</i> > <i>ssm_soot_collision_model</i> > <i>efficiency_model</i> = 0.
< <i>species name</i> >	Collision effectiveness factor for nucleation. Repeat this line for each precursor species. Each species must be on a separate line.
<i>ssm_soot_condensation_model</i>	
<i>active</i>	0 = Do not use soot condensation model, 1 = Use non-reversible soot condensation model,

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

Parameter	Description
	2 = Use reversible condensation model.
<i>condensation_species</i>	This settings sub-block is used only when <i>ssm_soot.in > ssm_soot_collision_model > efficiency_model = 0</i> .
< <i>species name</i> >	Collision effectiveness factor for condensation.
	Repeat this line for each condensation species. Each species must be on a separate line.
<i>nu_factor</i>	Intermolecular vibrational wavenumber used to calculate the reversiblity of condensation (m^{-1}). Only used when <i>ssm_soot_condensation_model > active = 2</i> .
<i>ssm_soot_coagulation_model</i>	
<i>active</i>	0 = Do not use soot coagulation model, 1 = Use soot coagulation model.
<i>effective_factor</i>	Collision effectiveness factor for coagulation. Used only when <i>ssm_soot.in > ssm_soot_collision_model > efficiency_model = 0</i> .
<i>ssm_soot_surface_growth_model</i>	
<i>active</i>	0 = Do not use soot surface growth model, 1 = Use soot surface growth model with C_{soot}^* as the quasi-steady-state species, 2 = Use soot surface growth model with direct C_{soot}^* calculation.
<i>alpha_factor</i>	Proportion of active sites. Must be between 0 and 1.
<i>correlation_bkl_factor</i>	Correction factor for the number of sites per unit surface in order to agree with Appel et al. (2000) . We recommend a value of 1.
<i>steric_cut</i>	Mean particle diameter for the variation zone of the steric factor (<i>m</i>).
<i>steric_width</i>	Width of the variation zone of the steric factor (<i>m</i>).
<i>steric_min</i>	Value of the steric factor for the smallest particles.
<i>steric_max</i>	Value of the steric factor for the largest particles.
<i>haca_reaction_control</i> *	This sub-block describes the Hydrogen Abstraction Carbon Addition (HACA) reactions.
- <i>haca_reaction</i>	This sub-block specifies the parameters for a HACA reaction. Repeat this sub-block for each reaction.
<i>id</i>	Reaction ID.
<i>reactants</i>	This sub-block specifies the HACA reactants.
< <i>reactant species</i> >	Stoichiometric coefficient for the species.

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

Parameter	Description
	Repeat this line for each reactant species. You must include at least two reactants and <i>Csoot</i> must be one of them. Each reactant species must be on a separate line.
<i>products</i>	This sub-block specifies the HACA products.
< <i>product species</i> >	Stoichiometric coefficient for the species.
	Repeat this line for each product species. You must include at least two products and <i>Csootstar</i> must be one of them. Each product species must be on a separate line.
<i>cf_haca</i>	Indicate whether to include reaction in quasi-steady-state assumption for computing <i>Csoot/Csootstar</i> ratio. -1 = Do not use reaction to compute quasi-steady-state multiplier but apply the computed multiplier to the reaction, 0 = Do not use reaction to compute quasi-steady-state multiplier and do not apply computed multiplier to the reaction, 1 = Use reaction to compute quasi-steady-state multiplier.
<i>size</i>	Indicate how surface growth rate is calculated. 0 = Use normal Arrhenius equation, 1 = Use Arrhenius equation with probability factor (<i>alpha_factor * correlation_bkl_factor</i>).
<i>activation_e</i>	Activation energy for the forward reaction (E_a in Equation 25.88 below).
<i>a_factor</i>	Pre-exponential factor for the forward reaction (A in Equation 25.88 below).
<i>b_factor</i>	Exponent for the temperature in the pre-exponential factor of the forward reaction (b in Equation 25.88 below).
<i>ps</i>	Exponent associated with the number of carbon atoms in the soot for the forward reaction (ps in Equation 25.88 below).
<i>activation_e_rev</i>	Activation energy for the reverse reaction (E_a in Equation 25.88 below).
<i>a_factor_rev</i>	Pre-exponential factor for the reverse reaction (A in Equation 25.88 below).
<i>b_factor_rev</i>	Exponent for the temperature in the pre-exponential factor of the reverse reaction (b in Equation 25.88 below).
<i>ps_rev</i>	Exponent associated with the number of carbon atoms in the soot for the reverse reaction (ps in Equation 25.88 below).

$$k = AT^b \exp\left(-\frac{E_a}{RT}\right) (n_c)^{ps} \quad (25.88)$$

Chapter 25: Input and Data Files

Physical Models Emissions Modeling

Passive NOx Rate: **passive_nox_rate.dat**

To activate the user-specified NOx rate coefficient option, set *emissions.in* > *nox_model* > *nox_thermal* > *rate_flag* = 1 and include a *passive_nox_rate.dat* file in your case setup. CONVERGE evaluates each reaction rate as

$$k = A \cdot T_{mult} \cdot \exp(-E_R T). \quad (25.89)$$

```
# 1. O + N2 <-> NO + N      (rate_1_for, rate_1_rev)
FOR1  7.6e13  0.0  38000.0
REV1  1.6e13  0.0  0.0

# 2. N + O2 <-> NO + O      (rate_2_for, rate_2_rev)
FOR2  6.4e9   1    3150.0
REV2  1.5e9   1    19500.0

# 3. N + OH <-> NO + H      (rate_3_for, rate_3_rev)
FOR3  4.1e13  0    0
REV3  2e14   0    23650.0
```

Figure 25.114: An example *passive_nox_rate.dat* file with rate constants described in [Heywood \(1988\)](#) (recommended for ICE cases).

```
# 1. O + N2 <-> NO + N      (rate_1_for, rate_1_rev)
FOR1  1.8e14  0.0  38370.0
REV1  3.8e13  0.0  425.0

# 2. N + O2 <-> NO + O      (rate_2_for, rate_2_rev)
FOR2  1.8e10  1    4680.0
REV2  3.8e9   1    20820.0

# 3. N + OH <-> NO + H      (rate_3_for, rate_3_rev)
FOR3  7.1e13  0    450
REV3  1.7e14  0    24560.0
```

Figure 25.115: An example *passive_nox_rate.dat* file with rate constants derived from [Hanson and Salimian \(1984\)](#) (recommended for gas turbine cases).

Table 25.88: Format of *passive_nox_rate.dat*.

Column	Parameter	Units
1	FOR<ID> = Forward reaction rate for reaction number <ID>, REV<ID> = Reverse reaction rate for reaction number <ID>.	
2	Arrhenius pre-exponential factor $cm^3/mol\cdot s$ A .	
3	Multiplicative temperature factor T_{mult}	
4	Activation energy divided by the K universal gas constant E_R .	

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Turbulence: turbulence.in

To model [turbulence](#), set `inputs.in > solver_control > turbulence_solver = 1` and include a `turbulence.in` file in your case setup.

```
version: 3.1
---

turbulence_model: RANS_K_OMEGA_STD
RANS_models:
  k_eps_models:
    keps_cmu: 0.09
    keps_rpr_tke: 1.0
    keps_ceps1: 1.44
    keps_ceps2: 1.96
    keps_ceps3: -1.0
    keps_rpr_eps: 0.7692
    keps_rng_beta: 0.012
    keps_rng_eta0: 4.38
    keps_rng_gen_b0: 2.0725
    keps_rng_gen_b1: -0.3865
    keps_rng_gen_b2: 0.083
  v2f_zetaf_models:
    keps_v2f_cmu: 0.22
    keps_v2f_c1: 1.4
    keps_v2f_c2: 0.3
    keps_v2f_c1: 0.23
    keps_v2f_ceta: 70.0
    keps_zetaf_rpr_zeta: 0.8333
    keps_zetaf_c2prime: 0.65
  k_omega_models:
    komega_cmu: 0.09
    komega_rpr_tke: 0.85
    komega_rpr_omega: 0.5
    komega_alpha: 0.556
    komega_beta: 0.075
    komega_clim: 0.875
    komega_sst_a1: 0.31
    komega_rpr_tke_outer: 1.0
    komega_rpr_omega_outer: 0.856
    komega_alpha_outer: 0.44
    komega_beta_outer: 0.0828
    komega_transition_model: 0
  reynolds_stress_models:
    rsm_cmu: 0.0986
    rsm_diffusion_model: iso
    rsm_sigma_k: 1.0
    rsm_sigma_eps: 1.3
    rsm_cs: 0.22
    rsm_cs_eps: 0.18
    rsm_ceps1: 1.44
    rsm_ceps2: 1.83
    rsm_ceps3: -1.0
  LRR_model:
    lrr_c1: 1.8
    lrr_c2: 0.6
    lrr_wall_reflection: 1
    lrr_cw1: 0.5
    lrr_cw2: 0.3
    lrr_cw2_c1: -0.044
    lrr_cw2_c2: -0.08
    lrr_cw2_c3: 0.6
  SSG_model:
    ssg_c1: 3.4
    ssg_cls: 1.8
    ssg_c2: 4.2
    ssg_c3: 0.8
    ssg_c3s: 1.3
    ssg_c4: 1.25
```

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

```

    ssg_c5:          0.4
spalart_allmaras_model:
    sa_cb1:         0.1355
    sa_cb2:         0.622
    sa_cw2:         0.3
    sa_cw3:         2.0
    sa_cv1:         7.1
    sa_cs:          0.3
    sa_rpr_sigma:   1.5
DES_models:
    ddes_komegasst_cdes: 0.78
    ddes_komegasst_cdes_outer: 0.61
    ddes_komegasst_cd1: 20.0
    ddes_komegasst_cd2: 3.0
    iddes_komegasst_cw: 0.15
    iddes_komegasst_cl: 5.0
    iddes_komegasst_ct: 1.87
LES_models:
    les_rpr_sgs_ke: 1.0
    les_c_sgs_ke: 2.0
    les_c_sgs_ke_visc: 0.05
    les_c_sgs_eps: 2.0
    les_wall_model: 1
Wall_modeling:
    wall_dist_flag: 0
    heat_model: 2
    near_wall_treatment: ENHANCED
    law_kappa: 0.42
    law_c: 5.5
Physics_effects:
    discrete_c_s: 0.0
    discrete_c_ps: 0.03
    buoyancy_flag: 0
Turbulence_statistics:
    turb_stat_flag: 0
    turb_stat_start_time: 180.3
    turb_stat_end_time: 357.6
    turb_stat_tol: 0.0001

```

Figure 25.116: An example *turbulence.in* file.

Table 25.89: Format of *turbulence.in*.

Parameter	Description	Typical value
<i>turbulence_model</i>	<i>RANS_K_EPS_STD</i> = $k-\epsilon$, <i>RANS_K_EPS RNG</i> = RNG $k-\epsilon$, <i>RANS_K_EPS RNG RD</i> = Rapid Distortion RNG $k-\epsilon$, <i>RANS_K_EPS RNG GEN</i> = Generalized RNG $k-\epsilon$, <i>RANS_K_EPS REAL</i> = Realizable $k-\epsilon$, <i>RANS_K_EPS_V2F</i> = v^2-f model, <i>RANS_K_EPS_ZETAf</i> = $\zeta-f$ model, <i>RANS_K_OMEGA_STD_98</i> = Standard $k-\omega$ 1998, <i>RANS_K_OMEGA_STD</i> = Standard $k-\omega$ 2006, <i>RANS_K_OMEGA_SST</i> = $k-\omega$ SST, <i>RANS_RSM_SSG_EPS</i> = RSM SSG, <i>RANS_RSM_LRR_EPS</i> = RSM LRR, <i>RANS_SPALART_ALLMARAS</i> = Spalart-Allmaras, <i>DDES_K_OMEGA_SST</i> = Delayed detached eddy simulation,	

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Parameter	Description	Typical value
	<i>IDDES_K_OMEGA_SST</i> = Delayed detached eddy simulation with improved wall-modeling capabilities, <i>LES_UPWIND</i> = Upwind LES, <i>LES_ONE_EQN_VISC</i> = One-equation eddy viscosity LES, <i>LES_SMAG</i> = Smagorinsky model, <i>LES_DYN_SMAG</i> = Dynamic Smagorinsky model, <i>LES_DYN_STRUCT</i> = Dynamic structure model, <i>LES_CON_DYN_STRUCT</i> = Consistent dynamic structure model, <i>LES_SIGMA</i> = Sigma LES model, <i>LES_TWO_EQN</i> = Two-equation $k-\epsilon$ LES.	
<i>RANS_models</i>		
<i>k_eps_models</i>	RANS $k-\epsilon$ model constants.	
<i>keps_cmu</i>	Turbulent viscosity coefficient for $k-\epsilon$ models.	
<i>keps_rpr_tke</i>	Reciprocal tke Prandtl number for $k-\epsilon$ models.	
<i>keps_ceps1</i>	ϵ equation coefficient for $k-\epsilon$ models.	
<i>keps_ceps2</i>	ϵ equation coefficient for $k-\epsilon$ models.	
<i>keps_ceps3</i>	ϵ equation coefficient for $k-\epsilon$ models. CONVERGE ignores this parameter if <i>turbulence.in</i> > <i>turbulence_model</i> = <i>RANS_K_EPS_RNG_RD</i> .	- 1.0 to 1.0
<i>keps_rpr_eps</i>	Reciprocal ϵ Prandlt number for $k-\epsilon$ models.	
<i>keps_rng_beta</i>	ϵ equation coefficient for $k-\epsilon$ models.	0.012
<i>keps_rng_eta0</i>	ϵ equation coefficient for $k-\epsilon$ models.	4.38
<i>keps_rng_gen_b0</i>	Model coefficient for the generalized RNG $k-\epsilon$ model.	2.0725
<i>keps_rng_gen_b1</i>	Model coefficient for the generalized RNG $k-\epsilon$ model.	-0.3865
<i>keps_rng_gen_b2</i>	Model coefficient for the generalized RNG $k-\epsilon$ model.	0.083
<i>v2f_zetaf_models</i>	RANS v^2-f and $\zeta-f$ model constants.	
<i>keps_v2f_cmu</i>	v^2-f model equation coefficient.	0.22
<i>keps_v2f_c1</i>	Equation coefficient for v^2-f and $\zeta-f$ models.	
<i>keps_v2f_c2</i>	v^2-f model equation coefficient.	0.3
<i>keps_v2f_cl</i>	Equation coefficient for v^2-f and $\zeta-f$ models.	
<i>keps_v2f_ceta</i>	Equation coefficient for v^2-f and $\zeta-f$ models.	

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Parameter	Description	Typical value
<i>keps_zetaf_rpr_zeta</i>	$\zeta-f$ model equation coefficient.	0.83333
<i>keps_zetaf_c2prime</i>	$\zeta-f$ model equation coefficient.	0.65
<i>k_omega_models</i>	RANS $k-\omega$ model constants.	
<i>komega_cmu</i>	Turbulent viscosity coefficient for $k-\omega$ models.	0.9
<i>komega_rpr_tke</i>	Reciprocal tke Prandtl number for $k-\omega$ models.	
<i>komega_rpr_omega</i>	Reciprocal ω Prandtl number for $k-\omega$ models.	0.5
<i>komega_alpha</i>	Equation coefficient for $k-\omega$ models.	
<i>komega_beta</i>	ε equation coefficient for $k-\omega$ models.	
<i>komega_clim</i>	Eddy viscosity constant limit for standard $k-\omega$ 2006.	0.875
<i>komega_sst_a1</i>	Equation coefficient for $k-\omega$ SST models.	0.31
<i>komega_rpr_tke_outer</i>	Outer reciprocal tke Prandtl number for $k-\omega$ SST models.	1.0
<i>komega_rpr_omega_outer</i>	Outer reciprocal omega Prandtl number for $k-\omega$ SST models.	0.856
<i>komega_alpha_outer</i>	Outer equation coefficient for $k-\omega$ SST models.	0.44
<i>komega_beta_outer</i>	Outer equation coefficient for $k-\omega$ SST models.	0.0828
<i>komega_transition_model</i>	0 = Do not apply low-Reynolds-number correction factor (recommended), 1 = Apply low-Reynolds-number correction factor. This model is not recommended for use with <i>turbulence.in > turbulence_model</i> = <i>RANS_K_OMEGA_STD_98</i> or <i>RANS_K_OMEGA_SST</i> .	0
<i>reynolds_stress_models</i>	Reynolds Stress Model (RSM) model constants.	
<i>rsm_cmu</i>	Model constant for the RSMs.	0.1156 for LRR without wall reflection, 0.0655 for LRR with wall reflection, 0.0985 for SSG.
<i>rsm_diffusion_model</i>	Diffusion model flag for the RSMs. <i>iso</i> or 0 = Isotropic diffusion model, <i>aniso_rs</i> or 1 = Anisotropic diffusion model for the Reynolds stress only, <i>aniso_all</i> or 2 = Anisotropic diffusion model for both the Reynolds stress and <i>eps</i> .	
<i>rsm_sigma_k</i>	Model constant for the RSMs.	1.0

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Parameter	Description	Typical value
<i>rsm_sigma_eps</i>	Model constant for the RSMs.	1.3
<i>rsm_cs</i>	Model constant for the RSMs.	0.22
<i>rsm_cs_eps</i>	Model constant for the RSMs.	0.18
<i>rsm_ceps1</i>	Model constant for the RSMs.	1.44
<i>rsm_ceps2</i>	Model constant for the RSMs.	1.92 for LRR, 1.83 for SSG.
<i>rsm_ceps3</i>	Model constant for the RSMs.	-1.0
<i>LRR_model</i>		
<i>lrr_c1</i>	Model constant for the LRR model.	1.8
<i>lrr_c2</i>	Model constant for the LRR model.	0.6
<i>lrr_wall_reflection</i>	LRR wall reflection flag. 0 = Do not include the wall reflection term, 1 = Include the Gibson & Launder wall reflection model, 2 = Include the Craft wall reflection model.	
<i>lrr_cw1</i>	Model constant for the LRR model.	0.5
<i>lrr_cw2</i>	Model constant for the LRR model.	0.3
<i>lrr_cw2_c1</i>	Model constant for the LRR model.	-0.044
<i>lrr_cw2_c2</i>	Model constant for the LRR model.	-0.08
<i>lrr_cw2_c3</i>	Model constant for the LRR model.	0.6
<i>SSG_model</i>		
<i>ssg_c1</i>	Model constant for the SSG model.	3.4
<i>ssg_c1s</i>	Model constant for the SSG model.	1.8
<i>ssg_c2</i>	Model constant for the SSG model.	4.2
<i>ssg_c3</i>	Model constant for the SSG model.	0.8
<i>ssg_c3s</i>	Model constant for the SSG model.	1.3
<i>ssg_c4</i>	Model constant for the SSG model.	1.25
<i>ssg_c5</i>	Model constant for the SSG model.	0.4
<i>spalart_allmaras_model</i>		
<i>sa_cb1</i>	Model constant for the Spalart-Allmaras model.	0.1355
<i>sa_cb2</i>	Model constant for the Spalart-Allmaras model.	0.622
<i>sa_cw2</i>	Model constant for the Spalart-Allmaras model.	0.3
<i>sa_cw3</i>	Model constant for the Spalart-Allmaras model.	2.0

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Parameter	Description	Typical value
<i>sa_cv1</i>	Model constant for the Spalart-Allmaras model.	7.1
<i>sa_cs</i>	Model constant for the Spalart-Allmaras model.	0.3
<i>sa_rpr_sigma</i>	Model constant for the Spalart-Allmaras model.	1.5
<i>DES_models</i>	DES model constants ($k-\omega$ SST based).	
<i>ddes_komegasst_cdes</i>	Model constant for DDES and IDDES models.	0.78
<i>ddes_komegasst_cdes_outer</i>	Model constant for DDES and IDDES models.	0.61
<i>ddes_komegasst_cd1</i>	Model constant for DDES and IDDES models.	20.0
<i>ddes_komegasst_cd2</i>	Model constant for DDES and IDDES models.	3.0
<i>iddes_komegasst_cw</i>	Model constant for the IDDES model.	0.15
<i>iddes_komegasst_cl</i>	Model constant for the IDDES model.	5.0
<i>iddes_komegasst_ct</i>	Model constant for the IDDES model.	1.87
<i>LES_models</i>	LES model constants.	
<i>les_rpr_sgs_ke</i>	Reciprocal sgs_ke Prandtl number for one-equation LES models.	1.0
<i>les_c_sgs_ke</i>	Constant used in estimating sub-grid values.	1.0 to 5.0
<i>les_c_sgs_ke_visc</i>	LES sub-grid viscosity model constant.	0.05 (for the one-equation eddy viscosity and consistent dynamic structure models) 0.5 (for the dynamic structure and two-equation $k-\varepsilon$ models) 0.1 to 0.2 (for the Smagorinsky model)
<i>les_c_sgs_eps</i>	LES dissipation rate model constant. Used for one- and two-equation LES models.	1.0
<i>les_wall_model</i>	0 = Standard law-of-the-wall, 1 = Werner and Wengle wall model.	
<i>Wall_modeling</i>		
<i>wall_dist_flag</i>	Wall distance calculation scheme . 0 = Use 0.5 times the cell size,	

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Parameter	Description	Typical value
	1 = Use 0.3 times the cell size.	
<i>near_wall_treatment</i>	RANS wall treatment flag. <i>k-ε</i> wall treatment options STANDARD = Standard wall function, SCALABLE = Scalable wall function , NON_EQUILIBRIUM = Non-equilibrium wall function . ENHANCED = Enhanced wall treatment , ANALYTIC = Analytic wall treatment .	
	<i>v²-f</i> and <i>ζ-f</i> wall treatment options STANDARD = Standard wall function, ENHANCED = Enhanced wall treatment .	
	<i>k-ω</i> wall treatment options STANDARD = Standard wall function, AUTOMATIC = Automatic wall function , ASYMPTOTIC = Menter's wall boundary condition if <i>turbulence.in > turbulence_model</i> = RANS_K_OMEGA_SST, otherwise automatic wall function with Wilcox's low-Re corrections, ENHANCED = Enhanced wall treatment .	
	Reynolds Stress Model (RSM) wall treatment options STANDARD = Standard wall function. ENHANCED = Enhanced wall treatment .	
	Spalart-Allmaras wall treatment options STANDARD = Standard wall function .	
<i>heat_model</i>	0 = Amsden, 1 = Han and Reitz, 2 = Angelberger, 3 = GruMo-UniMORE, 4 = Jayatilleke.	
<i>law_kappa</i>	von Karman's constant.	0.42
<i>law_c</i>	Law-of-the-wall parameter.	5.5
<i>law_ck</i>	Slope of the log-law relation between y^+ and k^+ . Used only if <i>boundary.in > turbulence > tke > type = LAW_OF_WALL</i> .	-0.416
<i>law_bk</i>	Intercept of the log-law relation between y^+ and k^+ . Used only if <i>boundary.in > turbulence > tke > type = LAW_OF_WALL</i> .	8.366

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Parameter	Description	Typical value
<i>law_ceps2</i>	Constant from the dissipation term of the turbulence dissipation transport equation. Used only if boundary.in > turbulence > tke > type = LAW_OF_WALL.	1.9
<i>law_cvisc</i>	Integration constant from derivation of viscous sublayer solution of near wall tke. Used only if boundary.in > turbulence > tke > type = LAW_OF_WALL.	11.0
<i>Physics_effects</i>	Other physics effects.	
<i>discrete_c_s</i>	ε equation coefficient (a value of 0.0 turns off the spray/turbulence modulation term in both the k and ε equations).	0.0 - 1.5
<i>discrete_c_ps</i>	Drop turbulent dispersion constant.	0.0 - 0.16
<i>buoyancy_flag</i>	0 = No buoyancy effects, 1 = Buoyancy effects enabled for k transport equation. Uses inputs.in > property_control > gravity values. 2 = Buoyancy effects enabled for k and ε transport equations. Uses inputs.in > property_control > gravity values. CONVERGE ignores this parameter for non-RANS models.	
<i>Turbulence_statistics</i>		
<i>turb_stat_flag</i>	Activate the turbulence statistics calculation feature. 0 = Do not calculate turbulence statistics, 1 = Calculate mean turbulence values and fluctuations.	
<i>turb_stat_start_time\$</i>	The start time (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero) for the mean statistics and fluctuation calculations.	
<i>turb_stat_end_time\$</i>	The end time(in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero) for the mean statistics and fluctuation calculations.	
<i>turb_stat_tol\$</i>	Relative tolerance for turbulence statistics convergence = $abs(delta(mean)/mean)$. For monitoring purposes only.	

§ CONVERGE reads this parameter at each time-step when [inputs.in](#) > simulation_control > reread_input = 1.

Chapter 25: Input and Data Files

Physical Models Turbulence: turbulence.in

Table 25.90: Typical values of RANS k - ϵ parameters in *turbulence.in*.

Parameter name	Typical value for k - ϵ models	Typical value for RNG k - ϵ models	Typical value for v^2 - f model	Typical value for ζ - f model
<i>keps_cmu</i>	0.09	0.0845	0.09	0.09
<i>keps_rpr_tke</i>	1.0	1.39	1.0	1.0
<i>keps_ceps1</i>	1.44	1.42	1.4	1.4
<i>keps_ceps2</i>	1.92	1.68	1.9	1.9
<i>keps_v2f_c1</i>	-	-	1.4	0.4
<i>keps_v2f_cl</i>	-	-	0.23	0.36
<i>keps_v2f_ceta</i>	-	-	70	85

Table 25.91: Typical values of RANS k - ω parameters in *turbulence.in*.

Parameter name	Typical value for Standard k - ω 1998 model	Typical value for Standard k - ω 2006 model	Typical value for k - ω SST model
<i>komega_rpr_tke</i>	0.5	0.6	0.85
<i>komega_alpha</i>	13/25	13/25	5/9
<i>komega_beta</i>	0.072	0.0708	0.075

Source Modeling

This section describes input files that contain options for source modeling in CONVERGE.

Source/Sink: *source.in*

In CONVERGE, you can have [sources and sinks](#) for energy, momentum, turbulent kinetic energy, dissipation, species, passives, porous media, baffle media, batteries, and relaxation zones. To model sources, set [*inputs.in*](#) > *feature_control* > *source_flag* = 1 and include a *source.in* file in your case setup.

```
version: 3.1
---

- source:
  description: Source 1
  equation: ENERGY
  type: TOTAL_VALUE
  value: 0.02
  stream_id: 0
  temporal_control:
    type: CYCLIC
    cyclic_period: 720.0
    source_start_time: -15.0
    source_end_time: -14.5
    max_value: 50000.0
  shape:
    type: SPHERE
    x_center: [-0.003, 0.0, 0.0091]
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```
    radius:                      0.0005
    moving_control:
        moving_flag:             STATIONARY
        velocity:                [0.0,0.0,0.0]
        max_displace:            999999.0
        reset_source_flag:       DO_NOT_RESET
        mult_dt_source:          1.0
    - source:
        description:             Source 2
        equation:                 TKE
        type:                     PER_UNIT_VOLUME_TIME
        value:                   1.0
        stream_id:               0
        temporal_control:
            type:                  PERMANENT
            source_start_time:     -999999.0
            source_end_time:       -999999.0
        max_value:                100.0
        shape:
            type:                  BOX
            x_center:              [0.0, 0.0, 0.0]
            x_size:                [0.02, 0.02, 0.02]
        moving_control:
            moving_flag:           STATIONARY
            velocity:              [0.0,0.0,0.0]
            max_displace:          999999.0
            reset_source_flag:     DO_NOT_RESET
            mult_dt_source:         10.0
    - source:
        description:             Source 3
        equation:                 POROUS
        stream_id:               1
        temporal_control:
            type:                  PERMANENT
            source_start_time:     -999999.0
            source_end_time:       -999999.0
        shape:
            type:                  REGION
            region_id:              1
        porosity_data:
            alpha_coeff:            1.094
            alpha_cross_coeff:      1094.0
            beta_coeff:              1218.69
            beta_cross_coeff:       1218690.0
            is_directional:         1
            direction:              [0.242739, 0.000226191, -0.970092]
            eff_conductivity_flag:  0
            conductive_porosity:   0.5
            solid_conductivity:     1.0
            particulate_filter:
                active:               OFF
                filter_id:             1
            mult_dt_source:          1.0
    - source:
        description:             Source 4
        equation:                 ENERGY
        type:                     TOTAL_VALUE
        value:                   energy_data_table.dat
        stream_id:               [0,1]
        temporal_control:
            type:                  CYCLIC
            cyclic_period:          720.0
            source_start_time:      -15.0
            source_end_time:         -14.5
        max_value:                50000.0
        shape:
            type:                  SPHERE
            x_center:              [-0.003, 0.0, 0.0091]
            radius:                 0.0005
        moving_control:
            moving_flag:             STATIONARY
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```
velocity: [0.0,0.0,0.0]
max_displace: 999999.0
reset_source_flag: DO NOT_RESET
mult_dt_source: 1.0

- source:
  description: Source 5
  equation: TKE
  type: PER_UNIT_VOLUME_TIME
  value: turbulent_data_table.dat
  stream_id: ALL
  temporal_control:
    type: PERMANENT
    source_start_time: -999999.0
    source_end_time: -999999.0
  max_value: 100.0
  shape:
    type: BOX
    x_center: [0.0, 0.0, 0.0]
    x_size: [0.02, 0.02, 0.02]
  moving_control:
    moving_flag: STATIONARY
    velocity: [0.0,0.0,0.0]
    max_displace: 999999.0
    reset_source_flag: DO NOT_RESET
    mult_dt_source: 10.0

- source:
  description: Source 6
  equation: BAFFLE
  stream_id: 1
  temporal_control:
    type: PERMANENT
    source_start_time: -999999
    source_end_time: -999999
  shape:
    type: BOUNDARY
    boundary_id: 11
  porosity_data:
    alpha_coeff: 0.5
    beta_coeff: 0.5
  mult_dt_source: 1

- source:
  description: Source 7
  equation: U-EQ
  type: PER_UNIT_VOLUME_TIME
  value: 5
  stream_id: 0
  temporal_control:
    type: SEQUENTIAL
    source_start_time: 0
    source_end_time: 1
  max_value: 10
  shape:
    type: CYLINDER
    x1_center: [0, 0, 0]
    x2_center: [1, 0, 0]
    radius1: 0.01
    radius2: 1
  moving_control:
    moving_flag: PRESCRIBED_VELOCITY
    velocity: [1, 0, 0]
    max_displace: 0.1
    reset_source_flag: DO_NOT_RESET
    mult_dt_source: 1

- source:
  description: Source 8
  equation: EPS
  type: PER_UNIT_VOLUME_TIME
  value: 10
  stream_id: ALL
  temporal_control:
    type: PERMANENT
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```
source_start_time:          0
source_end_time:            1
max_value:                 10000
shape:
  type:                      BOX
  x_center:                  [0, 0, 0]
  x_size:                    [0.01, 0.01, 0.01]
moving_control:
  moving_flag:               MOVE_WITH_FLOW
  velocity:                  [1, 0, 0]
  max_displace:              1
  reset_source_flag:         DO_NOT_RESET
  mult_dt_source:            1
- source:
  description:               Source 9
  equation:                  C2H2
  type:                      PER_UNIT_TIME
  value:                     0
  stream_id:                 ALL
  species_control:
    temp:                     300
    velocity:                [10, 0, 0]
  temporal_control:
    type:                      PERMANENT
    source_start_time:         0
    source_end_time:           1
  max_value:                 0.5
  shape:
    type:                      LINE
    x1_center:                [0, 0, 0]
    x2_center:                [0, 0, 0.1]
    num_points:                10
  moving_control:
    moving_flag:               MOVE_WITH_FLOW
    velocity:                  [1, 0, 0]
    max_displace:              1
    reset_source_flag:         DO_NOT_RESET
    mult_dt_source:            1
- source:
  description:               Source 10
  equation:                  USER
  type:                      PER_UNIT_TIME
  value:                     0.1
  stream_id:                 ALL
  temporal_control:
    type:                      PERMANENT
    source_start_time:         0
    source_end_time:           1
  max_value:                 1
  shape:
    type:                      CIRCLE
    x_center:                  [0, 0, 0]
    num_points:                10
    radius_circle:             0.05
    normal_vector:              [0, 0, 1]
  moving_control:
    moving_flag:               MOVE_WITH_FLOW
    velocity:                  [1, 0, 0]
    max_displace:              1
    reset_source_flag:         RESET_LINE_CIRCLE
    mult_dt_source:            1
- source:
  description:               Source 11
  equation:                  ENERGY
  type:                      PRESSURE_TRACE
  value:                     ptrace.in
  stream_id:                 0
  temporal_control:
    type:                      SEQUENTIAL
    source_start_time:         0
    source_end_time:           1
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```
max_value: 5000
shape:
  type: BOUNDARY
  boundary_id: 1
mult_dt_source: 1
- source:
  description: Source 12
  equation: ENERGY
  type: HEAT_RELEASE_DATA
  value: HRR.in
  stream_id: 1
  temporal_control:
    type: SEQUENTIAL
    source_start_time: 0
    source_end_time: 1
  max_value: 5000
  shape:
    type: BOUNDARY
    boundary_id: [2, 4]
  mult_dt_source: 1
- source:
  description: Source 13
  equation: POROUS
  stream_id: 1
  temporal_control:
    type: PERMANENT
    source_start_time: -999999.0
    source_end_time: -999999.0
  shape:
    type: REGION
    region_id: 1
  porosity_data:
    coefficient_type: INERTIAL_VISCOUS_COEFFICIENTS
    inertial_viscous_coefficients:
      inertial_coeff: 1.094
      inertial_cross_coeff: 1094.0
      viscous_coeff: 1218.69
      viscous_cross_coeff: 1218690.0
    is_directional: 1
    direction: [0.242739, 0.000226191, -0.970092]
    eff_conductivity_flag: 0
    conductive_porosity: 0.5
    solid_conductivity: 1.0
    solid_transient_flag: LTNE
    solid_density: 5032.0
    solid_specific_heat: 502.0
    solid_init_temp: 300.0
    solid_htc_eff:
      type: HWANG
      value: 1.0
      lengthscale: 0.02
    mult_dt_source: 1.0
- source:
  description: rotor loss
  equation: ENERGY
  type: PER_UNIT_TIME
  value: 18.0652
  stream_id: ALL
  distribution: rotor_loss.in
  temporal_control:
    type: PERMANENT
    source_start_time: -999999
    source_end_time: -999999
  max_value: 50000
  shape:
    type: REGION
    region_id: 7
  moving_control:
    distribution_motion: MOVE_WITH_BOUNDARY
    distribution_boundary: 2
    distribution_boundary_side: FORWARD
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```
mult_dt_source: 1
- source:
  description: rotor loss
  equation: ENERGY
  type: PER_UNIT_TIME
  value: 18.0652
  stream_id: ALL
  temporal_control:
    type: PERMANENT
    source_start_time: -999999
    source_end_time: -999999
    max_value: 50000
  shape:
    type: BOX
    x_center: [0.0328125, 0, 0]
    x_size: [0.02, 0.005, 0.01]
  moving_control:
    moving_flag: MOVE_WITH_BOUNDARY
    boundary: 101
    max_displace: 0
    reset_source_flag: DO_NOT_RESET
  mult_dt_source: 1
- source:
  description: battery heat
  equation: BATTERY
  type: PER_UNIT_TIME
  value: 0.05
  stream_id: 1
  temporal_control:
    type: PERMANENT
    source_start_time: -999999
    source_end_time: -999999
    max_value: 94999
  shape:
    type: BOX
    x_center: [0.01, 0, 0]
    x_size: [0.005, 0.005, 0.005]
  moving_control:
    moving_flag: STATIONARY
    velocity: [0, 0, 0]
    max_displace: 999999
    reset_source_flag: DO_NOT_RESET
  mult_dt_source: 1
  battery_data:
    n_rc_pairs: 1
    rc_pair_file_name: battery_properties1.dat
    initial_soc: 1
    max_soc_permitted: 1.5
    battery_capacity: 2.1
    current_history: current_history_CD.in
    soc_vocv: vocvl.dat
- source:
  description: battery thermal runaway
  equation: ENERGY
  type: REN_MODEL
  value: 1e-14
  stream_id: 0
  temporal_control:
    type: PERMANENT
    source_start_time: -999999
    source_end_time: -999999
    max_value: 5000
  shape:
    type: REGION
    region_id: 0
  mult_dt_source: 1
- source:
  description: RelZoneBottom
  equation: RELAXATION_ZONE
  temporal_control:
    type: PERMANENT
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```

source_start_time:          0.0
source_end_time:           999999
shape:
  type:                      BOX
  x_center:                  [12.151, 1.0, 0]
  x_size:                   [24.302, 2, 1]
relaxation_zone_data:
  blending_direction:        [0, 1, 0]
  blending_function:
    function_type:          EXPONENTIAL
    exp_coefficient:       3.5
zone_field_data:
  type:                      SPECIFIED
  velocity:                 [0., 0., 0.]
  water_level:              10.0
  interface_normal:         FROM_GRAVITY
- source:
  description:               RelZoneInFlow
  equation:                  RELAXATION_ZONE
  temporal_control:
    type:                     SEQUENTIAL
    source_start_time:        0.0
    source_end_time:         999999
  shape:
    type:                     BOUNDARY
    boundary_id:              [1]
    distance_from_boundary:  4.703587
  relaxation_zone_data:
    blending_function:
      function_type:          EXPONENTIAL
      exp_coefficient:       3.5
  zone_field_data:
    type:                     USER_RZ_OUTFLOW
    velocity:                 [0., 0., 0.]
    water_level:              10.0
    interface_normal:         FROM_GRAVITY

```

Figure 25.117: An example *source.in* file.

Table 25.92: Format of *source.in*.

Parameter	Description
- <i>source</i>	This settings block specifies the configuration for a source. Repeat this block through <i>mult_dt_source</i> for each source.
<i>description</i> *	Optional description of the source.
<i>equation</i>	Equation for this source. <i>ENERGY</i> = Thermal source, <i>U-EQ</i> = Momentum source in the x direction, <i>V-EQ</i> = Momentum source in the y direction, <i>W-EQ</i> = Momentum source in the z direction, <i>TKE</i> = Turbulent kinetic energy source, <i>EPS</i> = Turbulent dissipation source, <i><species name></i> = Species source, <i><passive name></i> = Passive source, <i>POROUS</i> = Porous media source, <i>BAFFLE</i> = Baffle media source, <i>BATTERY</i> = Battery heat source (requires <i>type</i> = <i>PER_UNIT_TIME</i>), <i>RELAXATION_ZONE</i> = Relaxation zone source (requires <i>inputs.in</i> > <i>feature_control</i> > <i>vof_flag</i> = 1).

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<i>type</i>	<p>Source type. <i>PER_UNIT_VOLUME_TIME</i> = Per unit volume per time, <i>PER_UNIT_TIME</i> = Per unit time, <i>TOTAL_VALUE</i> = Total source, <i>PRESSURE_TRACE</i> = Pressure trace (requires <i>equation</i> = <i>ENERGY</i> above), <i>HEAT_RELEASE_DATA</i> = Heat release data (requires <i>equation</i> = <i>ENERGY</i> above), <i>HATCHARD_KIM_MODEL</i> = Hatchard-Kim thermal runaway model (requires <i>equation</i> = <i>ENERGY</i> above and <i>thermal runaway model.in</i>), <i>REN_MODEL</i> = Ren thermal runaway model (requires <i>equation</i> = <i>ENERGY</i> above and <i>thermal runaway model.in</i>), <i>user_<UDF name></i> = Source from a user-defined function (UDF) based on the <i>CONVERGE_SOURCE</i> macro (refer to the CONVERGE 3.1 UDF Manual for more information).</p> <ul style="list-style-type: none">• Not used when <i>equation</i> is <i>POROUS</i>, <i>BAFFLE</i>, or <i>RELAXATION_ZONE</i>.• For sources of <i>PERMANENT</i> temporal type, <i>TOTAL_VALUE</i> is not available.• When <i>type</i> = <i>PRESSURE_TRACE</i>, you must include a <u>pressure trace file</u>. The pressure trace represents the pressure change due to combustion. You can obtain the pressure trace data from a single cycle of a CONVERGE simulation that includes combustion, experimental data, or a 1D simulation. The pressure trace file must contain the keyword <i>PRESSURE_CURVE</i>, and the pressure data must be in <i>Pascals</i>.• When <i>type</i> = <i>HEAT_RELEASE_DATA</i>, you must include a file that contains <u>heat release rate data</u>. You can obtain these data from a single cycle of a CFD simulation that includes combustion, experimental data, or a 1D simulation. The heat release data file must contain the keyword <i>HEAT_RELEASE</i>, and the heat release rate data must be in <i>J/s</i> (if <i>inputs.in > simulation_control > crank_flag</i> = 0) or <i>J/deg</i> (if <i>crank_flag</i> is non-zero). This option allows you to avoid the computationally expensive chemistry calculations for simulations in which the details of the combustion are already known.

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<i>value</i>	For transient simulations (<i>i.e.</i> , when inputs.in > <i>solver_control</i> > <i>steady_solver</i> = 0), source value based on <i>type</i> . For steady-state simulations (<i>i.e.</i> , when inputs.in > <i>steady_solver</i> = 1), if <i>type</i> = <i>TOTAL_VALUE</i> , source value per unit time (<i>i.e.</i> , when <i>type</i> = <i>TOTAL_VALUE</i> , CONVERGE will read <i>type</i> as <i>PER_UNIT_TIME</i> for a steady-state simulation). Not used for when <i>equation</i> = <i>POROUS</i> , <i>BAFFLE</i> , <i>BATTERY</i> , or <i>RELAXATION_ZONE</i> , or when <i>type</i> = <i>HATCHARD_KIM_MODEL</i> or <i>REN_MODEL</i> .
	Total source value per unit volume per unit time: $J/(m^3 \cdot s)$ (<i>ENERGY</i>), $kg/(m^2 \cdot s^2)$ (<i>U-EQ</i> , <i>V-EQ</i> , <i>W-EQ</i>), $m^{-1} \cdot s^{-3}$ (<i>TKE</i>), $m^{-1} \cdot s^{-4}$ (<i>EPS</i>), $kg/(m^3 \cdot s)$ (<i>species</i>), $m^{-3} \cdot s^{-1}$ (<i>user</i>), $m^{-3} \cdot s^{-1}$ (<i>passive</i>). Total source value per unit time units: J/s (<i>ENERGY</i>), $kg \cdot m/s^2$ (<i>U-EQ</i> , <i>V-EQ</i> , <i>W-EQ</i>), m^2/s^3 (<i>TKE</i>), m^2/s^4 (<i>EPS</i>), kg/s (<i>species</i>), $[source\ value\ units]/s$ (<i>user</i>), $[source\ value\ units]/s$ (<i>passive</i>). Total source value units: J (<i>ENERGY</i>), $kg \cdot m/s$ (<i>U-EQ</i> , <i>V-EQ</i> , <i>W-EQ</i>), m^2/s^2 (<i>TKE</i>), m^2/s^3 (<i>EPS</i>), kg (<i>species</i>), $[source\ value\ units]$ (<i>user</i>), $[source\ value\ units]$ (<i>passive</i>). Pressure: Pa (<i>ENERGY</i>). Heat release: J/s when inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or J/CAD when <i>crank_flag</i> is non-zero (<i>ENERGY</i>).
	For <i>ENERGY</i> sources, when <i>type</i> = <i>PRESSURE_TRACE</i> the <i>value</i> parameter gives the name of the file that contains pressure trace data. For <i>ENERGY</i> sources, when <i>type</i> = <i>HEAT_RELEASE_DATA</i> the <i>value</i> parameter gives the name of the file that contains heat release data.
<i>stream_id</i>	ID(s) of the stream(s) to which the source applies. Specify <i>ALL</i> if the source is valid for all streams.
<i>distribution*</i>	Specify the name of a file that contains the non-normalized spatial distribution of the source intensity. Available only if <i>type</i> = <i>PER_UNIT_TIME</i> or <i>TOTAL_VALUE</i> . Not used for <i>equation</i> = <i>BATTERY</i> .
<i>species_control</i>	Specify this sub-block only for <i>equation</i> = < <i>species name</i> >.
<i>temp</i>	Temperature of the species source (in <i>K</i>).
<i>velocity</i>	Velocity of the species source (in <i>m/s</i>). Specify the velocity vector as $[u, v, w]$.
<i>temporal_control</i>	
<i>type</i>	<i>SEQUENTIAL</i> , <i>PERMANENT</i> , or <i>CYCLIC</i> .
<i>cyclic_period</i>	The <i>CYCLIC</i> period for this source (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<code>source_start_time</code>	Start time for source/sink (in seconds if inputs.in > <code>simulation_control</code> > <code>crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero).
<code>source_end_time</code>	End time for source/sink (in seconds if inputs.in > <code>simulation_control</code> > <code>crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero).
<code>max_value</code>	Maximum value that solution variable associated with a source can attain. Not used when <code>source.in</code> > <code>source > equation</code> is <code>POROUS</code> or <code>BATTERY</code> . Units: K/s (<code>source > type = HATCHARD_KIM_MODEL</code> or <code>REN_MODEL</code>), temperature K (all other <code>ENERGY</code> sources), absolute value of velocity m/s ($U-EQ$, $V-EQ$, $W-EQ$), m^2/s^2 (<code>TKE</code>), m^2/s^3 (<code>EPS</code>), mass fraction (species), maximum passive value (passive).
	For sources with a negative <code>value</code> , CONVERGE uses this parameter as the minimum value this solution variable can attain.
<code>shape</code>	
<code>type</code>	Shape of the source: <code>BOX</code> , <code>REGION</code> (i.e., the size and shape of a source is an entire region), <code>LINE</code> , <code>CIRCLE</code> , <code>CYLINDER</code> , <code>SPHERE</code> , <code>BOUNDARY</code> , or <code>PROXIMITY</code> (requires proximity_boundaries.in).
<code>x_center</code>	Center of the source (x, y, and z coordinates). Used only for <code>shape > type = BOX</code> , <code>CIRCLE</code> , or <code>SPHERE</code> .
<code>x1_center</code>	Starting point of the source (x, y, and z coordinates). Used only for <code>shape > type = LINE</code> or <code>CYLINDER</code> .
<code>x2_center</code>	Ending point of the source (x, y, and z coordinates). Used only for <code>shape > type = LINE</code> or <code>CYLINDER</code> .
<code>num_points</code>	Number of evenly spaced points on the source. Used only for <code>shape > type = LINE</code> .
<code>radius</code>	Radius of the source (m). Used only for <code>shape > type = SPHERE</code> .
<code>radius_circle</code>	Radius of the source (m). Used only for <code>shape > type = CIRCLE</code> .
<code>radius1</code>	Radius of the first circle (m). Used only for <code>shape > type = CYLINDER</code> .
<code>radius2</code>	Radius of the second circle (m). Used only for <code>shape > type = CYLINDER</code> .
<code>x_size</code>	Half of x, y, and z dimensions of the source (m). Used only for <code>shape > type = BOX</code> .
<code>normal_vector</code>	Normal vector for the circle. Used only for <code>shape > type = CIRCLE</code> .
<code>region_id</code>	Region ID number (not name) for the source. Used only for <code>shape > type = REGION</code> or <code>PROXIMITY</code> .

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<i>boundary_id</i>	Boundary ID for the source. Used only for <i>shape > type</i> = BOUNDARY. For INTERFACE boundaries, use the positive-signed boundary ID to apply the source on the forward side of the boundary and/or a negative-signed boundary ID to apply the source to the reverse side.
<i>distance_from_boundary</i>	Distance by which the source extends from the boundary. Used only when <i>equation</i> = RELAXATION_ZONE and <i>shape > type</i> = BOUNDARY.
<i>moving_control</i>	
<i>moving_flag</i>	STATIONARY = Source is not moving, PRESCRIBED_VELOCITY = Source is moving at a specified velocity, MOVE_WITH_FLOW = Source is moving with the flow, MOVE_WITH_BOUNDARY = Source is moving with the specified boundary (requires <i>reset_source_flag</i> = 0), <i>user_<UDF name></i> = Source motion is controlled by a user-defined function (UDF) based on the CONVERGE_SHAPE_MOVE macro (refer to the CONVERGE 3.1 UDF Manual for more information).
<i>boundary*</i>	ID of the boundary with which the source moves. Must be a WALL or WALL-type INTERFACE boundary. Applies only when <i>moving_flag</i> = MOVE_WITH_BOUNDARY.
<i>boundary_side*</i>	FORWARD or REVERSE. Applies only if the boundary specified above is an INTERFACE boundary.
<i>velocity</i>	Prescribed velocity. Used only when <i>moving_flag</i> = PRESCRIBED_VELOCITY.
<i>max_displace</i>	Maximum distance that the source can move. Used only when <i>moving_flag</i> = PRESCRIBED_VELOCITY.
<i>reset_source_flag</i>	0 = Do not move source back to original location, 1 = Return source to original location when <i>max_displace</i> is reached, 2 = Return source when any point in LINE or CIRCLE exceeds <i>max_displace</i> (only for LINE and CIRCLE sources).
<i>distribution_motion*</i>	STATIONARY = Source distribution is not moving, MOVE_WITH_BOUNDARY = Source distribution is moving with the specified boundary (see below), and MOVE_WITH_SOURCE = Source distribution is moving with the source.
<i>distribution_boundary*</i>	ID of the boundary with which the source distribution moves. Must be a WALL or WALL-type INTERFACE boundary. Applies only when <i>distribution_motion</i> = MOVE_WITH_BOUNDARY.
<i>distribution_boundary_side*</i>	FORWARD or REVERSE. Applies only if the boundary specified above is an INTERFACE boundary.

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<i>porosity_data</i>	Specify this sub-block only for <i>source.in > source > equation = POROUS</i> .
<i>coefficient_type*</i>	0 = Specify coefficients in the <i>alpha_beta_coefficients</i> sub-block below, 1 = Specify coefficients in the <i>inertial_viscous_coefficients</i> sub-block below. If you do not include this parameter, CONVERGE will read coefficients from the <i>alpha_beta_coefficients</i> sub-block, not from the <i>inertial_viscous_coefficients</i> sub-block.
<i>alpha_beta_coefficients</i>	Specify this sub-block only for <i>coefficient_type</i> = 0. (CONVERGE ignores these values if <i>coefficient_type</i> = 1.)
<i>alpha_coeff</i>	Inertia resistivity (kg/m^4) in the flow direction.
<i>alpha_cross_coeff</i>	Inertia resistivity (kg/m^4) in the cross-flow direction.
<i>beta_coeff</i>	Viscous resistivity ($\text{kg}/\text{m}^3\text{-s}$) in the flow direction.
<i>beta_cross_coeff</i>	Viscous resistivity ($\text{kg}/\text{m}^3\text{-s}$) in the cross-flow direction.
<i>inertial_viscous_coefficients</i>	Specify this sub-block only for <i>coefficient_type</i> = 1. (CONVERGE ignores these values if <i>coefficient_type</i> = 0 or if <i>coefficient_type</i> is not included.)
<i>inertial_coeff</i>	Inertial resistivity factor (m^{-1}) in the flow direction.
<i>inertial_cross_coeff</i>	Inertial resistivity factor in cross-flow direction (m^{-1}).
<i>viscous_coeff</i>	Viscous resistivity factor (m^{-2}) in the flow direction.
<i>viscous_cross_coeff</i>	Viscous resistivity factor in cross-flow direction (m^{-2}).
<i>is_directional</i>	0 = Isotropic (only <i>*_coeff</i> values are used for calculating velocity), 1 = Orthotropic (<i>*_coeff</i> and <i>*_cross_coeff</i> values are used for calculating velocity).
<i>direction</i>	Direction vector in <i>i, j, k</i> .
<i>eff_conductivity_flag</i>	0 = CONVERGE assumes the thermal conductivity of the porous region is equal to the thermal conductivity of the fluid in the porous region, 1 = CONVERGE calculates the effective thermal conductivity of the porous region .
<i>conductive_porosity</i>	The porosity of the porous medium (used only when <i>eff_conductivity_flag</i> = 1 or when <i>solid_transient_flag</i> = LTE or LTNE).
<i>solid_conductivity</i>	The thermal conductivity of the solid phase in the porous medium (used only when <i>eff_conductivity_flag</i> = 1 or when <i>solid_transient_flag</i> = LTNE).
<i>solid_transient_flag*</i>	OFF = Do not consider fluid-solid coupling in the energy equation,

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
	<i>LTE</i> = Use the local thermal equilibrium (LTE) model to add fluid-solid coupling in the transient term of the fluid energy equation, <i>LTNE</i> = Use the local thermal non-equilibrium (LTNE) model to couple the fluid and solid energy equations and solve for fluid and solid temperatures separately.
<i>solid_density</i>	The density (kg/m^3) of the solid in the porous medium (used only when <i>solid_transient_flag</i> = <i>LTE</i> or <i>LTNE</i>).
<i>solid_specific_heat</i>	The specific heat ($\text{J}/\text{K}\cdot\text{kg}$) of the solid in the porous medium (used only when <i>solid_transient_flag</i> = <i>LTE</i> or <i>LTNE</i>).
<i>solid_init_temp</i>	The initial temperature (K) of the solid in the porous medium (used only when <i>solid_transient_flag</i> = <i>LTNE</i>).
<i>solid_htc_eff</i>	Sub-block for the effective heat transfer coefficient (used only when <i>solid_transient_flag</i> = <i>LTNE</i>).
<i>type</i>	Model for the effective heat transfer coefficient. <i>CONSTANT</i> = Specify a constant value, <i>HWANG</i> = Use the model of Hwang et al. (1995) , <i>WAKAO</i> = Use the model of Wakao et al. (1979) , <i>KUWAHARA</i> = Use the model of Kuwahara et al. (2001) , <i>DIXON_CRESSWELL</i> = Use the model of Dixon and Cresswell (1979) .
<i>value</i>	Value ($\text{W}/\text{K}\cdot\text{m}^3$) of the effective heat transfer coefficient (used only when <i>type</i> = <i>CONSTANT</i>).
<i>lengthscale</i>	Characteristic length scale (m) for the diameter of the solid particles in the porous medium (used only when <i>type</i> ≠ <i>CONSTANT</i>).
<i>particulate_filter</i>	Sub-block for the particulate filter model.
<i>active</i>	<i>OFF</i> = No particulate filter model, <i>ON</i> = Particulate filter model is active (requires particulate_filter.in).
<i>filter_id</i>	Filter ID for this source (from particulate_filter.in).
<i>mult_dt_source</i>	Time-step limiter for source magnitude.
<i>battery_data</i>	Specify this sub-block only when <i>source > equation</i> = <i>BATTERY</i> .
<i>n_rc_pairs</i>	Number of resistor-capacitor (R-C) pairs. Include 0, 1, 2, or 3 R-C pairs.
<i>rc_pair_filename</i>	File name (e.g., battery_properties.dat) of the file that contains data for series resistance and the resistors and capacitors. The file must contain data for the number of R-C pair specified above. Include this file in your case setup.
<i>initial_soc</i>	Initial state of charge (SOC) of the battery. Allows double values $0 \leq \text{initial_soc} \leq 1$, where 0 indicates the battery is fully discharged and 1 indicates the battery is fully charged.

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<i>max_soc_permitted</i>	Maximum SOC permitted. A value greater than 1 indicates an overcharged battery.
<i>battery_capacity</i>	Battery capacity (<i>ampere-hour</i>).
<i>current_history</i>	Electrical current value (<i>amperes</i>) or a file name for the electrical current history (e.g., <i>current_history_CD.in</i>). The electrical current history file is a temporal file with time (in <i>seconds</i> if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>seconds</i> or <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) and current data (<i>amperes</i>). Positive current values indicate a discharging battery and negative current values indicate a charging battery. If using a file, include this file in your case setup.
<i>soc_vocv</i>	V _{OCV} value (<i>volts</i>) or the name (e.g., vocv.dat) of the file that contains SOC versus V _{OCV} data with discharge and charge profiles.
<i>relaxation_zone_data</i>	Specify this sub-block only for <i>source.in</i> > <i>source</i> > <i>equation</i> = RELAXATION_ZONE.
<i>blending_direction</i>	Direction in which the relaxation zone field is blended with the CFD solution.
<i>blending_function</i>	
<i>function_type</i>	EXPONENTIAL = Exponential blending function.
<i>exp_coefficient</i>	Blending function exponent.
<i>zone_field_data</i>	
<i>type</i>	Type of field used for the theoretical solution in the relaxation zone. SPECIFIED = Use specified velocity and water level (see below), FROM_INITIAL_FIELD = Use the initial velocity and species/void fraction from initialize.in , <i>user_<UDF name></i> = Use the velocity and species/void fraction calculated by a user-defined function (UDF) based on the CONVERGE_SOURCE macro (refer to the CONVERGE 3.1 UDF Manual for more information).
<i>velocity</i>	Velocity in the relaxation zone. Used only for <i>type</i> = SPECIFIED.
<i>water_level</i>	Location of fluid-fluid interface in the direction specified in <i>interface_normal</i> . Used only for <i>type</i> = SPECIFIED.
<i>interface_normal</i>	Direction normal to the fluid-fluid interface. Specify a direction vector or enter FROM_GRAVITY to use the direction opposite to inputs.in > <i>property_control</i> > <i>gravity</i> . Used only for <i>type</i> = SPECIFIED.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Thermal Runaway Model: thermal_runaway_model.in

In CONVERGE, you can model thermal runaway for a source. To model a thermal runaway source, set [inputs.in](#) > *feature_control* > *source_flag* = 1, specify *HATCHARD_KIM_MODEL* or *REN_MODEL* for [source.in](#) > *source* > *type* and include a *thermal_runaway_model.in* file in your case setup.

```
version: 3.1
---

hatchard_kim_model:
    sei_init:          0.15
    neg_init:          0.75
    alpha_init:         0.04
    electrolyte_init:   1.0
    t_sei_init:        0.033

    sei_order:         1.0
    neg_order:         1.0
    alpha_order:        1.0
    one_minus_alpha_order: 1.0
    electrolyte_order:  1.0

    sei_decomp_factor: 1.667e15
    negative_solvent_factor: 2.5e13
    positive_solvent_factor: 6.667e13
    electrolyte_decomp_factor: 5.14e25

    sei_decomp_e:      1.3508e5
    negative_solvent_e: 1.3508e5
    positive_solvent_e: 1.396e5
    electrolyte_decomp_e: 2.74e5

    sei_decomp_specific_heat: 257
    negative_solvent_specific_heat: 1714
    positive_solvent_specific_heat: 314
    electrolyte_decomp_specific_heat: 155

    specific_carbon_content: 6.104e5
    specific_positive_content: 1.221e6
    specific_electrolyte_content: 4.069e5

ren_model:
    sei_init:          0.15
    anode_electrolyte_init: 0.75
    anode_bind_init:       0.04
    cathode_anode_init:    1.0
    cathode_bind_init:     0.033
    cathode_init:          0.01

    sei_order:         1.0
    anode_electrolyte_order: 1.0
    anode_bind_order:    1.0
    cathode_anode_order: 1.0
    cathode_bind_order:   1.0
    cathode_order:        1.0

    sei_decomp_factor: 1.667e15
    anode_electrolyte_factor: 2.5e13
    anode_bind_factor:    6.667e13
    cathode_anode_factor: 5.14e25
    cathode_bind_factor:  5.14e25
    cathode_factor:       5.14e25

    sei_decomp_e:      1.3508e5
    anode_electrolyte_e: 1.3508e5
    anode_bind_e:       1.396e5
    cathode_anode_e:    2.74e5
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```
cathode_bind_e:          2.74e5
cathode_e:               2.74e5

sei_decomp_specific_heat: 257
anode_electrolyte_specific_heat: 1714
anode_bind_specific_heat: 314
cathode_anode_specific_heat: 155
cathode_bind_specific_heat: 155
cathode_specific_heat: 155

mass_reactant_content: 6.104e5
bind_gamma_factor: 1.221e6
lumped_temperature_flag: 0
```

Figure 25.118: An example *thermal_runaway_model.in* file.

Table 25.93: Format of the *hatchard_kim_model* settings block.

Parameter	Description
<i>hatchard_kim_model</i>	
<i>sei_init</i>	Initial amount of lithium-containing meta-stable species in the solid electrolyte interface (SEI) (dimensionless).
<i>neg_init</i>	Initial amount of lithium that is intercalated within the carbon of the anode (dimensionless).
<i>alpha_init</i>	Initial value of α , which is the fraction degree of conversion for the cathode (dimensionless).
<i>electrolyte_init</i>	Initial concentration of electrolyte (dimensionless).
<i>t_sei_init</i>	Initial value of the measure of the SEI layer thickness that reflects the amount of lithium in the SEI.
<i>sei_order</i>	Reaction order for the amount of SEI (c_{SEI}).
<i>neg_order</i>	Reaction order for the amount of lithium that is intercalated (c_{An}).
<i>alpha_order</i>	Reaction order for α .
<i>one_minus_alpha_order</i>	Reaction order for $(1-\alpha)$.
<i>electrolyte_order</i>	Reaction order for the concentration of electrolyte (c_E).
<i>sei_decomp_factor</i>	SEI decomposition reaction pre-exponential factor (s^{-1}).
<i>negative_solvent_factor</i>	Anode and electrolyte reaction pre-exponential factor (s^{-1}).
<i>positive_solvent_factor</i>	Cathode and electrolyte reaction pre-exponential factor (s^{-1}).
<i>electrolyte_decomp_factor</i>	Electrolyte decomposition reaction pre-exponential factor (s^{-1}).
<i>sei_decomp_e</i>	SEI decomposition reaction activation energy (J/mol).
<i>negative_solvent_e</i>	Anode and electrolyte reaction activation energy (J/mol).
<i>positive_solvent_e</i>	Cathode and electrolyte reaction activation energy (J/mol).
<i>electrolyte_decomp_e</i>	Electrolyte decomposition reaction activation energy (J/mol).

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<i>sei_decomp_specific_heat</i>	SEI decomposition reaction specific heat release (J/kg).
<i>negative_solvent_specific_heat</i>	Anode and electrolyte reaction specific heat release (J/kg).
<i>positive_solvent_specific_heat</i>	Cathode and electrolyte reaction specific heat release (J/kg).
<i>electrolyte_decomp_specific_heat</i>	Electrolyte decomposition reaction specific heat release (J/kg).
<i>specific_carbon_content</i>	Specific carbon content in the jelly roll (kg/m ³).
<i>specific_positive_content</i>	Specific positive-active content in the jelly roll (kg/m ³).
<i>specific_electrolyte_content</i>	Specific electrolyte content in the jelly roll (kg/m ³).

Table 25.94: Format of the *ren_model* settings block.

Parameter	Description	Typical value
<i>ren_model</i>		
<i>sei_init</i>	Initial normalized concentration of solid electrolyte interface (SEI) film.	1.0
<i>anode_electrolyte_init</i>	Initial normalized concentration of anode active material.	1.0
<i>anode_bind_init</i>	Initial normalized concentration of binder material.	1.0
<i>cathode_anode_init</i>	Initial normalized concentration of anode active material.	1.0
<i>cathode_bind_init</i>	Initial normalized concentration of binder material.	1.0
<i>cathode_init</i>	Initial normalized concentration of cathode active material.	1.0
<i>sei_order</i>	Reaction order for normalized concentration of SEI film (c_{SEI}) in the SEI decomposition reaction.	
<i>anode_electrolyte_order</i>	Reaction order for the normalized concentration of anode active material (c_{An-E}) in the anode and electrolyte reaction.	
<i>anode_bind_order</i>	Reaction order for the normalized concentration of binder material (c_{An-B}) in the anode and binder reaction.	
<i>cathode_anode_order</i>	Reaction order for the normalized concentration of anode active material (c_{Cat-An}) in the cathode and anode reaction.	
<i>cathode_bind_order</i>	Reaction order for the normalized concentration of the binder material (c_{Cat-B}) in the cathode and binder material reaction.	

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description	Typical value
<i>cathode_order</i>	Reaction order for the normalized concentration of cathode (c_{Cat}) in the cathode decomposition reaction.	
<i>sei_decomp_factor</i>	SEI decomposition reaction pre-exponential factor (s^{-1}).	
<i>anode_electrolyte_factor</i>	Anode and electrolyte reaction pre-exponential factor (s^{-1}).	
<i>anode_bind_factor</i>	Anode and binder reaction pre-exponential factor (s^{-1}).	
<i>cathode_anode_factor</i>	Cathode and anode reaction pre-exponential factor (s^{-1}).	
<i>cathode_bind_factor</i>	Cathode and binder reaction pre-exponential factor (s^{-1}).	
<i>cathode_factor</i>	Cathode decomposition reaction pre-exponential factor (s^{-1}).	
<i>sei_decomp_e</i>	SEI decomposition reaction activation energy (J/mol).	
<i>anode_electrolyte_e</i>	Anode and electrolyte reaction activation energy (J/mol).	
<i>anode_bind_e</i>	Anode and binder reaction activation energy (J/mol).	
<i>cathode_anode_e</i>	Cathode and anode reaction activation energy (J/mol).	
<i>cathode_bind_e</i>	Cathode and binder reaction activation energy (J/mol).	
<i>cathode_e</i>	Cathode active material decomposition reaction activation energy (J/mol).	
<i>sei_decomp_specific_heat</i>	SEI decomposition reaction specific heat release (J/kg).	
<i>anode_electrolyte_specific_heat</i>	Anode and electrolyte reaction specific heat release (J/kg).	
<i>anode_bind_specific_heat</i>	Anode and binder reaction specific heat release (J/kg).	
<i>cathode_anode_specific_heat</i>	Cathode and anode reaction specific heat release (J/kg).	
<i>cathode_bind_specific_heat</i>	Cathode and binder reaction specific heat release (J/kg).	

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description	Typical value
<i>cathode_specific_heat</i>	Cathode decomposition reaction specific heat release (J/kg).	
<i>mass_reactant_content</i>	Mass density of the reactants (kg/m^3).	
<i>bind_gamma_factor</i>	Mass ratio of binder in the anode and the cathode.	
<i>lumped_temperature_flag</i>	0 = Use cell temperature, 1 = Use domain-averaged temperature.	

Particulate Filter Model: particulate_filter.in

To include a particulate filter model for a porous media source, set [source.in](#) > *source* > *porosity_data* > *particulate_filter* > *active* = *ON* and include a *particulate_filter.in* file in your case setup.

```
version: 3.1
---

particulate_filters:
  flow_control:
    - stream:
        stream_ids: ALL
        coupling_period: 0.001
        coupling_dt_max: 1.e-7
        mom_source_update_interval: 0.02
        inflow_soot:
          particle_density: 2000.0
          particle_size_distribution:
            - [51.7947, 0.020793003]
            - [71.2548, 0.059345514]
            - [89.6151, 0.113703955]
            - [110.4816, 0.150607912]
            - [130.2323, 0.15459729]
            - [148.2503, 0.175376265]
            - [170.4519, 0.115505318]
            - [189.2589, 0.096304576]
            - [210.141, 0.043844059]
            - [229.8645, 0.034147263]
            - [247.7076, 0.019698718]
    filter_list:
      - filter:
          id: 0
          properties:
            washcoat_flag: 1
            initial:
              porosity: 0.33
              unit_collector_diameter: 3.64e-5
              permeability: 1.4e-13
              pore_diameter: 1.2e-5
            bare:
              porosity: 0.5
              unit_collector_diameter: 3.3e-5
            soot_cake:
              porosity: 0.9
              unit_collector_diameter: 2.1e-7
              permeability: 8.e-15
              wall_packing_density: 10.0
              cake_packing_density: 90.0
              percolation_control_constant: 0.965
            ash_layer:
              active: 1
```

Chapter 25: Input and Data Files

Physical Models Source Modeling

```
porosity:          0.8
unit_collector_diameter: 2.3e-7
permeability:      2.5e-15
thickness:         1.e-6
filtration_efficiency:
velocity:
option:           SUPERFICIAL
scaling_factors:
sf_overall:       1.0
sf_diffusion:     1.6
sf_inertia:       15.0
```

Figure 25.119: An example *particulate_filter.in* file.

Table 25.95: Format of *particulate_filter.in*.

Parameter	Description
<i>particulate_filters</i>	
<i>flow_control</i>	
<i>stream</i>	Repeat this sub-block for each set of streams with unique flow control settings.
<i>stream_ids</i>	List of stream IDs or <i>ALL</i> for all streams.
<i>coupling_period</i>	Period of time for which CONVERGE runs at a small fixed time-step after updating the porous resistivity (<i>seconds</i>).
<i>coupling_dt_max</i>	Maximum time-step during the coupling period (<i>seconds</i>). If this value is larger than <i>inputs.in</i> > <i>temporal_control</i> > <i>dt_max</i> , CONVERGE will use <i>dt_max</i> instead.
<i>mom_source_update_interval</i>	Interval at which CONVERGE updates the porous resistivity in the momentum source term (<i>seconds</i>).
<i>inflow_soot</i>	
<i>particle_density</i>	Density of fine soot particulates at inlet (kg/m^3).
<i>particle_size_distribution</i>	Size distribution of soot particulates, specified as [<i>diameter (nm)</i> , <i>mass fraction</i>].
<i>filter_list</i>	
<i>filter</i>	Repeat this sub-block for each filter.
<i>id</i>	Filter ID.
<i>properties</i>	
<i>washcoat_flag</i>	0 = Bare filter, 1 = Washcoated filter.
<i>initial</i>	
<i>porosity</i>	Initial wall porosity.
<i>unit_collector_diameter</i>	Initial wall unit collector diameter (<i>m</i>).
<i>permeability</i>	Initial wall permeability (m^2).

Chapter 25: Input and Data Files

Physical Models Source Modeling

Parameter	Description
<i>pore_diameter</i>	Initial wall mean pore diameter (<i>m</i>).
<i>bare</i>	
<i>porosity</i>	Bare filter porosity.
<i>unit_collector_diameter</i>	Bare filter unit collector diameter (<i>m</i>).
<i>soot_cake</i>	
<i>porosity</i>	Soot cake porosity.
<i>unit_collector_diameter</i>	Soot cake unit collector diameter (<i>m</i>).
<i>permeability</i>	Soot cake permeability (<i>m</i> ²).
<i>wall_packing_density</i>	Soot packing density in porous wall (kg/m ³).
<i>cake_packing_density</i>	Soot packing density in soot cake (kg/m ³).
<i>percolation_control_constant</i>	Percolation control constant for calculation of partition coefficient.
<i>ash_layer</i>	
<i>active</i>	0 = Filter does not have ash layer, 1 = Filter has ash layer.
<i>porosity</i>	Ash layer porosity.
<i>unit_collector_diameter</i>	Ash layer unit collector diameter (<i>m</i>).
<i>permeability</i>	Ash layer permeability (<i>m</i> ²).
<i>thickness</i>	Ash layer thickness (<i>m</i>).
<i>filtration_efficiency</i>	
<i>velocity</i>	
<i>option</i>	Velocity that will be used in filtration efficiency calculations. <i>SUPERFICIAL</i> = Velocity from CFD solution, <i>PHYSICAL</i> = Superficial velocity divided by porosity (accounts for contraction effects of pores in the substrate).
<i>scaling_factors</i>	
<i>sf_overall</i>	Scaling factor for overall filtration efficiency (<i>E</i>).
<i>sf_diffusion</i>	Scaling factor for efficiency of filtration by diffusion (<i>E_D</i>).
<i>sf_inertia</i>	Scaling factor for efficiency of filtration by inertia (<i>E_I</i>).

Volume of Fluid Modeling

This section describes input files that contain options for volume of fluid modeling in CONVERGE.

Volume of Fluid: vof.in

To activate [volume of fluid \(VOF\) modeling](#), set *inputs.in* > *feature_control* > *vof_flag* = 1 and include a *vof.in* file in your case setup.

```
version: 3.1
---

front_capture_model: NO_FRONT_CAPTURE
face_density_approximation: ARITHMETIC_AVERAGE
reference_state:
    ref_temp: 300
    ref_pres: 100000.0
wall_adhesion_model:
    wall_adhesion_flag: 0
    contact_angle: 0.0
cavitation_model:
    cavitation_flag: 0
    cav_liquid: [IC14H30, IIC14H30]
    cav_gas: [C14H30, gC14H30]
    theta_0: [3.84e-07, 3.84e-07]
    condensation_time_factor: [5000, 5000]
    psi_vof_min: [1e-05, 1e-05]
    power_alpha: [-0.54, -0.54]
    power_phi: [-1.76, -1.76]
vof_spray_flag: 0
dissolved_gas_model:
    dissolved_gas_flag: 1
    dissolved_gases:
        - dissolved_gas:
            gas: N2
            passive: PASSIVE1
            henry_constant: 10000.0
            time_scale: 0.1
        - dissolved_gas:
            gas: O2
            passive: PASSIVE2
            henry_constant: 20000.0
            time_scale: 0.2
    liquid: IC14H30
    gas: N2
mixture_model:
    mixture_model_flag: 0
    particle_diameter: 1e-04
    inertial_effect_flag: 0
    inertial_limit: 98
    max_cfl_drift: 0.3
```

Figure 25.120: An example *vof.in* file that includes cavitation, wall adhesion modeling, and the mixture model.

Table 25.96: Format of *vof.in*.

Parameter	Description	Typical value
<i>front_capture_model</i>	<i>NO_FRONT_CAPTURE</i> = No front tracking, <i>HRIC</i> = High resolution interface capturing (HRIC) scheme, <i>PLIC</i> = Piecewise linear interface calculation (PLIC) scheme, <i>FCT</i> = Flux-corrected transport (FCT) scheme.	

Chapter 25: Input and Data Files

Physical Models Volume of Fluid Modeling

Parameter	Description	Typical value
<i>face_density_approximation</i>	<i>ARITHMETIC_AVERAGE</i> = Approximate cell face density with an arithmetic average of density, <i>HARMONIC_AVERAGE</i> = Approximate cell face density with a harmonic average of density.	
<i>reference_state</i>		
<i>ref_temp</i>	Reference temperature (K) used in the equation of state to calculate gas and incompressible liquid density.	
<i>ref_pressure</i>	Reference pressure (Pa) used in the equation of state to calculate gas density. (For the density of a compressible liquid, CONVERGE uses the <i>RefPressure</i> from liquid.dat .)	
<i>wall_adhesion_model</i>		
<i>wall_adhesion_flag</i>	0 = Do not use the wall adhesion model, 1 = Use the wall adhesion model.	
<i>contact_angle</i>	The contact angle (in <i>degrees</i>) that the fluid is assumed to make with the wall. Used to adjust the normal vectors of the cells near the wall.	
<i>cavitation_model</i>		
<i>cavitation_flag</i>	0 = Do not use the cavitation model, 1 = Use the cavitation model .	
<i>cav_liquid</i>	Name of the liquid species (must be defined in species.in and included in liquid.dat). You may specify more than one species in flow format . If you specify multiple species, each of the parameters in this settings block (except <i>cavitation_flag</i>) must have a list of the same length, and the <i>n</i> -th element of each list corresponds to the <i>n</i> -th element of every other list.	
<i>cav_gas</i>	Name of the gas (vapor) species (must be available in the thermodynamic data file). See <i>cav_liquid</i> for flow formatting instructions.	
<i>theta_0</i>	Time-scale coefficient . See <i>cav_liquid</i> for flow formatting instructions.	
<i>condensation_time_factor</i>	Condensation time factor , <i>F</i> . Used to lower the condensation rate. See <i>cav_liquid</i> for flow formatting instructions.	
<i>psi_vof_min</i>	Minimum value allowed for φ . See <i>cav_liquid</i> for flow formatting instructions.	
<i>power_alpha</i>	Power index of α . See <i>cav_liquid</i> for flow formatting instructions.	

Chapter 25: Input and Data Files

Physical Models Volume of Fluid Modeling

Parameter	Description	Typical value
<i>power_phi</i>	Power index of ϕ. See <i>cav_liquid</i> for flow formatting instructions.	
<i>vof_spray_flag</i>	0 = Do not write out the <i>vof_spray.out</i> file, 1 = Write out the <i>vof_spray.out</i> file (used for VOF-spray one-way coupling). If 1, <i>vof_spray.in</i> is required.	
<i>dissolved_gas_model</i>		
<i>dissolved_gas_flag</i>	0 = No dissolved gas modeling, 1 = Dissolved gas modeling.	
<i>dissolved_gases</i>		
- <i>dissolved_gas</i>	This settings sub-block specifies the dissolved gas configuration. Repeat this sub-block for each dissolved gas.	
<i>gas</i>	Name of gas species in which to model dissolution.	
<i>passive</i>	Dissolved gas passive name.	
<i>henry_constant</i>	Henry constant.	
<i>time_scale</i>	Time scale.	
<i>liquid</i>	The liquid species name that represents the dissolved gas in solution.	
<i>gas</i>	The corresponding gas species name (CONVERGE uses the gas species to obtain the molecular weight of the liquid).	
<i>mixture_model</i>		
<i>mixture_model_flag</i>	0 = No mixture model, 1 = Use the mixture model for all cells, 2 = Use the long-scale interface model to determine which cells are solved with the mixture model and which cells are solved using the VOF method without the mixture model.	
<i>particle_diameter</i>	Particle diameter (<i>m</i>) for dispersed phase.	1e-4
<i>inertial_effect_flag</i>	0 = Do not include inertial acceleration, 1 = Include inertial acceleration.	
<i>inertial_limit</i>	Maximum value (<i>m/s²</i>) of inertial acceleration.	98.0
<i>max_cfl_drift</i>	Maximum CFL number based on drift velocity.	0.3

VOF-Spray One-Way Coupling: `vof_spray.in`

If `vof.in` > `vof_spray_flag` = 1, you must include `vof_spray.in`. This file specifies how CONVERGE will write the `vof_spray.out` file, which can be used for [VOF-spray one-way coupling](#).

```
version: 3.1
---

vof_spray_injections:
  - stream:
    stream_id: 0
    twrite_vof_spray: 1.0e-8
    injection_list:
      - injection:
        vof_spray_injector_name: injector1
        nozzles:
          - nozzle:
            name: nozzle1
            region_pairs: [1, 2]
  - stream:
    stream_id: 1
    twrite_vof_spray: 2.0e-8
    injection_list:
      - injection:
        vof_spray_injector_name: injector2
        nozzles:
          - nozzle:
            name: nozzle2
            region_pairs: [11, 12]
```

Figure 25.121: An example `vof_spray.in` file.

Chapter 25: Input and Data Files

Physical Models Volume of Fluid Modeling

Table 25.97: Format of *vof_spray.in*.

Parameter	Description
<i>vof_spray_injections</i>	
- <i>stream</i>	Stream sub-block. If <i>vof_spray.in</i> is in the Case Directory, include a separate sub-block for each relevant stream. If <i>vof_spray.in</i> is in a stream-specific subdirectory, include one sub-block for only that stream.
<i>stream_id</i>	Stream ID for the region pairs in this sub-block.
<i>twrite_vof_spray</i>	Frequency (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) with which CONVERGE will write data to <i>vof_spray.out</i> for this stream.
<i>injection_list</i>	
- <i>injection</i>	Injection sub-block. Repeat for each injection in the stream sub-block.
<i>vof_spray_injector_name</i>	Name of the VOF-spray injector. CONVERGE writes this name to the <i>vof_spray.out</i> file to identify the data corresponding to the injector.
<i>nozzles</i>	
- <i>nozzle</i>	Nozzle sub-block. Repeat for each nozzle.
<i>name</i>	Name of the nozzle. CONVERGE writes this name to the <i>vof_spray.out</i> file to identify the data corresponding to the nozzle.
<i>region_pairs</i>	A pair of adjacent regions (identified by ID numbers) corresponding to this nozzle. CONVERGE will write data from the interface between these regions to <i>vof_spray.out</i> .

Super-Cycling: **supercycle.in**

To activate [super-cycling](#), set [inputs.in](#) > *feature_control* > *cht_supercycle_flag* = 1 and include a *supercycle.in* file in your case setup.

Chapter 25: Input and Data Files

Physical Models Super-Cycling: *supercycle.in*

```
version: 3.1
---

time_control:
  supercycle_start_time:          -360.0
  supercycle_stage_interval:      60.0
  supercycle_num_stages:          12
  supercycle_length:              -1.0
  stream_dependent_data:
    - stream_data:
        stream_id:                1
        supercycle_stage_interval: 120.0
        supercycle_num_stages:      6
    - stream_data:
        stream_id:                2
        supercycle_stage_interval: 240.0
        supercycle_num_stages:      3
  numerics_control:
    supercycle_cflk:               100.0
    supercycle_energy_tol:         1e-8
    supercycle_energy_omega:       1.0
    supercycle_steady_coeff:       0.0001
  supercycle_surface_map_flag:     1
  supercycle_points:
    - [0.0001, 0.0002, 0.0003]
    - [0.0006, 0.0005, 0.0004]
```

Figure 25.122: An example *supercycle.in* file.

Table 25.98: Format of *supercycle.in*.

Parameter	Description	Typical value
<i>time_control</i>		
<i>supercycle_start_time</i>	The start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) for the super-cycle model.	
<i>supercycle_stage_interval</i>	The time interval (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) of a super-cycle stage. During this interval, CONVERGE calculates and stores heat transfer coefficient and near-wall temperature data. At the end of a <i>supercycle_stage_interval</i> , CONVERGE averages these values for each cell at the solid/fluid interface.	60 seconds
<i>supercycle_num_stages</i>	Total number of <i>stages</i> over which CONVERGE will average data.	12
<i>supercycle_length</i>	If negative, CONVERGE uses a steady-state solver to solve the heat transfer in the solid. If positive, CONVERGE uses a transient solver and the magnitude of this parameter represents the length of time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is	

Chapter 25: Input and Data Files

Physical Models Super-Cycling: supercycle.in

Parameter	Description	Typical value
	non-zero) for the transient calculation. Use the transient approach to obtain a time-accurate solution.	
<i>stream_dependent_data</i> *	CONVERGE does not read this sub-block if <i>supercycle.in</i> is in a directory specified in <i>stream_settings.in</i> > <i>stream_list</i> > <i>stream</i> > <i>stream_overwrite</i> .	
- <i>stream_data</i>	This settings sub-block controls the timing for a stream. Repeat this sub-block for each stream.	
<i>stream_id</i>	Stream ID.	
<i>supercycle_stage_interval</i>	The time interval (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) of a super-cycle stage for the designated stream. During this interval, CONVERGE calculates and stores heat transfer coefficient and near-wall temperature data. At the end of a <i>supercycle_stage_interval</i> , CONVERGE averages these values for each cell at the solid/fluid interface.	
<i>supercycle_num_stages</i>	Number of <i>stages</i> over which CONVERGE will average data for the designated stream ID.	
<i>numerics_control</i>		
<i>supercycle_cflk</i>	CFL number for solid sensible internal energy (sie) solver.	100.0
<i>supercycle_energy_tol</i>	Sensible internal energy tolerance for the solver in the solid region. This parameter dictates the convergence criterion for a steady-state solid heat transfer calculation.	1.0e-07
<i>supercycle_energy_omega</i>	Sensible internal energy under-relaxation factor for the solver in the solid region.	1.4
<i>supercycle_steady_coeff</i>	Coefficient used to check if the transient super-cycle procedure has reached a steady temperature.	
<i>supercycle_surface_map_flag</i>	0 = No cylinder duplication, 1 = Cylinder duplication (<i>supercycle_surface_map.in</i> required).	
<i>supercycle_points</i> *	List of monitor points to monitor temperature data in the solid region. On the rows that follow, list the (x,y,z) coordinates of each monitor point. Use the <i>supercycle_points</i> parameter to output temperature only during the super-cycle sequence. CONVERGE	

Chapter 25: Input and Data Files

Physical Models Super-Cycling: *supercycle.in*

Parameter	Description	Typical value
	writes monitor point data to <i>supercycle_point<number>.out</i> .	
<i><point coordinates></i>	Super-cycle monitor point coordinates. Repeat for each desired point.	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Surface Duplication for CHT: *supercycle_surface_map.in*

To activate surface duplication for heat transfer mapping (e.g., cylinder duplication for a multi-cylinder [conjugate heat transfer](#) [CHT] case), set *supercycle.in* > *supercycle_surface_map_flag* = 1 and include a *supercycle_surface_map.in* file in your case setup.

```
version: 3.1
---

primaries:
  - primary:
      bound_ids: [37, 10025, 26, 10010]
      copies:
        - copy:
            transformation:
              mirror_plane: [0.0, 0.0, 0.0, 0.0]
              translation: [0, -0.865252, 0]
              rotation:
                rot_angle: 0
                orig_xyz: [0, 0, 1.0]
                vector_xyz: [0, 0, 0]
            forced_pairs:
              - [10032, 100058]
              - [25465, 32655]
        - copy:
            transformation:
              mirror_plane: [0, 0, 0, 0]
              translation: [0, -0.865252, 0]
              rotation:
                rot_angle: 0
                orig_xyz: [0, 0, 1.0]
                vector_xyz: [0, 0, 0]
      - primary:
          bound_ids: [38, 10026, 27, 10011]
          copies:
            - copy:
                transformation:
                  mirror_plane: [0, 0, 0, 0]
                  translation: [0, -0.865252, 0]
                  rotation:
                    rot_angle: 0
                    orig_xyz: [0, 0, 1.0]
                    vector_xyz: [0, 0, 0]
                forced_pairs:
                  - [23232, 5465454]
            - copy:
                transformation:
                  mirror_plane: [0, 0, 0, 0]
                  translation: [0, -0.865252, 0]
                  rotation:
                    rot_angle: 0
                    orig_xyz: [0, 0, 1.0]
                    vector_xyz: [0, 0, 0]
```

Figure 25.123: An example *supercycle_surface_map.in* file.

Chapter 25: Input and Data Files

Physical Models Surface Duplication for CHT: `supercycle_surface_map.in`

Table 25.99: Format of `supercycle_surface_map.in`.

Parameter	Description
<i>primaries</i>	
- <i>primary</i>	This settings sub-block specifies the configuration of a primary cylinder. Repeat this sub-block through <i>forced_pairs</i> for each primary cylinder.
<i>bound_ids</i>	The boundary IDs of the boundaries making up the current main cylinder.
<i>copies</i>	
- <i>copy</i>	This settings sub-block specifies the configuration of a copy cylinder. Repeat this sub-block through <i>forced_pairs</i> for each copy cylinder
<i>transformation</i>	
<i>mirror_plane</i>	Equation for the mirror plane that reflecting the duplicate cylinder across results in the duplicate overlapping the main exactly. Enter the <i>a</i> , <i>b</i> , <i>c</i> , and <i>d</i> coefficients of the equation that describes this mirror plane ($ax + by + cz + d = 0$).
<i>translation</i>	The <i>z</i> , <i>y</i> , and <i>z</i> translation amount necessary to translate the duplicate cylinder onto the main cylinder.
<i>rotation</i>	
<i>rot_angle</i>	Angle with which to rotate the duplicate cylinder in the counter-clockwise direction.
<i>orig_xyz</i>	The <i>x</i> , <i>y</i> , and <i>z</i> coordinates of the rotation origin.
<i>vector_xyz</i>	The <i>x</i> , <i>y</i> , and <i>z</i> components of the direction vector about which to rotate the duplicate cylinder.
<i>forced_pairs</i> *	Forced pairs are used if CONVERGE cannot determine the duplicate boundary on which to map the data (possibly due to imperfect geometries) based on the given transformation information. The two specified boundary IDs become forced pairs and CONVERGE copies the information from main to duplicate, regardless of the transformation. Supply the main boundary ID and the duplicate boundary ID.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

1D Conjugate Heat Transfer Modeling

This section describes input files that contain options for 1D conjugate heat transfer modeling in CONVERGE.

1D Conjugate Heat Transfer: `cht1d.in`

When you set up a WALL boundary type with the [1D CHT](#) model in conjunction with the temperature boundary condition, you must include a `cht1d.in` file in your case setup. Note that you can use 1D CHT only with a law-of-the-wall or Dirichlet temperature boundary condition.

Chapter 25: Input and Data Files

Physical Models 1D Conjugate Heat Transfer Modeling

```
version: 3.1
---

boundaries:
  - boundary:
    boundary_id: 3
    htc_scale_factor: 1.0
    solid_bulk_temp: 1.0
    solid_layers:
      - solid_layer:
        solid_layer_species: metal
        solid_layer_thickness: 0.005
        num_sub_layers: 1
        contact_resistance: 0
      - solid_layer:
        solid_layer_species: metal2
        solid_layer_thickness: 0.005
        num_sub_layers: 1
        contact_resistance: 0
  - boundary:
    boundary_id: 5
    htc_scale_factor: 1.0
    solid_bulk_temp: 0.0
    solid_layers:
      - solid_layer:
        solid_layer_species: metal
        solid_layer_thickness: 0.005
        num_sub_layers: 1
        contact_resistance: 0
```

Figure 25.124: An example *cht1d.in* file.

Chapter 25: Input and Data Files

Physical Models 1D Conjugate Heat Transfer Modeling

Table 25.100: Format of *cht1d.in*.

Parameter	Description
<i>boundaries</i>	
- <i>boundary</i>	This settings sub-block specifies the boundary configuration. Repeat this sub-block for each boundary configured with the 1D CHT boundary condition.
<i>boundary_id</i>	The boundary ID of the boundary on which CONVERGE will model 1D CHT.
<i>htc_scale_factor</i>	Scaling factor applied to the heat transfer coefficient (HTC) after it is calculated from the fluid heat flux. The recommended value is 1.0.
<i>solid_bulk_temp</i>	Solid bulk temperature. This entry can be a constant temperature (in K) or, for spatially varying quantities, a file name (e.g., <i>solid_bulk_temp.in</i>).
<i>solid_layers</i>	
- <i>solid_layer</i>	This settings sub-block specifies the configuration for a solid layer between the fluid and the bulk solid. Repeat this sub-block for each solid layer.
<i>solid_layer_species</i>	Species name of the solid layer between the fluid and the bulk solid. The specified solid must be defined in <i>species.in</i> .
<i>solid_layer_thickness</i>	Thickness (in m) of the solid layer between the fluid and the bulk solid.
<i>num_sub_layers</i>	Number of sublayers into which CONVERGE divides the solid layer. CONVERGE treats each sublayer as a computational cell.
<i>contact_resistance</i>	Thermal contact resistance (in $m^2\text{-K/W}$) between the solid layer and the solid bulk temperature.

Solid Bulk Temperature Profile: *solid_bulk_temp.in*

To set up a spatially and/or temporally varying 1D CHT temperature boundary condition, specify a file name (e.g., *solid_bulk_temp.in*) for [*cht1d.in*](#) > *boundaries* > *boundary* > *solid_bulk_temp* and include that file in your case setup.

Chapter 25: Input and Data Files

Physical Models 1D Conjugate Heat Transfer Modeling

```
SPATIAL
1.0      scale_xyz
-0.0     trans_x
-0.0     trans_y
-0.0     trans_z
z        rot_axis
0.0      rot_angle

0.0      second
x y z   bulk_solid_temp
-2e-2   0.0   1e-2   341.54
0.0     1.0   1e-2   400.00
2e-2   1.0   1e-2   598.71

1.0      second
x y z   bulk_solid_temp
-2e-2   0.0   1e-2   441.54
0.0     1.0   1e-2   500.00
2e-2   1.0   1e-2   698.71
```

Figure 25.125: An example of a file (e.g., *solid_bulk_temp.in*) that contains a spatially and temporally varying solid bulk temperature profile.

Table 25.101: Format of *solid_bulk_temp.in*.

Parameter	Description
<i>scale_xyz</i>	Scaling to be applied to the <i>x</i> , <i>y</i> , and <i>z</i> coordinates of the solid bulk temperature profile (see below).
<i>trans_x</i>	Translation to be applied to <i>x</i> coordinates. CONVERGE scales the solid bulk temperature profile before translating it, so the <i>trans_x</i> units should be consistent with the scaled units.
<i>trans_y</i>	Translation to be applied to <i>y</i> coordinates. CONVERGE scales the solid bulk temperature profile before translating it, so the <i>trans_y</i> units should be consistent with the scaled units.
<i>trans_z</i>	Translation to be applied to <i>z</i> coordinates. CONVERGE scales the solid bulk temperature profile before translating it, so the <i>trans_z</i> units should be consistent with the scaled units.
<i>rot_axis</i>	Axis about which the coordinates and velocity of the data below will be rotated.
<i>rot_angle</i>	Rotation angle (in <i>degrees</i>) about the <i>rot_axis</i> . Use the right hand rule to determine the direction of rotation about <i>rot_axis</i> .
<i>second</i>	Include for file formatting reasons. Set to 0.0.
<i>bulk_solid_temp</i>	Specify <i>x y z</i> for this entry. Each subsequent line should contain a location (<i>x</i> , <i>y</i> , <i>z</i> coordinates) followed by the solid bulk temperature (in <i>K</i>) at that location.

Chapter 25: Input and Data Files

Physical Models Radiation: radiation.in

Radiation: radiation.in

To model [radiation](#), set `inputs.in > feature_control > radiation_flag = 1` and include a `radiation.in` file in your case setup. In addition, you must define two [non-transport passives](#)—`RADIATION` and `RADIATION_SRC`—in [species.in](#) for any simulation that includes radiation modeling.

Chapter 25: Input and Data Files

Physical Models Radiation: radiation.in

```
version: 3.1
---

radiation_model: P1
radiation_spray_coupling:
    rad_spray_coupling_flag: 0
    parcel_emissivity: 0.512
radiation_energy_coupling:
    rad_energy_coupling_flag: SEG
    rad_solve_frequency: 100
scattering:
    scatter_function_flag: 0
    aniso_constant: 1.081
    de_constant: 1.987
    scatter_coeff: 1.121
absorption:
    absorption_coeff_model: 0
    absorption_coeff: 1.912
refractive_index: 1.192
nongray_radiation:
    nongray_model: BAND
    wsrgg:
        num_gray_gases: 2
    band:
        - band_properties:
            lower_bound: 0.534
            upper_bound: 1.216
            absorption:
                absorption_coeff_model: 0
                absorption_coeff: 0.978
                scatter_coeff: 0.876
        - band_properties:
            lower_bound: 0.111
            upper_bound: 1.872
            absorption:
                absorption_coeff_model: 0
                absorption_coeff: 1.276
                scatter_coeff: 1.782
        - band_properties:
            lower_bound: 1.120
            upper_bound: 1.323
            absorption:
                absorption_coeff_model: 0
                absorption_coeff: 1.763
                scatter_coeff: 1.371
radiation_boundaries:
    - boundary:
        boundary_id: 1
        emissivity: 1.172
        transmissivity: 0.012
        diffuse_fraction: 1.816
        bc_type: TEMP
        irradiance: 1.901
    - boundary:
        boundary_id: 2
        emissivity: 1.981
        transmissivity: 0.634
        diffuse_fraction: 1.912
        bc_type: IRRAD
        irradiance: 1.012
    - boundary:
        boundary_id: 3
        side: FORWARD
        emissivity: 1.230
        transmissivity: 0.198
        diffuse_fraction: 1.1
        bc_type: TEMP
        irradiance: 1.012
radiation_regions: [0, 1]
```

Figure 25.126: An example *radiation.in* file.

Chapter 25: Input and Data Files

Physical Models Radiation: radiation.in

Table 25.102: Format of *radiation.in*.

Parameter	Description	Typical value
<i>radiation_model</i>	<i>FVM</i> = Discrete Ordinates-style method-FVM, <i>FTnFVM</i> = Discrete Ordinates-style method-FTnFVM (works well with anisotropic cases). <i>P1</i> = Spherical harmonics reduced-order method.	
<i>discretization</i>		
<i>ordinates_in_octant</i>		
<i>num_theta</i>	Number of polar divisions per octant.	1 - 4
<i>num_phi</i>	Number of azimuthal divisions per octant.	1 - 4
<i>pixels_in_ordinate</i>		
<i>num_theta_pix</i>	Number of polar pixels per ordinate.	1 - 10
<i>num_phi_pix</i>	Number of azimuthal pixels per ordinate.	1 - 10
<i>radiation_spray_coupling</i>		
<i>rad_spray_coupling_flag</i>	Radiation/parcel interaction. 0 = Decoupled, 1 = Coupled.	
<i>parcel_emissivity</i>	Parcel surface emissivity.	
<i>radiation_energy_coupling</i>		
<i>rad_energy_coupling_flag</i>	<i>SEG</i> = Decoupled, <i>i.e.</i> , radiation and energy equations are solved in a sequential manner. This option may be good for optically thinner cases. <i>COUPLE</i> = Radiation and energy equations are solver in a coupled manner.	
<i>rad_solve_frequency</i>	CONVERGE solves radiation every <i>rad_solve_frequency</i> time-steps. For a transient case, 10 may be appropriate. For a steady-state case, 50 may be appropriate.	
<i>scattering</i>		
<i>scatter_function_flag</i>	Scattering function. <i>OFF</i> = Non-scattering, <i>ISO</i> = Isotropic scattering, <i>ANISO</i> = Linear anisotropic scattering, <i>DELTA_EDD</i> = Delta-Eddington model.	
<i>aniso_constant</i>	Asymmetry constant. Used if <i>scatter_function_flag</i> = <i>ISO</i> or <i>ANISO</i> . This parameter varies from -1 to 1.	
<i>de_constant</i>	Delta-Eddington constant. Used if <i>scatter_function_flag</i> = <i>DELTA_EDD</i> .	

Chapter 25: Input and Data Files

Physical Models Radiation: radiation.in

Parameter	Description	Typical value
<i>scatter_coeff</i>	Gray gas scattering coefficient.	
<i>absorption</i>		
<i>absorption_coeff_model</i>	Method for evaluating absorption coefficient value. 0 = Constant value, 1 = Function of gas-phase properties.	
<i>absorption_coeff</i>	Gray gas absorption coefficient ($1/m$).	
<i>refractive_index</i>	Refractive index.	
<i>nongray_radiation</i>		
<i>nongray_model</i>	Type of gas model. GRAY = Gray gas, WSGG = Weighted Sum of Gray Gases, BAND = Band model of gray gas.	
<i>wsgg</i>		
<i>num_gray_gases</i>	Number of gray gases.	
<i>band</i>		
<i>- band_properties</i>	This settings sub-block specifies the band properties. Repeat this sub-block for each band.	
<i>lower_bound</i>	The lower spectral bound.	
<i>upper_bound</i>	The upper spectral bound.	
<i>absorption</i>		
<i>absorption_coeff_model</i>	Method for evaluating absorption coefficient value. 0 = Constant value, 1 = Function of gas-phase properties.	
<i>absorption_coeff</i>	Gray gas absorption coefficient ($1/m$).	
<i>scatter_coeff</i>	Band scattering coefficient.	
<i>radiation_boundaries</i>	Boundaries not specified here will be assumed of <i>bc_type</i> = TEMP with <i>emissivity</i> and <i>diffuse_fraction</i> = 1.	
<i>- boundary</i>	This settings sub-block specifies the boundary configuration. Repeat this sub-block for each boundary.	
<i>boundary_id</i>	Boundary ID.	
<i>side</i>	<i>FORWARD</i> or <i>REVERSE</i> . Specify only for INTERFACE boundaries. Indicates the fluid side of the boundary specified by <i>boundary_id</i> .	
<i>emissivity</i>	Emissivity in the boundary specified by <i>boundary_id</i> .	

Chapter 25: Input and Data Files

Physical Models Radiation: radiation.in

Parameter	Description	Typical value
<i>transmissivity</i>	Transmissivity in the boundary specified by <i>boundary_id</i> .	
<i>diffuse_fraction</i>	Diffuse fraction in the boundary specified by <i>boundary_id</i> .	
<i>bc_type</i>	TEMP or IRRAD. Use IRRAD to model non-thermal radiation.	
<i>irradiance</i>	Irradiance (in W/m^2) on the boundary specified by <i>bound_id</i> . Read only if <i>bc_type</i> = IRRAD.	
<i>radiation_regions</i>	List of all regions in which radiation is active.	

Fluid-Structure Interaction Modeling

This section describes input files that contain options for fluid-structure interaction modeling in CONVERGE.

Fluid-Structure Interaction: fsi.in

To use the [fluid-structure interaction \(FSI\) model](#) in CONVERGE, set [*inputs.in* > feature_control > fsi_flag](#) = 1 and include an *fsi.in* file in your case setup.

Parameters for [FSI events](#) are located in the [*events.in*](#) file.

```
version: 3.1
---

implicit_fsi:
  implicit_fsi_flag: 1
  tolerance:
    force: 1e-3
    moment: 1e-5
  relaxation_factor:
    maximum: 1.0
    minimum: 0.01
  number_of_iterations:
    maximum: 10
    minimum: 2

rigid_fsi_objects:
  - object:
      object_id: check_valve
      boundary_id: [1, 2]
      gti_fsi_flag: 0
      fsi_start_time: -999999.0
      center_of_mass_flag: GEO_CENTER
      center_of_mass: [0.0, 0.0, 0.0]
      mass_flag: 0
      object_mass: 0.0
      object_density: 7900.0
      object_vel_init: [-1.0, 0.0, 0.0]
      object_rot_vel_init: 5.0
      rot_vel_init_axis: [2.0, 0.0, 0.0]
      applied_force: [0.0, 10.5, 0.0]
      applied_moment: 15.0
      applied_moment_axis: [1.0, 0.0, 0.0]
```

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

```
moment_of_inertia_flag:          1
moment_of_inertia_1:            [1.0, 0.0, 0.0]
moment_of_inertia_2:            [0.0, 1.0, 0.0]
moment_of_inertia_3:            [0.0, 0.0, 1.0]
1dof_constraint_flag:          TRANSLATE
axis_point:                    [0.0, 0.0, -1.0]
axis:                          [0.0, -1.0, 0.0]
min_displacement:              1e-38
max_displacement:              1e38
constrained_mom_of_inertia:    1.0
general_constrained_motion:
  active:                      1
  translation:
    type:                        ONE_D
    axis:                        [0, 0, 1]
  rotation:
    type:                        TWO_D
    axis:                        [0, 0, 1]

- object:
  object_id:                   check_valve_2
  boundary_id:                 [3, 4]
  gti_fsi_flag:                0
  fsi_start_time:              -999999.0
  center_of_mass_flag:         GEO_CENTER
  center_of_mass:              [1.0, 0.0, 0.5]
  mass_flag:                   0
  object_mass:                 0.0
  object_density:              7900.0
  object_vel_init:             [0.0, 0.0, 0.0]
  object_rot_vel_init:         0.0
  rot_vel_init_axis:           [0.0, 0.0, 0.0]
  applied_force:               fsi_force.in
  applied_moment:              0.0
  applied_moment_axis:         [0.0, 0.0, 0.0]
  moment_of_inertia_flag:      0
  1dof_constraint_flag:        ROTATE
  axis_point:                  [0.0, 0.0, 0.0]
  axis:                        [0.0, 0.0, 0.0]
  min_displacement:            1e-10
  max_displacement:            1e10
  constrained_mom_of_inertia: 1.0

beam_objects:
- object:
  object_id:                   suction_valve
  boundary_ids:                [6, 5, 4]
  variable_properties_flag:    1
  length:                      2.2585e-2
  property_file:               beam_profile_1.in
  c_k:                         0.0
  c_m:                         0.0
  fluid_load_type:              VARIABLE_LOADING
  beam_origin:                 [0.0, 1.123627E-02, 3.250000E-04]
  beam_axis:                   [0.0, -1.0, 0.0]
  beam_normal:                 [0.0, 0.0, -1.0]

  num_elements:                50
  beam_type:                   CANTILEVER
  external_load_type:          NONE
  initial_deflection_flag:     0
  contact_model:               1
  constraints:
    - constraint:              [0.0, 0.0, 1.0, -3.2500E-04]
    - constraint:              [0.0, 0.0, 1.0, 0.013675]
  constraint_bound_ids:        [9]
  monitor_lines:
    - monitor_line:            [0, 0.02, 11]
  monitor_variables:           [velocity, acceleration, load]

- object:
```

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

```
object_id: discharge_valve
boundary_ids: [12,11,10]
variable_properties_flag: 1
length: 2.88e-2
property_file: beam_profile_2.in
c_k: 0.0
c_m: 0.0
fluid_load_type: VARIABLE_LOADING
beam_origin: [1.389898E-02, 5.750001E-03, 1.804971E-03]
beam_axis: [-1.0, 0.0, 0.0]
beam_normal: [0.0, 0.0, 1.0]

num_elements: 50
beam_type: CLAMPED_CLAMPED
external_load_type: NONE
initial_deflection_flag: 0
contact_model: 1
resting_contact: 0
coefficient_of_restitution: 0.5
constraints:
  - constraint: [0.0]
  - constraint: beam_limit_max_2.in
  - constraint: [0.0, 0.0, 1.0, -1.804969E-03]
constraint_bound_ids: [9]
monitor_points:
  - monitor_point: 0.0144
  - monitor_point: 0.02585
monitor_lines:
  - monitor_line: ALL NODES
monitor_variables: [displacement, bending_stress, shear_stress]

beam_solver_settings:
  beam_solver: AUTO
  penalty_multiplier_mmmt: 1
  time_step_multiplier: 1

spring_flag: 1
stiction_flag: 0
```

Figure 25.127: An example *fsi.in* file.

Table 25.103: Format of *fsi.in*.

Parameter	Description
<i>implicit_fsi</i>	This settings block is used only for rigid FSI objects.
<i>implicit_fsi_flag</i>	0 = Do not use implicit FSI, 1 = Use implicit FSI.
<i>tolerance</i>	
<i>force</i>	Tolerance for the convergence of the force.
<i>moment</i>	Tolerance for the convergence of the moment.
<i>relaxation_factor</i>	
<i>maximum</i>	Maximum value for the force relaxation factor.
<i>minimum</i>	Minimum value for the force relaxation factor.
<i>number_of_iterations</i>	
<i>maximum</i>	Maximum number of iterations in a single time-step.
<i>minimum</i>	Minimum number of iterations in a single time-step.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
<i>rigid_fsi_objects</i>	
- <i>object</i>	This settings sub-block specifies the configuration of a rigid FSI object. Repeat this sub-block through <i>constrained_mom_of_inertia</i> (and optionally through <i>rotation > axis</i>) for each rigid FSI object.
<i>object_id</i>	A string used to identify an FSI object. For GT-SUITE/FSI coupling, the <i>object_id</i> must match the name given in GT-SUITE.
<i>boundary_id</i>	The boundary ID(s) (from boundary.in) of a boundary in the FSI object identified by the <i>object_id</i> above.
<i>gti_fsi_flag</i>	Flag for simulating FSI in a coupled GT-SUITE/CONVERGE simulation. 0 = CONVERGE will not model FSI, 1 = CONVERGE will model FSI.
<i>fsi_start_time</i>	The start time of FSI computations (in seconds if inputs.in > <i>simulation_control > crank_flag</i> = 0 or in CAD if <i>crank_flag</i> is non-zero).
<i>center_of_mass_flag</i>	GEO_CENTER or 0 = CONVERGE will use the geometric center of the FSI object as the center of mass. Use for uniform density FSI objects. CONVERGE will ignore the <i>center_of_mass</i> parameter if <i>center_of_mass_flag</i> = 0. OFFSET_GEO_CENTER or 1 = CONVERGE will offset the geometric center of the FSI object by the distance and amount specified by <i>center_of_mass</i> (see below) to determine the center of mass. Use for FSI objects with non-uniform density. CENTER_IN_GLOBAL_COORD or 2 = CONVERGE will set the center of mass to be the location specified by <i>center_of_mass</i> (see below). Use for FSI objects with non-uniform density. If <i>boundary_id</i> contains a fluid-fluid INTERFACE boundary, you must set <i>center_of_mass_flag</i> = 2.
<i>center_of_mass</i>	The x, y, z coordinates for center of mass. Used only when <i>center_of_mass_flag</i> = 1 or 2 (see above).
<i>mass_flag</i>	Flag for calculation of mass of the FSI object. 0 = CONVERGE will calculate the mass of the object after reading the average density from <i>object_density</i> (see below), 1 = CONVERGE will calculate the density of the object after reading the total mass from <i>object_mass</i> (see below). If <i>boundary_id</i> contains a fluid-fluid INTERFACE boundary, you must set <i>mass_flag</i> = 1.
<i>object_mass</i>	Mass (kg) of the FSI object. Used when <i>mass_flag</i> = 1.
<i>object_density</i>	Uniform density (kg/m^3) of the FSI object. Used when <i>mass_flag</i> = 0.
<i>object_vel_init</i>	Initial velocity (u, v, w) of the FSI object. Units are m/s.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
<i>object_rot_vel_init</i>	Initial angular velocity (<i>rad/s</i>) about the axis given by <i>rot_vel_init_axis</i> (see below).
<i>rot_vel_init_axis</i>	Initial axis vector (<i>x, y, z</i>) for rotation for the FSI object.
<i>applied_force</i>	Constant applied force vector (<i>x, y, z</i>). Units are <i>N</i> . To set up temporally varying force vectors, specify a file name (<i>e.g.</i> , <i>fsi_force.in</i>).
<i>applied_moment</i>	Constant applied moment (<i>N-m</i>).
<i>applied_moment_axis</i>	The axis (<i>x, y, z</i>) about which the <i>applied_moment</i> (see above) is applied.
<i>moment_of_inertia_flag</i>	0 = CONVERGE will compute the moment of inertia (constant density assumed). 1 = CONVERGE will read the moment of inertia tensor from the next three lines, or, if <i>1dof_constraint_flag</i> = 2 (see below), CONVERGE will use the value specified by <i>constrained_mom_of_inertia</i> (see below). If <i>boundary_id</i> contains a fluid-fluid INTERFACE boundary, you must set <i>moment_of_inertia_flag</i> = 1.
<i>moment_of_inertia_1*</i>	User-supplied input for the first row of moment of inertia tensor. Used when <i>moment_of_inertia_flag</i> = 1. Units are <i>kg-m²</i> .
<i>moment_of_inertia_2*</i>	User-supplied input for the second row of moment of inertia tensor. Used when <i>moment_of_inertia_flag</i> = 1. Units are <i>kg-m²</i> .
<i>moment_of_inertia_3*</i>	User-supplied input for the third row of moment of inertia tensor. Used when <i>moment_of_inertia_flag</i> = 1. Units are <i>kg-m²</i> .
<i>1dof_constraint_flag</i>	Degrees of freedom. 0 = The FSI object will have six degrees of freedom (three for translation and three for rotation), 1 = The FSI object will be constrained to one translational degree of freedom; and the direction and minimum and maximum distances of translation will be defined by <i>axis</i> , <i>min_displacement</i> , and <i>max_displacement</i> (see below), 2 = The FSI object will be constrained to one rotational degree of freedom; the rotation will be about the axis defined by <i>axis_point</i> and <i>axis</i> ; and the minimum and maximum angles of rotation will be defined by <i>min_displacement</i> and <i>max_displacement</i> . If you set <i>1dof_constraint_flag</i> to non-zero, you cannot activate general constrained motion.
<i>axis_point</i>	The <i>x, y, z</i> coordinates for the origin that defines the rotational axis. Used only when <i>1dof_constraint_flag</i> = 2.
<i>axis</i>	The axis vector (<i>x, y, z</i>) for defining the rotational axis. Used only when <i>1dof_constraint_flag</i> = 1 or 2.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
<i>min_displacement</i>	The minimum displacement for translationally or rotationally constrained cases. If <i>1dof_constraint_flag</i> = 1, units are <i>m</i> . If <i>1dof_constraint_flag</i> = 2, units are <i>degrees</i> .
<i>max_displacement</i>	The maximum displacement for translationally or rotationally constrained cases. If <i>1dof_constraint_flag</i> = 1, units are <i>m</i> . If <i>1dof_constraint_flag</i> = 2, units are <i>degrees</i> .
<i>constrained_mom_of_inertia</i>	If <i>moment_of_inertia_flag</i> = 0, CONVERGE will compute the moment of inertia. if <i>moment_of_inertia_flag</i> = 1, CONVERGE will use the value (in <i>kg-m²</i>) entered here. Used only when <i>1dof_constraint_flag</i> = 2.
<i>general_constrained_motion</i> *	Settings sub-block for general constrained motion.
<i>active</i>	0 = Do not constrain motion with this feature, 1 = Constrain motion with this feature. If you set <i>active</i> = 1, you cannot set <i>fsi.in > rigid_fsi_objects > object > 1dof_constraint_flag</i> to non-zero.
<i>translation</i>	Translational motion settings sub-block.
<i>type</i>	<i>LOCKED</i> = Do not allow translation, <i>ONE_D</i> = Allow translation in one dimension, <i>TWO_D</i> = Allow translation in two dimensions, <i>UNCONSTRAINED</i> = Allow translation in all directions.
<i>axis</i>	The x, y, and z coordinates of the vector which restricts translation. If <i>type</i> = <i>ONE_D</i> , this is the vector along which translation is allowed. If <i>type</i> = <i>TWO_D</i> , this is the normal vector of the plane across which translation is allowed.
<i>rotation</i>	Rotational motion settings sub-block.
<i>type</i>	<i>LOCKED</i> = Do not allow rotation, <i>ONE_D</i> = Allow rotation in one dimension, <i>TWO_D</i> = Allow rotation in two dimensions, <i>UNCONSTRAINED</i> = Allow rotation in all directions.
<i>axis</i>	The x, y, and z coordinates of the vector which restricts rotation. If <i>type</i> = <i>ONE_D</i> , this is the vector along which rotation is allowed. If <i>type</i> = <i>TWO_D</i> , this is the normal vector of the plane across which rotation is allowed.
<i>hinge_point</i> *	The x, y, and z coordinates of a fixed hinge point. The FSI object will rotate about that hinge point. You can alternately specify <i>CENTER_OF_MASS</i> for the hinge point.
<i>beam_objects</i>	

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
<code>- object</code>	This settings sub-block specifies the configuration of a beam object. Repeat this sub-block through <code>contact_model</code> (and optionally through <code>constraint_bound_ids</code> or <code>monitor_point</code>) for each FSI beam object.
<code>object_id</code>	A string used to identify an FSI object. For GT-SUITE/FSI coupling, the <code>object_id</code> must match the name given in GT-SUITE.
<code>boundary_ids</code>	The boundary ID(s) (from boundary.in) of a boundary in the FSI object identified by the <code>object_id</code> above.
<code>variable_properties_flag</code>	0 = Specify beam properties below, 1 = Specify beam properties in a property file (see <code>property_file</code> below).
<code>length</code>	Length (<i>m</i>) of the beam. Used only when <code>variable_properties_flag</code> = 0.
<code>breadth</code>	Breadth (<i>m</i>) of the beam. Used only when <code>variable_properties_flag</code> = 0. If you provide <code>area</code> and <code>height</code> , CONVERGE will automatically calculate <code>breadth</code> .
<code>height</code>	Height (<i>m</i>) of the beam. Used only when <code>variable_properties_flag</code> = 0. If you provide <code>area</code> and <code>breadth</code> , CONVERGE will automatically calculate <code>height</code> .
<code>area</code>	Cross-sectional area (<i>m</i> ²) of the beam. Used only when <code>variable_properties_flag</code> = 0. If you provide <code>height</code> and <code>breadth</code> , CONVERGE will automatically calculate <code>area</code> .
<code>density</code>	Density (<i>kg/m</i> ³) of the beam. Used only when <code>variable_properties_flag</code> = 0.
<code>youngs_modulus</code>	Young's modulus (<i>Pa</i>) of the beam. Used only when <code>variable_properties_flag</code> = 0.
<code>property_file</code>	Beam property file name (e.g. beam_profile.in). Used only when <code>variable_properties_flag</code> = 1.
<code>c_k</code>	Stiffness damping coefficient (<i>m</i>).
<code>c_m</code>	Mass damping coefficient (<i>m</i> ⁻¹).
<code>fluid_load_type</code>	0 = Uniform fluid loading, 1 = Variable fluid loading.
<code>beam_origin</code>	Coordinates of the origin of the beam. For <code>beam_type</code> = CANTILEVER, this is the fixed end.
<code>beam_axis</code>	Vector of the beam centerline.
<code>beam_normal</code>	Vector in the direction of beam deflection.
<code>num_elements</code>	Number of discrete material elements for the beam.
<code>beam_type</code>	Type of beam support.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
	<i>CANTILEVER</i> = Origin end of the beam fixed in location and rotation, <i>CLAMPED_CLAMPED</i> = Both ends of the beam fixed in location and rotation, <i>SIMPLY_SUPPORTED</i> = Both ends of the beam fixed in location but free in rotation.
<i>external_load_type</i>	Type of external (<i>i.e.</i> , non-fluid) load on the beam. <i>NONE</i> = No external load, <i>UNIFORM_LOAD</i> = Uniform load along the beam, <i>VARIABLE_LOAD</i> = Variable load along the beam, <i>FORCE_AT_END</i> = Impose a force at the non-origin end of the beam, <i>MOMENT_AT_END</i> = Impose a moment at the non-origin end of the beam.
<i>external_load_value</i>	Magnitude of the external load (<i>N</i> or <i>N-m</i>), Magnitude of the external load (<i>N</i> or <i>N-m</i>), or specify a file name (<i>e.g.</i> , <i>external_load_value.in</i>).
<i>external_load_temporal_type</i>	Temporal type for the external load. Available options are <i>CYCLIC</i> , <i>SEQUENTIAL</i> , and <i>PERMANENT</i> .
<i>external_load_period</i>	The period for a <i>CYCLIC</i> external load (in <i>seconds</i> if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).
<i>external_load_start_time</i>	Start time (in <i>seconds</i> if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for a <i>CYCLIC</i> or <i>SEQUENTIAL</i> external load.
<i>external_load_end_time</i>	End time (in <i>seconds</i> if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for a <i>CYCLIC</i> or <i>SEQUENTIAL</i> external load.
<i>initial_deflection_flag</i>	0 = No initial deflection, 1 = Specify initial deflection from a file (<i>e.g.</i> , <i>initial_deflection.in</i>), 2 = Calculate initial deflection from a specified initial load.
<i>initial_load_type</i>	Type of initial load on the beam. <i>NONE</i> = No initial load, <i>UNIFORM_LOAD</i> = Uniform load along the beam, <i>VARIABLE_LOAD</i> = Variable load along the beam, <i>FORCE_AT_END</i> = Impose a force at the non-origin end of the beam, <i>MOMENT_AT_END</i> = Impose a moment at the non-origin end of the beam.
<i>initial_load_value</i>	Magnitude of the initial load (<i>N</i> or <i>N-m</i>), or specify a file name (<i>e.g.</i> , <i>external_load_value.in</i>).
<i>initial_load_file</i>	File name that specifies the initial load.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
<i>contact_model</i>	0 = No contact model, 1 = Check for constraints on FSI beam motion (resting contact model or other constraints that you specify using the <i>constraints</i> sub-block).
<i>resting_contact</i> *	Available only when <i>contact_model</i> = 1 and <i>beam_solver</i> = MMT or AUTO. 0 = No resting contact model, 1 = Enable a resting contact model to prevent deflection of the beam in the direction opposite to <i>beam_normal</i> .
<i>coefficient_of_restitution</i> *	Controls the rebound velocity of the beam when it hits a constraint. Specify a value between 0 and 1 (0 = beam has no rebound velocity, 1 = beam rebounds elastically). Default value is 0.5. Used only when <i>contact_model</i> = 1 and <i>beam_solver</i> = MMT or AUTO.
<i>constraints</i> *	
- <i>constraint</i>	Constraint definition. Specify four real values that define a constraint plane as $ax + by + cz + d = 0$, or specify a file name (e.g., beam_limit.in). Repeat for each constraint.
<i>constraint_bound_ids</i> *	Specify boundary ID(s) which the FSI beam object cannot intersect.
<i>monitor_points</i> *	
- <i>monitor_point</i>	Location along the beam axis (<i>m</i>) for this beam monitor point. Repeat for each monitor point.
<i>monitor_lines</i> *	
- <i>monitor_line</i>	For each monitor line, specify the beginning location along the beam axis (<i>m</i>), the end location along the beam axis (<i>m</i>), and the number of points, or enter ALL_NODES to generate output for all computational nodes on the beam.
<i>monitor_variables</i> *	List the variables to monitor. The available variables are <i>displacement</i> , <i>velocity</i> , <i>acceleration</i> , <i>load</i> , <i>bending_stress</i> , and <i>shear_stress</i> . If you do not specify this parameter, all of the available variables are monitored by default.
<i>beam_solver_settings</i> *	
<i>beam_solver</i> *	AUTO or 0 = CONVERGE chooses the appropriate beam solver based on case setup (HHT if <i>contact_model</i> = 0, MMT if <i>contact_model</i> = 1), HHT or 1 = CONVERGE solves the beam motion with the Hilber-Hughes-Taylor method, MMT or 2 = CONVERGE solves the beam motion with Moreau's midpoint time-stepping scheme.
<i>penalty_multiplier_mmt</i> *	Value by which the penalty calculated by the MMT beam solver is multiplied (decreased penalty corresponds to decreased impulse)

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
	and vice versa). Used only when <i>beam_solver</i> = MMT or when <i>contact_model</i> = 1 and <i>beam_solver</i> = AUTO. Recommended value is 1. Must be greater than 0. Note that CONVERGE may further adjust the penalty after applying this multiplier if the beam solver does not converge.
<i>time_step_multiplier</i> *	Value by which the time-step for the MMT beam solver is multiplied (applies only to the beam solver time-step, which is different from the fluid solver time-step). Used only when <i>beam_solver</i> = MMT or when <i>contact_model</i> = 1 and <i>beam_solver</i> = AUTO. Recommended value is 1. Must be greater than 0. Note that CONVERGE may further adjust the beam solver time-step after applying this multiplier if needed for simulation stability.
<i>spring_flag</i>	Flag for FSI spring model. 0 = Do not use FSI spring model. 1 = Use the FSI spring model (<i>spring.in</i> required).
<i>stiction_flag</i>	Flag for FSI stiction model. 0 = Do not use FSI stiction model. 1 = Use the FSI stiction model (<i>stiction.in</i> required).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

FSI Beam Variable Loading: **beam_loading.in**

To specify a variable loading for an FSI beam, specify a file name (e.g., *beam_loading.in*) for [*fsi.in*](#) > *beam_objects* > *object* > *external_load_value* or *initial_load_value* and include that file in your case setup.

```
3  NUMBER_OF_POINTS
0    0.0
1    0.005
2    0.01
```

Figure 25.128: An example *beam_loading.in* file.

Table 25.104: Description of *beam_loading.in*.

Column	Description
1	Axial location of this beam station.
2	Load at this beam station.

FSI Beam Limit: **beam_limit.in**

To specify a deflection limit for an FSI beam, specify a file name (e.g., *beam_limit.in*) for [*fsi.in*](#) > *beam_objects* > *object* > *constraints* > *constraint* and include that file in your case setup.

The first row of *beam_limit.in* specifies the total number of points used to define the limits for the beam. For each point, include a row with the format shown below.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

```
3    NUM_CONSTRAIN_POINTS
0.0    0.0
0.10   0.005
0.20   0.01
```

Figure 25.129: An example *beam_limit.in* file.

Table 25.105: Description of *beam_limit.in*.

Column	Description
1	Axial location of this beam station.
2	Beam deflection constraint at this location.

FSI Beam Initial Deflection: *initial_deflection.in*

To specify an initial deflection for an FSI beam, set [*fsi.in*](#) > *beam_objects* > *object* > *initial_deflection_flag* = 1 and include an *initial_deflection.in* file in your case setup.

The first row of *initial_deflection.in* specifies the total number of points used to define the initial deflection of the beam. For each point, include a row with the format shown below.

```
10 number_of_points
0.000000  0.000000
0.001000  0.000000
0.002000  0.000001
0.003000  0.000002
0.004000  0.000004
0.005000  0.000006
0.006000  0.000009
0.007000  0.000012
0.008000  0.000016
0.009000  0.000020
```

Figure 25.130: An example *initial_deflection.in* file.

Table 25.106: Description of *initial_deflection.in*.

Column	Description
1	Axial location on the beam (m).
2	Initial displacement in a direction normal to the beam at this location (m).

FSI Beam Properties: *beam_profile.in*

To specify variable properties for an FSI beam, set [*fsi.in*](#) > *beam_objects* > *object* > *variable_properties_flag* = 1, specify a file name (e.g., *beam_profile.in*) for [*fsi.in*](#) > *beam_objects* > *object* > *property_file*, and include the beam properties file in your case setup.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

6 NUMBER_OF_POINTS						
0.000E+00	5.25E-03	2.40E-04	1.26E-06	7.90E+03	2.10E+11	6.04E-15
6.150E-04	5.07E-03	2.40E-04	1.21E-06	7.90E+03	2.10E+11	5.84E-15
1.230E-03	4.90E-03	2.40E-04	1.17E-06	7.90E+03	2.10E+11	5.64E-15
1.844E-03	4.73E-03	2.40E-04	1.13E-06	7.90E+03	2.10E+11	5.44E-15
2.463E-03	4.63E-03	2.40E-04	1.11E-06	7.90E+03	2.10E+11	5.33E-15
3.082E-03	4.53E-03	2.40E-04	1.08E-06	7.90E+03	2.10E+11	5.21E-15

Figure 25.131: An example *beam_profile.in* file.

Table 25.107: Description of *beam_profile.in*.

Column	Header (units)	Description
1	x (m)	Axial location of this beam station.
2	B(x) (m)	Breadth of the beam at this station.
3	H(x) (m)	Height of the beam at this station.
4	A(x) (m^2)	Area of the beam at this station.
5	RHO(x) (kg/m^3)	Density of the beam at this station.
6	E_m(x) (Pa)	Young's modulus of the beam at this station.
7	I_m(x) (m^2)	Moment of inertia of the beam at this station.

FSI Stiction: *stiction.in*

To activate the FSI stiction model, set [*fsi.in*](#) > *stiction_flag* = 1 and include a *stiction.in* file in your case setup.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

```
version: 3.1
---

- stiction:
    stiction_name: stiction1
    fsi_object_name: suction_valve
    stiction_model: DEFAULT
    displacement_type: 0
    stiction_distance: 1e-05
    stiction_force:
        magnitude: 1.0
        direction: [1.0, 0.0, 0.0]
    stiction_moment:
        magnitude: 0.0
        direction: [1.0, 0.0, 0.0]

- stiction:
    fsi_object_name: discharge_valve
    stiction_model: DEFAULT
    displacement_type: 1
    stiction_distance: 1e-05
    stiction_force:
        magnitude: 1.0
        direction: [1.0, 0.0, 0.0]
    stiction_moment:
        magnitude: 0.0
        direction: [1.0, 0.0, 0.0]

- stiction:
    fsi_object_name: discharge_valve
    stiction_model: DEFAULT
    displacement_type: 1
    displacement_type: 1
    stiction_distance: 1e-03
    stiction_force:
        magnitude: 0.0
        direction: [0.0, -1.0, 0.0]
    stiction_moment:
        magnitude: 32.6
        direction: [1.0, 0.0, 0.0]
```

Figure 25.132: An example *stiction.in* file.

Table 25.108: Format of *stiction.in*.

Parameter	Description
- <i>stiction</i>	This settings block specifies the stiction configuration for an FSI object. Repeat this block for each FSI object.
<i>stiction_name</i> *	Name of the stiction object.
<i>fsi_object_name</i>	A string used to identify an FSI object subject to this stiction force, corresponding to FSI object name in fsi.in .
<i>stiction_model</i>	Stiction model type. Currently only the DEFAULT model is available.
<i>displacement_type</i>	0 = Linear displacement calculated along the <i>stiction.in > stiction_force > direction</i> , 1 = Angular displacement calculated about the <i>stiction.in > stiction_moment > direction</i> .
<i>stiction_distance</i>	Maximum distance over which stiction force/moment acts (in <i>meters</i> if <i>displacement_type</i> = 0 or in <i>degrees</i> if <i>displacement_type</i> = 1), defined positive.
<i>stiction_force</i>	Required if <i>displacement_type</i> = 0.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
<i>magnitude</i>	Stiction force (N), defined positive.
<i>direction</i>	Direction of stiction force.
<i>stiction_moment</i>	Required if <i>displacement_type</i> = 1.
<i>magnitude</i>	Stiction moment (N-m). For <i>displacement_type</i> = 1, defined positive.
<i>direction</i>	Direction of stiction moment.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Variable Spring Constant: `spring_constant.in`

For an FSI simulation that includes a spring with variable spring constant, specify a file name (e.g., `spring_constant.in`) for `spring.in` > `spring` > `spring_constant` and include that file in your case setup. Figure 25.133 below shows an example `spring_constant.in` file. The first row contains two columns headings: *displacement* and *force*. Subsequent rows should contain spring force values (in N) as a function of displacement (m). The *displacement* column must contain monotonically increasing displacement values.

```
displacement    force
-0.00511      2.51E+00
-0.00501      2.42E+00
-0.00494      2.12E+00
-0.00478      1.71E+00
-0.00461      1.28E+00
-0.00428      1.01E+00
-0.00410      9.33E-01
-0.00372      6.55E-01
-0.00218      5.10E-01
-0.00146      3.48E-01
0              0.00E+00
.
.
```

Figure 25.133: An example of a variable spring constant file (e.g., `spring_constant.in`).

Table 25.109: Format of `spring_constant.in`.

Column	Description	Units
1	Displacement	m
2	Force	N

FSI Spring: `spring.in`

To activate the FSI spring model, set `fsi.in` > `spring_flag` = 1 and include a `spring.in` file in your case setup.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

```
version: 3.1
---

- spring:
    spring_name: spring1
    fsi_object_name: suction_valve
    spring_constant: 2000.0
    damping_constant: 0.0
    initial_deformation: 0.0005
    detached_flag: 0
    detachment_distance: 0.0
    end_position_fsi: [-0.0219 , 0.0, 0.00606]
    end_position_fixed: [-0.01838, 0.0, 0.00606]
    zero_angle_direction: [0.0, 0.0, 1.0]
    subsprings:
        - subspring:
            subspring_radius: 0.015
            subspring_angle: 0.0
        - subspring:
            subspring_radius: 0.015
            subspring_angle: 90.0
        - subspring:
            subspring_radius: 0.015
            subspring_angle: 90.0
        - subspring:
            subspring_radius: 0.015
            subspring_angle: 90.0

    - spring:
        fsi_object_name: discharge_valve
        spring_constant: spring_constant.in
        damping_constant: 0.0
        initial_deformation: 0.0005
        detached_flag: 1
        detachment_distance: 1e-04
        end_position_fsi: [0.01952, 0.0, 0.00606]
        end_position_fixed: [0.02304, 0.0, 0.00606]
        zero_angle_direction: [0.0, 0.0, 1.0]
```

Figure 25.134: An example *spring.in* file.

Table 25.110: Format of *spring.in*.

Parameter	Description
- <i>spring</i>	This settings block specifies the configuration of a spring. Repeat this sub-block through <i>zero_angle_direction</i> (or optionally through <i>subspring_angle</i>) for each spring.
<i>spring_name</i> *	Name of the spring object.
<i>fsi_object_name</i>	A string used to identify an FSI spring object, corresponding to FSI object name in <i>fsi.in</i> .
<i>spring_constant</i>	Spring constant of the spring object (N/m), defined greater than zero. Specify a file name for a case with a variable spring constant .
<i>damping_constant</i>	Damping constant of the spring object ($N\cdot s/m$), defined greater than zero.
<i>initial_deformation</i>	Initial preload deformation of the spring object (m).
<i>detached_flag</i>	Flag for spring attachment. 0 = Spring attached at both ends.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Parameter	Description
	1 = Spring attached from one end.
<i>detachment_distance</i>	Distance from detached end of spring to FSI object. Used only when <i>detached_flag</i> = 1.
<i>end_position_fsi</i>	Coordinates of spring end position on FSI object.
<i>end_position_fixed</i>	Coordinates of spring end position on fixed object.
<i>zero_angle_direction</i>	Unit vector pointing along zero axis for sub-spring coordinate system.
<i>subsprings*</i>	Specify sub-spring objects to represent the fact that the physical spring has a finite coil diameter.
- <i>subspring</i>	This settings sub-block specifies the configuration of a sub-spring. Repeat this sub-block for each sub-spring.
<i>subspring_radius</i>	Sub-spring radius (m).
<i>subspring_angle</i>	Sub-spring radial angle (deg).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

FSI Forces: *fsi_force.in*

To set up a temporally varying applied force for a rigid-body FSI object, specify a file name (e.g., *fsi_force.in*) for [*fsi.in*](#) > *rigid_fsi_objects* > *object* > *applied_force* and include that file in your case setup.

Figure 25.135 shows an excerpt of an example *fsi_force.in* file. The first line of the header contains the keyword TEMPORAL. The second line specifies the temporal type (CYCLIC or SEQUENTIAL) and the period (if CYCLIC). The third line specifies the column headers for the subsequent rows.

```
TEMPORAL
SEQUENTIAL
second    fx      fy      fz
0          0.0     0.0     0
0.000001  0.0     0.0     0
0.000002  0.0     0.0     0
0.000003  0.0     0.0     0
0.000004  0.0     0.0     0
0.000005  0.0     0.0     0
0.000006  0.0     0.0     0
0.000007  0.0     0.0     0
0.000008  0.0     0.0     0
0.000009  0.0     0.0     0
0.000010  0.0     0.0     0
0.000011  0.0     0.0     0.000982848
0.000012  0.0     0.0     0.001281136
0.000013  0.0     0.0     0.001618892
0.000014  0.0     0.0     0.001996108
0.000015  0.0     0.0     0.002412784
.
.
```

Figure 25.135: An excerpt of an example *fsi_force.in* file.

Chapter 25: Input and Data Files

Physical Models Fluid-Structure Interaction Modeling

Table 25.111: Format of rows four+ in *fsi_force.in*.

Column	Description
1	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Force (N) in the x direction.
3	Force (N) in the y direction.
4	Force (N) in the z direction.

Surface Chemistry Modeling

This section describes input files that contain options for surface chemistry modeling in CONVERGE.

Surface Chemistry: *surface_chemistry.in*

To activate CONVERGE's surface chemistry model, set [combust.in](#) > *surface_chemistry_flag* = 1 and include a *surface_chemistry.in* file in your case setup.

You can simulate surface chemistry on one of two types of surfaces: a non-moving WALL boundary that is specified in [boundary.in](#) or a porous medium that is specified as a region in [initialize.in](#). The *surface_chemistry.in* file must contain a block of parameters for each boundary or porous region on which surface chemistry modeling will occur. For a porous region with surface chemistry, you must also set up a REGION source in [source.in](#).

```
version: 3.1
---

surface_chemistry_control:
    mechanism_filename: surface_mech.dat
    thermodynamic_filename: surface_therm.dat
    region_flag: 1
    temporal_type: CYCLIC
    period: 720
    start_time: 0
    end_time: 120
    coupled_solver_flag: 0
    temp_cutoff: 300
    analyt_jac: 0
    rel_tol: 1e-08
    abs_tol: 1e-18
    reaction_multiplier: 1.0
    volume_units_flag: 1
    boundary_cell_temp_flag: 1
boundary_control:
    - boundary:
        boundary_id: 11
        catalytic_species:
            ZNH3: 0.00100000
            Z: 0.99900000
        surface_area_to_volume_ratio: 100
        catalytic_geometric_area_ratio: 1.0
        effectiveness_factor_control:
            effectiveness_model: OFF
            diffusion_model: MIXED
            knudsen_constant: 1.0
            washcoat_control:
                species: O2
```

Chapter 25: Input and Data Files

Physical Models Surface Chemistry Modeling

```
porosity: 0.279
tortuosity: 3.0
thickness: 0.0001
pore_diameter: 1.229e-08
- boundary:
  boundary_id: 12
  catalytic_species:
    ZNH3: 0.00100000
    Z: 0.99900000
  surface_area_to_volume_ratio: 100
  catalytic_geometric_area_ratio: 1.0
  effectiveness_factor_control:
    effectiveness_model: THIELE
    diffusion_model: MOLECULAR
    knudsen_constant: 1.0
    washcoat_control:
      species: O2
      porosity: 0.279
      tortuosity: 3.0
      thickness: 0.0001
      pore_diameter: 1.229e-08
- boundary:
  boundary_id: 18
  catalytic_species:
    ZNH3: 0.00100000
    Z: 0.99900000
  surface_area_to_volume_ratio: 100
  catalytic_geometric_area_ratio: 5.0
  effectiveness_factor_control:
    effectiveness_model: OFF
```

Figure 25.136: An example *surface_chemistry.in* file with *boundary_control*.

```
version: 3.1
---

surface_chemistry_control:
  mechanism_filename: surface_mech.dat
  thermodynamic_filename: surface_therm.dat
  region_flag: 1
  temporal_type: CYCLIC
  period: 720
  start_time: 0
  end_time: 120
  coupled_solver_flag: 0
  temp_cutoff: 300
  analyt_jac: 0
  rel_tol: 1e-08
  abs_tol: 1e-18
  reaction_multiplier: 1.0
  volume_units_flag: 1
  boundary_cell_temp_flag: 1
porous_region_control:
  - region:
    region_id: 1
    catalytic_species:
      Z: 0.99997500
      ZNH3: 0.00002500
    surface_area_to_volume_ratio: 100
    catalytic_geometric_area_ratio: 1.0
    effectiveness_factor_control:
      effectiveness_model: OFF
      diffusion_model: KNUDSEN
      knudsen_constant: 1.0
      washcoat_control:
        species: O2
        porosity: 0.279
        tortuosity: 3.0
        thickness: 0.0001
```

Chapter 25: Input and Data Files

Physical Models Surface Chemistry Modeling

```
pore_diameter: 1.229e-08
- region:
  region_id: 2
  catalytic_species:
    Z: 0.99997500
    ZNH3: 0.00002500
  surface_area_to_volume_ratio: 100
  catalytic_geometric_area_ratio: 1.0
  effectiveness_factor_control:
    effectiveness_model: THIELE
    diffusion_model: KNUDSEN
    knudsen_constant: 1.0
    washcoat_control:
      species: O2
      porosity: 0.279
      tortuosity: 3.0
      thickness: 0.0001
    pore_diameter: 1.229e-08
- region:
  region_id: 5
  catalytic_species:
    Z: 0.99997500
    ZNH3: 0.00002500
  surface_area_to_volume_ratio: 100
  catalytic_geometric_area_ratio: 1.0
  effectiveness_factor_control:
    effectiveness_model: OFF
```

Figure 25.137: An example *surface_chemistry.in* file with *porous_region_control*.

Table 25.112: Format of *surface_chemistry.in*.

Parameter	Description	Typical value
<i>surface_chemistry_control</i>		
<i>mechanism_filename</i>	The name of the surface chemical mechanism file.	
<i>thermodynamic_filename</i>	The name of the surface thermodynamic properties file.	
<i>region_flag</i>	0 = Surface chemistry model is not region dependent and uses temporal type, <i>start_time</i> and <i>end_time</i> to start and stop the surface chemistry calculations in the specified regions, 1 = Surface chemistry model is region dependent and <i>surface_chemistry_region.in</i> must be saved in your case setup.	
<i>temporal_type</i>	SEQUENTIAL, PERMANENT or CYCLIC. For CYCLIC, supply the period after the <i>period</i> parameter.	
<i>period</i>	The CYCLIC period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Surface chemistry modeling start time. Used only if <i>temporal_type</i> is not PERMANENT.	

Chapter 25: Input and Data Files

Physical Models Surface Chemistry Modeling

Parameter	Description	Typical value
<i>end_time</i>	Surface chemistry modeling end time. Used only if <i>temporal_type</i> is not PERMANENT.	
<i>coupled_solver_flag</i>	0 = Surface chemistry gas and surface species are decoupled during the solution procedure, 1 = Surface chemistry gas and surface species are coupled during the solution procedure,	
<i>temp_cutoff</i>	Minimum cell or boundary temperature (K) for surface chemistry activation.	300
<i>analyt_jac</i>	0 = Solve Jacobian matrix numerically, 1 = Solve Jacobian matrix analytically.	
<i>rel_tol</i>	Relative iteration error for each species in the surface chemistry solver.	1e-4
<i>abs_tol</i>	Absolute iteration error for each species in the surface chemistry solver.	1e-14
<i>reaction_multiplier</i>	Reaction rate multiplier for each surface (site) species.	Default: 1.0
<i>volume_units_flag</i>	Switch units from area (default, = 0) to volume based (= 1), porous surface chemistry only.	Default: 0
<i>boundary_cell_temp_flag</i>	For temperature in boundary surface chemistry: 0 = Use boundary temperature, 1 = Use temperature of the cell attached to the boundary.	Default: 1
<i>boundary_control</i> or <i>porous_region_control</i>	This settings block specifies the configuration of the boundary or porous region. You can specify one <i>boundary_control</i> settings block or one <i>porous_region_control</i> settings block but not both.	
- <i>boundary</i> or - <i>region</i>	This settings sub-block specifies the configuration of a boundary or porous region on which surface chemistry is modeled. Repeat this sub-block through <i>pore_diameter</i> for each boundary or porous region.	
<i>boundary_id</i> or <i>region_id</i>	Boundary or region ID for a boundary or porous region included in the surface chemistry model.	
<i>catalytic_species</i>	Catalytic species (open sites).	
< <i>species name</i> >	Catalytic species name(s) and composition ratio(s). The species listed here must be included in <i>surface mech.dat</i> . Note that the composition ratios must sum to one. CONVERGE will not normalize the ratios.	

Chapter 25: Input and Data Files

Physical Models Surface Chemistry Modeling

Parameter	Description	Typical value
<code>surface_area_to_volume_ratio</code>	Geometric surface area per unit volume (m^{-1}). This value is used only in porous media.	Default: 1.0
<code>catalytic_gemoetric_area_ratio</code>	Ratio of active catalytic surface area to geometric surface area.	
<code>effectiveness_factor_control</code>		
<code>effectiveness_model</code>	<code>OFF</code> = Off (Do not use washcoat parameters), <code>THIELE</code> = Use Thiele modulus approach (CONVERGE calculates the effectiveness factor only for cells in a particular region or boundary in that region or boundary, and all washcoat parameters must be greater than zero).	Default: OFF
<code>diffusion_model</code>	<code>KNUDSEN</code> = Knudsen diffusion model, <code>MOLECULAR</code> = Molecular diffusion model, <code>MIXED</code> = Mixed diffusion model. When <code>diffusion_model</code> = <code>KNUDSEN</code> or <code>MOLECULAR</code> , you must include transport.dat and set <code>inputs.in > species_diffusion_model</code> = 1.	
<code>knudsen_constant</code>	Knudsen diffusion constant.	4.0
<code>washcoat_control</code>		
<code>species</code>	The single gas phase species that reacts with the catalyst and determines diffusion into the washcoat used in Thiele Modulus calculation. The gas phase species listed here must be listed in mech.dat .	Default: O2
<code>porosity</code>	Fractional volume of voids in washcoat used in Thiele Modulus calculation.	Default: 0
<code>tortuosity</code>	Quantitative deviation of the washcoat topology from a flat surface used in Thiele Modulus calculation.	Default: 0
<code>thickness</code>	Thickness of washcoat (m) used in Thiele Modulus calculation.	Default: 1e-3
<code>pore_diameter</code>	Pore diameter of washcoat (m) used in Thiele Modulus calculation.	Default: 1e-9

Surface Species Reaction Data: `surface_mech.dat`

The `surface_mech.dat` file contains surface species reaction information in the CHEMKIN format. Note that the names in `surface_mech.dat` must be consistent with the names in `surface_therm.dat`. CONVERGE does not list elements in `surface_mech.dat`. List all species related to surface chemistry, such as the empty site species, in the [reaction mechanism file](#).

Chapter 25: Input and Data Files

Physical Models Surface Chemistry Modeling

```
!MATERIAL CATALYST
SITE/1/SDEN/2.7063E-9/
C3H6_S/2/ PT_S/1/ H_S/1/ O_S/1/ OH_S/1/
H2O_S/1/ CH3_S/1/ CH2_S/1/ CH_S/1/
C_S/1/ CO_S/1/ CO2_S/1/
END

REACTIONS   JOULES/MOLE
H2 + 2PT_S => 2H_S           FORD/PT_S 1/
2H_S => H2 + 2PT_S           COV/H_S
O2 + 2PT_S => 2O_S           DUPLICATE
O2 + 2PT_S => 2O_S           STICK MOTZ
H2O + PT_S => H2O_S          DUPLICATE
                           STICK
END
```

Excerpt of a *surface_mech.dat* file.

As shown previously in Figure 25.137 above, *surface_mech.dat* lists the surface density and species before the reactions. In the first row, SITE/1/SDEN/2.7063E-9/, the last number is the surface site density (*mol/cm²*). Note the format of this line, including the / symbols. This line cannot contain spaces. Below the first row, list the surface species. The surface species names cannot contain symbols [e.g., # or ()].

By default, the site occupancy for each surface species listed in *surface_mech.dat* is one. If the species has a site occupancy of two or more, include / after the name of the species, followed by the site occupancy for that species and then /. Figure 25.137 above shows an example, C3H6_S/2/, of a species with a site occupancy of two. Specify the end of the list of species at the site by using the keyword *END*.

The next section specifies the reactions at the surface sites. The first line contains the keyword *REACTIONS* followed by the units (here *J/mol*). You can activate surface chemistry options by including optional keywords after the units on this line:

- The keyword *SURF_SP_UNIT* will force the rate calculation to use coverage (a percentage, *unitless*) rather than surface density (*mol/cm²*).
- The keyword *SURF_GAS_UNIT* tells CONVERGE to expect SI units for the gaseous species (otherwise, [it conforms to the CHEMKIN format](#)).
- The keyword *MWON* tells CONVERGE to apply the Motz-Wise correction to every *STICK* reaction (*i.e.*, every reaction labeled with the *STICK* or *STICK MOTZ* keyword).

You can specify several keywords for individual reactions: *COV* for coverage-dependent, *STICK* for sticking probability, and *STICK MOTZ* for sticking probability with the Motz-Wise correction, as well as *REV*, *FORD*, *USER*, and *DUP*, which are described in the subsections of [Species Data and Reaction Mechanism: mech.dat](#).

For coverage-dependent reactions, specify the species name after *COV/*, followed by the three coverage parameters (χ_{ji} , μ_{ji} , and ε_{ji}) in order, and then close the statement with /, as shown above in Figure 25.137.

Surface Species Thermodynamic Data: `surface_therm.dat`

The `surface_therm.dat` file contains thermodynamic information for surface species and place holders for empty sites. The file format is similar to that of the [thermodynamic data file](#).

```
THERMO
300.0 1000.0 3000.0
OH_S 92491 O 1 H 1 Pt 1 I 300.00 3000.00 1000.00
 0.18249973E+01 0.32501565E-02 -0.31197541E-06 -0.34603206E-09 0.79171472E-13
 -0.26685492E+05 -0.12280891E+02 -0.20340881E+01 0.93662683E-02 0.66275214E-06
 -0.52074887E-08 0.17088735E-11 -0.25319949E+05 0.89863186E+01
Pt Pt 1 S 300.0 3000.0 1000.0
 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
.
.
END
```

Figure 25.138: Excerpt of a `surface_therm.dat` file.

Note that `surface_therm.dat` must be in [NASA 7](#) format. The first and second rows are the same as those previously described for [NASA 7](#) format. The third row contains the surface species (e.g., OH_S in Figure 25.138), followed by a date (e.g., 92491) and the atomic symbols and number of each atom type. Next, the catalytic species and number of open sites (e.g., Pt 1) are listed. Then, species are denoted with either *I* or *S*. The surface species that are formed during the chemistry on the surface are denoted with *I*, and catalytic species, empty sites, or, if applicable, extra species found in the DETCHEM CHEMKIN format (e.g., Pt and Rh) are denoted with *S*. For species designated with *I*, the last element listed before the *I* must be the catalytic element that the species adheres to on the surface. The number of sites each empty site occupies on the surface is listed in place of an element for that species. If the species adheres to a molecule on the surface and not an element, the *I* designation cannot be used and this space must be left empty (*i.e.*, list the temperatures immediately after atomic symbols and number of each atom type). The gas species are listed in the [reaction mechanism file](#) (e.g., `mech.dat`).

Region-Dependent Surface Chemistry: `surface_chemistry_region.in`

The `surface_chemistry_region.in` file designates the start and end time that CONVERGE uses when the surface chemistry model is region-dependent. CONVERGE reads this file when the [`surface_chemistry.in > surfchem_region_flag = 1`](#).

Chapter 25: Input and Data Files

Physical Models Surface Chemistry Modeling

```
version: 3.1
---

surface_chemistry_region_control:
  - region:
      region_id: 0
      temporal_type: CYCLIC
      period: 720
      start_time: 0
      end_time: 100.0
  - region:
      region_id: 1
      temporal_type: SEQUENTIAL
      start_time: 0
      end_time: 100
```

Figure 25.139: Sample of a *surface_chemistry_region.in* file.

Table 25.113: Format of *surface_chemistry_region.in*.

Parameter	Description
<i>surface_chemistry_region_control</i>	
- <i>region</i>	This settings sub-block specifies the configuration of a surface chemistry region. Repeat this sub-block for each region.
<i>region_id\$</i>	The region in which surface chemistry calculations will be performed. The value given here must also be a <i>region_id</i> in <i>initialize.in</i> .
<i>temporal_type</i>	Temporal type: SEQUENTIAL, CYCLIC, or PERMANENT.
<i>period\$</i>	The CYCLIC period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>start_time\$</i>	Surface chemistry model start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>end_time\$</i>	Surface chemistry model end time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).

§ CONVERGE reads this parameter at each time-step when [*inputs.in*](#) > *simulation_control* > *reread_input* = 1.

Nucleate Boiling: *nucleate_boiling.in*

To activate the [nucleate boiling model](#), set [*inputs.in*](#) > *feature_control* > *nucleate_boiling_flag* = 1 and include a *nucleate_boiling.in* file in your case setup. This file contains parameters for the Rohsenow correlation ([Rohsenow, 1952](#)) used in the nucleate boiling model.

Chapter 25: Input and Data Files

Physical Models Nucleate Boiling: nucleate_boiling.in

```
version: 3.1
---
saturation_temp: 381.15
vapor_density: 0.804
c_sf: 0.013
nb_exponent: 0.0
```

Figure 25.140: An example *nucleate_boiling.in* file.

Table 25.114: Format of *nucleate_boiling.in*.

Parameter	Description
<i>saturation_temp</i>	Saturation temperature (K) of the liquid that experiences nucleate boiling.
<i>vapor_density</i>	Density of the vapor phase of the liquid that experiences nucleate boiling.
<i>c_sf</i>	Constant that depends on the surface-fluid interface.
<i>nb_exponent</i>	Nucleate boiling exponent given by <i>m</i> in the Rohsenow correlation.

Urea: urea.in

To activate [urea modeling](#), set *inputs.in* > *feature_control* > *urea_flag* = 1 and include a *urea.in* file in your case setup.

To invoke the [urea/water injection option](#), set *urea.in* > *urea_model* = 1. To invoke the [molten solid approach for urea decomposition](#), set *urea.in* > *urea_model* = 2. To invoke the [detailed decomposition model](#), set *urea.in* > *urea_model* = 3.

```
version: 3.1
---

urea_model: 3
urea_hdcmp: 3088000.0
molten_solid_A_E: [4.20E-001, 6.90E+008]
detailed_decomposition_A_E:
  R1: [8.450E+006, 8.40E+004]
  R2: [1.50E+002, 4.00E+004]
  R3: [6.20E+011, 1.00E+004]
  R4: [7.00E+032, 1.15E+005]
  R5: [1.50E+024, 2.50E+005]
  R6: [2.50E+036, 1.50E+005]
  R7: [3.00E+018, 2.60E+005]
  R8: [3.10E+023, 3.50E+004]
  R9: [6.00E+014, 2.20E+005]
  R10: [1.80E+010, 8.40E+004]
  R11: [5.00E+027, 5.90E+004]
  R12: [3.50E+032, 1.15E+005]
  R1_clip: [1.16E+046, 4.89E+005]
  R2_clip: [4.00E+035, 3.86E+005]
  R3_clip: [6.35E+034, 2.48E+005]
scaling_factor_A: 2.0
scaling_factor_E: 0.95
```

Figure 25.141: An example *urea.in* file.

Table 25.115: Format of *urea.in*.

Chapter 25: Input and Data Files

Physical Models Urea: urea.in

Parameter	Description	Typical value
<i>urea_model</i>	1 = Urea-water solution depletion calculation with the multi-component model, 2 = Urea-water solution depletion calculation with the molten solid approach 3 = Urea-water solution depletion calculation with the detailed decomposition approach.	
<i>urea_hdcmp</i>	Enthalpy change due to urea decomposition (J/kg). Used only when <i>urea_model</i> = 2 or 3.	3.088e6
<i>molten_solid_A_E</i>	Pre-exponential ($kg/m\cdot s$) and activation energy ($J/kmol$) for the Arrhenius correlation for $\text{urea} \rightarrow \text{HNCO} + \text{NH}_3$. Used only when <i>urea_model</i> = 2. Note that the units are $J/kmol$ for the activation energy in this file.	4.2e-1 and 6.9e7, respectively
<i>detailed_decomposition_A_E</i>	All of the parameters in this section are used only when <i>urea_model</i> = 3.	
R1	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (1) (see Chapter 14 for list of reactions).	8.450e6, 8.40e4
R2	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (2).	1.50e2, 4.00e4
R3	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (3).	6.20e11, 1.00e4
R4	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (4).	7.00e32, 1.15e5
R5	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (5).	1.50e24, 2.50e5
R6	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (6).	2.50e36, 1.50e5
R7	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (7).	3.00e18, 2.60e5
R8	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (8).	3.10e23, 3.50e4
R9	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (9).	6.00e14, 2.20e5
R10	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (10).	1.80e10, 8.40e4
R11	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (11).	5.00e27, 5.90e4
R12	Arrhenius pre-exponential (s^{-1}) and activation energy (J/mol) for reaction (12).	3.50e32, 1.15e5

Chapter 25: Input and Data Files

Physical Models Urea: urea.in

Parameter	Description	Typical value
<i>R1_clip</i>	Altered reaction coefficients for reaction (1) clipping equation. We recommend not altering this value.	1.16e46, 4.89e5
<i>R2_clip</i>	Altered reaction coefficients for reaction (2) clipping equation. We recommend not altering this value.	4.00e35, 3.86e5
<i>R3_clip</i>	Altered reaction coefficients for reaction (3) clipping equation. We recommend not altering this value.	6.35e34, 2.48e5
<i>scaling_factor_A</i>	Global Arrhenius prefactor scaling factor. Used only when <i>urea.in > urea_model</i> = 3.	2.0
<i>scaling_factor_E</i>	Global Arrhenius activation energy scaling factor. Used only when <i>urea.in > urea_model</i> = 3.	0.95

Aeroacoustics: acoustics.in

To activate [aeroacoustic modeling](#), set *inputs.in > feature_control > acoustics_flag* = 1 and include an *acoustics.in* file in your case setup.

```
version: 3.1
---

acoustic_models: [INTEGRAL_FWH, BROADBAND_PROUTMAN, BROADBAND_CURLE,
LIGHTHILL_STRESS_TENSOR]
farfield_properties:
  pressure: 101325.0
  density: 1.325
  sound_speed: 342.0
fwh_acoustic_stream_id: 0
fwh_sources:
  source_surfaces:
    surface_type: PERMEABLE
    boundary_ids: [3, 7]
    bounded_region_id: 0
  source_volumes:
    region_ids: [1]
fwh_receivers:
  - receiver:
    id: 1
    name: receiver1
    xx: [1.850, 0.00, 0.00]
    uu: [0.00, 0.00, 0.00]
  - receiver:
    id: 2
    name: receiver2
    xx: [0.235, 0.00, 0.00]
    uu: [0.00, 0.00, 0.00]
max_spectral_freq: 50000
ref_pressure: 2.0E-5
ref_acoustic_power: 1.0E-12
decibel_units_flag: 0
```

Figure 25.142: An example *acoustics.in* file.

Table 25.116: Format of *acoustics.in*.

Chapter 25: Input and Data Files

Physical Models Aeroacoustics: acoustics.in

Parameter	Description	Typical value
<i>acoustic_models</i>	List of aeroacoustic models to activate for this simulation. These options are independent and can be specified in any combination. <i>INTEGRAL_FWH</i> = Ffowcs Williams-Hawkins (FW-H) far-field tonal model, <i>BROADBAND_PROUDMAN</i> = Proudman near-field broadband model, <i>BROADBAND_CURLE</i> = Curle near-field broadband model, <i>LIGHTHILL_STRESS_TENSOR</i> = Write the Lighthill stress tensor to <i>post.h5</i> .	
<i>farfield_properties</i>		
<i>pressure</i>	Far-field pressure (<i>Pa</i>) for the FW-H model.	101,325
<i>density</i>	Far-field density (<i>kg/m³</i>) for the FW-H model.	1.325
<i>sound_speed</i>	Sound speed (<i>m/s</i>) for the FW-H model.	342.0
<i>fwh_acoustic_stream_id</i>	Stream ID for the FW-H model.	
<i>fwh_sources</i>		
<i>source_surfaces</i>		
<i>surface_type</i>	<i>PERMEABLE</i> = Permeable FW-H surface, <i>IMPERMEABLE</i> = Impermeable FW-H surface.	
<i>boundary_ids</i>	List of boundary IDs for the FW-H surface. For PERMEABLE surfaces, these boundaries should be flow-through INTERFACE. For IMPERMEABLE surfaces, these boundaries should be WALL type.	
<i>bounded_region_id</i>	For PERMEABLE FW-H surfaces, the region ID bounded by the flow-through INTERFACE boundaries.	
<i>source_volumes</i>		
<i>region_ids</i>	For FW-H volume sources, the region IDs of the volume sources.	
<i>fwh_receivers</i>		
- <i>receiver</i>	Repeat this sub-block for each FW-H receiver. CONVERGE writes output for this receiver to <i>acoustic_receiver <ID>_stream <ID>_pressure.out</i> .	
<i>id</i>	FW-H receiver ID.	
<i>name</i>	FW-H receiver name.	
<i>xx</i>	The x, y, and z location (<i>m</i>) of the receiver.	
<i>uu</i>	The u, v, and w components (<i>m/s</i>) of the receiver's motion. This velocity is applied within the FW-H	

Chapter 25: Input and Data Files

Physical Models Aeroacoustics: acoustics.in

Parameter	Description	Typical value
	calculation, but the source does not actually move during the simulation.	
<i>max_spectral_freq</i>	For the FW-H model, the maximum frequency (Hz) considered.	5000
<i>ref_pressure</i>	For the FW-H model, the reference acoustic pressure (Pa).	2e-5
<i>ref_acoustic_power</i>	For the Proudman and Curle models, the reference acoustic power (W).	1e-12
<i>decibel_units_flag</i>	For the Proudman and Curle models, 0 = Print units in W/m^2 or W/m^3 , 1 = Print units in decibels.	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Erosion: erosion.in

To activate erosion modeling, set *inputs.in* > *feature_control* > *erosion_flag* = 1 and include an *erosion.in* file in your case setup.

```
version: 3.1
---

erosion_model_groups:
  - erosion_model_group:
      erosion_model: generic_model
      boundary_id: [7, 10]
      material_density: 8500
      erosion_start_time: 0.5
      erosion_end_time: 5.5
      model_group_override:
        genm_erosion_constant: 1.72e-7
        genm_angle_factor: angle_factor_temp.dat
  - erosion_model_group:
      erosion_model: generalized_ecrc_model
      boundary_id: [1, 4]
      material_density: 9500
      erosion_start_time: 0.8
      erosion_end_time: 4.5

erosion_parcel_shape_factors:
  sand: 0.2
  crude: 0.8

erosion_model_parameters:
  generic_model:
    genm_erosion_constant: 1.22e-7
    genm_velocity_exponent: 2.41
    genm_angle_factor: angle_factor.dat
  arabnejad_model:
    aram_velocity_exponent: 2.41
    aram_c_1: 4.58e-08
    aram_c_2: 5.56e-08
    aram_k: 0.4
    aram_u_tsh: 5.8
  zhang_model:
    zhdm_brinnel_hardness: 178.917
    zhdm_brinnel_exponent: -0.59
    zhdm_velocity_exponent: 2.41
    zhdm_erosion_constant: 2.17e-07
```

Chapter 25: Input and Data Files

Physical Models Erosion: erosion.in

```
zham_shape_factor_parameters: [1.4234, -6.3283, 10.9327, -10.1068, 5.3983]
generalized_ecrc_model:
    ecrcm_velocity_exponent : 2.41
    ecrcm_c_1: 4.58e-08
    ecrcm_c_2: 5.56e-08
    ecrcm_k: 0.4
    ecrcm_u_tsh: 3.5
    ecrcm_c_st: 0.24
    ecrcm_exp_st: -0.503
oka_model:
    okam_k_p: 65
    okam_Hv: 0.5
    okam_k1: -0.12
    okam_k3: 0.19
    okam_s1: 0.71
    okam_s2: 2.4
    okam_q1: 0.14
    okam_q2: -0.94
    okam_v_p: 104
    okam_D_p: 3.26e-6
finnie_model:
    finm_velocity_exponent: 2
    finm_p: 2
    finm_psi: 2
    finm_K: 2
mclaury_model:
    mclm_A: 1.1
    mclm_Fs: 1.2
    mclm_Fp: 1.3
    mclm_qsd: 1.4
    mclm_velocity_exponent: 1.5
    mclm_brinnel_hardness: 1.6
    mclm_brinnel_exponent: 1.7
    mclm_d: 1.8
    mclm_cstd: 1.9
```

Figure 25.143: An example *erosion.in* file.

Table 25.117: Format of *erosion.in*.

Parameter	Description	Typical value
<i>erosion_model_groups</i>		
- <i>erosion_model_group</i>	This settings block provides erosion model settings for a particular erosion model group. Repeat this block for each erosion model group.	
<i>erosion_model</i>	Erosion model for this erosion model group. <i>generic_model</i> , <i>arabnejad_model</i> , <i>zhang_model</i> , <i>generalized_ecrc_model</i> , <i>oka_model</i> , <i>finnie_model</i> , or <i>mclaury_model</i> .	
<i>boundary_id</i>	List of boundary IDs to which the erosion model applies.	
<i>material_density</i>	Boundary material density (kg/m^3).	
<i>erosion_start_time</i>	Erosion model start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank</i>	

Chapter 25: Input and Data Files

Physical Models Erosion: erosion.in

Parameter	Description	Typical value
	<i>angle degrees if crank_flag is non-zero).</i>	
<i>erosion_end_time</i>	Erosion model end time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).	
<i>model_group_override*</i>	If you want to override the global parameters (as defined in the <i>erosion_model_parameters</i> settings block) for this erosion model, list the overridden parameters in this sub-block.	
<i><parameter name></i>	Override value for this parameter.	
<i>erosion_parcel_shape_factors</i>	This settings block defines shape factors of the erosion species. List each erosion parcel species.	
<i><parcel species name></i>	Shape factor for this parcel species. Specify a value between 0 and 1. Repeat this line for each erosion parcel species. Each species must be on a separate line.	1 for sharp, 0.53 for semi-rounded, and 0.2 for fully rounded
<i>erosion_model_parameters</i>	Settings block for global erosion model parameters.	
<i>generic_model</i>		
<i>genm_erosion_constant</i>	Erosion constant for the generic model.	
<i>genm_velocity_exponent</i>	Velocity exponent for the generic model.	
<i>genm_angle_factor</i>	Name of the angle factor data file (e.g., <i>angle_factor.dat</i>) for the generic model.	
<i>arabnejad_model</i>		
<i>aram_velocity_exponent</i>	Velocity exponent for the Arabnejad model.	2.41
<i>aram_c_1</i>	Erosion constant for cutting erosion in the Arabnejad model.	
<i>aram_c_2</i>	Erosion constant for deformation erosion in the Arabnejad model.	
<i>aram_k</i>	Angle K (radians) for the Arabnejad model.	0.4
<i>aram_u_tsh</i>	Threshold velocity u_{tsh} for the Arabnejad model.	
<i>zhang_model</i>		
<i>zham_brinnel_hardness</i>	Brinell hardness of the boundary material.	
<i>zham_brinnel_exponent</i>	Brinell hardness exponent for the Zhang model.	-0.59
<i>zham_velocity_exponent</i>	Velocity exponent for the Zhang model.	2.41
<i>zham_erosion_constant</i>	Erosion constant for the Zhang model.	

Chapter 25: Input and Data Files

Physical Models Erosion: erosion.in

Parameter	Description	Typical value
<i>zham_shape_factor_parameters</i>	Vector containing the polynomial coefficients used to calculate the Zhang model angle factor. The vector can be any length.	
<i>generalized_ecrc_model</i>		
<i>ecrcm_velocity_exponent</i>	Velocity exponent for the Generalized E/CRC model.	2.41
<i>ecrcm_c_1</i>	Erosion constant for cutting erosion in the Generalized E/CRC model.	
<i>ecrcm_c_2</i>	Erosion constant for deformation erosion in the Generalized E/CRC model.	
<i>ecrcm_k</i>	Angle K (radians) for the Generalized E/CRC model.	0.4
<i>ecrcm_u_tsh</i>	Threshold velocity u_{tsh} for the Generalized E/CRC model.	
<i>ecrcm_c_st</i>	Stokes factor constant for the Generalized E/CRC model.	0.24
<i>ecrcm_exp_st</i>	Stokes factor exponent for the Generalized E/CRC model.	-0.503
<i>oka_model</i>		
<i>okam_k_P</i>	Particle property factor for the Oka model.	
<i>okam_Hv</i>	Initial Vickers material hardness pyramid number for the boundary material (GPa).	
<i>okam_k1</i>	Hardness number exponent for the Oka model.	
<i>okam_k3</i>	Particle size exponent for the Oka model.	
<i>okam_s1</i>	Oka model constant s_1 .	
<i>okam_s2</i>	Oka model constant s_2 .	
<i>okam_q1</i>	Oka model exponent q_1 .	
<i>okam_q2</i>	Oka model exponent q_2 .	
<i>okam_v_p</i>	Reference particle velocity (m/s) for the Oka model.	
<i>okam_D_p</i>	Reference particle diameter (m) for the Oka model.	
<i>finnie_model</i>		
<i>finm_velocity_exponent</i>	Velocity exponent for the Finnie model.	2

Chapter 25: Input and Data Files

Physical Models Erosion: erosion.in

Parameter	Description	Typical value
<i>finm_p</i>	Plastic flow stress exerted by boundary material on the particle (Pa).	
<i>finm_psi</i>	Ratio of depth of contact to depth of cut.	
<i>finm_K</i>	Ratio of vertical (normal) component of force on the particle face to horizontal component.	Very close to 2
<i>mclaury_model</i>		
<i>mclm_A</i>	Coefficient A for the McLaury model.	
<i>mclm_Fs</i>	Sand sharpness factor for the McLaury model.	
<i>mclm_Fp</i>	Penetration factor for the McLaury model.	
<i>mclm_qsd</i>	Sand production rate (m^3/s).	
<i>mclm_velocity_exponent</i>	Velocity exponent for the McLaury model.	1.73
<i>mclm_brinnel_hardness</i>	Brinell hardness of the pipe.	
<i>mclm_brinnel_exponent</i>	Brinell hardness exponent for the McLaury model.	0.59
<i>mclm_d</i>	Pipe diameter (m).	
<i>mclm_cstd</i>	Ratio of radius of curvature to pipe diameter (r/d) for a standard elbow.	

* Note: Blocks or parameters indicated with an asterisk are optional.

25.8 Grid Control

This section describes the input files that contain information about controlling the grid for your CONVERGE simulation.

Adaptive Mesh Refinement: amr.in

[Adaptive Mesh Refinement](#) (AMR) automatically refines the grid based on fluctuating and moving conditions such as temperature and velocity. To enable AMR, set [*inputs.in*](#) > *grid_control* > *amr_flag* = 1 and include an *amr.in* file in your case setup.

```
version: 3.1
---

amr_settings:
  cycle_steady:          100
  max_cells:              1500000
  min_cells:              1
  embed_frequency:        10
  release_frequency:      10

amr_groups:
  - amr_group:
    amr_regions:           [0, 1]
    amr_velocity:
      active:                1
```

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

```
embed_scale:          3
sgs_embed:           1.0
temporal_type:       GRIDSCALE
starting_gridscale: 2
parcel_embed:        50

amr_temperature:
  active:             1
  embed_scale:        3
  sgs_embed:          2.5
  temporal_type:     CYCLIC
  period:             720
  start_time:         -17.0
  end_time:           131.0
  amr_frequency:      1
  amr_type:           EMBED_BETWEEN
  embed_criterion:   [1, 2]
  hold_criterion:    [0.5, 2.5]

amr_pressure:
  active:             1
  embed_scale:        2
  sgs_embed:          0.1
  temporal_type:     PERMANENT
  start_time:         -10.0
  end_time:           135.0

amr_density:
  active:             1
  embed_scale:        2
  sgs_embed:          1.0
  temporal_type:     PERMANENT
  start_time:         -10.0
  end_time:           135.0

amr_void:
  active:             0
  embed_scale:        3
  sgs_embed:          0.001
  temporal_type:     PERMANENT

amr_inlaid_neighbors:
  active:             1
  embed_scale:        6
  temporal_type:     PERMANENT
  amr_type:           EMBED_ABOVE
  embed_criterion:   4

amr_species:
  species_active:    1
  default_embed_scale: 2
  species_list:
    - species:
        active:          1
        embed_scale:     2
        name:            H
        sgs_embed:       0.001
        temporal_type:  CYCLIC
        period:          720
        start_time:      -10.0
        end_time:        10.0
    - species:
        active:          1
        embed_scale:     3
        name:            CO
        sgs_embed:       0.001
        temporal_type:  SEQUENTIAL
        start_time:      -10.0
        end_time:        10.0
        amr_type:         EMBED_BELOW
        embed_criterion: 1

amr_passive:
  passives_active:   1
  default_embed_scale: 2
  passive_list:
    - passive:
        active:          1
```

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

```
    embed_scale:      2
    name:            NOX
    sgs_embed:       0.001
    temporal_type: CYCLIC
    period:          720
    start_time:     5
    end_time:        25
  - passive:
    active:          0
    embed_scale:     3
    name:            HIROY_SOOT
    sgs_embed:       0.001
    temporal_type: SEQUENTIAL
    start_time:     5
    end_time:        25

  - amr_group:
    amr_regions:      [2]
    amr_velocity:
      active:          0
      embed_scale:     3
      sgs_embed:       1.0
      temporal_type: PERMANENT
      parcel_embed:   50
    amr_temperature:
      active:          0
      embed_scale:     3
      sgs_embed:       2.5
      temporal_type: PERMANENT
    amr_void:
      active:          1
      embed_scale:     3
      sgs_embed:       0.001
      temporal_type: GRIDSCALE
      starting_gridscale: 1

  -----
# Boundary AMR
-----
amr_bounds:
  - boundary:
    boundary_id:      1
    side:             FORWARD
    embed_scale:      2
    yplus:            30.0
    temporal_type:   PERMANENT
    starting_gridscale: -2
  - boundary:
    boundary_id:      3
    side:             REVERSE
    embed_scale:      2
    yplus:            30.0
    temporal_type:   SEQUENTIAL
    start_time:      -10.0
    end_time:        10.0

  -----
# Boundary release AMR
-----
amr_yplus_restrict:
  - boundary:
    boundary_id:      3
    side:             FORWARD
    yplus_target:    30.0
    yplus_ratio:     0.33333
    temporal_type:  SEQUENTIAL
    start_time:      -10.0
    end_time:        10.0
  - boundary:
    boundary_id:      1
    side:             REVERSE
```

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

```
yplus_target:          30.0
yplus_ratio:           0.33333
temporal_type:         PERMANENT

-----
# Proximity AMR
-----
amr_proximity:
  - boundary:
    boundary_id:      2
    side:             FORWARD
    embed_scale:      2
    prox_dist:        0.1
    temporal_type:   PERMANENT
```

Figure 25.144: An example *amr.in* file.

Table 25.118: Format of *amr.in*.

Parameter	Description	Typical value
<i>amr_settings</i>		
<i>cycle_steady</i> \$	Number of cycles between AMR calculations for embedding or releasing cells for steady-state simulations. A small number may result in poor convergence. You can use a file for temporally varying cycles (e.g., <i>amr_cycle_steady.in</i>).	50-200
<i>max_cells</i> \$	Maximum number of cells in the domain. This number should be larger than the base number of cells when AMR is not enabled. You can use a file for a temporally varying maximum number of cells (e.g., <i>amr_max_cells.in</i>).	
<i>min_cells</i> \$	Minimum number of cells in the domain. You can use a file for temporally varying minimum number of cells (e.g., <i>amr_min_cells.in</i>).	
<i>embed_frequency</i> *\$	Number of cycles between AMR embedding events for transient simulations. If this parameter is not included, the default value is 1.	
<i>release_frequency</i> *\$	Number of cycles between AMR releasing events for transient simulations. If this parameter is not included, the default value is 25.	
<i>amr_groups</i> *		
- <i>amr_group</i>	This settings sub-block specifies the AMR configuration for a group. Repeat this sub-block through the last	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
	parameter of each <i>amr_*</i> settings sub-block that you configure for each group.	
<i>amr_regions</i>	List of regions where AMR is active for this group.	
<i>amr_velocity</i>	Velocity-based AMR settings for this group.	
<i>active</i>	0 = No velocity AMR, 1 = Velocity AMR enabled.	
<i>embed_scale</i>	Maximum embedding scale for velocity AMR.	
<i>sgs_embed</i>	Sub-grid velocity above which a cell will be embedded (m/s). Required only when <i>amr_type</i> = SGS.	0.1% to 10% of the characteristic velocity in domain.
<i>temporal_type</i>	Specify whether the velocity AMR is PERMANENT, SEQUENTIAL, CYCLIC, or GRIDSCALE.	
<i>period</i>	The CYCLIC period (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero).	
<i>start_time</i>	Start time (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero) for velocity AMR for temporal_type = CYCLIC or SEQUENTIAL.	
<i>end_time</i>	End time (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero) for velocity AMR for temporal_type = CYCLIC or SEQUENTIAL. For steady-state simulations, CONVERGE will freeze AMR at this time.	
<i>starting_gridscale</i>	Gridscale at which to start the velocity AMR for temporal_type = GRIDSCALE.	
<i>parcel_embed</i>	Maximum number of parcels in a cell before CONVERGE will embed a cell via AMR.	
<i>amr_type</i> *	Specify the type of AMR. SGS = Sub-grid scale,	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
	<p><i>EMBED_BELOW</i> = Embed values less than the specified criterion, <i>EMBED_BETWEEN</i> = Embed values between the specified criterion, <i>EMBED_ABOVE</i> = Embed values greater than the specified criterion.</p> <p>If you omit this parameter, CONVERGE will set <i>amr_type</i> = <i>SGS</i> by default.</p>	
<i>embed_criterion</i> *	<p>Either a single value (for <i>EMBED_BELOW</i> or <i>EMBED_ABOVE</i>) or a vector of values (for <i>EMBED_BETWEEN</i>). Required when <i>amr_type</i> = <i>EMBED_BELOW</i>, <i>EMBED_BETWEEN</i>, or <i>EMBED_ABOVE</i>. Values less than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_BELOW</i>. Values greater than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_ABOVE</i>. Values between the <i>embed_criterion</i> are embedded for <i>amr_type</i> = <i>EMBED_BETWEEN</i>.</p>	
<i>amr_temperature</i>	Temperature-based AMR settings for this group.	
<i>active</i>	0 = No temperature AMR, 1 = Temperature AMR enabled.	
<i>embed_scale</i>	Maximum embedding scale for temperature AMR.	
<i>sgs_embed</i>	Sub-grid temperature above which a cell will be embedded (K).	0.1% to 10% of characteristic temperature in the domain.
<i>temporal_type</i>	Specify whether the temperature AMR is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The CYCLIC period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>start_time</i>	Start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for temperature AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>end_time</i>	End time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for temperature AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>starting_gridscale</i>	Gridscale at which to start the temperature AMR for <i>temporal_type</i> = <i>GRIDSCALE</i> .	
<i>amr_type</i> *	Specify the type of AMR. <i>SGS</i> = Sub-grid scale, <i>EMBED_BELOW</i> = Embed values less than the specified criterion, <i>EMBED_BETWEEN</i> = Embed values between the specified criterion, <i>EMBED_ABOVE</i> = Embed values greater than the specified criterion. If you omit this parameter, CONVERGE will set <i>amr_type</i> = <i>SGS</i> by default.	
<i>embed_criterion</i> *	Either a single value (for <i>EMBED_BELOW</i> or <i>EMBED_ABOVE</i>) or a vector of values (for <i>EMBED_BETWEEN</i>). Required when <i>amr_type</i> = <i>EMBED_BELOW</i> , <i>EMBED_BETWEEN</i> , or <i>EMBED_ABOVE</i> . Values less than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_BELOW</i> . Values greater than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_ABOVE</i> . Values between the <i>embed_criterion</i> are embedded for <i>amr_type</i> = <i>EMBED_BETWEEN</i> .	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>amr_pressure</i>	Pressure-based AMR settings for this group.	
<i>active</i>	0 = No pressure AMR, 1 = pressure AMR enabled.	
<i>embed_scale</i>	Maximum embedding scale for pressure AMR.	
<i>sgs_embed</i>	Sub-grid pressure above which a cell will be embedded (<i>Pa</i>).	0.1% to 10% of characteristic pressure in the domain.
<i>temporal_type</i>	Specify whether the pressure AMR is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The <i>CYCLIC</i> period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for pressure AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>end_time</i>	End time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for pressure AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>starting_gridscale</i>	Gridscale at which to start the pressure AMR for <i>temporal_type</i> = <i>GRIDSCALE</i> .	
<i>amr_type*</i>	Specify the type of AMR. <i>SGS</i> = Sub-grid scale, <i>EMBED_BELOW</i> = Embed values less than the specified criterion, <i>EMBED_BETWEEN</i> = Embed values between the specified criterion, <i>EMBED_ABOVE</i> = Embed values greater than the specified criterion. If you omit this parameter, CONVERGE will set <i>amr_type</i> = <i>SGS</i> by default.	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>embed_criterion</i> *	Either a single value (for <i>EMBED_BELOW</i> or <i>EMBED ABOVE</i>) or a vector of values (for <i>EMBED_BETWEEN</i>). Required when <i>amr_type</i> = <i>EMBED_BELOW</i> , <i>EMBED_BETWEEN</i> , or <i>EMBED ABOVE</i> . Values less than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_BELOW</i> . Values greater than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED ABOVE</i> . Values between the <i>embed_criterion</i> are embedded for <i>amr_type</i> = <i>EMBED_BETWEEN</i> .	
<i>amr_density</i>	Density-based AMR settings for this group.	
<i>active</i>	0 = No density AMR, 1 = Density AMR enabled.	
<i>embed_scale</i>	Maximum embedding scale for density AMR.	
<i>sgs_embed</i>	Sub-grid density above which a cell will be embedded (kg/m^3).	0.1% to 10% of characteristic density in the domain.
<i>temporal_type</i>	Specify whether the density AMR is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The <i>CYCLIC</i> period (in seconds if <i>inputs.in > simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Start time (in seconds if <i>inputs.in > simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for density AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>end_time</i>	End time (in seconds if <i>inputs.in > simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for density AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>starting_gridscale</i>	Gridscale at which to start the density AMR for <i>temporal_type</i> = <i>GRIDSCALE</i> .	
<i>amr_type</i> *	Specify the type of AMR. <i>SGS</i> = Sub-grid scale, <i>EMBED_BELOW</i> = Embed values less than the specified criterion, <i>EMBED_BETWEEN</i> = Embed values between the specified criterion, <i>EMBED_ABOVE</i> = Embed values greater than the specified criterion.	
	If you omit this parameter, CONVERGE will set <i>amr_type</i> = <i>SGS</i> by default.	
<i>embed_criterion</i> *	Either a single value (for <i>EMBED_BELOW</i> or <i>EMBED_ABOVE</i>) or a vector of values (for <i>EMBED_BETWEEN</i>). Required when <i>amr_type</i> = <i>EMBED_BELOW</i> , <i>EMBED_BETWEEN</i> , or <i>EMBED_ABOVE</i> . Values less than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_BELOW</i> . Values greater than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_ABOVE</i> . Values between the <i>embed_criterion</i> are embedded for <i>amr_type</i> = <i>EMBED_BETWEEN</i> .	
<i>amr_void</i>	Void-fraction-based AMR settings for this group.	
<i>active</i>	0 = No void fraction AMR, 1 = Void fraction AMR enabled.	
<i>embed_scale</i>	Maximum embedding scale for void fraction AMR.	
<i>sgs_embed</i>	Sub-grid void fraction above which a cell will be embedded.	
<i>temporal_type</i>	Specify whether the void fraction AMR is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The <i>CYCLIC</i> period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>start_time</i>	Start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for void fraction AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>end_time</i>	End time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for void fraction AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>starting_gridscale</i>	Gridscale at which to start the void fraction AMR for <i>temporal_type</i> = <i>GRIDSCALE</i> .	
<i>amr_type</i> *	Specify the type of AMR. <i>SGS</i> = Sub-grid scale, <i>EMBED_BELOW</i> = Embed values less than the specified criterion, <i>EMBED_BETWEEN</i> = Embed values between the specified criterion, <i>EMBED_ABOVE</i> = Embed values greater than the specified criterion. If you omit this parameter, CONVERGE will set <i>amr_type</i> = <i>SGS</i> by default.	
<i>embed_criterion</i> *	Either a single value (for <i>EMBED_BELOW</i> or <i>EMBED_ABOVE</i>) or a vector of values (for <i>EMBED_BETWEEN</i>). Required when <i>amr_type</i> = <i>EMBED_BELOW</i> , <i>EMBED_BETWEEN</i> , or <i>EMBED_ABOVE</i> . Values less than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_BELOW</i> . Values greater than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_ABOVE</i> . Values between the <i>embed_criterion</i> are embedded for <i>amr_type</i> = <i>EMBED_BETWEEN</i> .	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>amr_inlaid_neighbors</i>	AMR for this group based on the inlaid cell neighbor count.	
<i>active</i>	0 = No inlaid neighbor AMR, 1 = Inlaid neighbor AMR enabled.	
<i>embed_scale</i>	Maximum embedding scale for inlaid neighbor AMR.	
<i>temporal_type</i>	Specify whether the inlaid neighbor AMR is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The <i>CYCLIC</i> period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for inlaid neighbor AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>end_time</i>	End time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for inlaid neighbor AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>starting_gridscale</i>	Gridscale at which to start the inlaid neighbor AMR for <i>temporal_type</i> = <i>GRIDSCALE</i> .	
<i>amr_type</i>	Specify the type of AMR. <i>EMBED ABOVE</i> = Embed Cartesian cells for which the number of neighboring inlaid cells exceeds the specified criterion.	
<i>embed_criterion</i>	Number of neighboring inlaid cells above which AMR will be activated.	
<i>amr_species</i>	Species-based AMR settings for this group.	
<i>species_active</i>	1 = Activate AMR for all species provided below,	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
	0 = Do not activate AMR for all species provided below.	
<i>default_embed_scale</i>	Use this embedding scale for AMR for all species provided below, unless they have individual embedding scales specified.	
<i>species_list</i>	List of species for species-based AMR.	
- <i>species</i>	This settings sub-block specifies the species AMR configuration. Repeat this sub-block through <i>temporal_type</i> (if <i>temporal_type</i> = PERMANENT), <i>end_time</i> (if <i>temporal_type</i> = SEQUENTIAL or CYCLIC), or <i>starting_gridscale</i> (if <i>temporal_type</i> = GRIDSCALE) (and optionally through <i>amr_type</i> or <i>embed_criterion</i>) for each desired species.	
<i>active</i> *	0 = Do not activate AMR for this species, 1 = Individually activate AMR for this species (overrides <i>species_active</i>).	
<i>embed_scale</i> *	Use this embedding scale for AMR for this species (overrides <i>amr.in > amr_groups > amr_group > amr_species > default_embed_scale</i>)	
<i>name</i>	Species name.	
<i>sgs_embed</i>	Sub-grid species mass fraction above which a cell will be embedded.	0.1% to 10% of characteristic species mass fraction in the domain.
<i>temporal_type</i>	Specify whether the species AMR is PERMANENT, SEQUENTIAL, CYCLIC, or GRIDSCALE.	
<i>period</i>	The CYCLIC period (in seconds if <i>inputs.in > simulation_control > crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Start time (in seconds if <i>inputs.in > simulation_control > crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) for species AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = PERMANENT.	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>end_time</i>	End time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank_angle_degrees</i> if <i>crank_flag</i> is non-zero) for species AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>starting_gridscale</i>	Gridscale at which to start the species AMR for <i>temporal_type</i> = <i>GRIDSCALE</i> .	
<i>amr_type</i> *	Specify the type of AMR. <i>SGS</i> = Sub-grid scale, <i>EMBED_BELOW</i> = Embed values less than the specified criterion, <i>EMBED_BETWEEN</i> = Embed values between the specified criterion, <i>EMBED_ABOVE</i> = Embed values greater than the specified criterion.	
	If you omit this parameter, CONVERGE will set <i>amr_type</i> = <i>SGS</i> by default.	
<i>embed_criterion</i> *	Either a single value (for <i>EMBED_BELOW</i> or <i>EMBED_ABOVE</i>) or a vector of values (for <i>EMBED_BETWEEN</i>). Required when <i>amr_type</i> = <i>EMBED_BELOW</i> , <i>EMBED_BETWEEN</i> , or <i>EMBED_ABOVE</i> . Values less than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_BELOW</i> . Values greater than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_ABOVE</i> . Values between the <i>embed_criterion</i> are embedded for <i>amr_type</i> = <i>EMBED_BETWEEN</i> .	
<i>amr_passive</i>	Passive-based AMR settings for this group.	
<i>passives_active</i>	1 = Activate AMR for all passives provided below, 0 = Do not activate AMR for all passives provided below.	
<i>default_embed_scale</i>	Use this embedding scale for AMR for all passives provided below, unless	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
	they have individual embedding scales specified.	
<i>passive_list</i>	List of passives for passive-based AMR.	
- <i>passive</i>	This settings sub-block specifies the passive AMR configuration. Repeat this sub-block through <i>temporal_type</i> (if <i>temporal_type</i> = <i>PERMANENT</i>), <i>end_time</i> (if <i>temporal_type</i> = <i>SEQUENTIAL</i> or <i>CYCLIC</i>), or <i>starting_gridscale</i> (if <i>temporal_type</i> = <i>GRIDSCALE</i>) (and optionally through <i>amr_type</i> , <i>embed_criterion</i> , or <i>hold_criterion</i>) for each desired passive.	
<i>active</i> *	1 = Individually activate AMR for this passive (overrides <i>passives_active</i>), 0 = Do not activate AMR for this passive.	
<i>embed_scale</i> *	Use this embedding scale for AMR for this passive (overrides <i>default_embed_scale</i>)	
<i>name</i>	Variable name. Any variable in the active variables log that is designated for AMR can be used.	
<i>sgs_embed</i>	Sub-grid passive value above which a cell will be embedded.	0.1% to 10% of characteristic passive value in the domain.
<i>temporal_type</i>	Specify whether the passive AMR is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The CYCLIC period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) for passive AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>end_time</i>	End time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for passive AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>starting_gridscale</i>	Gridscale at which to start the passive AMR for <i>temporal_type</i> = <i>GRIDSCALE</i> .	
<i>amr_type</i> *	Specify the type of AMR. <i>SGS</i> = Sub-grid scale, <i>EMBED_BELOW</i> = Embed values less than the specified criterion, <i>EMBED_BETWEEN</i> = Embed values between the specified criterion, <i>EMBED_ABOVE</i> = Embed values greater than the specified criterion.	If you omit this parameter, CONVERGE will set <i>amr_type</i> = <i>SGS</i> by default.
<i>embed_criterion</i> *	Either a single value (for <i>EMBED_BELOW</i> or <i>EMBED_ABOVE</i>) or a vector of values (for <i>EMBED_BETWEEN</i>). Required when <i>amr_type</i> = <i>EMBED_BELOW</i> , <i>EMBED_BETWEEN</i> , or <i>EMBED_ABOVE</i> . Values less than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_BELOW</i> . Values greater than <i>embed_criterion</i> are embedded when <i>amr_type</i> = <i>EMBED_ABOVE</i> . Values between the <i>embed_criterion</i> are embedded for <i>amr_type</i> = <i>EMBED_BETWEEN</i> .	
<i>hold_criterion</i> *	Optional when <i>amr_type</i> = <i>EMBED_BELOW</i> , <i>EMBED_BETWEEN</i> , or <i>EMBED_ABOVE</i> . Requires the same number of inputs as <i>embed_criterion</i> . This parameter should be used for AMR variables with values that are changed by embedding. This value should be less than or equal to <i>embed_criterion</i> when <i>amr_type</i> =	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
	<i>EMBED_BETWEEN</i> or <i>EMBED_ABOVE</i> . This value should be greater than or equal to <i>embed_criterion</i> when <i>amr_type</i> = <i>EMBED_BELOW</i> . Cells that have values that fall between the <i>hold_criterion</i> and <i>embed_criterion</i> will maintain their current level of embedding.	
<i>amr_bounds*</i>		
- <i>boundary</i>	This settings sub-block specifies the AMR boundary configuration. Repeat this sub-block for each AMR boundary you wish to configure.	
<i>boundary_id</i>	Boundary on which CONVERGE will use y+ AMR.	
<i>side*</i>	Side (<i>FORWARD</i> or <i>REVERSE</i>) for INTERFACE boundary	
<i>embed_scale</i>	Maximum embedding scale for y+ AMR.	
<i>yplus</i>	y+ value above which CONVERGE will activate y+ AMR.	
<i>temporal_type</i>	Specify whether the y+ AMR is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The <i>CYCLIC</i> period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Start time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for y+ AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	
<i>end_time</i>	End time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for y+ AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>amr_yplus_restrict*</i>		
- <i>boundary</i>	This settings sub-block specifies the y^+ AMR boundary restrictions. Repeat this sub-block for each boundary you wish to configure.	
<i>boundary_id</i>	Boundary on which CONVERGE will activate the y^+ AMR restriction.	
<i>yplus_target</i>	Target y^+ value. If the y^+ value of a cell near the specified boundary is less than the specified target, then at that time-step CONVERGE will remove one level of refinement from that cell. CONVERGE does not remove more than one level of refinement in a single time-step even if the resulting y^+ value is still less than the target. It is important to ensure that the target y^+ value is based on the turbulence model in your simulation.	For k-ϵ models : 30 - 300 For k-ω models with <i>turbulence.in > Wall_modeling > near_wall_treatment_flag</i> = STANDARD: 30 - 300, ENHANCED or AUTOMATIC: 1-300, ASYMPTOTIC: 3 - 300
<i>yplus_ratio</i>	Designated y^+ ratio. If the y^+ value of a cell near the specified boundary is less than the product of the y^+ target and the y^+ ratio, then at that time-step CONVERGE will remove one level of refinement on all of the neighboring cells whose AMR level is equal to or greater than that of the original cell. CONVERGE does not remove more than one level of refinement in a single time-step even if the resulting y^+ value is still less than the product of the y^+ target and the y^+ ratio.	
<i>temporal_type</i>	Specify whether the y^+ AMR restriction is <i>PERMANENT</i> , <i>SEQUENTIAL</i> , <i>CYCLIC</i> , or <i>GRIDSCALE</i> .	
<i>period</i>	The <i>CYCLIC</i> period (in seconds if <i>inputs.in > simulation_control > crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).	

Chapter 25: Input and Data Files

Grid Control Adaptive Mesh Refinement: amr.in

Parameter	Description	Typical value
<i>start_time</i>	Start time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for y+ AMR restriction. CONVERGE ignores this parameter if <i>temporal_type</i> = PERMANENT.	
<i>end_time</i>	End time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for y+ AMR restriction. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = PERMANENT.	
<i>amr_proximity*</i>		
- <i>boundary</i>	This settings sub-block specifies the AMR proximity boundary configuration. Repeat this sub-block for each AMR proximity boundary.	
<i>boundary_id</i>	Boundary on which CONVERGE will activate proximity AMR.	
<i>embed_scale</i>	Maximum embedding scale for proximity AMR.	
<i>prox_dist</i>	Distance (<i>m</i>) between proximity groups below which CONVERGE activates proximity AMR.	
<i>temporal_type</i>	Specify whether the proximity AMR is PERMANENT, SEQUENTIAL, CYCLIC, or GRIDSCALE.	
<i>period</i>	The CYCLIC period (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero).	
<i>start_time</i>	Start time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for proximity AMR. CONVERGE ignores this parameter if <i>temporal_type</i> = PERMANENT.	

Parameter	Description	Typical value
<i>end_time</i>	End time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) for proximity AMR. For steady-state simulations, CONVERGE will freeze AMR at this time. CONVERGE ignores this parameter if <i>temporal_type</i> = PERMANENT.	

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

§ CONVERGE reads this parameter at each time-step when [inputs.in](#) > *simulation_control* > *reread_input* = 1.

Fixed Embedding: embedded.in

To include [fixed embedding](#) in a simulation, set [inputs.in](#) > *grid_control* > *embedded_flag* = 1 and include an *embedded.in* file in your case setup.

```
version: 3.1
---

- embedding:
    embed_type: BOX
    x_center: [0.0375, 0.0, 0.005]
    x_size: [0.0375, 0.0074, 0.005]
    embed_scale: 1
    temporal_type: PERMANENT
    start_time: -1.0
    end_time: 100.0
- embedding:
    embed_type: CYLINDER
    first_end:
        x_center: [0.0, 0.0, 0.0]
        radius: 0.00635
    second_end:
        x_center: [0.0, 0.0, 0.01]
        radius: 0.00635
    embed_scale: 4
    temporal_type: PERMANENT
    start_time: -1.0
    end_time: 100.0
- embedding:
    embed_type: BOUND
    boundary_id: 14
    embed_scale: 3
    num_embed: 2
    temporal_type: PERMANENT
    start_time: -999999.0
    end_time: -999999.0
- embedding:
    embed_type: SPHERE
    x_center: [-0.0034, 0.0, 0.0091]
    radius: 0.003
    embed_scale: 4
    temporal_type: CYCLIC
    period: 720
    start_time: -16.0
    end_time: 0.0
- embedding:
    embed_type: REGION
    region_id: 0
```

Chapter 25: Input and Data Files

Grid Control Fixed Embedding: embedded.in

```
embed_scale:      3
temporal_type:   PERMANENT
start_time:      -16.0
end_time:        0.0
- embedding:
  embed_type:    NOZZLE
  injector_name: injector_aa
  nozzle_name:   nozzle_aa
  radius_1:      0.001
  radius_2:      0.003
  length:        0.01
  embed_scale:   2
  temporal_type: SEQUENTIAL
  start_time:   -12.0
  end_time:     15.0
- embedding:
  embed_type:    INJECTOR
  injector_name: injector_bb
  radius_1:      0.001
  radius_2:      0.003
  length:        0.01
  embed_scale:   2
  temporal_type: CYCLIC
  period:        720
  start_time:   -12.0
  end_time:     15.0
- embedding:
  embed_type:    BOUND
  boundary_id:  15
  forward:
    embed_scale: 3
    num_embed:   2
    temporal_type: CYCLIC
    period:      720
    start_time: -999999.0
    end_time:   -999999.0
  reverse:
    embed_scale: 3
    num_embed:   2
    temporal_type: CYCLIC
    period:      720
    start_time: -999999.0
    end_time:   -999999.0
  description:  bound embedding for interface bdy 15
```

Figure 25.145: An example *embedded.in* file.

Table 25.119: Format of the *embedding* settings block for all embedding types.

Embedding type	Parameter	Description
- <i>embedding</i>		This settings block specifies the embedding configuration. Repeat this settings block for each embedding.
REGION	<i>embed_type</i>	Embedding type.
	<i>region_id</i>	Region identifier where embedding will be applied.
	<i>embed_scale</i>	Embedding scale for the region embedding.

Chapter 25: Input and Data Files

Grid Control Fixed Embedding: embedded.in

Embedding type	Parameter	Description
	<i>temporal_type</i> and <i>period</i>	Temporal type for the region embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if <i>inputs.in</i> > <i>simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) must be specified on the following line.
	<i>start_time</i>	Start time of region embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>end_time</i>	End time of region embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>description*</i>	Description of the <i>embedding</i> settings block.
BOX	<i>embed_type</i>	Embedding type.
	<i>x_center</i>	(Three values) x, y, and z coordinates for the center of the box.
	<i>x_size</i>	(Three values) Half of the box length in the x, y, and z directions.
	<i>embed_scale</i>	Embedding scale for the box embedding.
	<i>temporal_type</i> and <i>period</i>	Temporal type for the box embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if <i>inputs.in</i> > <i>simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) must be specified on the following line.
	<i>start_time</i>	Start time of box embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>end_time</i>	End time of box embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>description*</i>	Description of the <i>embedding</i> settings block.
CYLINDER	<i>embed_type</i>	Embedding type.
	<i>first_end</i>	
	<i>x_center</i>	Center of the first end of the embedding cylinder or truncated cone.
	<i>radius</i>	Radius (m) of the first end of the embedding cylinder or truncated cone.
	<i>second_end</i>	
	<i>x_center</i>	Center of the second end of the embedding cylinder or truncated cone.

Chapter 25: Input and Data Files

Grid Control Fixed Embedding: embedded.in

Embedding type	Parameter	Description
	<i>radius</i>	Radius (<i>m</i>) of the second end of the embedding cylinder or truncated cone.
	<i>embed_scale</i>	Embedding scale for the cylinder embedding.
	<i>temporal_type</i> and <i>period</i>	Temporal type for the cylinder embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if <i>inputs.in</i> > <i>simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) must be specified on the following line.
	<i>start_time</i>	Start time of cylinder embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>end_time</i>	End time of cylinder embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>description*</i>	Description of the <i>embedding</i> settings block.
SPHERE	<i>embed_type</i>	Embedding type.
	<i>x_center</i>	Center of the sphere.
	<i>radius</i>	Radius (<i>m</i>) of the sphere.
	<i>embed_scale</i>	Embedding scale for the sphere embedding.
	<i>temporal_type</i> and <i>period</i>	Temporal type for the sphere embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if <i>inputs.in</i> > <i>simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) must be specified on the following line.
	<i>start_time</i>	Start time of sphere embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>end_time</i>	End time of sphere embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>description*</i>	Description of the <i>embedding</i> settings block.
INJECTOR	<i>embed_type</i>	Embedding type.
	<i>injector_no</i>	Injector number.
	<i>radius</i>	Radius (<i>m</i>) of the first end at the nozzle (circle).
	<i>radius</i>	Radius (<i>m</i>) of the second end at the nozzle (circle).
	<i>length</i>	Length (<i>m</i>) of the embedding.
	<i>embed_scale</i>	Embedding scale for the injector embedding.

Chapter 25: Input and Data Files

Grid Control Fixed Embedding: embedded.in

Embedding type	Parameter	Description
	<i>temporal_type</i> and <i>period</i>	Temporal type for the injector embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if inputs.in > <i>simulation_control > crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) must be specified on the following line.
	<i>start_time</i>	Start time of injector embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>end_time</i>	End time of injector embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>description*</i>	Description of the <i>embedding</i> settings block.
NOZZLE	<i>embed_type</i>	Embedding type.
	<i>injector_no</i>	Injector number.
	<i>nozzle_no</i>	Nozzle number.
	<i>radius_1</i>	Radius (m) of the first end at the nozzle (circle).
	<i>radius_2</i>	Radius (m) of the second end at the nozzle (circle).
	<i>length</i>	Length (m) of the embedding.
	<i>embed_scale</i>	Embedding scale for the nozzle embedding.
	<i>temporal_type</i> and <i>period</i>	Temporal type for the nozzle embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if inputs.in > <i>simulation_control > crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) must be specified on the following line.
	<i>start_time</i>	Start time of nozzle embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>end_time</i>	End time of nozzle embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>description*</i>	Description of the <i>embedding</i> settings block.
BOUND	<i>embed_type</i>	Embedding type.
	<i>boundary_id</i>	Boundary ID specified in boundary.in .
	<i>embed_scale</i>	Embedding scale for the boundary embedding.
	<i>num_embed</i>	Number of layers of embedding to be added to the boundary. For example, if <i>num_embed</i> = 2, CONVERGE will refine the specified boundary plus two additional layers of cells.

Chapter 25: Input and Data Files

Grid Control Fixed Embedding: embedded.in

Embedding type	Parameter	Description
	<i>temporal_type</i> and <i>period</i>	Temporal type for the boundary embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if <i>inputs.in</i> > <i>simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) must be specified on the following line.
	<i>start_time</i>	Start time of boundary embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>end_time</i>	End time of boundary embedding. CONVERGE ignores this parameter if <i>temporal_type</i> = <i>PERMANENT</i> .
	<i>description*</i>	Description of the <i>embedding</i> settings block.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

For boundary embedding, you can specify an *embed_scale* and other information for the entire boundary, as shown in the previous table. If you have an INTERFACE boundary, you can include the *forward* and *reverse* settings blocks to specify embedding information that is specific to the forward and reverse sides of the boundary.

Table 25.120: Format of the *forward* and *reverse* settings blocks (for boundary embedding).

Parameter	Description
<i>forward</i>	
<i>embed_scale</i>	Embedding scale for the forward region of the INTERFACE boundary.
<i>num_embed</i>	Number of layers of embedding to be added to the forward region of the INTERFACE boundary. For example, if <i>num_embed</i> = 2, CONVERGE will refine the specified boundary plus two additional layers of cells.
<i>temporal_type</i> and <i>period</i>	Temporal type for the forward boundary embedding (<i>PERMANENT</i> , <i>SEQUENTIAL</i> , or <i>CYCLIC</i>). If <i>CYCLIC</i> , the period (in seconds if <i>inputs.in</i> > <i>simulation_control > crank_flag</i> = 0 or in <i>crank angle degrees</i> if <i>crank_flag</i> is non-zero) must be specified on the following line.
<i>start_time</i>	Start time of boundary embedding. CONVERGE ignores this parameter if the temporal type is <i>PERMANENT</i> .
<i>end_time</i>	End time of boundary embedding. CONVERGE ignores this parameter if the temporal type is <i>PERMANENT</i> .
<i>reverse</i>	
<i>embed_scale</i>	Embedding scale for the reverse region of the INTERFACE boundary.

Chapter 25: Input and Data Files

Grid Control Fixed Embedding: embedded.in

Parameter	Description
<i>num_embed</i>	Number of layers of embedding to be added to the reverse region of the INTERFACE boundary. For example, if <i>num_embed</i> = 2, CONVERGE will refine the specified boundary plus two additional layers of cells.
<i>temporal_type</i> and <i>period</i>	Temporal type for the reverse boundary embedding (PERMANENT, SEQUENTIAL, or CYCLIC). If CYCLIC, the period (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero) must be specified on the following line.
<i>start_time</i>	Start time of boundary embedding. CONVERGE ignores this parameter if the temporal type is PERMANENT.
<i>end_time</i>	End time of boundary embedding. CONVERGE ignores this parameter if the temporal type is PERMANENT.

Grid Scaling: gridscale.in

To direct CONVERGE to adjust the base grid size ([inputs.in](#) > grid_control > base_grid) as a function of time, specify a file name (e.g., *gridscale.in*) for [inputs.in](#) > grid_control > grid_scale. This procedure is known as [grid scaling](#).

```
version: 3.1
---
- gridscale:
    time:          1.1
    value:         -1
    monitor_filename: monitor_steady_state1.in
- gridscale:
    time:          AUTO
    value:          0
    monitor_filename: monitor_steady_state2.in
```

Figure 25.146: An example *gridscale.in* file.

Table 25.121: Format of the *gridscale* settings block.

Parameter	Description
- <i>gridscale</i>	This settings block specifies the gridscale configuration. Repeat this sub-block for each grid scale setting.
<i>time</i>	Time (in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero) at which to apply this level of grid scaling.
<i>value</i>	An integer that coarsens or refines the grid via the following equation: scaled grid = $dx_{base} / 2^{value}$. For example, if <i>value</i> = -1, CONVERGE will coarsen the base grid by a factor of two. Values of <i>value</i> must be integers that increase monotonically as the simulation progresses.
<i>monitor_filename</i> *	Steady-state monitor file name associated with this grid scaling (for steady-state simulations).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Inlaid Mesh: *inlaid_mesh.in*

CONVERGE can create an [inlaid mesh](#) from user-specified boundary IDs. To enable inlaid meshing, set *inputs.in* > *grid_control* > *inlaid_mesh_flag* = 1 and include an *inlaid_mesh.in* file in your case setup.

```
version: 3.1
---

boundary_ids:
  - 30
  - 31

inlaid_mesh_settings:
  - inlaid_mesh:
    boundary_ids: 30
    cell_pairing:
      active: 1
      criterion: VOLUME
      target_value: AUTO
      auto_scaling_ratio: 0.5
    deformation:
      active: 0
  - inlaid_mesh:
    boundary_ids: 31
    cell_pairing:
      active: 1
      criterion: VOLUME
      target_value: AUTO
      auto_scaling_ratio: 0.3
    deformation:
      active: 0
```

Figure 25.147: An example *inlaid_mesh.in* file.

Chapter 25: Input and Data Files

Grid Control Inlaid Mesh: `inlaid_mesh.in`

Table 25.122: Format of the *inlaid_mesh.in*.

Parameter	Description
<code>boundary_ids</code>	List of boundary IDs of the coupled flow-through INTERFACE boundaries that are internal to the inlaid grid region.
<code>inlaid_mesh_settings</code>	This settings block describes the configuration for cell pairing.
- <code>inlaid_mesh</code>	Repeat this sub-block for each cell pairing configuration.
<code>boundary_ids</code>	Boundary ID to which cell pairing settings are applied. This boundary must also be listed in the <code>boundary_ids</code> settings block above.
<code>cell_pairing</code>	
<code>active</code>	0 = Cell pairing is inactive (default), 1 = Cell pairing is active.
<code>criterion</code>	<code>VOLUME</code> = Pair cells according to volume ratios.
<code>target_value</code>	Ideal value for specified criterion. Enter a value or enter <code>AUTO</code> . When the <code>target_value = AUTO</code> , the target value is configured as the mean of all inlaid cells per inlaid mesh.
<code>auto_scaling_ratio</code>	Cells whose criterion value is less than <code>auto_scaling_ratio * target_value</code> will be paired. Only used when <code>target_value = AUTO</code> .
<code>deformation*</code>	
<code>active</code>	Deprecated in CONVERGE 3.1.6. To allow deforming motion, it is sufficient to set <code>boundary.in > boundary_conditions > boundary > motion = DEFORM</code> for the deforming boundaries. 0 = Do not allow deforming motion, 1 = Calculate deforming motion from radial basis function for boundaries with <code>boundary.in > boundary_conditions > boundary > motion = DEFORM</code> .

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Minimum Volume Ratio: `min_vol_ratio.in`

To specify a minimum volume ratio for Cartesian cell pairing that varies in space and/or time, set `inputs.in > grid_control > min_vol_ratio = min_vol_ratio.in` and include the `min_vol_ratio.in` file in your case setup. The parameters in this file apply only to the Cartesian grid.

```
version: 3.1
---

- min_vol_ratio:
    type: CYLINDER
    first_end:
        x_center: [-1.0, 2.0, 0.0]
        radius: 0.5
    second_end:
        x_center: [1.0, 2.0, 0.0]
        radius: 1.0
    value: 0.05
    temporal_type: SEQUENTIAL
```

Chapter 25: Input and Data Files

Grid Control Minimum Volume Ratio: min_vol_ratio.in

```
start_time:      0.05
end_time:       0.1

- min_vol_ratio:
  type:          BOUND
  boundary_id:   4
  value:         0.5
  temporal_type: PERMANENT

- min_vol_ratio:
  type:          BOUND
  boundary_id:   7
  value:         0.6
  temporal_type: SEQUENTIAL
  start_time:    0.0
  end_time:     1.0

- min_vol_ratio:
  type:          SPHERE
  x_center:      [0.0, -2.0, 0.0]
  radius:        1.0
  value:         0.2
  temporal_type: CYCLIC
  period:        0.15
  start_time:   0.05
  end_time:    0.08

- min_vol_ratio:
  type:          BOX
  x_center:      [-0.375, 3.0, 0.0]
  x_size:        [0.25, 0.5, 1.0]
  value:         0.15
  temporal_type: CYCLIC
  period:        0.15
  start_time:   0.02
  end_time:    0.10

- min_vol_ratio:
  type:          REGION
  region_id:    0
  value:         0.8
  temporal_type: SEQUENTIAL
  start_time:   0.0
  end_time:    0.03
```

Figure 25.148: An example *min_vol_ratio.in* file.

Chapter 25: Input and Data Files

Grid Control Minimum Volume Ratio: min_vol_ratio.in

Table 25.123: Format of the *min_vol_ratio* settings block.

Parameter	Description
- <i>min_vol_ratio</i>	This settings block specifies the minimum volume ratio configuration. Repeat this block for each minimum volume ratio type.
<i>type</i>	Type of minimum volume ratio shape specification. <i>CYLINDER</i> = Supply the <i>x_center</i> and <i>radius</i> for the <i>first_end</i> of the cylinder and the <i>second_end</i> of the cylinder, <i>BOUND</i> = Embedding on a boundary. Supply the <i>boundary_id</i> below, <i>SPHERE</i> = Embedding within a sphere. Supply the <i>x_center</i> (sphere center) and <i>radius</i> (sphere radius), <i>BOX</i> = Embedding within a box. Supply the <i>x_center</i> (box center point) and <i>x_size</i> (box half size), <i>REGION</i> = Embedding within a region. Supply the <i>region_id</i> below.
<i>value</i>	Minimum volume ratio value.
<i>temporal_type</i>	Temporal type for embedding. <i>SEQUENTIAL</i> = Sequential embedding. Supply the <i>start_time</i> and <i>end_time</i> below, <i>CYCLIC</i> = Cyclic embedding. Supply the <i>period</i> , <i>start_time</i> , and <i>end_time</i> below, <i>PERMANENT</i> = Permanent embedding.
<i>period</i>	The CYCLIC period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>start_time</i>	Start time for embedding (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>end_time</i>	End time for embedding (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).

Proximity Boundaries: proximity_boundaries.in

If *inputs.in* > *feature_control* > *prox_check_flag* is non-zero, the *proximity_boundaries.in* file must be included in your case setup.

For each proximity group, specify the group ID and the boundary IDs that make up each group. You must define at least two groups.

```
version: 3.1
---

seal_tolerance:      1.0e-4
proximity_groups:
  - group_id:      1
    boundary_ids: [1, 2, 3]
  - group_id:      2
    boundary_ids: [4, 5]
```

Figure 25.149: An example *proximity_boundaries.in* file.

Chapter 25: Input and Data Files

Grid Control Proximity Boundaries: proximity_boundaries.in

Table 25.124: Format of *proximity_boundaries.in*.

Parameter	Description
<i>seal_tolerance</i>	The spacing (<i>m</i>) of a gap between proximity groups below which CONVERGE flags cells for proximity-based Adaptive Mesh Refinement and proximity-shaped sources.
<i>proximity_groups</i>	
- <i>group_id</i>	Unique ID of the proximity group. This settings sub-block specifies the group and boundaries for proximity groups. Repeat this sub-block for each proximity group.
<i>boundary_ids</i>	Boundary IDs of the boundaries that make up the proximity group. For INTERFACE boundaries, use the positive-signed boundary ID to specify proximity grouping on the forward side of the INTERFACE boundary. Use the negative-signed boundary ID to apply the proximity grouping on the reverse side of the boundary.

25.9 Pre-Processing

This section describes the input files for pre-processing utilities.

Make Surface: *makesurface.in*

The *makesurface.in* file contains input parameters for the [*make_surface*](#) utility.

```
0.13716 bore
0.1651 stroke
0.1 conrod
0 wrist_pin_offset
ADJUST target_comp_ratio
0.01 target_squish_height
NO_USE target_crevice_width
NO_USE target_crevice_depth
60.00 theta
0 create_profile
6 nforces
2.0 kspring
6.0 xmax
-4.0 ymin
0.001 minsquish
```

Figure 25.150: An example *makesurface.in* file.

Chapter 25: Input and Data Files

Pre-Processing Make Surface: makesurface.in

Table 25.125: Format of *makesurface.in*.

Parameter	Description
<i>bore</i>	Engine cylinder bore (<i>m</i>).
<i>stroke</i>	Engine cylinder stroke (<i>m</i>).
<i>conrod</i>	Engine connecting rod length (<i>m</i>).
<i>wrist_pin_offset</i>	Engine wrist pin offset (<i>m</i>). This parameter can be negative as the absolute value is used.
<i>target_comp_ratio</i>	Target compression ratio. Enter a value or set to <i>ADJUST</i> .*
<i>target_squish_height</i>	Target squish height (<i>m</i>). Enter a value or set to <i>ADJUST</i> .*
<i>target_crevice_width</i>	Target crevice width (<i>m</i>). Enter a value or set to <i>ADJUST</i> if a crevice is to be generated. Set to <i>NO_USE</i> if no crevice is to be generated.*
<i>target_crevice_depth</i>	Target crevice depth (<i>m</i>). Enter a value or set to <i>ADJUST</i> if a crevice is to be generated. Set to <i>NO_USE</i> if no crevice is to be generated.*
<i>theta</i>	Sector angle in degrees.
<i>create_profile</i>	0 = Profile file is provided, 1 = CONVERGE creates the profile from <i>forces.in</i> .
<i>nforces</i>	Number of spline points. Used only if <i>create_profile</i> = 1.
<i>kspring</i>	Spring constant (<i>k</i>). Used only if <i>create_profile</i> = 1.
<i>xmax</i>	Maximum displacement along x axis (<i>m</i>). Used only if <i>create_profile</i> = 1.
<i>ymin</i>	Minimum displacement along y axis (<i>m</i>). Used only if <i>create_profile</i> = 1.
<i>minsquish</i>	Minimum squish height (<i>m</i>). Used only if <i>create_profile</i> = 1.

*The [make_surface](#) utility can adjust at most one of the following four parameters: *target_comp_ratio*, *target_squish_height*, *target_crevice_width*, and *target_crevice_depth*. The other three parameters must be assigned a fixed value (or set to *NO_USE* if no crevice is to be generated). Note that both crevice parameters must be set to *NO_USE* if one of them is set to *NO_USE*.

To have the [make_surface](#) utility create the piston bowl and head profiles using the method of forces, set *create_profile* = 1. Then enter values for *nforces*, *kspring*, *xmax*, *ymin*, and *minsquish*. The method of forces calculates displacements due to the number of forces (*nforces*) acting on a flat spring that has a stiffness *kspring*. CONVERGE generates the head and bowl profiles due to the displacements from these forces. You must also include in your case setup a *forces.in* input file that specifies the magnitude of each force, as shown below in Figure 25.151. Note that *nforces* must be equal to the number of forces specified in the *forces.in* input file.

```
0.035
0.037
0.036
0.041
0.045
0.039
```

Chapter 25: Input and Data Files

Pre-Processing Make Surface: makesurface.in

Figure 25.151: An example *forces.in* file.

To use a flat surface for the head and/or bowl profiles, set *create_profile* = 0. If you do not provide a *bowl_profile* file, the [*make_surface*](#) utility will create a flat profile for the piston bowl. This profile will be based on the value of the *bore* parameter in *makesurface.in*. If you do not provide a *head_profile* file, the [*make_surface*](#) utility will create a flat profile for the head. Again, this profile will be based on the value of the *bore* parameter in *makesurface.in*.

To provide your own head and/or bowl profile information, set *create_profile* = 0. Save the files containing basic x, z coordinate information for the head and bowl as *head_profile* and *bowl_profile*, respectively. The file format is shown below in Figure 25.152. The first point in each profile must be at the axis (*x* = 0.0). If the last point in the profile is not at the cylinder radius, the [*make_surface*](#) utility will create an additional point with the *x* value equal to the cylinder radius and the *z* value equal to the last point specified in the profile. The *x* coordinates are absolute, while the *z* coordinates are relative and will be adjusted based on stroke and squish. The units of all values must be in *meters*. With these *head_profile* and/or *bowl_profile* files present, the [*make_surface*](#) utility will generate a surface geometry file that is ready for simulation.

```
0.000000e+00 -1.741080e-01
5.700000e-03 -1.762060e-01
8.400000e-03 -1.772170e-01
1.110000e-02 -1.781950e-01
1.380000e-02 -1.791890e-01
1.659200e-02 -1.802050e-01
1.940700e-02 -1.812610e-01
2.160000e-02 -1.820600e-01
2.375000e-02 -1.828310e-01
2.575000e-02 -1.836180e-01
2.810000e-02 -1.844820e-01
3.000000e-02 -1.851750e-01
3.233300e-02 -1.860120e-01
3.410000e-02 -1.866530e-01
3.733800e-02 -1.872210e-01
4.000000e-02 -1.868740e-01
4.216700e-02 -1.859010e-01
4.413300e-02 -1.844430e-01
4.568300e-02 -1.824290e-01
4.663900e-02 -1.801370e-01
4.699000e-02 -1.775690e-01
4.699000e-02 -1.693210e-01
6.858000e-02 -1.693210e-01
```

Figure 25.152: An example *bowl_profile* file.

25.10 Output/Post-Processing

This section describes the input files that contain output and post-processing parameters for your CONVERGE simulation.

Post Variable Selection: *post.in*

The *post.in* file, which is required for all simulations, specifies which quantities CONVERGE writes to the binary output files for post-processing and visualization.

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

```
version: 3.1
---

cells:
  general: [ pressure, total_pressure, density,
    mol_cond, mol_visc, mol_diff_<species name>,
    vol_frac, diff_pres, grad_p, pressure_ratio, sie,
    sensible_sie, temperature, total_temp, dt,
    velocity, monotone_upwind, prod_monotone_upwind,
    flux_limiter_mom, flux_limiter_global,
    q_criterion, porous_cell_index,
    radiation, radiation_src, sat_temp, source_value,
    kim_passives, ren_passives, specific_entropy ]

  geometry: [ area, xcen, level, logic_ijk, rank, idreg,
    center_flag, min_flag, min_vol_ratio,
    prox_dist, prox_flag, bl_cell,
    mesh_quality, stream_id ]

  combustion: [ damkohler, equiv_ratio, lambda,
    overall_equiv_ratio, react_lambda,
    react_ratio, tur_flamespeed, flame_index,
    reaction_progress, zmean, zvar, cmean ]

  turbulence: [ eps, viscosity_ratio, omega, conductivity, tke,
    turb_length, turb_velocity, visc, yplus,
    les_mode ]

  boundary: [ bound_htc, cell_htc, conv_htc, bound_t_ref,
    bnd_mode, bound_temp, attach, sc_avg_htc,
    sc_avg_temp, sc_wall_temp, moving, bound,
    io_mass_flow_rate ]

  film: [ film_ht, film_mass, film_temp ]

  soot: [ soot_pi, soot_sg, soot_fr, soot_ox, soot_con,
    soot_coag, soot_mf, soot_vf, soot_size,
    soot_mass, soot_num_density ]

  species_massfrac: [ h20, o2 ]
  species_molefrac: [ h20, o2 ]
  film_species_massfrac: [ h20, o2 ]

  passive: [ INTAKE, EXHAUST, CYLINDER ]
  user: [ deposit_risk ]
  surfcov: [ INTAKE, EXHAUST, CYLINDER ]
  sootsect_vf: [ INTAKE, EXHAUST, CYLINDER ]
  sootsect_mf: [ INTAKE, EXHAUST, CYLINDER ]
  sootsect_mass: [ INTAKE, EXHAUST, CYLINDER ]

  parcels:
    liquid_parcels:
      physical: [ density, mass, velocity, node_xx, triangle, temp,
        surf_temp, radius, num_drop, from_nozzle,
        from_injector, parent, film_flag,
        film_thickness, film_area, radiation_energy ]

      model: [ distort, distor_dot, shed_mass, film_shed, k_kuhnke,
        shed_num_drop, tbreak_kh, tbreak_rt, t_turb,
        time_turb_accum, uprime, vprime, wprime,
        distant, weber ]

    massfrac: [ h20, o2 ]
```

Figure 25.153: An example *post.in* file.

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

The *post.in* file contains two blocks, *cells* and *parcels*. Each block contains a set of sub-blocks, listed below in Table 25.126, each of which list some of the variables that CONVERGE can write to the [binary output files](#).

Table 25.126: Available settings blocks and sub-blocks in *post.in*.

Block	Allowed sub-blocks
<i>cells</i> *	general geometry combustion turbulence boundary film soot species_diffusion_velocity species_massfrac species_molefrac film_species_massfrac passive user surfcov sootsect_vf sootsect_mf sootsect_mass
<i>parcels</i> *	liquid_parcels physical model massfrac solid_parcels physical model massfrac

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

CONVERGE saves the HDF5-formatted output files (*post*.h5*) to a directory named *output*, which is within the Case Directory. After the simulation is complete, use the [post convert](#) utility to convert the data in the *post*.h5* files to a format that can be used by a visualization program.

If you change the *post.in* file prior to restarting a simulation, CONVERGE will create a new output directory for the restart post files. The new directory will be named *output<number>* (e.g., *output1*), where *<number>* is *restart_number* from *inputs.in*.

To generate averaged quantities from *post*.h5* files, you must include the following variables in the *post.in* file before starting a simulation: *logic_ijk*, *level*, and *volume*.

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Table 25.154 describes some of the species/[passive](#) or region-specific sub-blocks in *post.in*. Each of Tables 25.155 through 25.164 below describes the variables from one of the other sub-blocks in *post.in*.

Note that these variables must be spelled exactly as shown in the tables. CONVERGE frequently updates the list of acceptable variables in *post.in* (for more information on version-to-version changes, consult the [release notes](#)). To ensure that you can use all the variables listed in this section, please install the latest version of CONVERGE.

Table 25.127: Sub-blocks used to list species/passive/user properties. These sub-blocks, if included, must be in the *cells* settings block.

Sub-block	Allowed values	Description
<i>species_diffusion_velocity</i>	< <i>species</i> >. The species name must match the name specified in <i>initialize.in</i> , the reaction mechanism file, and/or <i>species.in</i> .	Diffusion velocity of species.
<i>species_massfrac</i>	< <i>species</i> >. The species name must match the name specified in <i>initialize.in</i> , the reaction mechanism file, and/or <i>species.in</i> .	Mass fraction of the species. When using the RIF model , you can specify in <i>post.in</i> any species included in the mechanism, regardless of whether it is a <i>rif_transport_species</i> , and CONVERGE will provide mass fraction data for that species.
<i>species_molefrac</i>	< <i>species</i> >. The species name must match the name specified in <i>initialize.in</i> , the reaction mechanism file, and/or <i>species.in</i> .	Mole fraction of species.
<i>film_species_massfrac</i>	< <i>species</i> >. The species name must match the name specified in <i>initialize.in</i> , the reaction mechanism file, and/or <i>species.in</i> .	Mass fraction of film species. When using the RIF model , you can specify in <i>post.in</i> any species included in the mechanism, regardless of whether it is a <i>rif_transport_species</i> , and CONVERGE will provide mass fraction data for that species.
<i>passive</i>	< <i>passive</i> >. The passive name must match the name specified in <i>initialize.in</i> and <i>species.in</i> .	Value of the passive. Some internal passives are listed in the species.in section.

Chapter 25: Input and Data Files

Output/Post-Processing

Post Variable Selection: post.in

Sub-block	Allowed values	Description
<i>user</i>	< <i>custom variable</i> >. The custom variable name must match the name of the CONVERGE_POST macro.	Value of a custom post-processing variable from a user-defined function (UDF) based on the CONVERGE_POST macro. Refer to the CONVERGE 3.1 UDF Manual for more information.
<i>species_density</i>	< <i>species</i> >. The species name must match the name specified in <i>initialize.in</i> , the reaction mechanism file, and/or <i>species.in</i> .	Cell species density (kg/m^3).
<i>surfcov</i>	< <i>species</i> >. The species name must match the name specified in <i>surface_mech.dat</i> .	Surface coverage fraction of the surface species.
<i>sootsect_mass</i>	INTAKE, EXHAUST, CYLINDER	PSM soot mass (kg) in each section and each cell of the specified region.
<i>sootsect_mf</i>	INTAKE, EXHAUST, CYLINDER	PSM soot mass fraction in each section and each cell of the specified region.
<i>sootsect_vf</i>	INTAKE, EXHAUST, CYLINDER	PSM soot volume fraction in each section and each cell of the specified region.

Table 25.128: Cell property variables in *post.in*. These variables, if included, must be in the *cells > general settings* sub-block.

Variable name	Description
<i>absorption_coef</i>	Radiation absorption coefficient ($1/m$).
<i>acentric_factor</i>	Acentric factor.
<i>acoustic_mesh_freq_cuto ff</i>	The highest acoustic frequency resolved with the current mesh.
<i>alpha</i>	Cell void fraction. Volume of fluid modeling must be active (inputs.in > feature_control > vof_flag = 1).
<i>cav_rate</i>	Cell cavitation rate, <i>i.e.</i> , the finite rate at which liquid evaporates or vapor condenses at a given cell pressure ($kg/m^3\text{-s}$). Must have inputs.in > feature_control > vof_flag = 1 and vof.in > cavitation_model > cavitation_flag = 1 .
<i>cp</i>	Specific heat capacity at constant pressure in the cell ($J/kg\text{-K}$).
<i>cv</i>	Specific heat capacity at constant volume in the cell ($J/kg\text{-K}$).
<i>cfl_mach</i>	Mach CFL number .

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>cfl_nu</i>	Viscosity CFL number.
<i>cfl_u</i>	Convection CFL number.
<i>density</i>	Cell density (kg/m^3)
<i>diff_pres</i>	Cell pressure minus the average region pressure (N/m^2).
<i>div</i>	Divergence in the cell (1/s), i.e., $\frac{\partial u_i}{\partial x_i}$.
<i>efield_current</i>	Electric current density J_i (A/m^2). Valid only when inputs.in > <i>feature_control</i> > <i>electric_potential_flag</i> = 1.
<i>elsa_mfrac_liq_<species></i>	Fraction of liquid mass (combined Eulerian and Lagrangian) of this species to total mass in the cell. Only valid for an ELSA simulation.
<i>elsa_vfrac_liq_<species></i>	Fraction of liquid volume (combined Eulerian and Lagrangian) of this species to total volume in the cell. Only valid for an ELSA simulation.
<i>epot_phi</i>	Electric potential (V). Valid only when inputs.in > <i>feature_control</i> > <i>electric_potential_flag</i> = 1.
<i>epot_sie_src</i>	Source term in the solid energy equation due to ohmic heat dissipation (W/m^3). Valid only when inputs.in > <i>feature_control</i> > <i>electric_potential_flag</i> = 1.
<i>erosion_rate</i>	Erosion rate $\dot{m}_{er}/(\rho_s A)$ (m/s), where ρ_s and A are the density and area of the surface, respectively. Must have inputs.in > <i>feature_control</i> > <i>erosion_flag</i> = 1.
<i>flux_limiter_mom</i>	The sum of the flux limiter values for the momentum equation in the x, y, and z directions. Ranges from 0-3, where 3 indicates that flux is not limited in any direction.
<i>flux_limiter_global</i>	The sum of the flux limiter values for the global equation in the x, y, and z directions. Ranges from 0-3, where 3 indicates that flux is not limited in any direction.
<i>gamma</i>	Ratio of specific heats at constant pressure and constant volume in the cell.
<i>grad_p[<i>]</i>	Gradient of pressure (N/m^3) in the cell in the <i>i</i> -th direction. 1 represents x, 2 represents y, and 3 represents z.
<i>kim_passives</i>	Passives specified in the Hatchard-Kim model . These are written to <i>post*.h5</i> as KIM_SEI, KIM_AN, KIM_T_SEI, KIM_ALPHA, and KIM_E.
<i>dt</i>	Value of cell dt (time-step) in seconds.
<i>mach</i>	Cell Mach number.
<i>mass</i>	Cell mass (kg).

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>mass_eroded</i>	Total mass that has eroded from the boundary (kg). Must have <i>inputs.in</i> > <i>feature_control</i> > <i>erosion_flag</i> = 1.
<i>mass_impinged</i>	Total mass that has impinged on the boundary (kg). Must have <i>inputs.in</i> > <i>feature_control</i> > <i>erosion_flag</i> = 1.
<i>mol_cond</i>	Cell molecular conductivity (W/m-K).
<i>mol_visc</i>	Cell molecular viscosity (N-s/m ²).
<i>mol_diff_<species name></i>	Molecular diffusivity of the specified species. Repeat for as many species as exist in the simulation. This feature requires <i>inputs.in</i> > <i>property_control</i> > <i>species_diffusion_model</i> greater than 0.
<i>monotone_upwind</i>	Indicates the numerical scheme used in the cell in all three directions (typically based on <i>solver.in</i> > <i>Numerical_Schemes</i> > <i>monotone_tolerance</i>), where the solver used the 0 = First-order upwind scheme, 1 = Numerical scheme specified in <i>solver.in</i> .
<i>omega_fuel_gas</i>	Fuel gas source term (kg/s).
<i>omega_fuel_spray</i>	Fuel spray source term (kg/s).
<i>cell_pairs</i>	Cell pair volume (m ³) (if a cell is paired with another cell).
<i>parcel_source</i>	Drag on solid parcels (N). Must have <i>inputs.in</i> > <i>feature_control</i> > <i>erosion_flag</i> = 1 and <i>inputs.in</i> > <i>feature_control</i> > <i>fixed_flow_flag</i> = 1.
<i>porous_cell_index</i>	Index of porous media in the cell.
<i>pressure_ratio</i>	Density-to-pressure ratio in cell (kg/N-m).
<i>pressure</i>	Cell pressure (N/m ²).
<i>prod_monotone_upwind</i>	Identifies scheme used in boundary cells (which may be Cartesian cut cells) by calculating the product of <i>monotone_upwind</i> in the x, y, and z directions. Always 0 or 1.
<i>radiation</i>	Incident radiation (W/m ²).
<i>radiation_src</i>	Radiation source term (W/m ³).
<i>ren_passives</i>	Passives specified in the Ren model . These are written to <i>post*.h5</i> as <i>REN_SEI</i> , <i>REN_AN_E</i> , <i>REN_AN_B</i> , <i>REN_CAT</i> , <i>REN_CAT-AN</i> , and <i>REN_CAT-B</i> .
<i>sat_temp</i>	Saturation temperature (K). Must have <i>inputs.in</i> > <i>feature_control</i> > <i>vof_flag</i> = 1 and <i>vof.in</i> > <i>cavitation_model</i> > <i>cavitation_flag</i> = 1.
<i>sensible_sie</i>	Cell specific sensible value for internal energy (J/kg)
<i>sie</i>	Cell specific internal energy (J/kg).
<i>source_value</i>	Value of all sources specified in <i>source.in</i> . Variables are written to <i>post*.h5</i> as < <i>description</i> >_SOURCE_VALUE.

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>specific_entropy</i>	Cell specific entropy ($J/kg\cdot K$). If specified, CONVERGE will also write this value in thermo.out .
<i>temperature</i>	Cell temperature (K).
<i>temp_sgs</i>	Sub-grid scale temperature (K).
<i>total_pressure</i>	Total pressure in cell (N/m^2).
<i>total_temperature</i>	Total temperature in cell (K).
<i>two_layer_mode</i>	RANS enhanced wall treatment blending coefficient (requires k-ε turbulence model).
<i>vap_pres</i>	Cell vapor pressure (N/m^2).
<i>velocity</i>	All three components of cell velocity (m/s).
<i>vel_sgs</i>	Sub-grid scale velocity (m/s).
<i>vol_frac</i>	Liquid parcel volume divided by cell volume.
<i>volume</i>	Cell volume (m^3).
<i>vorticity</i>	Vorticity (s^{-1}) (generates three separate scalar components).
<i>wall_dist</i>	Distance to the nearest wall (m).

Table 25.129: Cell geometry and cell location variables in *post.in*. These variables, if included, must be in the *cells > geometry settings* sub-block.

Variable name	Description
<i>area[i]</i>	Area of the cell side (m^2), where side is any integer, <i>i</i> . For <i>i</i> , integers 0 through 5 correspond to left, right, front, back, bottom, or top, respectively.
<i>bl_cell</i>	Returns a 0 if this cell is part of a Cartesian mesh. Returns a 1 if this cell is part of an inlaid mesh.
<i>cell_pairs</i>	Paired cells.
<i>center_flag</i>	Cell center located outside of cell.
<i>idreg</i>	Cell region identification number.
<i>grid_level</i>	Cell embed level relative to the root cell (initial cell in which the geometry is immersed during grid generation).
<i>logic_ijk</i>	Cell node in <i>i</i> -th, <i>j</i> -th, and <i>k</i> -th directions. For inlaid cells, <i>i</i> = <i>j</i> = 1, and <i>k</i> is unique.
<i>max_grid_level</i>	Embed level of the cell. If this is part of a cell pair, the maximum embedding level for the entire pair.
<i>min_grid_level</i>	Embed level of the cell. If this is part of a cell pair, the minimum embedding level for the entire pair.

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>multi_grid_level</i>	Returns 0 if the cell is not paired, or if the entire cell pair is at the same embedding level. Returns 1 if the cell is part of a pair that has child cells at multiple embedding levels.
<i>mesh_quality</i>	CONVERGE will calculate and print the local values of the mesh quality metrics (recommended primarily for inlaid mesh simulations).
<i>prox_dist</i>	Minimum proximity distance (<i>m</i>).
<i>prox_flag</i>	Cells that have a distance less than tolerance specified in <i>proximity_boundaries.in</i> .
<i>rank</i>	Processor identifier for the cell.
<i>stream_id</i>	Cell stream identification number.
<i>xcen[direction]</i>	Cell center (in <i>m</i>), where direction is any integer 0, 1, or 2 corresponding to the x, y, or z direction respectively.

Table 25.130: Combustion variables in *post.in*. These variables, if included, must be in the *cells > combustion* settings sub-block.

Variable name	Description
<i>cmean</i>	Progress variable for the FGM model. This value ranges from 0 to 1.
<i>damkohler</i>	Cell Damkohler number. Combustion modeling must be active (inputs.in > feature_control > combustion_flag = 1).
<i>equiv_ratio</i>	Cell equivalence ratio.
<i>flame_index</i>	Normalized real-valued flame index. -1.0 = Indicates a fully non-premixed flame, 1.0 = Indicates a fully premixed flame.
<i>lam_flamespeed</i>	Cell laminar flamespeed (m/s). Combustion modeling must be active (inputs.in > feature_control > combustion_flag = 1). Uses the combust.in > sl_model settings block.
<i>lambda</i>	Cell relative air-fuel ratio.
<i>mix_frac</i>	Cell mixture fraction.
<i>az_id</i>	Identifier for adaptive zoning.
<i>overall_equiv_ratio</i>	Overall equivalence ratio (both gas and liquid phase) in the cell.
<i>radiation</i>	Monochromatic incident radiation in the cell (W/m ³)
<i>react_lambda</i>	Cell lambda value for the combustion reaction excluding CO ₂ and H ₂ O.
<i>react_ratio</i>	Cell equivalence ratio that does not include CO ₂ and H ₂ O in the calculation.
<i>reaction_progress</i>	Normalized progress variable in the cell for the FGM model. Always 0 if FGM model is not activated (combust.in > fgm_model > active ≠ 1).

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>tur_flamespeed</i>	Turbulent flamespeed (m/s) in the cell. Combustion modeling must be active (inputs.in > <i>feature_control</i> > <i>combustion_flag</i> = 1).
<i>zmean</i>	Mixture fraction in the cell for the ECFM, ECFM3Z, and FGM models.
<i>zvar</i>	Mixture fraction variance in the cell for the ECFM, ECFM3Z, and FGM models.

Table 25.131: Turbulence variables in *post.in*. These variables, if included, must be in the *cells > turbulence* settings sub-block.

Variable name	Description
<i>conductivity</i>	Cell thermal conductivity, including the turbulent component ($W/m\cdot K$).
<i>eps</i>	Cell turbulence dissipation rate (m^2/s^3).
<i>omega</i>	Cell specific turbulence dissipation rate (s^{-1})
<i>les_mode</i>	If a DES model is activated (<i>turbulence.in</i> > <i>turbulence_model</i> = <i>DDES_K_OMEGA_SST</i> or <i>IDDES_K_OMEGA_SST</i>), this variable specifies the regime that the cell is under. 0 = Cell has RANS behavior, 1 = Cell has LES behavior.
<i>strain_rate</i>	All the components ($s_{11}, s_{12}, s_{13}, s_{22}, s_{23}, s_{33}$) of the strain rate tensor for turbulence in the cell (s^{-1}).
<i>s11, s12, s13, s22, s23, s33</i>	Each component of the strain rate tensor for turbulence in the cell (s^{-1}).
<i>sgs_eps</i>	Sub-grid scale dissipation (m^2/s^3).
<i>sgs_ke</i>	Sub-grid scale kinetic energy (m^2/s^2).
<i>tke</i>	Cell turbulent kinetic energy (m^2/s^2).
<i>turb_length</i>	Turbulent length scale in the cell (m).
<i>turb_velocity</i>	Magnitude of turbulent velocity in the cell (m/s).
<i>turb_viscosity</i>	Cell viscosity (dynamic), including the turbulent component ($N\cdot s/m^2$).
<i>viscosity_ratio</i>	Ratio of turbulent viscosity and molecular viscosity.
<i>yplus</i>	Dimensionless wall distance.
<i>q_criterion</i>	Second invariant of the velocity gradient tensor.

Table 25.132: Boundary variables in *post.in*. These variables, if included, must be in the *cells > boundary* settings sub-block.

Variable name	Description
<i>attach</i>	Number of boundaries attached to the cell.

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>bound</i>	Indicates if a boundary is attached to the cell. 0 = No boundaries are attached to the cell, 1 = At least one boundary is attached to the cell.
<i>bound_flux</i>	Heat flux at wall boundary (W/m^2).
<i>bound_htc</i>	Heat transfer coefficient ($W/m^2\text{-}K$) for convection boundaries. This value should be equal to that specified in boundary.in .
<i>bound_t_ref</i>	Boundary reference temperature for convection boundaries.
<i>bound_temp</i>	Temperature at wall boundary (K).
<i>cell_htc</i>	Heat transfer coefficient at a wall boundary, given by the heat flux divided by the temperature difference ($W/m^2\text{-}K$). If 1D CHT is enabled, this value is multiplied by the scaling factor specified in cht1d.in > boundaries > boundary > <i>htc_scale_factor</i> . CONVERGE writes 0 for solid boundaries.
<i>conv_htc</i>	Convective heat transfer coefficient ($W/m^2\text{-}K$) at the fluid wall boundary of the cell based on the heat flux value when film heat transfer is neglected. In other words, this value considers only Eulerian field convection and, if inputs.in > <i>feature_control</i> > <i>radiation_flag</i> = 1, radiation. If 1D CHT is enabled, this value is multiplied by the scaling factor specified in cht1d.in > boundaries > boundary > <i>htc_scale_factor</i> .
<i>io_mass_flow_rate</i>	Net mass flow rate (kg/s) through the INFLOW and OUTFLOW boundaries.
<i>mag_wall_stress</i>	Magnitude of the wall stress at the specified boundary.
<i>moving</i>	Indicates if a moving boundary is attached to the cell. 0 = No moving boundaries are attached to the cell, 1 = At least one moving boundary is attached to the cell.
<i>rad_bound_flux</i>	Radiation heat flux at wall boundary (W/m^2)
<i>sc_avg_temp</i>	Time-averaged temperature (K) used as the boundary condition for super-cycling .
<i>sc_avg_htc</i>	Time-averaged heat transfer coefficient ($W/m^2\text{-}K$) used as the boundary condition for super-cycling .
<i>sc_wall_temp</i>	Super-cycling surface temperature (K).
<i>wall_stress</i>	Wall stress (N/m^2) at the specified boundary.

Table 25.133: Film variables in *post.in*. These variables, if included, must be in the *cells* > *film settings* sub-block.

Variable name	Description
<i>film_ht</i>	Cell liquid film thickness (m).
<i>film_mass</i>	Cell liquid film mass (kg).
<i>film_temp</i>	Cell liquid film temperature (K).

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Table 25.134: Detailed soot model variables in *post.in*. These variables, if included, must be in the *cells > soot* settings sub-block.

Variable name	Description
<i>pm_mom0</i>	Zeroth soot moment (mole of soot per kg of fuel) (PM model).
<i>pm_mom1</i>	First soot moment (mole of soot per kg of fuel) (PM model).
<i>soot_coag</i>	Soot mass density (kg/m^3) due to coagulation in each cell (PM, PSM, and SSM models).
<i>soot_con</i>	Soot mass density (kg/m^3) due to condensation in each cell (PM, PSM, and SSM models).
<i>soot_fr</i>	Soot mass density (kg/m^3) due to fragmentation in each cell (PM, PSM, and SSM models).
<i>soot_mass</i>	Soot mass density (kg/m^3) in each cell (PM, PSM, and SSM models).
<i>soot_mf</i>	Soot mass fraction in each cell (PM, PSM, and SSM models).
<i>soot_num_density</i>	Soot number density (m^{-3}) in each cell.
<i>soot_ox</i>	Soot mass density (kg/m^3) due to oxidation in each cell (PM, PSM, and SSM models).
<i>soot_pi</i>	Soot mass density (kg/m^3) due to particle inception (nucleation) (kg) in each cell (PM, PSM, and SSM models).
<i>soot_size</i>	Diameter of the soot particle (m), which is assumed to be a sphere.
<i>soot_sg</i>	Soot mass (kg/m^3) density due to surface growth in each cell (PM, PSM, and SSM models).
<i>soot_vf</i>	Soot volume fraction in each cell (PM, PSM, and SSM models).

Table 25.135: Physical properties of liquid parcel variables in *post.in*. These variables, if included, must be in the *parcels > liquid_parcels > physical* settings sub-block.

Variable name	Description
<i>density</i>	Parcel density (kg/m^3).
<i>film_flag</i>	0 = Parcel is not in wall film, 1 = Parcel is in wall film, 2 = Rebounded parcels, 3 = Splashed parcels, 4 = Separated parcels, 5 = Stripped parcels.
<i>film_thickness</i>	Thickness of the film associated with the parcel (m).
<i>from_injector</i>	Identifies the injector number from which the parcel originates.
<i>from_nozzle</i>	Identifies the nozzle number from which the parcel originates.
<i>mass</i>	Parcel mass (kg).

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>node_xx</i>	Cell position that contains the parcel.
<i>num_drop</i>	Number of drops per parcel.
<i>parent</i>	If the parcel is a parent drop (close to the injector inlet).
<i>radiation_energy</i>	Cumulative radiation energy absorbed by this parcel.
<i>radius</i>	Parcel radius (<i>m</i>).
<i>surf_temp</i>	Surface temperature of the film associated with the parcel (<i>K</i>).
<i>temp</i>	Parcel temperature (<i>K</i>).
<i>triangle</i>	Identifies the surface triangle(s) that the parcel is associated with.
<i>velocity</i>	All three components of parcel velocity (<i>m/s</i>).

Table 25.136: Model properties of liquid parcel variables in *post.in*. These variables, if included, must be in the *parcels > liquid parcels > model settings* sub-block.

Variable name	Description
<i>distant</i>	Unitless RT model constant.
<i>distort</i>	Cell drop distortion from TAB model.
<i>distort_dot</i>	Time-rate (<i>s⁻¹</i>) of drop distortion in the cell from TAB model.
<i>film_shed</i>	Accumulated parcel mass (<i>kg</i>) for stripped drop calculation.
<i>k_kuhnke</i>	K number from the Kuhnke splash model.
<i>shed_mass</i>	Accumulated parcel mass (<i>kg</i>) for child drop calculation.
<i>shed_num_drop</i>	Number of drops used in <i>shed_mass</i> calculation.
<i>t_turb</i>	Turbulent time scale for fluctuating velocity (<i>s</i>).
<i>tbreak_kh</i>	Drop breakup time (<i>s</i>) for the KH model.
<i>tbreak_rt</i>	Drop breakup time (<i>s</i>) for the RT model.
<i>time_turb_accum</i>	Accumulated turbulent time (<i>s</i>) for the parcel.
<i>uprime</i>	Turbulent x component of parcel velocity (<i>m/s</i>).
<i>vprime</i>	Turbulent y component of parcel velocity (<i>m/s</i>).
<i>wprime</i>	Turbulent z component of parcel velocity (<i>m/s</i>).
<i>weber</i>	Weber number of parcels.

Table 25.135: Physical properties of solid parcel variables in *post.in*. These variables, if included, must be in the *parcels > solid parcels > physical settings* sub-block.

Variable name	Description
<i>density</i>	Parcel density (<i>kg/m³</i>).

Chapter 25: Input and Data Files

Output/Post-Processing Post Variable Selection: post.in

Variable name	Description
<i>from_injector</i>	Identifies the injector number from which the parcel originates.
<i>from_nozzle</i>	Identifies the nozzle number from which the parcel originates.
<i>mass</i>	Parcel mass (kg).
<i>node_xx</i>	The cell that contains the parcel.
<i>num_drop</i>	Number of drops per parcel.
<i>parent</i>	If the parcel is a parent drop (close to the injector inlet).
<i>radiation_energy</i>	Cumulative radiation energy absorbed by this parcel.
<i>radius</i>	Parcel radius (m).
<i>surf_temp</i>	Surface temperature of the film associated with the parcel (K).
<i>temp</i>	Parcel temperature (K).
<i>triangle</i>	Identifies the surface triangle(s) that the parcel is associated with.
<i>velocity</i>	All three components of parcel velocity (m/s).

Table 25.136: Model properties of solid parcel variables in *post.in*. These variables, if included, must be in the *parcels > solid_parcel* settings sub-block.

Variable name	Description
<i>uprime</i>	Turbulent x component of parcel velocity (m/s).
<i>vprime</i>	Turbulent y component of parcel velocity (m/s).
<i>wprime</i>	Turbulent z component of parcel velocity (m/s).

Table 25.137: Sub-block used to list properties of species in parcels in *post.in*. This sub-block, if included, must be in the *parcels > liquid_parcel* or *parcels > solid_parcel* settings block.

Sub-block	Allowed values	Description
<i>massfrac</i>	< <i>species</i> >. The species name must match <i>species.in</i> .	Mass fraction of species. Spray modeling must be active (inputs.in > feature_control > parcel_mode > liquid_parcel or <i>solid_parcel</i> = 1).

Wall Output: *wall_output.in*

To control for which boundaries CONVERGE writes wall output and/or wall stress data, specify a file name (e.g., *wall_output.in*) for [*inputs.in > output_control > wall_output_flag*](#) and include that file in your case setup.

Figure 25.154 shows an example file. In this example, CONVERGE will write boundary information for boundaries 1, 3, and 5; stress information for boundaries 4 and 6; both boundary and stress information for boundaries 2 and 10; and power and torque only for a

Chapter 25: Input and Data Files

Output/Post-Processing Wall Output: wall_output.in

rotation object composed of boundaries 7 and 8 and a rotation object composed of boundary 9.

```
version: 3.1
---

write_wall_boundary:
    boundary_id: [1, 3, 5]

write_wall_stress:
    boundary_id: [4, 6]

write_wall_all:
    boundary_id: [2]

write_wall_power_torque:
    - rotation_object:
        boundary_id: [7, 8]
    - rotation_object:
        boundary_id: [9]
```

Figure 25.154: An example *wall_output.in* file.

Table 25.138: Format of *wall_output.in*.

Parameter	Description
<i>write_wall_boundary</i>	
<i>boundary_id</i>	List of boundary IDs for which CONVERGE will write wall output.
<i>write_wall_stress</i>	
<i>boundary_id</i>	List of boundary IDs for which CONVERGE will write wall stress.
<i>write_wall_all</i>	
<i>boundary_id</i>	List of boundary IDs for which CONVERGE will write wall output and stress.
<i>write_wall_power_torque</i>	
- <i>rotation_object</i>	This settings sub-block specifies the boundary ID of the rotation object. Repeat this sub-block for each rotation object.
<i>boundary_id</i>	List of boundary IDs which make up this rotation object.

Swirl, Tumble, and Angular Momentum: *dynamic.in*

To define parameters related to swirl, tumble, and inter-region angular momentum, set [inputs.in](#) > *output_control* > *dynamic_flag* = 1 and include the *dynamic.in* file in your case setup. Use this option when the cylinders are inclined with respect to the z axis in the surface geometry file (as in a V engine) or if you use multiple cylinders. Note that CONVERGE calculates and can write out the positive and negative components of swirl and tumble.

In general, if the piston motion vector is the z axis, the calculations for swirl ratio, tumble ratio, and angular momentum are straightforward. For cylinders aligned with the z axis, you do not need to use these dynamic input parameters. CONVERGE always calculates the swirl

Chapter 25: Input and Data Files

Output/Post-Processing Swirl, Tumble, and Angular Momentum: dynamic.in

and tumble ratio about the center of mass, except for sector cases where (0,0) is used for the x and y center.

If the piston motion is not along the z axis, CONVERGE can calculate the swirl based on either the piston's direction of motion (*piston_or_vector > rotation_matrix > bound_or_vector = 0*) or any user-specified vector (*bound_or_vector = 1*).

CONVERGE will calculate the tumble internally in manner consistent with the option chosen.

```
version: 3.1
---

piston_or_vector:
  - rotation_matrix:
    region_id: 2
    bound_or_vector: 0
    boundary_id: 9
  - rotation_matrix:
    region_id: 3
    bound_or_vector: 1
    vector_x: [1.0, -1.0, 0.0]
    vector_y: [-1.0, 1.0, -1.0]
    vector_z: [0.0, -1.0, 1.0]
ang_mom_flux:
  - output:
    ang_mom_flux_region_from: 0
    ang_mom_flux_region_to: 1
  - output:
    ang_mom_flux_region_from: 2
    ang_mom_flux_region_to: 3
```

Figure 25.155: An example *dynamic.in* file.

Chapter 25: Input and Data Files

Output/Post-Processing Swirl, Tumble, and Angular Momentum: *dynamic.in*

Table 25.139: Format of *dynamic.in*.

Parameter	Description
<i>piston_or_vector</i>	Each section describes a direction vector (or piston) used to calculate dynamic information for the specified region.
- <i>rotation_matrix</i>	This settings sub-block specifies the parameters to calculate dynamic information for a region. Repeat this sub-block for each vector (or piston) for which to calculate dynamic information.
<i>region_id</i>	ID of the region (must match a <i>region_id</i> in initialize.in) in which swirl and tumble ratio will be calculated.
<i>bound_or_vector</i>	0 = The tumble is calculated along a boundary motion vector, 1 = The tumble is calculated along a user-specified vector.
<i>boundary_id</i>	ID of the boundary (must match a <i>boundary_id</i> in boundary.in) whose motion vector is used to calculate tumble. Used only when <i>bound_or_vector</i> = 0.
<i>vector_x</i>	The x component of the tumble calculation vector. Used only when <i>bound_or_vector</i> = 1.
<i>vector_y</i>	The y component of the tumble calculation vector. Used only when <i>bound_or_vector</i> = 1.
<i>vector_z</i>	The z component of the tumble calculation vector. Used only when <i>bound_or_vector</i> = 1.
<i>ang_mom_flux</i>	Output angular momentum flux.
- <i>output</i>	This settings sub-block specifies the regions between which to calculate angular momentum flux. Repeat this sub-block for each region pair.
<i>ang_mom_flux_region_from</i>	ID of the region (must match a <i>region_id</i> in initialize.in) from which CONVERGE will calculate the angular momentum.
<i>ang_mom_flux_region_to</i>	ID of the region (must match a <i>region_id</i> in initialize.in) to which CONVERGE will calculate the angular momentum. This region ID corresponds to the ID of <i>dynamic_region<region ID>.out</i> .

Flow Between Regions: *regions_flow.in*

The *regions_flow.in* file specifies the inter-region flow parameters when [inputs.in](#) > *output_control* > *region_flow_flag* = 2. This file also controls the species and [passive](#) output for all INFLOW and OUTFLOW boundaries in a simulation.

Use the parameters in this file to customize the output for the mass flow of different species and passives between regions and across INFLOW and OUTFLOW boundaries.

Chapter 25: Input and Data Files

Output/Post-Processing Flow Between Regions: regions_flow.in

```
version: 3.1
---

output_allregions_for_streams: []
inter_regions:
  - region_ids:
      from_region: 1
      to_region: 2
  - region_ids:
      from_region: 3
      to_region: 4
output_allspecies_flag: 0
species:
  - CO2
  - H2O
output_allpassives_flag: 0
passives:
  - soot
output_allparcels_flag: 0
parcel_species:
  liquid:
    - H2O
  solid: []
output_tke_flag: 0
output_sgs_ke_flag: 0
output_eps_flag: 1
output_sgs_eps_flag: 0
output_ca_flag: 0
output_omega_flag: 0
```

Figure 25.156: An example *regions_flow.in* file.

Chapter 25: Input and Data Files

Output/Post-Processing Flow Between Regions: regions_flow.in

Table 25.140: Format of *regions_flow.in*.

Parameter	Description
<i>output_allregions_for_streams</i>	List the streams for which to output mass flows for all adjacent regions (or specify ALL to output mass flows for all adjacent regions in all streams). If the list is empty ([]), CONVERGE outputs mass flows only between regions listed in the <i>inter_regions</i> settings block.
<i>inter_regions</i>	
- <i>region_ids</i>	This settings sub-block specifies the pair of regions for inter-region flow rate calculation. Repeat this sub-block for each pair of regions.
<i>from_region:</i>	Calculate inter-region flow rate from this region.
<i>to_region:</i>	Calculate inter-region flow rate to this region.
<i>output_allspecies_flag</i>	0 = Output mass flows only for the species specified below, 1 = Output mass flows for all species in the domain.
<i>species*</i>	List of species to output.
<i>output_allpassives_flag</i>	0 = Output mass flows only for the passives specified below, 1 = Output mass flows for all passives in the domain.
<i>passives*</i>	List of passives to output.
<i>output_allparcels_flag</i>	0 = Output mass flows only for the parcel species specified below, 1 = Output mass flows for all parcel species in the domain.
<i>parcel_species*</i>	
<i>liquid</i>	List of liquid parcel species to output.
<i>solid</i>	List of solid parcel species to output.
<i>output_tke_flag</i>	For RANS simulations, 0 = Do not output tke flux between regions, 1 = Output tke flux between regions.
<i>output_sgs_ke_flag</i>	For LES simulations, 0 = Do not output sgs_ke flux between regions, 1 = Output sgs_ke flux between regions.
<i>output_eps_flag</i>	For RANS simulations, 0 = Do not output eps flux between regions, 1 = Output eps flux between regions.
<i>output_sgs_eps_flag</i>	For LES simulations, 0 = Do not output sgs_eps flux between regions, 1 = Output sgs_eps flux between regions.
<i>output_ca_flag</i>	0 = Do not output liquid area fraction, 1 = Output liquid area fraction (applies only when volume of fluid modeling is active).
<i>output_omega_flag</i>	0 = Do not output omega flux between regions, 1 = Output omega flux between regions.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

SMD and Drop Distribution: *smd_pdf.in*

To activate the Sauter mean diameter (SMD) and parcel distribution function output, set [*inputs.in*](#) > *output_control* > *smd_pdf_flag* = 1 and include an *smd_pdf.in* file in your case setup. CONVERGE will calculate the SMD and parcel diameter distribution on the plane you specify and write this information to [*smd_pdf.out*](#). You can specify the plane in an arbitrary location and direction, and the plane does not need to be collocated with the CONVERGE computational grid.

```
version: 3.1
---

smd_pdfs:
  - stream:
      stream_id: 0
      point: [0, 0, 0]
      normal: [0, 0, 1]
      thickness: 1e-30
      min_diameter: 1e-30
      max_diameter: 1e-5
      num_bins: 10
```

Figure 25.157: An example *smd_pdf.in* file.

Table 25.141: Format of *smd_pdf.in*.

Parameter	Description
<i>smd_pdfs</i>	
- <i>stream</i>	Stream sub-block. If the file is in the Case Directory, include a separate sub-block for each relevant stream. If the file is in a stream-specific subdirectory, include one sub-block for only that stream.
<i>stream_id</i>	ID of the stream for which to generate output.
<i>point</i>	The xyz location of a point on this plane.
<i>normal</i>	The xyz normal of this plane.
<i>thickness</i>	The thickness of the calculation region collocated with the plane (<i>m</i>).
<i>min_diameter</i>	Minimum parcel diameter (<i>m</i>) for SMD distribution calculation.
<i>max_diameter</i>	Maximum parcel diameter (<i>m</i>) for SMD distribution calculation.
<i>num_bins</i>	Number of bins into which the parcels will be distributed.

Parcel Radius Distribution: *radius_pdf.in*

To activate the parcel radius distribution output, set [*inputs.in*](#) > *output_control* > *smd_pdf_flag* = 3 and include a *radius_pdf.in* file in your case setup. CONVERGE calculates the parcel radius distribution in the region or box that you specify and writes this information to [*radius_pdf.out*](#).

Chapter 25: Input and Data Files

Output/Post-Processing Parcel Radius Distribution: `radius_pdf.in`

```
version: 3.1
---

radius_pdfs:
  - stream:
    stream_id: 0
    region_id: 1
    point_0: [0.028, -0.028, -0.01405]
    point_1: [0.028, 0.028, -0.01405]
    point_2: [-0.028, -0.028, -0.01405]
    height: 0.0141
    min_radius: 0.0
    max_radius: 8.0e-5
    num_bins: 8
```

Figure 25.158: An example `radius_pdf.in` file.

Table 25.142: Format of `radius_pdf.in`.

Parameter	Description
<code>radius_pdfs</code>	
<code>- stream</code>	Stream sub-block. If the file is in the Case Directory, include a separate sub-block for each relevant stream. If the file is in a stream-specific subdirectory, include one sub-block for only that stream.
<code>stream_id</code>	ID of the stream for which to generate output.
<code>region_id</code>	Region for which to output the parcel radius distribution. If you specify a valid region ID, <code>radius_pdf.out</code> contains data for this region. Otherwise, <code>radius_pdf.out</code> contains data for the box defined by <code>point_0</code> , <code>point_1</code> , <code>point_2</code> , and <code>height</code> (see below).
<code>point_0</code>	Point in the plane containing the base of the box.
<code>point_1</code>	Point in the plane containing the base of the box. The first edge vector is defined as $\text{vec1} = \text{point}_1 - \text{point}_0$.
<code>point_2</code>	Point in the plane containing the base of the box. The second edge vector is defined as $\text{vec2} = \text{point}_2 - \text{point}_0$.
<code>height</code>	Height of the box in the normal direction (m). The normal direction is defined by the cross product $\text{vec1} \times \text{vec2}$.
<code>min_radius</code>	Minimum parcel radius to include in the output (m).
<code>max_radius</code>	Maximum parcel radius to include in the output (m).
<code>num_bins</code>	Number of bins in which to divide the output. The <code>radius_pdf.out</code> file contains a column for each bin.

Phase Doppler Particle Analyzer: `pdpa.in`

The Phase Doppler Particle Analyzer (PDPA) feature calculates the size, number, velocity, and flux of parcels and particles through a set of monitored regions. To activate the PDPA feature, set [`inputs.in`](#) > `output_control` > `smd_pdf_flag` = 2 and include a `pdpa.in` file in your case setup. CONVERGE will calculate the parcel statistics on the circular planes you specify and write this information to [`user_PDPA_<time>.out`](#) in the `PDPA_output` subdirectory.

Chapter 25: Input and Data Files

Output/Post-Processing Phase Doppler Particle Analyzer: pdpa.in

CONVERGE also stores a *pdpa_restart*.rst* file in the *restart* subdirectory. The PDPA planes do not need to be collocated with the CONVERGE computational grid, but they must share a common normal and radius.

```
version: 3.1
---

pdpa_inputs:
  - stream:
    stream_id: 0
    normal: [1., 0., 0.]
    pdpa_radius: 0.00025
    pdpa_t_start: 0.005
    pdpa_t_write: 0.005
    pdpa_points:
      - [0.063837362,-0.032004,0]
      - [0.063837362,-0.032004,0]
      - [0.063837362,-0.030004,0]
      - [0.063837362,-0.028004,0]
      - [0.063837362,-0.026003,0]
      - [0.063837362,-0.024003,0]
      - [0.063837362,-0.022003,0]
      - [0.063837362,-0.020003,0]
      - [0.063837362,-0.018002,0]
      - [0.063837362,-0.016002,0]
```

Figure 25.159: An example *pdpa.in* file.

Table 25.143: Format of *pdpa.in*.

Parameter	Description
<i>pdpa_inputs</i>	
- <i>stream</i>	Stream sub-block. If the file is in the Case Directory, include a separate sub-block for each relevant stream. If the file is in a stream-specific subdirectory, include one sub-block for only that stream.
<i>stream_id</i>	ID of the stream for which to generate output.
<i>normal</i>	The xyz normal of the planes.
<i>pdpa_radius</i>	The radius (<i>m</i>) of the planes.
<i>pdpa_t_start</i>	PDPA output start time for this stream (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>pdpa_t_write</i>	PDPA output write interval for this stream (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>pdpa_points</i>	
- [<i>x</i> , <i>y</i> , <i>z</i>]	The xyz coordinates (<i>m</i>) of the center of this plane. Repeat for each plane.

Monitor Points: *monitor_points.in*

Monitor points are locations in the domain at which CONVERGE collects data during the simulation. There are three configuration options: point, line, and cloud. You specify the location of a point, the endpoints of a line and the number of points on the line, or the name

Chapter 25: Input and Data Files

Output/Post-Processing Monitor Points: monitor_points.in

of a file containing spatial-temporal data for a set of points (*i.e.*, cloud). If you specify a box size, CONVERGE will report the average value of each monitored quantity within a cube of that size surrounding the monitor point. You can specify whether the values will be mass-averaged, volume-averaged, or taken from the nearest point on a boundary.

To activate this feature, set *inputs.in* > *output_control* > *monitor_points_flag* = 1 and include a *monitor_points.in* file.

```
version: 3.1
---

monitor_point_coords_out      1
monitor_points:
  - point:
    coordinates: [0.0400, 0.00000, 0.00000]
    box_size: 0.00100
    stream_id: 0
  - point:
    coordinates: [0.04000, 0.01000, 0.00000]
    box_size: 0.001000
    stream_id: 0

monitor_clouds:
  - cloud:
    file_name: "cloud.in"
    box_size: 0
    stream_id: 0

monitor_lines:
  - line:
    num_points_in_line: 4
    end_points:
      - point:
        coordinates: [0.00000, 0.00000, 0.00000]
        box_size: 0.00100
        stream_id: 0
        boundary_id: 1
      - point:
        coordinates: [0.08000, 0.00000, 0.00000]
        box_size: 0.00100
        stream_id: 0
        boundary_id: 3

variable_and_statistical_type:
  pressure: [VOL_AVG, MASS_AVG, BOUND_AVG]
  cp: [VOL_AVG, MASS_AVG]
  temperature: [VOL_AVG, MASS_AVG]
  equiv_ratio: [VOL_AVG, MASS_AVG]
  passives:
    NOX: [VOL_AVG, MASS_AVG]
    G_EQN: [VOL_AVG, MASS_AVG]
  species_massfrac:
    H2O: [VOL_AVG, MASS_AVG]
    CO2: [VOL_AVG, MASS_AVG]
  species_molefrac:
    H2O: [VOL_AVG, MASS_AVG]
    CO2: [VOL_AVG, MASS_AVG]
```

Figure 25.160: An example *monitor_points.in* file.

Table 25.144: Format of *monitor_points.in*.

Parameter	Description
<i>monitor_point_coords_out</i> *	0 = CONVERGE will not output the coordinates of the monitor point,

Chapter 25: Input and Data Files

Output/Post-Processing Monitor Points: monitor_points.in

Parameter	Description
	1 = CONVERGE will output the coordinates of the monitor point.
<i>monitor_points</i> *	
- <i>point</i>	Sub-block for a monitor point. Repeat for each monitor point.
<i>coordinates</i>	The x, y, and z coordinates of this monitor point (<i>m</i>).
<i>box_size</i>	The bounding cube side length (<i>m</i>) for this monitor point. To monitor a single point, set this to zero.
<i>stream_id</i>	The stream ID of the region to be monitored.
<i>boundary_id</i> *	The boundary ID associated with this monitor point.
<i>monitor_clouds</i> *	
- <i>cloud</i>	Sub-block for a monitor cloud. Repeat for each monitor cloud.
<i>file_name</i>	Name of the cloud file (e.g., <i>cloud.in</i>).
<i>box_size</i>	The bounding cube side length (<i>m</i>) for the points in the cloud. To monitor a single point at each location, set this to zero.
<i>stream_id</i>	The stream ID of the region to be monitored.
<i>monitor_lines</i> *	
- <i>line</i>	Sub-block for a monitor line. Repeat for each monitor line.
<i>num_points_in_line</i>	The number of points in this monitor line.
<i>end_points</i>	
- <i>point</i>	Sub-block for an end point of a monitor line. Include two of these sub-blocks (one for each end point).
<i>coordinates</i>	The x, y, and z coordinates of this monitor line end point (<i>m</i>).
<i>box_size</i>	The bounding cube side length (<i>m</i>) for the points on this monitor line. To monitor a single point at each location, set this to zero.
<i>stream_id</i>	The stream ID of the region to be monitored. The two end points can have different stream IDs.
<i>boundary_id</i> *	The boundary ID associated with this end point.
<i>variable_and_statistical_type</i>	
< <i>variable name</i> >*	The corresponding statistical type for this variable. Available options are <i>VOL_AVG</i> , <i>MASS_AVG</i> , and <i>BOUND_AVG</i> . Repeat for all desired variables from Table 25.145.
<i>passives</i>	
< <i>passive name</i> >	The corresponding statistical type for this passive variable. Available options are <i>VOL_AVG</i> , <i>MASS_AVG</i> , and <i>BOUND_AVG</i> . Repeat for all desired passives.

Chapter 25: Input and Data Files

Output/Post-Processing Monitor Points: monitor_points.in

Parameter	Description
<i>species_massfrac</i>	
< <i>species name</i> >	The corresponding statistical type for this species. Available options are VOL_AVG, MASS_AVG, and BOUND_AVG. Repeat for all desired species.
<i>species_molefrac*</i>	
< <i>species name</i> >	The corresponding statistical type for this species. Available options are VOL_AVG, MASS_AVG, and BOUND_AVG. Repeat for all desired species.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Table 25.145: Variables (except for species and passives) that can be monitored via monitor_points.in.

Quantity (units)	Variable name
Pressure (Pa)	<i>pressure</i>
Volume (m^3)*	<i>volume</i>
Mass (kg)*	<i>mass</i>
Density (kg/m^3)	<i>density</i>
Heat capacity at constant pressure (J/kg-K)	<i>cp</i>
Heat capacity at constant volume (J/kg-K)	<i>cv</i>
Ratio of specific heats	<i>gamma</i>
Mach number	<i>mach</i>
Specific internal energy (J/kg)	<i>sie</i>
Equivalence ratio	<i>equiv_ratio</i>
React ratio	<i>reac_ratio</i>
Temperature (K)	<i>temperature</i>
Turbulent kinetic energy (m^2/s^2)	<i>tke</i>
Turbulence dissipation rate (m^2/s^3)	<i>eps</i>
Specific dissipation rate (s^{-1})	<i>omega</i>
U velocity component (m/s)	<i>u</i>
V velocity component (m/s)	<i>v</i>
W velocity component (m/s)	<i>w</i>
Mixture fraction	<i>mix_frac</i>
Conductivity (W/m-K)	<i>cond</i>
Viscosity ($N\cdot s/m^2$)	<i>visc</i>

Chapter 25: Input and Data Files

Output/Post-Processing Monitor Points: monitor_points.in

Quantity (units)	Variable name
Dimensionless wall distance	<i>yplus</i>
Void fraction	<i>alpha</i>

* Note: CONVERGE prints the total value of this variable in the monitored region, regardless of whether it is specified MASS_AVG or VOL_AVG. BOUND_AVG is not available for this variable.

Note that CONVERGE contains two other monitor point options: a UDF monitor point option and a [super-cycle monitor point option](#). Refer to the CONVERGE 3.1 UDF Manual for information about using UDFs to set up monitor points.

In addition to monitor points you specify in CONVERGE, you can set up monitor points through the Probe option in Tecplot after you run a simulation. Please refer to the Tecplot User Manual for instructions on how to set up a probe.

Monitor Cloud Data: *cloud.in*

The monitor cloud file (*e.g.*, *cloud.in*) specifies the locations of the points in a monitor cloud. Enter the name of this file in [*monitor_points.in*](#) > *monitor_clouds* > *cloud* > *file_name*.

Figures 25.161 and 25.162 below show example cloud files. In these examples, the locations of the points vary in time. The temporal variation is sequential in Figure 25.161 and cyclic in Figure 25.162. To provide location data for a set of stationary points, use the sequential format with data for a single time.

Sequential Temporal Variation

For sequential temporal variation, enter the keyword *SPATIAL* in the first row. In the next six rows, enter the scaling value (*scale_xyz*); translation values for x (*trans_x*), y (*trans_y*), and z (*trans_z*); and the rotation axis (*rot_axis*) and angle (*rot_angle*).

The subsequent blocks of data contain the x, y, and z coordinates of the points in the monitor cloud at specified points in time. In the first row of each block, specify the time value followed by *second* (if [*inputs.in*](#) > *simulation_control* > *crank_flag* = 0) or *crank* (if *crank_flag* is non-zero). In the next row, enter *x y z*. In each of the following rows, specify the x, y, and z coordinates of each point in the cloud at that time. The blocks must be listed in order from the minimum time to the maximum time. Each block must contain the same number of rows.

To determine the position of a given point at a given time, CONVERGE interpolates between the positions at the minimum time and the maximum time. For times earlier than the minimum time, CONVERGE uses the position at the minimum time. For times later than the maximum time, CONVERGE uses the position at the maximum time.

```
SPATIAL
1.0      scale_xyz
0.0      trans_x
0.0      trans_y
0.0      trans_z
x        rot_axis
0.0      rot_angle

0.0      second
x        y          z
0.005   0.005     0.00
0.005   0.005     0.025
0.005   0.005     0.05

0.01    second
x        y          z
0.005   0.005     0.075
0.005   0.005     0.10
0.005   0.005     0.125
.
.
```

Figure 25.161: Excerpt of a cloud file that varies sequentially in time.

Cyclic Temporal Variation

For cyclic temporal variation, enter the keyword *SPATIAL_CYCLIC* in the first row, followed by the period of the cycle in *seconds* (if *inputs.in* > *simulation_control* > *crank_flag* = 0) or *crank angle degrees* (if *crank_flag* is non-zero). In the next six rows, enter the scaling value (*scale_xyz*); translation values for x (*trans_x*), y (*trans_y*), and z (*trans_z*); and the rotation axis (*rot_axis*) and angle (*rot_angle*).

The subsequent blocks of data contain the x, y, and z coordinates of the points in the monitor cloud at different points in the cycle. In the first row of each block, specify the time value followed by *second* (if the period is specified in *seconds*) or *crank* (if the period is specified in *crank angle degrees*). In the next row, enter x y z. In each of the following rows, specify the x, y, and z coordinates of each point in the cloud at that time. The blocks must be listed in order of increasing time. The difference between the time for the last block and the time for the first block must be equal to the period, and the data in the last block must be identical to the data in the first block. Each block must contain the same number of rows.

To determine the position of a given point at a given time, CONVERGE interpolates between the positions at the two nearest times based on where the given time lies within the cycle. For example, with the data shown below in Figure 25.162, at a time of 0.015 *seconds*, CONVERGE determines that the time lies in the first part of the cycle and interpolates between [0.0, 0.0, 0.0] and [0.0, 0.0, 0.099999].

```

SPATIAL_CYCLIC 0.01
1           scale_xyz
0.0         trans_x
0.0         trans_y
0.0         trans_z
x           rot_axis
0.0         rot_angle

0.0         second
x           y           z
0.0         0.0         0.0

0.0099999  second
x           y           z
0.0         0.0         0.099999

0.01        second
x           y           z
0.0         0.0         0.0
.
.
```

Figure 25.162: Excerpt of a cloud file that varies cyclically in time.

Uniformity Index: uniformity_index.in

To activate the uniformity index output, set [inputs.in](#) > *output_control* > *uniformity_index_output_flag* = 1 and include a *uniformity_index.in* file in your case setup. CONVERGE will calculate uniformity index on the planes you specify and write this information to [planes_flow.out](#). You can specify the planes in arbitrary locations and directions, and the planes do not need to be collocated with the CONVERGE computational grid.

There are two ways to calculate the uniformity index. To calculate the uniformity index using the L1 norm,

$$Y_{L1,\bar{\phi}} = 1 - \frac{1}{2} \sum_{cell=1}^{Ncells} \frac{A_{cell} |\bar{\phi}_{cell} - \bar{\phi}_{mean}|}{A_{plane} \bar{\phi}_{mean}}. \quad (25.90)$$

To calculate the uniformity index using the L2 norm,

$$Y_{L2,\bar{\phi}} = 1 - \frac{1}{2\bar{\phi}_{mean}} \sqrt{\sum_{cell=1}^{Ncells} \frac{A_{cell}}{A_{plane}} (\bar{\phi}_{cell} - \bar{\phi}_{mean})^2}. \quad (25.91)$$

In the above equations

$$\bar{\phi}(x) = \frac{1}{T_2 - T_1} \int_{T_1}^{T_2} \phi(x, t) dt \quad (25.92)$$

Chapter 25: Input and Data Files

Output/Post-Processing Uniformity Index: uniformity_index.in

is calculated by turbulent statistics and $\bar{\phi}$ can be species mass fraction, species mole fraction, or normal velocity.

```
version: 3.1
---

output_control:
    uniformity_index_option: L2
    mean_max_min_flag: 1
slice_definition:
    - slice:
        name: uniformity_slice_1
        region_id: 0
        point: [0, 0, 0]
        normal: [0, 0, 1]
    - slice:
        name: uniformity_slice_2
        region_id: 0
        point: [0, 0, 0.1]
        normal: [0, 1, 1]
cells:
    geometry: idreg
    general: [velocity, pressure, density, temperature]
    turbulence: [tke, eps, visc]
```

Figure 25.163: An example *uniformity_index.in* file.

Table 25.146: Format of the *output_control* settings block.

Parameter	Description
<i>output_control</i>	
<i>uniformity_index_option</i>	0 = Do not calculate the uniformity index, L1 = Calculate the uniformity index using the L1 norm, L2 = Calculate the uniformity index using the L2 norm.
<i>mean_min_max_flag</i>	0 = Do not output mean, maximum, and minimum of each cell quantity on each plane, 1 = Output mean, maximum, and minimum of each cell quantity on each plane.

Chapter 25: Input and Data Files

Output/Post-Processing Uniformity Index: uniformity_index.in

Table 25.147: Format of the *slice_definition* settings block.

Parameter	Description
<i>slice_definition</i>	
- <i>slice</i>	This settings sub-block specifies a slice for which to calculate uniformity index. Repeat this sub-block for each slice.
<i>name*</i>	Name of this slice.
<i>region_id*</i>	Region ID for this slice.
<i>point</i>	The xyz location of a point on this slice.
<i>normal</i>	The xyz normal of this slice.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

The *cells* settings block is formatted identically to the [cells](#) setting block in [post.in](#). Refer to the [post.in](#) section for information about this settings block.

For the *cells > general* sub-block of *uniformity_index.in*, CONVERGE offers several additional uniformity index variables that are not available in *post.in*.

Specify *bar_velocity* to calculate the time-averaged velocity calculated from turbulence statistics. This variable is available only when *Turbulence_statistics > turb_stat_flag = 1* in [turbulence.in](#) and the following non-transport passives are included in [species.in](#): *bar_u*, *bar_v*, and *bar_w*.

Specify *vol_frac_uniformity_index* to calculate the volume fraction for spray cases. This variable requires at least one of *inputs.in > feature_control > parcel_control > liquid_parcel*, *solid_parcel*, or *gas_parcel = 1* and *output_control > uniformity_index_output_flag = 1*.

Custom Species Output: *species_output.in*

To direct CONVERGE to generate customized output files containing results for species total mass, mass fraction, standard deviation of mass fraction, and mole fraction, set [inputs.in > output_control > species_output_flag = species_output.in](#) and include a *species_output.in* file in your case setup.

```
version: 3.1
---
total_mass:
  - N2
  - O2
total_vol:
  - all
mass_fraction_std:
  - O2
mole_fraction:
  - all
```

Figure 25.164: An example *species_output.in* file. This file will generate total mass data for N2 and O2, standard deviation of the mass fraction for O2, and total volume and mole fraction data for all of the species in [species.in](#) or the [reaction mechanism file](#).

Chapter 25: Input and Data Files

Output/Post-Processing Custom Species Output: species_output.in

Table 25.148: Format of *species_output.in*.

Parameter	Description
<i>total_mass</i> *	<species name> CONVERGE will output the total mass of this species to <i>species_mass.out</i> . Repeat for each species. To output every species, specify <i>all</i> .
<i>total_vol</i> *	<species name> CONVERGE will output the total volume of this species to <i>species_vol.out</i> . Repeat for each species. To output every species, specify <i>all</i> . Only VOF species are available.
<i>mass_fraction</i> *	<species name> CONVERGE will output the mass fraction of this species to <i>species_mass_frac.out</i> . Repeat for each species. To output every species, specify <i>all</i> .
<i>mass_fraction_std</i> *	<species name> CONVERGE will output the standard deviation of the mass fraction of this species to <i>species_std_masfrac.out</i> . Repeat for each species. To output every species, specify <i>all</i> .
<i>mole_fraction</i> *	<species name> CONVERGE will output the mole fraction of this species to <i>species_mole_frac.out</i> . Repeat for each species. To output every species, specify <i>all</i> .
<i>total_mass_src</i> *	<species name> CONVERGE will output the mass source of this species to <i>species_mass_src.out</i> . Repeat for each species. To output every species, specify <i>all</i> .

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

After the name of the settings block, list the name(s) of the species for which you would like CONVERGE to record this quantity or type the word *all* to include output for all of the species listed in [*species.in*](#) or the [*reaction mechanism file*](#).

If you do not include a settings block, or if you specify the term *none*, CONVERGE will not write output for this quantity.

Mapping File Frequency: `write_map.in`

When [*inputs.in*](#) > *output_control* > *write_map_flag* = 1 in , CONVERGE writes [*map_<time>.h5*](#) file(s) at the simulation time(s) specified in *write_map.in*. You can use these output files to initialize a new simulation. If a simulation has discrete phase modeling (*i.e.*, if [*inputs.in*](#) > *feature_control* > *parcel_mode* > *liquid_parcel*, *solid_parcel*, or *gas_parcel* = 1), CONVERGE also writes [*map_parcel_<time>.h5*](#) file(s) at the simulation time(s) specified in *write_map.in*. For example, if you direct CONVERGE to write file(s) at 100 crank angle degrees, CONVERGE will

Chapter 25: Input and Data Files

Output/Post-Processing

Mapping File Frequency: write_map.in

write *map_1.000000e+02.h5* and (for a simulation with discrete phase modeling) *map_parcel_1.000000e+02.h5*.

When *inputs.in* > *output_control* > *write_map_flag* = 1, CONVERGE also writes *map_bound<boundary ID><time>.h5* files (e.g., *map_bound2_1.000000e+02.h5*) for each INFLOW and OUTFLOW boundary.

```
version: 3.1
---

temporal_type: CYCLIC
temporal_period: 360
map_write_time: [0.00035, 0.00050, 0.00075, 0.00100]
```

Figure 25.165: An example *write_map.in* file.

Table 25.149: Format of *write_map.in*.

Parameter	Description
<i>temporal_type</i>	SEQUENTIAL or CYCLIC. If CYCLIC, supply the <i>period</i> after <i>temporal_period</i> .
<i>temporal_period</i>	The CYCLIC period (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
<i>map_write_time</i>	Time(s) (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) at which CONVERGE will write a map file.

Proper Orthogonal Decomposition: pod.in

When you perform a [proper orthogonal decomposition](#) (POD) with the *pod* utility, the utility automatically generates a *pod.in* file. You can alter this file for subsequent utility runs.

```
<input type>
pod
<pod domain center>
0
0
0
<pod domain size>
1e+25
1e+25
1e+25
<pod modes output>
6
<pod modes reconstruct>
0
<cell variables>
logic_ijk
level
pressure
temp
<files>
post000765 +8.88257e+00.out
post000766 +8.88302e+00.out
post000767 +8.88359e+00.out
post000768 +8.88401e+00.out
post000769 +8.88454e+00.out
post000770 +8.88507e+00.out
```

Figure 25.166: Sample *pod.in* file.

Chapter 25: Input and Data Files

Output/Post-Processing Proper Orthogonal Decomposition: pod.in

Table 25.150: Format of the *pod.in* file.

Parameter	Description
<i>input type</i>	Input type <i>pod</i> to perform the POD calculation. Other values reserved for future use.
<i>pod domain center</i>	Coordinate x, y, and z of the center of the domain for the POD calculation. By default, this is the origin.
<i>pod domain size</i>	Domain extent in x, y, and z for the POD calculation. The calculation domain is a box of these dimensions. By default, these extents are 1e25.
<i>pod modes output</i>	The number of modes to calculate. By default, this is equal to the number of files provided. You can reduce, but not increase, this number.
<i>pod modes reconstruct</i>	The number of modes to use to reconstruct the flow field. The default value is zero, which will not perform the reconstruction.
<i>cell variables</i>	Cell variables requested for the POD calculation. Refer to the <i>post.in</i> section for variable options.
<i>files</i>	List of <i>post<output number>_<time>.out</i> files on which to perform the POD calculation.

In Situ Post-Processing: *paraview_catalyst.in*

In situ post-processing (also referred to as co-processing) allows you to post-process results in real time during a CONVERGE simulation. This co-processing capability includes extracting and saving a subset of the domain (e.g., a slice) and image export. Because co-processing is performed on data already in memory rather than reading from disk, it can substantially reduce user interaction time and storage requirements for some workflows.

CONVERGE uses ParaView Catalyst in situ post-processing. This feature is available only when running a simulation on Linux. To activate in situ post-processing, set [*inputs.in*](#) > *output_control* > *paraview_catalyst_flag* = 1 and include a *paraview_catalyst.in* file in your case setup. Refer to the ParaView Catalyst for CONVERGE Setup Guide for information about running a simulation with ParaView Catalyst.

Chapter 25: Input and Data Files

Output/Post-Processing In Situ Post-Processing: paraview_catalyst.in

```
version: 3.1
---
output_surfaces: 1
scripts:
  - script:
      script_name: full_volume_output.py
      timing_control:
        temporal_type: SEQUENTIAL
        start_time: 0.001
        end_time: 0.005
        twrite: 0.1
        trigger_at_finalize: 1
  - script:
      script_name: spray.py
      timing_control:
        temporal_type: CYCLIC
        start_time: -30.0
        end_time: 0.0
        period: 720.0
        twrite: 0.1
  - script:
      script_name: slice.py
      timing_control:
        temporal_type: FIXED TIME
        fixed_time: [0, 5, 22]
```

Figure 25.167: An example *paraview_catalyst.in* file.

Chapter 25: Input and Data Files

Output/Post-Processing In Situ Post-Processing: `paraview_catalyst.in`

Table 25.151: Format of `paraview_catalyst.in`

Parameter	Description
<code>output_surfaces</code>	0 = Output 3D field only, 1 = Output 3D field and surfaces.
<code>scripts</code>	
<code>script</code>	Block to add in situ post-processing script.
<code>script_name</code>	Name of in situ post-processing script.
<code>timing_control</code>	
<code>temporal_type</code>	<code>SEQUENTIAL</code> , <code>CYCLIC</code> or <code>FIXED_TIME</code> .
<code>period</code>	The <code>CYCLIC</code> period (in seconds if <code>inputs.in > simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero). Used only when <code>temporal_type</code> = <code>CYCLIC</code> .
<code>start_time</code>	Start time for co-processing execution (in seconds if <code>inputs.in > simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero). Used only when <code>temporal_type</code> = <code>SEQUENTIAL</code> or <code>CYCLIC</code> .
<code>end_time</code>	End time for co-processing execution (in seconds if <code>inputs.in > simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero). Used only when <code>temporal_type</code> = <code>SEQUENTIAL</code> or <code>CYCLIC</code> .
<code>twrite</code>	Interval for co-processing execution (in seconds if <code>inputs.in > simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero). Used only when <code>temporal_type</code> = <code>SEQUENTIAL</code> or <code>CYCLIC</code> .
<code>fixed_time</code>	List of timestamps for co-processing execution (in seconds if <code>inputs.in > simulation_control > crank_flag</code> = 0 or in crank angle degrees if <code>crank_flag</code> is non-zero).
<code>trigger_at_finalize*</code>	0 = Off, 1 = Execute co-processing script at the end of simulation (default).

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

If you include a `paraview_catalyst.in` file in the root directory, CONVERGE will execute the situ post-processing scripts on all streams. For executing in situ post-processing scripts on a specific stream, you must include `paraview_catalyst.in` file in that stream's sub-directory.

We recommend generating ParaView Catalyst in situ post-processing scripts (e.g., `slice.py`) using a graphical user interface. There are two ways to do this. You can specify a basic function (e.g., slices or isosurfaces) in CONVERGE Studio. Alternately, you can create basic or advanced scripts in Studio ParaView or a stand-alone instance of ParaView.

To set up a Catalyst co-processing script in Paraview, you must first generate a CONVERGE *post*.h5* file for this Case Setup, or locate an output file from a similar setup. Load *post*.h5* in the ParaView GUI and set up a "pipeline" of post-processing commands. Once the desired post-processing is complete, save the pipeline as a Catalyst script (*File > Save Catalyst State*).

Refer to the ParaView documentation for how to set up post-processing pipelines.

25.11 User-Defined Functions

This section describes the input files that contain information about user-defined functions (UDFs) for your CONVERGE simulation.

User-Defined Functions: *udf.in*

You can write user-defined functions (UDFs) to add models or features to CONVERGE. Before running a simulation with UDFs, consult the CONVERGE 3.1 UDF Manual, which describes how to write, compile, and execute UDF code.

To activate UDFs, set *inputs.in* > *feature_control* > *udf_flag* = 1 and include a *udf.in* file in your case setup. Many parameters are available for this file and they are all optional. For information about the parameters used to activate specific UDFs, refer to the CONVERGE 3.1 UDF Manual.

```
version: 3.1
---

user_post_flag:    1
```

Figure 25.168: An example *udf.in* file.

25.12 Chemistry

This section describes the input files related to the chemistry utilities in CONVERGE.

Zero-Dimensional Chemistry Utilities

This section describes the input files related to the 0D chemistry utilities in CONVERGE.

0D Solver: *zero_d_solver.in*

To run the zero-dimensional chemistry solver, CONVERGE requires both *zero_d_cases.in* (or *zero_d_template.in*) and *zero_d_solver.in*. The *zero_d_solver.in* file contains the required 0D solver input parameters.

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

```
version: 3.1
---

zero_d_input_control:
    mechanism_filename: mech.dat
    thermodynamic_filename: therm.dat
    case_type: CONSTANT_VOLUME
    species_fraction_input_type: MOLE
    lhv_flag: 0
    ron_mon_table_input_flag: 0

zero_d_solver_control:
    ode_solver: DENSE
    analyt_jac: 0
    rel_tol: 1.e-08
    abs_tol: 1.e-20
    reaction_multiplier: 1.0
    end_time_flag: END
    case_time_limit: 600

zero_d_output_control:
    output_file_flag: 0
    end_time_species: [NC12H26, IC8H18, PHC3H7, C9H12, O2, CO, CO2]
    species_fraction_output_type: MASS
    double_ignition_delay_flag: 0
    ga_flag: 0
    generate_tki_table_flag: 0
    no_filter_output_flag: 0
    failed_cases_output_flag: 0
    cvode_error_output_flag: 0
    ron_mon_table_output_control:
        low: 0
        high: 100
        iso_octane: IC8H18
        n_heptane: NC7H16

zero_d_egr_ceq_species_control:
    subset_species_flag: 1
    subset_species:
        - CO
        - N2

zero_d_sensitivity_control:
    sensitivity_flag: ADJOINT
    adjoint_control:
        adjoint_rel_tol: 0.0001
        adjoint_abs_tol: 1e-14
        adjoint_variables:
            - TEMPERATURE
            - CH2

# optional block for WSR
well_stirred_reactor_control:
    steady_solver: 0
    energy_solver: 1
    find_turning: OFF
    surface_chemistry: 0
    surface_mechanism_filename: surface_mech.dat
    surface_thermodynamic_filename: surface_therm.dat
```

Figure 25.169: An example *zero_d_solver.in* file.

Table 25.152: Format of *zero_d_solver.in*.

Parameter	Description
<i>zero_d_input_control</i>	
<i>mechanism_filename</i>	Mechanism file name (e.g., <i>mech.dat</i>).

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
<i>thermodynamic_filename</i>	Thermodynamic data file name (e.g., <i>therm.dat</i>).
<i>case_type</i>	<ul style="list-style-type: none">• CONSTANT_VOLUME• CONSTANT_PRESSURE• CONSTANT_TEMPERATURE_PRESSURE• CEQ_CONSTANT_ENTHALPY_PRESSURE• CEQ_CONSTANT_TEMPERATURE_PRESSURE• VARIABLE_VOLUME• ENGINE• ENGINE_RON• ENGINE_MON• ENGINE_RON_MON• ENGINE_RON_MON_TABLE• WELL_STIRRED_REACTOR• CONSTANT_PRESSURE_PLUG_FLOW_REACTOR• CVODE_CONSTANT_VOLUME: When <i>combust.in</i> > <i>sage_model</i> > <i>option</i> = CONSTANT_VOLUME and <i>sage_model</i> > <i>cvode_error_output_flag</i> = 1, CONVERGE will write this case type. This case type allows the 0D solver to read without truncation high-precision variables (integration time, temperature, pressure, and custom species mass fractions) from cells with CVODE errors.• CVODE_CONSTANT_PRESSURE: When <i>combust.in</i> > <i>sage_model</i> > <i>option</i> = CONSTANT_PRESSURE and <i>sage_model</i> > <i>cvode_error_output_flag</i> = 1, CONVERGE will write this case type. This case type allows the 0D solver to read without truncation high-precision variables (integration time, temperature, pressure, and custom species mass fractions) from cells with CVODE errors.• CVODE_CONSTANT_TEMPERATURE_PRESSURE: When <i>combust.in</i> > <i>sage_model</i> > <i>cvode_error_output_flag</i> = 1 and <i>sage_model</i> > <i>ode_solver</i> > <i>solve_temp</i> = 0, CONVERGE will write this case type. This case type allows the 0D solver to read without truncation high-precision variables (integration time, temperature, pressure, and custom species mass fractions) from cells with CVODE errors.
<i>species_fraction_input_type</i>	<i>MASS</i> = Input is in mass fraction format, <i>MOLE</i> = Input is in mole fraction format (recommended if <i>zero_d_solver.in</i> > <i>zero_d_output_control</i> > <i>generate_tki_table_flag</i> = 1).
<i>lhw_flag</i>	0 = Use default species lower heating value (LHV), 1 = Input the species LHV (requires <i>lhw.in</i>).
<i>ron_mon_table_input_flag</i>	Specify the correlation table to use for RON and/or MON determination. Used only when <i>case_type</i> = ENGINE_RON, ENGINE_MON, or ENGINE_RON_MON. 0 = Use internal RON/MON correlation table,

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
	1 = Read <i>ron_mon_table.dat</i> .
<i>zero_d_solver_control</i>	
<i>ode_solver</i>	Options for different ordinary differential equation (ODE) solvers: <i>DENSE</i> = CVODE dense solver (recommended when the total number of species is no greater than 100), <i>ITERATIVE</i> = CVODE with preconditioned iterative solver, <i>ITERATIVE_SUPERLU</i> = CVODE with preconditioned iterative solver with SuperLU (SuperLU, 2011), <i>ITERATIVE_CONVERGE</i> = CVODE with optimized preconditioned iterative solver (recommended when the total number of species is greater than 100).
<i>analyt_jac</i>	Flag to specify if Jacobian matrix is solved analytically or numerically in the SAGE solver. 0 = Calculate the derivatives in the Jacobian matrix numerically, 1 = Calculate the derivatives in the Jacobian matrix analytically (recommended because the simulation typically runs faster).
<i>rel_tol</i>	Relative iteration error for each species. Recommended value is 1e-8.
<i>abs_tol</i>	Absolute iteration error for each species. Recommended value is 1e-20.
<i>reaction_multiplier</i>	Scaling factor for reaction rates.
<i>end_time_flag</i>	<i>IGN</i> = Halt the 0D simulation after ignition, <i>END</i> = Halt the 0D simulation at <i>zero_d_cases.in</i> > <i>zero_d_cases</i> > <i>case</i> > <i>integration_time</i> or <i>zero_d_template.in</i> > <i>integration_time</i> .
<i>case_time_limit</i>	Time limit in <i>seconds</i> for each case. If the simulation reaches the time limit, the case exits (fails).
<i>zero_d_output_control</i>	
<i>output_file_flag</i>	0 = Write only <i>ignition_det.out</i> , 1 = Write all 0D *.out files. Note that for <i>zero_d_solver.in</i> > <i>zero_d_input_control</i> > <i>case_type</i> = <i>CEQ_*</i> , CONVERGE will always write <i>zero_d_sol_case<case ID>.out</i> .
<i>end_time_species</i> *	List of species for which CONVERGE writes end-of-simulation mass fractions (if <i>species_fraction_output_type</i> = <i>MASS</i>) or mole fractions (if <i>species_fraction_output_type</i> = <i>MOLE</i>) to <i>zero_d_end_time_species.out</i> .
<i>species_fraction_output_type</i>	<i>MASS</i> = Write species information in output files in mass fraction, <i>MOLE</i> = Write species information in output files in mole fraction.
<i>double_ignition_delay_flag</i>	0 = Do not write double ignition delay information,

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
	1 = Enable output of double ignition delay information (requires <i>zero_d_solver.in</i> > <i>zero_d_sensitivity_control</i> > <i>sensitivity_flag</i> = OFF).
<i>ga_flag</i>	0 = Do not write output for a genetic algorithm simulation, 1 = Write output for a genetic algorithm simulation (required for mechanism tune with CONGO).
<i>generate_tki_table_flag</i>	0 = Do not activate tabulated kinetics of ignition (TKI) table setup and generation, 1 = Activate TKI table setup and generation and read <i>zero_d_template.in</i> instead of <i>zero_d_cases.in</i> (requires <i>zero_d_solver.in</i> > <i>zero_d_input_control</i> > <i>case_type</i> = CONSTANT_PRESSURE [recommended] or CONSTANT_VOLUME. Note that if <i>zero_d_input_control</i> > <i>case_type</i> = CONSTANT_VOLUME, we do not recommend using the TKI table file for a 3D simulation).
<i>no_filter_output_flag</i>	0 = Do not filter output, 1 = Filter output.
<i>failed_cases_output_flag</i>	0 = Do not write a <i>zero_d_cases_FAILED.in</i> file for cases that fail, 1 = Write a <i>zero_d_cases_FAILED.in</i> file with structure identical to <i>zero_d_cases.in</i> containing case data for all the files that failed to converge in <i>zero_d_solver.in</i> > <i>zero_d_solver_control</i> > <i>case_time_limit</i> .
<i>cicode_error_output_flag</i> *	0 = Do not write species and reaction information for cases with CVODE errors, 1 = Write species and reaction information for each case that ran with CVODE errors. CVODE errors do not always terminate the case.
<i>ron_mon_table_output_control</i> *	Required when <i>zero_d_solver.in</i> > <i>zero_d_input_control</i> > <i>case_type</i> = ENGINE_RON_MON_TABLE.
<i>low</i>	Low PRF number for table generation, minimum is 0.
<i>high</i>	High PRF number for table generation, maximum is 100.
<i>iso_octane</i>	Isooctane species name found in mechanism file.
<i>n_heptane</i>	n-Heptane species name found in mechanism file.
<i>zero_d_egr_ceq_species_control</i>	
<i>subset_species_flag</i>	0 = Do not limit the number of species used in the CEQ calculations of the EGR mixture, 1 = Enable a limit on the number of species used in the CEQ calculations of the EGR mixture.
<i>subset_species</i>	This settings sub-block contains the list of species to include in the CEQ calculation of the EGR mixture. This settings sub-block

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
	is used if zero_d_cases.in > zero_d_cases > case > egr_mixture_type = CEQ and subset_species_flag = 1 above.
<subset species name>	Subset species name. Repeat this line for each species. Each species must be on a separate line.
zero_d_sensitivity_control	This settings block is used only when zero_d_solver.in > zero_d_input_control > case_type = CONSTANT_PRESSURE or CONSTANT_VOLUME and zero_d_output_control > generate_tki_table = 0.
sensitivity_flag	<i>OFF</i> = Do not perform a sensitivity analysis, <i>ADJOINT</i> = Perform an adjoint sensitivity analysis (requires zero_d_solver.in > zero_d_solver_control > ode_solver = DENSE), <i>FORWARD</i> = Perform a forward (time-varying) sensitivity analysis (requires zero_d_solver.in > zero_d_solver_control > ode_solver = DENSE).
adjoint_control	This settings block is used only when sensitivity_flag = ADJOINT above.
adjoint_rel_tol	Backward relative iteration error for each species for adjoint sensitivity analysis. Recommended value is 1e-6.
adjoint_abs_tol	Backward absolute iteration error for each species for adjoint sensitivity analysis. Recommended value is 1e-18
adjoint_variables	List the variables for which CONVERGE will perform adjoint sensitivity analysis. The variable options are TEMPERATURE and the species name from mech.dat . Typical value is TEMPERATURE. Note that for a mechanism tune simulation, adjoint_variables = TEMPERATURE by default and cannot be changed.
well_stirred_reactor_control	This settings block is used only when zero_d_solver.in > zero_d_input_control > case_type = WELL_STIRRED_REACTOR.
steady_solver	0 = Transient WSR solver, 1 = Steady-state WSR solver.
energy_solver	0 = Do not solve the energy equation and enforce constant temperature, 1 = Solve the energy equation.
find_turning	<i>OFF</i> , <i>EXTINCTION</i> = Find extinction turning point in the S curve, <i>EXTINCTION_AND_IGNITION</i> = Find extinction and ignition turning point in the S curve.
surface_chemistry	0 = Do not couple surface chemistry, 1 = Couple surface chemistry (requires surface_mech.dat).

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
<i>surface_mechanism_filename</i>	Surface chemistry mechanism file name (e.g., <i>surface_mech.dat</i>). Used only when <i>surface_chemistry</i> = 1 above.
<i>surface_thermodynamic_filename</i>	Surface chemistry thermodynamic data file name (e.g., <i>surface_therm.dat</i>). Used only when <i>surface_chemistry</i> = 1 above.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

0D Cases: zero_d_cases.in

To run the zero-dimensional chemistry utility, CONVERGE requires both *zero_d_cases.in* (or [zero_d_template.in](#)) and [zero_d_solver.in](#). The *zero_d_cases.in* file contains the case settings and the format will differ depending on the case type.

```
version: 3.1
---

zero_d_cases:
#-----
# Zero-D Case 0
# CUSTOM species case
#-----
- case:
  case_name: CUSTOM species
  integration_time: 1.0
  temperature: 800.00
  pressure: 2000000.0
  heat_loss_control:
    heat_loss_flag: 0
  mixture_type: CUSTOM
  egr_ratio: 0
  equivalence_ratio: 0
  species:
    IC8H18: 0.008333
    O2: 0.208333
    N2: 0.78333

#-----
# Zero-D Case 1
# SPECIFIED species case
#-----
- case:
  case_name: SPECIFIED species without EGR
  integration_time: 1.1
  temperature: 900.00
  pressure: 2100000.0
  heat_loss_control:
    heat_loss_flag: 0
  mixture_type: SPECIFIED
  equivalence_ratio: 1.0
  fuel:
    IC8H18: 1.0
  oxidizer:
    O2: 0.21
    N2: 0.79

#-----
# Zero-D Case 2
# SPECIFIED species with CEQ EGR
#-----
- case:
  case_name: SPECIFIED species with CEQ EGR
  integration_time: 1.2
  temperature: 1000.00
```

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

```
pressure: 2200000.0
heat_loss_control:
    heat_loss_flag: 0
mixture_type: SPECIFIED
equivalence_ratio: 1.0
fuel:
    IC8H18: 1.0
oxidizer:
    O2: 0.21
    N2: 0.79
egr_ratio: 0.2
egr_ratio_definition: WITH_FUEL
egr_mixture_type: CEQ

-----
# Zero-D Case 3
# SPECIFIED species with USER EGR
-----
- case:
    case_name: SPECIFIED species with USER EGR
    integration_time: 1.3
    temperature: 1100.00
    pressure: 2300000.0
    heat_loss_control:
        heat_loss_flag: 0
    mixture_type: SPECIFIED
    equivalence_ratio: 1.0
    fuel:
        IC8H18: 1.0
    oxidizer:
        O2: 0.21
        N2: 0.79
    egr_ratio: 0.2
    egr_ratio_definition: WITH_FUEL
    egr_mixture_type: USER
    egr_species:
        N2: 0.7
        CO: 0.3

-----
# Zero-D Case 4
# VARIABLE_VOLUME with CUSTOM species
# no heat loss, heat_loss_flag=0
-----
- case:
    case_name: VARIABLE_VOLUME with heat loss flag 0
    integration_time: 0.8
    temperature: 300.0
    pressure: 151987.5
    heat_loss_control:
        heat_loss_flag: 0
    volume_profile_filename: volume_profile.dat
    mixture_type: CUSTOM
    egr_ratio: 0
    equivalence_ratio: 0
    species:
        N2: 3.76
        O2: 1.0
        NC7H16: 0.090909

-----
# Zero-D Case 5
# VARIABLE_VOLUME with CUSTOM species
# constant heat loss, heat_loss_flag=1
-----
- case:
    case_name: VARIABLE_VOLUME with heat loss flag 1
    integration_time: 0.8
    temperature: 300.0
    pressure: 151987.5
    volume_profile_filename: volume_profile.dat
```

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

```
heat_loss_control:
    heat_loss_flag:          1
    heat_loss_rate:          1.0
mixture_type:           CUSTOM
species:
    N2:      3.76
    O2:      1.0
    NC7H16: 0.090909
equivalence_ratio:      0
egr_ratio:              0

-----
# Zero-D Case 6
# VARIABLE_VOLUME with CUSTOM species
# heat loss with heat_transfer_coefficient, heat_loss_flag=2
-----
- case:
    case_name:             VARIABLE_VOLUME with heat loss flag 2
    integration_time:      0.8
    temperature:           300.0
    pressure:              151987.5
    volume_profile_filename: volume_profile.dat
    heat_loss_control:
        heat_loss_flag:      2
        wall_temperature:   400
        heat_transfer_coefficient: 1.0
        surface_volume_ratio: 0.5
    mixture_type:           CUSTOM
    species:
        N2:      3.76
        O2:      1.0
        NC7H16: 0.090909
    egr_ratio:              0
    equivalence_ratio:      0

-----
# Zero-D Case 7
# VARIABLE_VOLUME with SPECIFIED species
# EGR is also available when SPECIFIED is on
# heat loss with heat_transfer_coefficient, heat_loss_flag=2
-----
- case:
    case_name:             VARIABLE_VOLUME with heat loss flag 2
    integration_time:      0.8
    temperature:           300.0
    pressure:              151987.5
    volume_profile_filename: volume_profile.dat
    heat_loss_control:
        heat_loss_flag:      2
        wall_temperature:   400
        heat_transfer_coefficient: 1.0
        surface_volume_ratio: 0.5
    mixture_type:           SPECIFIED
    equivalence_ratio:      1.0
    fuel:
        IC8H18: 1.0
    oxidizer:
        O2: 0.21
        N2: 0.79
    egr_ratio:              0

-----
# Zero-D Case 8
# CVODE_CONSTANT_VOLUME case type
# generated from a SAGE simulation
-----
zero_d_cvode_cases:
    - case:
        case_name:          ncyc 2
        integration_time: 6.2499999999999945e-07
        temperature:        9.99999300753408988e+02
```

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

```
pressure:          5.91999585772301443e+06
dpdt:             0
heat_loss_control:
    heat_loss_flag: 0
mixture_type:     CUSTOM
species:
    O2:            1.28000000000000502e-01
    N2:            5.7599999999999845e-01
    CO2:           7.49999999999995670e-02
    H2O:            2.09999999999998035e-02
    C12H26:         2.00000000000000205e-01
equivalence_ratio: 0
egr_ratio:         0
```

Figure 25.170: An example *zero_d_cases.in* file.

Table 25.153: Format of *zero_d_cases.in*.

Parameter	Description
<i>zero_d_cases</i> (<i>zero_d_cvode_cases</i>) if this file originated as <i>zero_d_cases_rank<number>.out</i>	
- <i>case</i>	This settings sub-block specifies 0D case settings. Repeat this sub-block through < <i>species name</i> > (if <i>mixture_type</i> = CUSTOM), <i>egr_ratio</i> (if <i>mixture_type</i> = SPECIFIED) or < <i>egr_species</i> > (if <i>mixture_type</i> = SPECIFIED and <i>egr_ratio</i> > 0) for each case.
<i>case_name</i>	0D case name.
<i>integration_time</i>	Final time (seconds) for integration. The solver will integrate only to the time at which the reactants reach equilibrium.
<i>temperature</i>	Initial temperature (K).
<i>pressure</i>	Initial pressure (Pa).
<i>dpdt</i>	Change in pressure over time. Used only if this file originated as <i>zero_d_cases_rank<number>.out</i> .
<i>volume_profile_filename</i>	Name of the file containing the volume profile. Used only when <i>zero_d_solver.in</i> > <i>zero_d_input_control</i> > <i>case_type</i> = VARIABLE_VOLUME.
<i>heat_loss_control</i>	The following parameters through <i>surface_volume_ratio</i> are for 0D heat loss calculations.
<i>heat_loss_flag</i>	0 = No heat loss ($dQ = 0.0$), 1 = Constant heat loss, 2 = Global equation.
<i>heat_loss_rate</i>	For <i>heat_loss_flag</i> = 1, constant heat loss rate (W/m^3).
<i>wall_temperature</i>	Wall temperature (K).
<i>heat_transfer_coefficient</i>	Heat transfer coefficient ($W/m^2\text{-}K$).
<i>surface_volume_ratio</i>	Ratio to convert area-based energy source to volume-based energy source for heat loss calculations.

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
<i>mixture_type</i>	Specify how to supply the mixture. <i>CUSTOM</i> = Supply <i>species</i> names and mass or mole fractions, <i>SPECIFIED</i> = Supply the <i>equivalence_ratio</i> , <i>fuel</i> species, and <i>oxidizer</i> species; optionally, supply the EGR mixture and species.
<i>fuel</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
< <i>fuel species name</i> >	Fuel species mass or mole fraction. Repeat this line for each fuel species. Each species must be on a separate line.
<i>oxidizer</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
< <i>oxidizer species name</i> >	Oxidizer species mass or mole fraction. Repeat this line for each oxidizer species. Each species must be on a separate line
<i>species</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>CUSTOM</i> .
< <i>species name</i> >	Custom species mass or mole fraction. Repeat this line for each custom species. Each species must be on a separate line.
<i>equivalence_ratio</i>	Equivalence ratio of fuel and oxidizer. Used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
<i>egr_ratio</i>	The EGR ratio. Used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
<i>egr_ratio_definition</i>	<i>NO_FUEL</i> = Do not include the fuel charge term in the EGR calculation, <i>WITH_FUEL</i> = Include the fuel charge term in the EGR calculation. Used only when <i>egr_ratio</i> is greater than 0.
<i>egr_mixture_type</i>	<i>USER</i> = Supply the EGR mixture, <i>CEQ</i> = CONVERGE computes the EGR mixture with CEQ. Used only when <i>egr_ratio</i> is greater than 0.
<i>egr_species</i>	EGR species mass or mole fraction.
< <i>egr species</i> >	Repeat this line for each EGR species. Each species must be on a separate line. Used only when <i>egr_ratio</i> is greater than 0.

0D HCCI Engine Cases

To quickly simulate single-cycle homogeneous charge compression ignition (HCCI) engines, set [`zero_d_solver.in`](#) > `zero_d_input_control` > `case_type` = `ENGINE` and include a `zero_d_engine_cases` settings block in the [`zero_d_cases.in`](#) file.

```
version: 3.1
---

zero_d_engine_cases:
#-----
# Zero-D Engine Case 1
# ENGINE case with SPECIFIED
# species and adiabatic
#-----
- case:
  case_name: ENGINE 1-SPECIFIED species
  start_time: -142.0
  end_time: 115.0
  temperature: 300.0
  pressure: 151987.5
  mixture_type: SPECIFIED
  fuel:
    NC7H16: 1
  oxidizer:
    N2: 0.79
    O2: 0.21
  egr_ratio: 0
  engine_control:
    bore: 0.13716
    stroke: 0.1651
    connecting_rod: 0.263
    crank_offset: 0.0
    rpm: 1600.0
    compression_ratio: 16.5
  heat_transfer_control:
    heat_transfer_option: 0
#-----
# Zero-D Engine Case 2
# ENGINE case with SPECIFIED
# species and adiabatic
#-----
- case:
  case_name: ENGINE 2-SPECIFIED species
  start_time: -142.0
  end_time: 115.0
  temperature: 300.0
  pressure: 151987.5
  mixture_type: SPECIFIED
  equivalence_ratio: 0
  fuel:
    NC7H16: 1
  oxidizer:
    N2: 0.79
    O2: 0.21
  egr_ratio: 0
  engine_control:
    bore: 0.13716
    stroke: 0.1651
    connecting_rod: 0.263
    crank_offset: 0.0
    rpm: 1600.0
    compression_ratio: 16.5
  heat_transfer_control:
    heat_transfer_option: 1
    htc_profile_filename: htc2.dat
#-----
```

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

```
# Zero-D Engine Case 3
# ENGINE case with CUSTOM species
# with Woschni Heat Transfer Model
#-----
- case:
  case_name:                                ENGINE 3-CUSTOM species
  start_time:                               -142.0
  end_time:                                 115.0
  temperature:                             300.0
  pressure:                                151987.5
  mixture_type:                            CUSTOM
  species:
    N2:          3.76
    O2:          1.0
    NC7H16:     0.090909
  egr_ratio:                                0
  equivalence_ratio:                      0
  engine_control:
    bore:                                     0.13716
    stroke:                                    0.1651
    connecting_rod:                           0.263
    crank_offset:                            0.0
    rpm:                                      1600.0
    compression_ratio:                      16.5
  heat_transfer_control:
    heat_transfer_option:                   2
    heat_transfer_model_control:
      wall_temperature:                     300
      head_bore_area_ratio:                1.0
      piston_bore_area_ratio:              1.0
      heat_transfer_multiplier:            1.0
      heat_transfer_model:                 WOSCHNI
#-----
# Zero-D Engine Case 4
# ENGINE case with SPECIFIED species
# with Woschni Heat Transfer Model
#-----
- case:
  case_name:                                ENGINE 4-SPECIFIED species
  start_time:                               -142.0
  end_time:                                 115.0
  temperature:                             300.0
  pressure:                                151987.5
  mixture_type:                            SPECIFIED
  equivalence_ratio:                      1.0
  fuel:
    NC7H16:       1
  oxidizer:
    N2:          0.79
    O2:          0.21
  egr_ratio:                                0
  engine_control:
    bore:                                     0.13716
    stroke:                                    0.1651
    connecting_rod:                           0.263
    crank_offset:                            0.0
    rpm:                                      1600.0
    compression_ratio:                      16.5
  heat_transfer_control:
    heat_transfer_option:                   3
    heat_transfer_model_control:
      wall_temperature:                     300
      head_bore_area_ratio:                1.0
      piston_bore_area_ratio:              1.0
      heat_transfer_multiplier:            1.0
      heat_transfer_model:                 WOSCHNI
      heat_transfer_equation_control:
        scaling_factor:                  3.26
        length_type:                   CYLINDER_BORE
        length_exponent:                -0.2
```

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

```
    pressure_exponent:          0.8
    temperature_exponent:       -0.53
    velocity_exponent:          0.8
    c1:                          2.28
    c2:                          0.00324
-----
# Zero-D Engine Case 5
# ENGINE case with SPECIFIED species
# and EGR with Chang Heat Transfer Model
-----
- case:
    case_name:                  ENGINE 5-SPECIFIED species + EGR
    start_time:                 -142.0
    end_time:                   115.0
    temperature:                300.0
    pressure:                   151987.5
    mixture_type:               SPECIFIED
    equivalence_ratio:          1.0
    fuel:
        NC7H16: 1
    oxidizer:
        N2: 0.79
        O2: 0.21
    egr_ratio:                  0.2
    egr_ratio_definition:      WITH_FUEL
    egr_mixture_type:          USER
    egr_species:
        N2: 0.7
        CO: 0.3
    engine_control:
        bore:                      0.13716
        stroke:                     0.1651
        connecting_rod:            0.263
        crank_offset:              0.0
        rpm:                        1600.0
        compression_ratio:         16.5
    heat_transfer_control:
        heat_transfer_option:      3
        heat_transfer_model_control:
            wall_temperature:      300
            head_bore_area_ratio:  1.0
            piston_bore_area_ratio: 1.0
            heat_transfer_multiplier: 1.0
            heat_transfer_model:    CHANG
            heat_transfer_equation_control:
                scaling_factor:      3.4
                length_type:          CHAMBER_HEIGHT
                length_exponent:      -0.2
                pressure_exponent:    0.8
                temperature_exponent: -0.73
                velocity_exponent:   0.8
                c1:                   2.28
                c2:                   0.00054
    -----
# Zero-D Engine Case 6
# ENGINE case with SPECIFIED species
# and Hohenberg Heat Transfer Model
-----
- case:
    case_name:                  ENGINE 6-SPECIFIED species
    start_time:                 -142.0
    end_time:                   115.0
    temperature:                300.0
    pressure:                   151987.5
    mixture_type:               SPECIFIED
    equivalence_ratio:          1.0
    fuel:
        NC7H16: 1
    oxidizer:
        N2: 0.79
```

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

```
O2: 0.21
egr_ratio: 0
engine_control:
    bore: 0.13716
    stroke: 0.1651
    connecting_rod: 0.263
    crank_offset: 0.0
    rpm: 1600.0
    compression_ratio: 16.5
heat_transfer_control:
    heat_transfer_option: 3
    heat_transfer_model_control:
        wall_temperature: 300
        head_bore_area_ratio: 1.0
        piston_bore_area_ratio: 1.0
        heat_transfer_multiplier: 1.0
        heat_transfer_model: HOHENBERG
        heat_transfer_equation_control:
            scaling_factor: 1.0
            length_type: VOLUME
            length_exponent: -0.06
            pressure_exponent: 0.8
            temperature_exponent: -0.4
            velocity_exponent: 0.8
            c1: 130
            c2: 1.4

-----
# Zero-D Engine Case 7
# ENGINE case with SPECIFIED species
# and Annand Heat Tranfer Model
-----
- case:
    case_name: ENGINE 7-SPECIFIED species
    start_time: -142.0
    end_time: 115.0
    temperature: 300.0
    pressure: 151987.5
    mixture_type: SPECIFIED
    equivalence_ratio: 1.0
    fuel:
        NC7H16: 1
    oxidizer:
        N2: 0.79
        O2: 0.21
    egr_ratio: 0
    engine_control:
        bore: 0.13716
        stroke: 0.1651
        connecting_rod: 0.263
        crank_offset: 0.0
        rpm: 1600.0
        compression_ratio: 16.5
    heat_transfer_control:
        heat_transfer_option: 3
        heat_transfer_model_control:
            wall_temperature: 300
            head_bore_area_ratio: 1.0
            piston_bore_area_ratio: 1.0
            heat_transfer_multiplier: 1.0
            heat_transfer_model: ANNAND
            scaling_factor: 1.0
            length_type: CYLINDER_BORE
            length_exponent: -0.3
            pressure_exponent: 0.316
            velocity_exponent: 0.7
            c1: 0.25
            c2: 0.576
```

Figure 25.171: Sample `zero_d_engine_cases` settings block in `zero_d_cases.in`.

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Table 25.154: Format of the *zero_d_engine_cases* settings block.

Parameter	Description
<i>zero_d_engine_cases</i>	
- <i>case</i>	This settings sub-block specifies the 0D engine case parameters. Repeat this sub-block through <i>wall_temperature</i> (for <i>heat_transfer_option</i> = 0 or 1), <i>heat_transfer_model</i> (for <i>heat_transfer_option</i> = 2), or <i>c2</i> (for <i>heat_transfer_option</i> = 3) for each case.
<i>case_name</i>	0D HCCI engine case name.
<i>start_time</i>	Case start time (s).
<i>end_time</i>	Case end time (s).
<i>temperature</i>	Initial cylinder temperature (K).
<i>pressure</i>	Initial pressure (Pa).
<i>mixture_type</i>	Specify how to supply the mixture. <i>CUSTOM</i> = Supply <i>species</i> names and mass or mole fractions, <i>SPECIFIED</i> = Supply the <i>equivalence_ratio</i> , <i>fuel</i> species, and <i>oxidizer</i> species; optionally, supply the EGR mixture and species.
<i>fuel</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
< <i>fuel species name</i> >	Fuel species mass or mole fraction. Repeat this line for each fuel species. Each species must be on a separate line.
<i>oxidizer</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
< <i>oxidizer species name</i> >	Oxidizer species mass or mole fraction. Repeat this line for each oxidizer species. Each species must be on a separate line.
<i>species</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>CUSTOM</i> .
< <i>species name</i> >	Custom species mass or mole fraction. Repeat this line for each custom species. Each species must be on a separate line.
<i>equivalence_ratio</i>	Equivalence ratio of fuel and oxidizer. Used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
<i>egr_ratio</i>	The EGR ratio. Used only when <i>mixture_type</i> = <i>SPECIFIED</i> .

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
<i>egr_ratio_definition</i>	<i>NO_FUEL</i> = Do not include the fuel charge term in the EGR calculation, <i>WITH_FUEL</i> = Include the fuel charge term in the EGR calculation. Used only when <i>egr_ratio</i> is greater than 0.
<i>egr_mixture_type</i>	<i>USER</i> = Supply the EGR mixture, <i>CEQ</i> = CONVERGE computes the EGR mixture with CEQ. Used only when <i>egr_ratio</i> is greater than 0.
<i>egr_species</i>	EGR species mass or mole fraction.
< <i>egr species</i> >	Repeat this line for each EGR species. Each species must be on a separate line. Used only when <i>egr_ratio</i> is greater than 0.
<i>engine_control</i>	
<i>bore</i>	Engine bore diameter (m).
<i>stroke</i>	Engine stroke (m).
<i>connecting_rod</i>	Engine connecting rod length (m).
<i>crank_offset</i>	Crank offset (m).
<i>rpm</i>	Engine RPM.
<i>compression_ratio</i>	Engine compression ratio.
<i>heat_transfer_control</i>	
<i>heat_transfer_option</i>	0 = Adiabatic, 1 = Specify a heat transfer coefficient file, 2 = Heat transfer model, 3 = Heat transfer model with equation control.
<i>htc_profile_filename</i>	Name of the HTC profile file. Used only for <i>heat_transfer_option</i> = 1.
<i>heat_transfer_model_control</i>	
<i>wall_temperature</i>	Wall temperature (K).
<i>head_bore_area_ratio</i>	Ratio of the head area to the bore area. Used only when <i>heat_transfer_option</i> = 2 or 3.
<i>piston_bore_area_ratio</i>	Ratio of the piston area to the bore area. Used only when <i>heat_transfer_option</i> = 2 or 3.
<i>heat_transfer_multiplier</i>	Heat transfer scaling factor. Used only when <i>heat_transfer_option</i> = 2 or 3.

Parameter	Description
<i>heat_transfer_model</i>	Used only when <i>heat_transfer_option</i> = 2 or 3. <i>WOSCHNI</i> = Woschni heat transfer model, <i>CHANG</i> = Chang heat transfer model, <i>HOHENBERG</i> = Hohenberg heat transfer model, <i>ANNAND</i> = Annand heat transfer model.
<i>heat_transfer_equation_control</i>	This settings sub-block is used only when <i>heat_transfer_option</i> = 3.
<i>scaling_factor</i>	Scaling factor to match a specific engine geometry.
<i>length_type</i>	Length scaling factor type. <i>CYLINDER_BORE</i> = Characteristic cylinder bore, <i>CHAMBER_HEIGHT</i> = Characteristic chamber height, <i>VOLUME</i> = Characteristic volume.
<i>length_exponent</i>	Characteristic length exponent value.
<i>pressure_exponent</i>	Pressure exponent value.
<i>temperature_exponent</i>	Temperature exponent value.
<i>velocity_exponent</i>	Velocity exponent value.
<i>c1</i>	Constant used in <i>heat_transfer_model</i> .
<i>c2</i>	Constant used in <i>heat_transfer_model</i> .

0D RON/MON Calculations

To calculate the research octane number (RON) or motor octane number (MON) for a fuel, set `zero_d_solver.in` > `zero_d_input_control` > `case_type` = `ENGINE_RON`, `ENGINE_MON`, or `ENGINE_RON_MON` and supply a `zero_d_engine_ron_mon_cases` settings block in the `zero_d_cases.in` file. Note that `zero_d_cases.in` needs to specify only the fuel composition.

```
version: 3.1
---

zero_d_engine_ron_mon_cases:
#-----
# ENGINE_RON or ENGINE_MON case
# only require fuel composition
# specify ENGINE_RON, ENGINE_MON
# or ENGINE_RON_MON as case type
# in zero_d_solver.in file
#-----
- case:
  case_name: PRF
  fuel:
    IC8H18: 0.87
    NC7H16: 0.13
- case:
  case_name: PRF
  fuel:
    IC8H18: 0.95
    NC7H16: 0.05
```

Figure 25.172: Sample input for `zero_d_engine_ron_mon_cases` block in the `zero_d_cases.in` file.

Table 25.155: Format of the *zero_d_engine_ron_mon_cases* settings block.

Parameter	Description
<i>zero_d_engine_ron_mon_cases</i>	
- <i>case</i>	This settings sub-block specifies 0D RON/MON calculation parameters. Repeat this sub-block through <i>fuel</i> for each case.
<i>case_name</i>	0D RON/MON case name.
<i>fuel</i>	
< <i>fuel species name</i> >	Fuel species mass or mole fraction.
	Repeat this line for each fuel species. Each species must be on a separate line.

0D Well-Stirred Reactor Cases

To generate ignition delay data from a [well-stirred reactor](#) (WSR) model, set [*zero_d_solver.in*](#) > *zero_d_input_control* > *case_type* = WELL_STIRRED_REACTOR and include a *zero_d_wsr_cases* settings block in the [*zero_d_cases.in*](#) file.

```
version: 3.1
---

zero_d_wsr_cases:
  - case:
    case_name: WSR_Case_0
    integration_time: 0.4
    pressure: 2.0e6
    volume: 1e-5
    surface_area: 1e-4
    mass_flow_rate: 1e-3
    ambient_temperature: 300.0
    heat_transfer_coefficient: 12.0
    initial_mixture:
      temperature: 700.0
      mixture_type: CUSTOM
      species:
        IC8H18: 0.008333
        O2: 0.208333
        N2: 0.78333
      equivalence_ratio: 0
      egr_ratio: 0
    inlet_mixture:
      temperature: 850.0
      mixture_type: SPECIFIED
      equivalence_ratio: 1.0
      fuel:
        IC8H18: 1.0
      oxidizer:
        O2: 0.21
        N2: 0.79
      egr_ratio: 0
      surface_species:
        PT_: 1.0
```

Figure 25.173: Sample WSR block in *zero_d_cases.in*.

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Table 25.156: Format of the `zero_d_wsr_cases` settings block.

Parameter	Description
<code>zero_d_wsr_cases</code>	
<code>- case</code>	This settings sub-block specifies 0D case parameters. Repeat this sub-block through <code><species name></code> (if <code>mixture_type = CUSTOM</code>), <code>egr_ratio</code> (if <code>mixture_type = SPECIFIED</code>), <code><egr_species></code> (if <code>mixture_type = SPECIFIED</code> and <code>egr_ratio > 0</code>), or <code>surface_species</code> (if <code>zero_d_solver.in > well_stirred_reactor_control > surface_chemistry = 1</code>) for each case.
<code>case_name</code>	0D WSR case name.
<code>integration_time</code>	Unsteady WSR integration time (s).
<code>pressure</code>	Initial pressure (Pa).
<code>volume</code>	Initial volume (m^3).
<code>surface_area</code>	Surface area (m^2).
<code>mass_flow_rate</code>	Mass flow rate (kg/s).
<code>ambient_temperature</code>	Ambient temperature (K).
<code>heat_transfer_coefficient</code>	Heat transfer coefficient ($W/m^2\text{-}K$).
<code>initial_mixture</code>	This settings sub-block specifies the initial mixture composition.
<code>temperature</code>	Initial temperature (K).
<code>mixture_type</code>	Specify how to supply the mixture. <code>CUSTOM</code> = Supply <i>species names</i> and mass or mole fractions, <code>SPECIFIED</code> = Supply the <i>equivalence_ratio</i> , <i>fuel species</i> , and <i>oxidizer species</i> ; optionally, supply the EGR mixture and species.
<code>fuel</code>	This settings sub-block is used only when <code>mixture_type = SPECIFIED</code> .
<code><fuel species name></code>	Fuel species mass or mole fraction. Repeat this line for each fuel species. Each species must be on a separate line.
<code>oxidizer</code>	This settings sub-block is used only when <code>mixture_type = SPECIFIED</code> .
<code><oxidizer species name></code>	Oxidizer species mass or mole fraction. Repeat this line for each oxidizer species. Each species must be on a separate line.
<code>species</code>	This settings sub-block is used only when <code>mixture_type = CUSTOM</code> .
<code><species name></code>	Custom species mass or mole fraction.

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
	Repeat this line for each custom species. Each species must be on a separate line.
<i>equivalence_ratio</i>	Equivalence ratio of fuel and oxidizer. Used only when <i>mixture_type</i> = SPECIFIED.
<i>egr_ratio</i>	The EGR ratio. Used only when <i>mixture_type</i> = SPECIFIED.
<i>inlet_mixture</i>	This settings sub-block specifies the inlet mixture composition.
<i>temperature</i>	Inlet mixture temperature (K).
<i>mixture_type</i>	Specify how to supply the mixture. <i>CUSTOM</i> = Supply species names and mass or mole fractions, <i>SPECIFIED</i> = Supply the <i>equivalence_ratio</i> , <i>fuel</i> species, and <i>oxidizer</i> species; optionally, supply the EGR mixture and species.
<i>fuel</i>	This settings sub-block is used only when <i>mixture_type</i> = SPECIFIED.
< <i>fuel species name</i> >	Fuel species mass or mole fraction.
	Repeat this line for each fuel species. Each species must be on a separate line.
<i>oxidizer</i>	This settings sub-block is used only when <i>mixture_type</i> = SPECIFIED.
< <i>oxidizer species name</i> >	Oxidizer species mass or mole fraction.
	Repeat this line for each oxidizer species. Each species must be on a separate line.
<i>species</i>	This settings sub-block is used only when <i>mixture_type</i> = CUSTOM.
< <i>species name</i> >	Custom species mass or mole fraction.
	Repeat this line for each custom species. Each species must be on a separate line.
<i>equivalence_ratio</i>	Equivalence ratio of fuel and oxidizer. Used only when <i>mixture_type</i> = SPECIFIED.
<i>egr_ratio</i>	The EGR ratio. Used only when <i>mixture_type</i> = SPECIFIED.
<i>surface_species</i>	This settings sub-block is used only when <i>zero_d_solver.in</i> > <i>well_stirred_reactor_control</i> > <i>surface_chemistry</i> = 1.
< <i>surface species name</i> >	Surface species mass or mole fraction.
	Repeat this line for each surface species. Each species must be on a separate line.

0D Plug Flow Reactor Cases

To generate ignition delay data from a [plug flow reactor \(PFR\) model](#), set `zero_d_solver.in > zero_d_input_control > case_type = CONSTANT_PRESSURE_PLUG_FLOW_REACTOR` and include an appropriately configured `zero_d_cases` settings block in the [`zero_d_cases.in`](#) file.

```
version: 3.1
---

zero_d_cases:
- case:
    case_name: ZEROD_PFR_1
    integration_time: 99.0
    mixture_type: CUSTOM
    pressure: 101325
    heat_loss_control:
        heat_loss_flag: 0
    species:
        H2: 2.0
        O2: 1.0
        AR: 0.1
    equivalence_ratio: 0
    egr_ratio: 0
    temperature: 1500.0
    plug_flow_control:
        domain_length: 1.5e-7
        inlet_velocity: 0.006
        cross_sectional_area: 1e-4

- case:
    case_name: ZEROD_PFR_2
    integration_time: 99.0
    mixture_type: CUSTOM
    pressure: 101325
    heat_loss_control:
        heat_loss_flag: 0
    species:
        H2: 2.0
        O2: 1.0
        AR: 0.1
    equivalence_ratio: 0
    egr_ratio: 0
    temperature: 1500.0
    plug_flow_control:
        domain_length: 1.5e-7
        inlet_velocity: 0.008
        cross_sectional_area: 1e-4

- case:
    case_name: ZEROD_PFR_3
    integration_time: 99.0
    mixture_type: CUSTOM
    pressure: 101325
    heat_loss_control:
        heat_loss_flag: 0
    species:
        H2: 2.0
        O2: 1.0
        AR: 0.1
    equivalence_ratio: 0
    egr_ratio: 0
    temperature: 1200.0
    plug_flow_control:
        domain_length: 3e-7
        inlet_velocity: 0.006
        cross_sectional_area: 1e-4
```

Figure 25.174: Sample `zero_d_cases` settings block for the PFR model in `zero_d_cases.in`.

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Table 25.157: Format of the *zero_d_cases* settings block for the PFR model

Parameter	Description
<i>zero_d_cases</i>	
<i>- case</i>	This settings sub-block specifies 0D case parameters. Repeat this sub-block through <i>cross_sectional_area</i> for each case.
<i>case_name</i>	0D plug flow case name.
<i>integration_time</i>	Unsteady plug reactor integration time (s).
<i>mixture_type</i>	Specify how to supply the mixture. <i>CUSTOM</i> = Supply species names and mass or mole fractions, <i>SPECIFIED</i> = Supply the <i>equivalence_ratio</i> , <i>fuel</i> species, and <i>oxidizer</i> species; optionally, supply the EGR mixture and species.
<i>pressure</i>	Initial pressure (Pa).
<i>heat_loss_control</i>	This settings sub-block (through <i>surface_volume_ratio</i>) is for 0D heat loss calculations.
<i>heat_loss_flag</i>	0 = No heat loss ($dQ = 0.0$), 1 = Constant heat loss, 2 = Global equation.
<i>heat_loss_rate</i>	For <i>heat_loss_flag</i> = 1, constant heat loss rate (W/m ³).
<i>wall_temperature</i>	Wall temperature (K).
<i>heat_transfer_coefficient</i>	Heat transfer coefficient (W/m ² -K).
<i>surface_volume_ratio</i>	Ratio to convert area-based energy source to volume-based energy source for heat loss calculations.
<i>fuel</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
<i><fuel species name></i>	Fuel species mass or mole fraction. Repeat this line for each fuel species. Each species must be on a separate line.
<i>oxidizer</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
<i><oxidizer species name></i>	Oxidizer species mass or mole fraction. Repeat this line for each oxidizer species. Each species must be on a separate line.
<i>species</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>CUSTOM</i> .
<i><species name></i>	Custom species mass or mole fraction. Repeat this line for each custom species. Each species must be on a separate line.
<i>equivalence_ratio</i>	Equivalence ratio of fuel and oxidizer. Used only when <i>mixture_type</i> = <i>SPECIFIED</i> .

Parameter	Description
<i>egr_ratio</i>	The EGR ratio. Used only when <i>mixture_type</i> = SPECIFIED.
<i>temperature</i>	Initial temperature (K).
<i>plug_flow_control</i>	
<i>domain_length</i>	Plug flow reactor domain length (m).
<i>inlet_velocity</i>	Plug flow reactor inlet velocity (m/s).
<i>cross_sectional_area</i>	Plug flow reactor cross-sectional area (m^2).

0D Template: zero_d_template.in

To generate a [0D TKI table](#), CONVERGE reads parameter ranges from *zero_d_template.in*.

```
version: 3.1
---

integration_time:          1.1
pressure_control:
  type_flag:               1
  min:                     100000.0
  max:                     500000.0
  interval:                200000.0
  values:                  [100000.0, 200000.0, 400000.0]
temperature_control:
  type_flag:               1
  min:                     300.0
  max:                     1100.0
  interval:                100.0
  values:                  [300.0, 500.0, 700.0]
equivalence_ratio_control:
  type_flag:               0
  min:                     0.5
  max:                     1.7
  interval:                0.1
  values:                  []
egr_control:
  active:                  0
  type_flag:               0
  min:                     0.0
  max:                     0.4
  interval:                0.1
  values:                  []
  ratio_definition_flag:   1
  ceq_flag:                 0
  species:
    CO2: 0.4
    CO: 0.4
    H2O: 0.1
    H2: 0.1
species_control:
  custom_species_flag:     1
  fuel_species:
    CH4: [0.7000000, 0.5000000]
    CO2: [0.3000000, 0.5000000]
  oxidizer_species:
    O2: [0.23, 0.5000000]
    N2: [0.77, 0.5000000]
```

Figure 25.175: An example *zero_d_template.in* file.

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Table 25.158: Format of *zero_d_template.in*.

Parameter	Description
<i>integration_time</i>	End time (s) for integration.
<i>pressure_control</i>	
<i>type_flag</i>	0 = Compute table over the listed range of pressures, 1 = Compute table on specified pressures.
<i>min</i>	Minimum pressure (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum pressure (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	Pressure interval (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	Flow-formatted list of specified pressures (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 1. Not used when <i>type_flag</i> = 0.
<i>temperature_control</i>	
<i>type_flag</i>	0 = Compute table over the listed range of temperatures, 1 = Compute table on specified temperatures.
<i>min</i>	Minimum temperature (K) for <i>temperature_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum temperature (K) for <i>temperature_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	Temperature interval (K) for <i>temperature_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	Flow-formatted list of specified temperatures (K) for <i>temperature_control</i> > <i>type_flag</i> = 1. Not used when <i>type_flag</i> = 0.
<i>equivalence_ratio_control</i>	
<i>type_flag</i>	0 = Compute table over the listed range of elemental equivalence ratios, 1 = Compute table on specified elemental equivalence ratios.
<i>min</i>	Minimum elemental equivalence ratio for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum elemental equivalence ratio for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	Elemental equivalence ratio interval for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	Flow-formatted list of specified elemental equivalence ratios for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 1. Not used when <i>type_flag</i> = 0.
<i>egr_control</i>	

Chapter 25: Input and Data Files

Chemistry Zero-Dimensional Chemistry Utilities

Parameter	Description
<i>active</i>	0 = Do not activate EGR, 1 = Activate EGR.
<i>type_flag</i>	0 = Compute table over the listed range of EGR ratios, 1 = Compute table on specified EGR ratios.
<i>min</i>	Minimum EGR ratio for <i>egr_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum EGR ratio for <i>egr_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	EGR ratio interval for <i>egr_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	List of specified EGR ratios for <i>egr_control</i> > <i>type_flag</i> = 1. Not read for <i>type_flag</i> = 0.
<i>ratio_definition_flag</i>	0 = Define EGR ratio without including fuel mass (recommended when generating TKI table file for ECFM/ECFM3Z simulation), 1 = Define EGR ratio including fuel mass.
<i>ceq_flag</i>	0 = User defined EGR species, 1 = Use CEQ equilibrium.
<i>species</i>	This settings sub-block is used for user-defined EGR species. This settings sub-block is not used if <i>ceq_flag</i> = 1 above.
<i><species name></i>	Ratio of this species. Repeat this line for each species. Each species must be on a separate line. Ratios must sum to 1.0.
<i>species_control</i>	
<i>custom_species_flag</i>	0 = Supply <i>species</i> names and mass or mole fractions, 1 = Supply the <i>fuel</i> species and <i>oxidizer</i> species.
<i>fuel_species</i>	
<i><fuel species name></i>	Mass fraction of this fuel species. To specify multiple compositions of fuel, provide a flow-formatted list of species mass fractions (e.g., [0.7, 0.5]). Repeat this line for an additional fuel species. More than two fuel species is not allowed. Each pair of mass fractions for a composition must sum to 1.0.
<i>oxidizer_species</i>	
<i><oxidizer species name></i>	Mass fraction of this oxidizer species. To specify multiple compositions of oxidizer, provide a flow-formatted list of species mass fractions (e.g., [0.23, 0.5]).

Parameter	Description
	Repeat this line for each oxidizer species. Each species must be on a separate line. Mass fractions for a composition must sum to 1.0.

One-Dimensional Chemistry Utilities

This section describes the input files related to the 1D chemistry utilities in CONVERGE.

1D Solver: `one_d_solver.in`

To run the one-dimensional [premixed laminar flamespeed](#) or [laminar counterflow flame](#) model utility, CONVERGE requires both [`one_d_cases.in`](#) (or [`one_d_template.in`](#)) and `one_d_solver.in`. The `one_d_solver.in` file contains the required solver input parameters.

```
version: 3.1
---

one_d_general_control:
    case_type: LAMINAR_COUNTERFLOW_FLAME
    input_control:
        mechanism_filename: mech.dat
        thermodynamic_filename: therm.dat
        init_from_restart_flag: 0
        species_fraction_input_type: MASS
        lhv_flag: 0
    solver_control:
        solver_type: NEWTON
        specify_anchor_temp_flag: 0
        anchor_temp: 0.0
        domain_length: 0.035
        reaction_multiplier: 1.0
        case_time_limit: 3600
        species_diffusion_model: 1
        schmidt_number: 0.98
    output_control:
        output_file_flag: 1
        end_distance_species: [CH4, O2, CO2, CO]
        generate_tlf_table_flag: 0
        sensor_species: [CH4]
        species_fraction_output_type: MOLE
        failed_cases_output_flag: 1
#sensor_species:
    egr_ceq_species_control:
        subset_species_flag: 0
        subset_species:
            - N2
#=====
one_d_newton_control:
    initialization_control:
        num_gridpoints: 10
        ramp_fraction: 0.1
    discretization_control:
        sparse_solver_flag: 0
        impl_species_diffusion_flag: 1
        central_difference_flag: 0
    steady_state_solver_control:
        jac_iterations: 5
        rel_tol: 1.e-6
        abs_tol: 1.e-10
    pseudo_transient_solver_control:
        jac_iterations: 5
        rel_tol: 1.e-6
        abs_tol: 1.e-10
        dt_start: 2.e-4
```

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

```
    num_time_steps:          10
    grid_control:
        max_slope:           0.1
        min_slope:           1.e-5
        max_curve:           0.1
        min_curve:           1.e-5
        max_ratio:           10.0
    output_control:
        log_file_flag:       1
        sensitivity_flag:   0
        ga_flag:             0
        schmidt_species_flag: 0
=====
one_d_piso_control:
    simulation_setup_control:
        end_time:            10.0
        sponge_length:       0.01
        dx_base:             0.001
    flame_anchoring_control:
        mass_flow_rate_damp_factor: 0.1
        relax_velocity_factor:      0.5
        prog_variables:
            - CO
            - CO2
    sage_model_control:
        ode_solver:          DENSE
        rel_tol:              1.e-4
        abs_tol:              1.e-14
    monitor_steady_state_control:
        sample_size:          50
        tol_avg:              1.e-3
        max_std:              1.e-2
    amr_stages_control:
        amr_temp_stages:
            - 5.0
            - 1.0
            - 0.1
    final_stage_control:
        active:               1
        max_cfl_nu:           5.0
        sample_size:           500
        tol_avg:              1.e-4
        max_std:              1.e-2
    output_control:
        screen_print_level:   2
        post_file_flag:       1
```

Figure 25.176: An example *one_d_solver.in* file.

Table 25.159: Format of the *one_d_general_control* settings block.

Parameter	Description
<i>one_d_general_control</i>	
<i>case_type</i>	<i>LAMINAR_FREELY_PROPAGATING_FLAME</i> = Use a laminar freely propagating flame with one inlet, <i>LAMINAR_COUNTERFLOW_FLAME</i> = Use a laminar counterflow flame in which two inlets are directed toward each other and an axisymmetric flow field is produced between the two inlets. If you do not specify a <i>case_type</i> , CONVERGE will assume a laminar freely propagating flame with one inlet.
<i>input_control</i>	

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
<i>mechanism_filename</i>	Mechanism file name (e.g., <i>mech.dat</i>).
<i>thermodynamic_filename</i>	Thermodynamic data file name (e.g., <i>therm.dat</i>).
<i>init_from_restart_flag</i>	0 = Begin a new 1D simulation, 1 = Initialize the 1D simulation from a restart file.
<i>species_fraction_input_type</i>	MASS = Input is in mass fraction format (recommended if <i>one_d_solver.in > output_control > generate_tlf_table_flag</i> = 1), MOLE = Input is in mole fraction format.
<i>lhv_flag</i>	0 = Use default species lower heating value (LHV), 1 = Specify the species LHV (requires <i>lhv.in</i>).
<i>solver_control</i>	
<i>solver_type</i>	One-dimensional solver type. <i>NEWTON</i> = Stand-alone 1D steady-state solver, <i>PISO</i> = PISO solver (CONVERGE transient solver), <i>HYBRID</i> = Hybrid: Begin with the PISO solver (CONVERGE transient solver) on a coarse mesh and end with the stand-alone Newton solver on a finer mesh.
<i>specify_anchor_temp_flag</i>	0 = CONVERGE automatically calculates an anchor temperature for each case, 1 = Specify the constant anchor temperature for all cases via <i>anchor_temp</i> below.
<i>anchor_temp</i>	Premixed flame anchoring temperature (K).
<i>domain_length</i>	Length of the 1D domain (m).
<i>reaction_multiplier</i>	Scaling factor for reaction rates.
<i>case_time_limit</i>	Maximum wall-clock time allocated to each case in seconds.
<i>species_diffusion_model</i>	Species diffusion model. 0 = Apply single species diffusion coefficient (requires <i>gas.dat</i>), 1 = Apply mixture-averaged diffusion coefficient when solving species and energy transport equations (requires <i>transport.dat</i>), 2 = Apply a different Schmidt number to each species (requires <i>gas.dat</i> and <i>schmidt_species.dat</i>).
<i>schmidt_number</i>	Schmidt number if <i>species_diffusion_model</i> = 0 above.
<i>output_control</i>	
<i>output_file_flag</i>	0 = Generate only <i>one_d_flamespeed*.out</i> (and <i>thermo.out</i> and <i>flamespeed.out</i>) if <i>one_d_solver.in > one_d_general_control > solver_control > solver_type</i> = PISO or HYBRID), 1 = Generate the files listed above as well as any output files that scale with the number of cases (i.e., <i>one_d_newton_restart_case<case ID>.rst</i> and <i>one_d_sol_case<case ID>.out</i>).

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
<i>end_distance_species</i> *	List of species for which CONVERGE writes end-of-simulation mass fractions (if <i>species_fraction_output_type</i> = MASS) or mole fractions (if <i>species_fraction_output_type</i> = MOLE) to <u>one_d_end_distance_species.out</u> .
<i>generate_tlf_table_flag</i>	0 = Do not generate flamespeed table, 1 = Generate <u>laminar flamespeed (TLF) table</u> (<i>tlf_table.h5</i>) and read <u>one_d_template.in</u> instead of <u>one_d_cases.in</u> (requires <i>one_d_solver.in</i> > <i>one_d_general_control</i> > <i>case_type</i> = LAMINAR_FREELY_PROPAGATING_FLAME and <i>one_d_general_control</i> > <i>solver_control</i> > <i>solver_type</i> = NEWTON).
<i>sensor_species</i>	<u>Flow-formatted list</u> of species that are used as reaction rate sensors in the TFM.
<i>species_fraction_output_type</i>	MASS = Output is in mass fraction format, MOLE = Output is in mole fraction format,
<i>failed_cases_output_flag</i>	0 = Do not write a record of non-converged cases, 1 = Write a <i>one_d_cases_fail.in</i> file containing non-converged cases (not available when <i>generate_tlf_table_flag</i> = 1 above).
<i>egr_ceq_species_control</i>	
<i>subset_species_flag</i>	0 = Do not limit the number of species used in the CEQ calculations of the EGR mixture, 1 = Enable a limit on the number of species used in the CEQ calculations of the EGR mixture.
<i>subset_species</i>	This settings sub-block is used only when <i>subset_species_flag</i> = 1.
<i><subset species name></i>	Repeat this line for each species used in CEQ calculations of the EGR mixture. Each species must be on a separate line.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

Table 25.160: Format of the *one_d_newton_control* settings block.

Parameter	Description
<i>one_d_newton_control</i>	
<i>initialization_control</i>	
<i>num_gridpoints</i>	Initial number of grid points.
<i>ramp_fraction</i>	The fraction of the domain in which the initial temperature profile increases linearly to the equilibrium value.
<i>discretization_control</i>	
<i>sparse_solver_flag</i>	0 = Do not use the sparse solver, 1 = Enable sparse solver in the Newton solver.
<i>impl_species_diffusion_flag</i>	0 = Do not evaluate species diffusion within the Newton-Raphson loop,

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
	1 = Evaluate species diffusion within the Newton-Raphson loop.
<i>central_difference_flag</i>	Specify the spatial discretization scheme: 0 = Upwind scheme, 1 = Central difference scheme after the upwind scheme converges.
<i>steady_state_solver_control</i>	
<i>jac_iterations</i>	Number of iterations between Jacobian evaluation for the steady-state solver.
<i>rel_tol</i>	Relative tolerance for the steady-state solver.
<i>abs_tol</i>	Absolute tolerance for the steady-state solver.
<i>pseudo_transient_solver_control</i>	
<i>jac_iterations</i>	Number of iterations between Jacobian evaluations for the pseudo time-step solver.
<i>rel_tol</i>	Relative tolerance for the pseudo time-step solver.
<i>abs_tol</i>	Absolute tolerance for the pseudo time-step solver.
<i>dt_start</i>	Initial time-step for the pseudo time-step solver (s).
<i>num_time_steps</i>	Number of time-steps for the pseudo time-step solver.
<i>grid_control</i>	
<i>max_slope</i>	Normalized slope criterion for refinement.
<i>min_slope</i>	Normalized slope criterion for coarsening.
<i>max_curve</i>	Normalized curvature criterion for refinement.
<i>min_curve</i>	Normalized curvature criterion for coarsening.
<i>max_ratio</i>	Maximum ratio of allowed adjacent grid sizes.
<i>output_control</i>	
<i>log_file_flag</i>	0 = Do not write log files, 1 = Write a log file (<i>one_d_case<case number>.log</i>) for each case. Used only when <i>one_d_solver.in > one_d_general_control > solver_control > solver_type</i> = NEWTON or HYBRID.
<i>sensitivity_flag</i>	0 = Do not perform a sensitivity analysis, 1 = Perform a sensitivity analysis.
<i>ga_flag</i>	0 = Do not write output for a genetic algorithm simulation, 1 = Write output for a genetic algorithm simulation (required for mechanism tune with CONGO).
<i>schmidt_species_flag</i>	0 = Do not calculate Schmidt numbers, 1 = Calculate Schmidt numbers for each species in each case and output to <i>schmidt_species_case<number>.dat</i> (requires

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
	<code>one_d_solver.in > one_d_general_control > case_type = LAMINAR_FREELY_PROPAGATING, one_d_general_control > solver_control > solver_type = NEWTON, and solver_control > species_diffusion_model = 1 above).</code>

Table 25.161: Format of the `one_d_piso_control` settings block.

Parameter	Description
<code>one_d_piso_control</code>	
<code>simulation_setup_control</code>	
<code>end_time</code>	End time (s).
<code>sponge_length</code>	Length of the sponge layer (<i>m</i>) to dampen acoustic fluctuation. Cannot be more than half of the domain length.
<code>dx_base</code>	Maximum cell size (<i>m</i>) in the axial direction.
<code>flame_anchoring_control</code>	
<code>mass_flow_rate_damp_factor</code>	Controls the forcing term in the momentum equation to anchor the premixed flame.
<code>relax_velocity_factor</code>	Under-relaxation factor for the update of inlet velocity to anchor the premixed flame.
<code>prog_variables</code>	List of progress variable species. On the lines below, enter a hyphen and the species name for each stage with one stage per line.
<code>sage_model_control</code>	
<code>ode_solver</code>	Options for different ordinary differential equation (ODE) solvers: <i>DENSE</i> = CVODE dense solver (recommended when the total number of species is no greater than 100), <i>ITERATIVE</i> = CVODE with preconditioned iterative solver, <i>ITERATIVE_SUPERLU</i> = CVODE with preconditioned iterative solver with SuperLU (SuperLU, 2011), <i>ITERATIVE_CONVERGE</i> = CVODE with optimized preconditioned iterative solver (recommended when the total number of species is greater than 100), <i>NONSTIFF_DENSE</i> = CVODE dense solver for non-stiff systems (recommended for single-step mechanisms).
<code>rel_tol</code>	Relative iteration error for each species in the SAGE solver.
<code>abs_tol</code>	Absolute iteration error for each species in the SAGE solver.
<code>monitor_steady_state_control</code>	
<code>sample_size</code>	Number of samples used to determine if the specified variable has reached a steady state.

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
<i>tol_avg</i>	Tolerance for the difference between the mean of two monitored samples.
<i>max_std</i>	Maximum allowed standard deviation in monitored samples.
<i>amr_stages_control</i>	
<i>amr_temp_stages</i>	List of temperature AMR stages. On the lines below, enter a hyphen and the temperature sub-grid scale value for each stage with one stage per line.
<i>final_stage_control</i>	
<i>active</i>	0 = Do not activate a final AMR stage with tighter settings, 1 = Activate a final AMR stage with tighter solver settings.
<i>max_cfl_nu</i>	Maximum CFL number based on viscosity for the final AMR stage.
<i>sample_size</i>	Number of samples used to determine if the specified variable has reached a steady state for the final AMR stage.
<i>tol_avg</i>	Tolerance for the difference between the mean of two monitored samples in the final AMR stage.
<i>max_std</i>	Maximum allowed standard deviation in monitored samples in the final AMR stage.
<i>output_control</i>	
<i>screen_print_level</i>	Screen print level.
<i>post_file_flag</i>	0 = Do not write post files, 1 = Write post files at the end of the simulation.

1D Cases: *one_d_cases.in*

To run the one-dimensional chemistry solver, CONVERGE requires both *one_d_cases.in* (or [one_d_template.in](#)) and [one_d_solver.in](#). The format of *one_d_cases.in* will differ depending on if you are performing a [1D premixed laminar flamespeed](#) or a [laminar counterflow](#) calculation.

Premixed Laminar Flamespeed

To run the [1D premixed laminar flamespeed model](#) utility, CONVERGE needs two files: *one_d_cases.in* and [one_d_solver.in](#). Figure 25.177 shows an example *one_d_cases.in* file, which lists initial conditions for temperature, pressure, and species.

```
version: 3.1
---

one_d_cases:
#-----
# One-D Case 0
#-----
- case:
  case_name: OneD Case 0
```

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

```
unburned_temp:          300.0
unburned_pres:          1000000.0
init_velocity:           0.3
cutoff_temp:             10.0
mixture_type:            CUSTOM
egr_ratio:                0
equivalence_ratio:        0
species:
    CH4: 0.077519379845
    O2: 0.193798449612
    N2: 0.728682170543
#-----
# One-D Case 1
#-----
- case:
    case_name:          OneD Case 1
    unburned_temp:        300.0
    unburned_pres:        1000000.0
    init_velocity:         0.3
    cutoff_temp:           10.0
    mixture_type:          CUSTOM
    egr_ratio:              0
    equivalence_ratio:        0
    species:
        CH4: 0.095057034220
        O2: 0.190114068441
        N2: 0.714828897338
#-----
# One-D Case 2
#-----
- case:
    case_name:          OneD Case 2
    unburned_temp:        300.0
    unburned_pres:        1000000.0
    init_velocity:         0.3
    cutoff_temp:           10.0
    mixture_type:          CUSTOM
    egr_ratio:              0
    equivalence_ratio:        0
    species:
        CH4: 0.111940298507
        O2: 0.186567164179
        N2: 0.701492537313
#-----
# One-D Case 3
#-----
- case:
    case_name:          OneD Case 3
    unburned_temp:        300.0
    unburned_pres:        1000000.0
    init_velocity:         0.3
    cutoff_temp:           10.0
    mixture_type:          SPECIFIED
    equivalence_ratio:        1.0
    fuel:
        CH4: 1.0
    oxidizer:
        O2: 0.21
        N2: 0.79
    egr_ratio:                0.1
    egr_ratio_definition:   NO_FUEL
    egr_mixture_type:        USER
    egr_species:
        N2: 0.7
        CO: 0.3
```

Figure 25.177: An example *one_d_cases.in* file for a 1D laminar freely propagating flame.

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Table 25.162: Format of the *one_d_cases* settings block for a 1D laminar freely propagating flame.

Parameter	Description
<i>one_d_cases</i>	
- <i>case</i>	This settings sub-block specifies 1D case parameters. Repeat this sub-block through < <i>species name</i> > (if <i>mixture_type</i> = <i>CUSTOM</i>), <i>egr_ratio</i> (if <i>mixture_type</i> = <i>SPECIFIED</i>) or < <i>egr_species</i> > (if <i>mixture_type</i> = <i>SPECIFIED</i> and <i>egr_ratio</i> > 0) for each case.
<i>case_name</i>	1D case name.
<i>unburned_temp</i>	Unburned temperature (K).
<i>unburned_pres</i>	Unburned pressure (Pa).
<i>init_velocity</i>	Initial velocity (m/s).
<i>cutoff_temp</i>	Combustion temperature cutoff (K) below which chemistry calculations are not solved.
<i>mixture_type</i>	Specify how to supply the mixture. <i>CUSTOM</i> = Supply <i>species names</i> and mass or mole fractions, <i>SPECIFIED</i> = Supply the <i>equivalence_ratio</i> , <i>fuel species</i> , and <i>oxidizer species</i> ; optionally, supply the EGR mixture and species.
<i>fuel</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
< <i>fuel species name</i> >	Fuel species mass or mole fraction. Repeat this line for each fuel species. Each species must be on a separate line.
<i>oxidizer</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
< <i>oxidizer species name</i> >	Oxidizer species mass or mole fraction. Repeat this line for each oxidizer species. Each species must be on a separate line.
<i>species</i>	This settings sub-block is used only when <i>mixture_type</i> = <i>CUSTOM</i> .
< <i>species name</i> >	Custom species mass or mole fraction. Repeat this line for each custom species. Each species must be on a separate line.
<i>equivalence_ratio</i>	Equivalence ratio of fuel and oxidizer. Used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
<i>egr_ratio</i>	The EGR ratio. Used only when <i>mixture_type</i> = <i>SPECIFIED</i> .
<i>egr_ratio_definition</i>	<i>NO_FUEL</i> = Do not include the fuel charge term in the EGR calculation, <i>WITH_FUEL</i> = Include the fuel charge term in the EGR calculation. Used only when <i>egr_ratio</i> is greater than 0.

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
<i>egr_mixture_type</i>	<i>USER</i> = Supply the EGR mixture, <i>CEQ</i> = CONVERGE computes the EGR mixture with CEQ.
	Used only when <i>egr_ratio</i> is greater than 0.
<i>egr_species</i>	
< <i>egr species</i> >	EGR species mass or mole fraction.
	Repeat this line for each EGR species. Each species must be on a separate line. Used only when <i>egr_ratio</i> is greater than 0.

Laminar Counterflow Calculation

To run a [laminar counterflow calculation](#), CONVERGE needs two files: *one_d_cases.in* and *one_d_solver.in*. Figure 25.178 shows an example *one_d_cases.in* file, which lists initial conditions for temperature, pressure, and species.

```
version: 3.1
---

one_d_cases:
#-----
# One-D Case 1
#-----
- case:
  case_name:          OneD Case 1 counterflow diffusion flame
  cutoff_temp:        10.0
  unburned_pres:     101325
  left_inlet_velocity: 0.1965
  left_inlet_temp:    300
  right_inlet_velocity: 0.61
  right_inlet_temp:   300
  init_type:          1
  mesh_type:          0
  #mesh_tol:           1e-6
  init_wmix:          0.001
  init_xcen:          0.006
  init_Tmax:          2000
  left_species:
    C2H6: 1.0
  right_species:
    O2: 0.21
    N2: 0.78
    Ar: 0.01
  product_species:
    H2O: 3.0
    CO2: 2.0
    N2: 11.28

#-----
# One-D Case 2
#-----
- case:
  case_name:          OneD Case 2 counterflow premixed flame
  cutoff_temp:        10.0
  unburned_pres:     101325
  left_inlet_velocity: 0.1965
  left_inlet_temp:    300
  right_inlet_velocity: 0.61
  right_inlet_temp:   300
  init_type:          0
```

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

```
mesh_type:          1
mesh_tol:           1e-6
init_wmix:          0.001
init_xcen:          0.006
init_Tmax:          2000
left_species:
    C2H6: 1.0
    O2:  0.21
    N2:  0.79
right_species:
    C2H6: 1.0
    O2:  0.21
    N2:  0.78
product_species:
    H2O: 3.0
    CO2: 2.0
    N2: 11.28
```

Figure 25.178: An example *one_d_cases.in* file for a laminar counterflow flame.

Table 25.163: Format of the *one_d_cases* settings block for a laminar counterflow flame.

Parameter	Description
<i>one_d_cases</i>	
- case	This settings sub-block specifies 1D case parameters. Repeat this sub-block through <i>product_species</i> for each case.
<i>case_name</i>	1D case name.
<i>cutoff_temp</i>	Combustion temperature cutoff (K). Source term is not evaluated for cell temperature lower than the given value.
<i>unburned_pres</i>	Unburned pressure (Pa).
<i>left_inlet_velocity</i>	Initial velocity at the left inlet (m/s).
<i>left_inlet_temp</i>	Initial temperature at the left inlet (K).
<i>right_inlet_velocity</i>	Initial velocity at the right inlet (m/s).
<i>right_inlet_temp</i>	Initial temperature at the right inlet (K).
<i>init_type</i>	0 = General plateau profile, 1 = Specific to diffusion counterflow flame (recommended when modeling a diffusion counterflow flame).
<i>mesh_type</i>	0 = Original mesh, 1 = CONVERGE mesh (useful for premixed counterflow flame).
<i>mesh_tol*</i>	Tuning parameter for CONVERGE mesh type.
<i>init_wmix</i>	Initial mixing width for plateau profile (m).
<i>init_xcen</i>	Initial center for plateau profile (m). Use a negative value to use the center of the domain.
<i>init_Tmax</i>	Initial maximum temperature for plateau profile (K).
<i>left_species</i>	
<i>species</i>	Species mass or mole fraction.

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
	Repeat this line for each of the left boundary inlet conditions. Each species must be on a separate line.
<i>right_species</i>	
<i>species</i>	Species mass or mole fraction.
	Repeat this line for each of the right boundary inlet conditions. Each species must be on a separate line.
<i>product_species</i>	
<i>species</i>	Initial estimations for product species mass or mole fraction.
	Repeat this line for each of the product estimations. Each species must be on a separate line.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

For a premixed counterflow flame, define a fuel and an oxidizer species in the left or right inlet (or both). For a diffusion counterflow flame, define one inlet with fuel species and one inlet with oxidizer species.

1D Template: *one_d_template.in*

To generate a [1D TLF](#) table, CONVERGE requires both [*one_d_template.in*](#) and [*one_d_solver.in*](#). The *one_d_template.in* file contains the ranges for the cases.

```
version: 3.1
---

pressure_control:
  type_flag: 1
  min: 100000.0
  max: 500000.0
  interval: 200000.0
  values: [100000.0, 200000.0, 400000.0]
temperature_control:
  type_flag: 1
  min: 300.0
  max: 1100.0
  interval: 100.0
  values: [300.0, 500.0, 700.0]
cutoff_temperature_control:
  active: 0
  type_flag: 1
  threshold: 600.0
  percent: 10.0
  value: 100.0
equivalence_ratio_control:
  type_flag: 0
  min: 0.5
  max: 1.7
  interval: 0.1
  values: []
egr_control:
  active: 0
  type_flag: 0
  min: 0.0
```

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

```
max: 0.4
interval: 0.1
values: []
ratio_definition_flag: 1
ceq_flag: 0
species:
    CO2: 0.4
    CO: 0.4
    H2O: 0.1
    H2: 0.1
species_control:
    custom_species_flag: 1
    fuel_species:
        CH4: [0.7000000, 0.5000000]
        CO2: [0.3000000, 0.5000000]
    oxidizer_species:
        O2: [0.2, 0.5000000]
        N2: [0.77, 0.5000000]
```

Figure 25.179: An example *one_d_template.in* file.

Table 25.164: Format of *one_d_template.in*.

Parameter	Description
<i>pressure_control</i>	
<i>type_flag</i>	0 = Compute table over the listed range of pressures, 1 = Compute table on specified pressures.
<i>min</i>	Minimum pressure (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum pressure (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	Pressure interval (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	Flow-formatted list of specified pressures (Pa) for <i>pressure_control</i> > <i>type_flag</i> = 1. Not read for <i>type_flag</i> = 0.
<i>temperature_control</i>	
<i>type_flag</i>	0 = Compute table over the listed range of temperatures, 1 = Compute table on specified temperatures.
<i>min</i>	Minimum temperature (K) for <i>temperature_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum temperature (K) for <i>temperature_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	Temperature interval (K) for <i>temperature_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	Flow-formatted list of specified temperatures (K) for <i>temperature_control</i> > <i>type_flag</i> = 1. Not read for <i>type_flag</i> = 0.
<i>cutoff_temperature_control</i>	
<i>active</i>	0 = Do not activate cutoff temperature,

Chapter 25: Input and Data Files

Chemistry One-Dimensional Chemistry Utilities

Parameter	Description
	1 = Activate cutoff temperature.
<i>type_flag</i>	0 = Specify cutoff temperature by percent, 1 = Specify cutoff temperature by value.
<i>threshold</i>	Minimum temperature threshold (K) for temperature cutoff activation.
<i>percent</i>	Percent added to the unburned temperature (if <i>cutoff_temperature_control</i> > <i>type_flag</i> = 0).
<i>value</i>	Value (K) added to the unburned temperature (if <i>cutoff_temperature_control</i> > <i>type_flag</i> = 1).
<i>equivalence_ratio_control</i>	
<i>type_flag</i>	0 = Compute table over the listed range of elemental equivalence ratios, 1 = Compute table on specified elemental equivalence ratios.
<i>min</i>	Minimum elemental equivalence ratio for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum elemental equivalence ratio for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	Elemental equivalence ratio interval for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	Flow-formatted list of specified elemental equivalence ratios for <i>equivalence_ratio_control</i> > <i>type_flag</i> = 1. Not read for <i>type_flag</i> = 0.
<i>egr_control</i>	
<i>active</i>	0 = Do not activate EGR, 1 = Activate EGR.
<i>type_flag</i>	0 = Compute table over the listed range of EGR ratios, 1 = Compute table on specified EGR ratios.
<i>min</i>	Minimum EGR ratio for <i>egr_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>max</i>	Maximum EGR ratio for <i>egr_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>interval</i>	EGR ratio interval for <i>egr_control</i> > <i>type_flag</i> = 0. Not read for <i>type_flag</i> = 1.
<i>values</i>	List of specified EGR ratios for <i>egr_control</i> > <i>type_flag</i> = 1. Not read for <i>type_flag</i> = 0.
<i>ratio_definition_flag</i>	0 = Define EGR ratio without including fuel mass (recommended when generating <i>tlf_table.h5</i> to be used in an ECFM/ECFM3Z simulation), 1 = Define EGR ratio including fuel mass.
<i>ceq_flag</i>	0 = User defined EGR species, 1 = Use CEQ equilibrium.

Parameter	Description
<i>species</i>	This settings sub-block is used for user-defined EGR species. This settings sub-block is not used if <i>ceq_flag</i> = 1 above.
< <i>species name</i> >	Ratio of this species. Repeat this line for each species. Each species must be on a separate line. Ratios must sum to 1.0.
<i>species_control</i>	
<i>custom_species_flag</i>	0 = Supply <i>species</i> names and mass or mole fractions, 1 = Supply the <i>fuel species</i> and <i>oxidizer</i> species.
<i>fuel_species</i>	
< <i>species name</i> >	Mass fraction of this fuel species. To specify multiple compositions of fuel, provide a flow-formatted list of species mass fractions (e.g., [0.7, 0.5]). Repeat this line for an additional fuel species. More than two fuel species is not allowed. Each pair of mass fractions for a composition must sum to 1.0.
<i>oxidizer_species</i>	
< <i>species name</i> >	Mass fraction of this oxidizer species. To specify multiple compositions of oxidizer, provide a flow-formatted list of species mass fractions (e.g., [0.2, 0.5]). Repeat this line for each oxidizer species. Each species must be on a separate line. Mass fractions for a composition must sum to 1.0.

Mechanism Reduction

This section describes the input files related to the [mechanism reduction utility](#) in CONVERGE.

Mechanism Reduction: mechanism_reduction.in

CONVERGE needs *mechanism_reduction.in* to perform [mechanism reduction](#) and generate a reduced mechanism, which is written to *mech_ske.dat*. Note that you can specify [temperature as a target](#) (with an associated error tolerance) for reduction, where the impact of the species on HRR is used to determine if the species is important and should be retained in the skeletal mechanism.

Chapter 25: Input and Data Files

Chemistry Mechanism Reduction

```
version: 3.1
---

dr gep_control:
    search_iterations: 2
    zero_d_control:
        active: 0
        ignition_delay_error: 0.2
    one_d_control:
        active: 0
        coefficient_flag: 0
sensitivity_analysis_control:
    subset_species_analyzed: 0.5
    zero_d_control:
        active: 0
        ignition_delay_error: 0.2
    one_d_control:
        active: 0
        flamespeed_error: 0.2
isomer_lumping_control:
    zero_d_control:
        active: 1
        ignition_delay_error: 0.3
one_d_normalized_flamespeed_error_flag: 1

target_species:
    nc7h16: 1.0
    ic8h18: 1.0
    c6h5ch3: 1.0
    n2: 1.0
    o2: 1.0
    oh: 0.01
    ho2: 0.01
    co: 0.01
    co2: 0.01
```

Figure 25.180: An example *mechanism_reduction.in* file.

Table 25.165: Format of *mechanism_reduction.in*.

Parameter	Description	Typical value
<i>dr gep_control</i>		
<i>search_iterations</i>	Control the search iterations for DRGEP. This value determines the number of groups the species in the comprehensive mechanism are divided into during the initial DRGEP calculations.	
<i>zero_d_control</i>		
<i>active</i>	0 = DRGEP is inactive, 1 = DRGEP is active in 0D mode.	
<i>ignition_delay_error</i>	Normalized maximum ignition delay error for DRGEP.	0.5
<i>one_d_control</i>		
<i>active</i>	0 = DRGEP is inactive, 1 = DRGEP is active in 1D mode.	

Chapter 25: Input and Data Files

Chemistry Mechanism Reduction

Parameter	Description	Typical value
<i>flamespeed_error</i>	Normalized or actual maximum flamespeed error for DRGEP (<i>m/s</i>).	0.2
<i>coefficient_flag</i>	0 = Consider only 0D solution in the calculation of the DRGEP coefficient, 1 = Consider 1D solution in the calculation of the DRGEP coefficient.	
<i>sensitivity_analysis_control</i>		
<i>subset_species_analyzed</i>	Fraction of species on which analysis will be performed.	
<i>zero_d_control</i>		
<i>active</i>	0 = Sensitivity analysis is inactive, 1 = Sensitivity analysis is active in 0D mode.	
<i>ignition_delay_error</i>	Normalized maximum ignition delay error.	
<i>one_d_control</i>		
<i>active</i>	0 = Sensitivity analysis is inactive, 1 = Sensitivity analysis is active 1D mode.	
<i>flamespeed_error</i>	Normalized or actual flamespeed error for sensitivity analysis (<i>m/s</i>).	0.05
<i>isomer_lumping_control</i>		
<i>zero_d_control</i>		
<i>active</i>	0 = Automatic isomer lumping is inactive, 1 = Automatic isomer lumping is active (requires <i>mechanism_reduction.in > sensitivity_analysis_control > zero_d_control > active = 1</i>). See the Isomer Lumping Reaction Option section for more information about isomer lumping.	
<i>ignition_delay_error</i>	Normalized maximum ignition delay error for isomer lumping. A larger value will lead to a smaller mechanism (<i>i.e.</i> , more lumping).	
<i>one_d_normalized_flamespeed_error_flag</i>	Control both DRGEP and sensitivity analysis. 0 = Flamespeed error (<i>m/s</i>), 1 = Normalized flamespeed error.	

Chapter 25: Input and Data Files

Chemistry Mechanism Reduction

Parameter	Description	Typical value
<i>target_species</i>	<p><<i>species name</i>></p> <p>Name of the target species, followed by the weight. Use weight 1.00 for all target species. Use a value between 0 and 1 (e.g., 0.05) for species that are not target species but should be retained in the final skeletal mechanism. Add a negative sign in front of the weight to direct CONVERGE to include only this species (not its connected species as well) in the reduced mechanism.</p> <p>Repeat this line for each target species. Each species must be on a separate line.</p>	

Dynamic Mechanism Reduction: sage_dmr.in

The DMR option will reduce the mechanism during the SAGE simulation, based on target weight and error propagation tolerance values you specify. To activate the [Dynamic Mechanism Reduction](#) utility, you must first activate the SAGE detailed chemistry solver by setting [*combust.in*](#) > *sage_model* > *active* = 1. Configure the [SAGE](#)-related parameters as needed. DMR may be used in conjunction with the SAGE adaptive zoning option.

Then, set [*combust.in*](#) > *sage_model* > *dmr_flag* = 1. CONVERGE will look for a file named *sage_dmr.in* in your case setup. Since DMR automatically removes species based on local thermo-chemical conditions, you may lose species that are important to your simulation, such as soot precursors for emission modeling. You can save species information by setting *sage_dmr.in* > *sage_dmr_species_flag* = 1 for no-target species, as shown in the following example *sage_dmr.in* file.

Target Temperature Accuracy

In some cases, species that are important for heat release rate can be lost. You can specify TEMP as a target along with the error tolerance. The change in heat release rate due to the loss of the species and its reactions is calculated and normalized. Species that have errors above the tolerance (and therefore are important for accurate calculation of HRR) are retained in the skeletal mechanism. Note that these HRR-relevant species may already be listed in the target list or identified by CONVERGE by its typical species-based approach. In such cases, you may want to require a tighter tolerance to see an appreciable difference in the reduced mechanism or corresponding calculations.

Chapter 25: Input and Data Files

Chemistry Mechanism Reduction

```
version: 3.1
---

dmr_target_species:
    C7H16: 1.0
    H2O: 1.0
    CO: 1.0
    CO2: 1.0
    CH4: 0.1
drgep_tolerance: 0.005
sage_dmr_species_flag: 1
sage_dmr_species:
    - A3R5
    - A3R5-
```

Figure 25.181: An example *sage_dmr.in* file.

Table 25.166: Format of *sage_dmr.in*.

Parameter	Description
<i>dmr_target_species</i>	This settings sub-block lists the target species that must be preserved during DMR.
<i><species name> or TEMP</i>	When <i><species name></i> is specified: Species weight. The weight is value from 0 to 1.0. Target species typically have a weight of 1.0. When <i>TEMP</i> is specified: Error tolerance. Typical value is 1e-3.
	Repeat this line for each target species to preserve during DMR. Each species must be on a separate line.
<i>drgep_tolerance</i>	Error propagation tolerance for the Directed Relation Graph.
<i>sage_dmr_species_flag</i>	0 = Do not save non-target species from being lost due to DMR, 1 = Save non-target species from being lost due to DMR.
<i>sage_dmr_species</i>	This settings sub-block is used to define the species that DMR cannot remove during a simulation. This settings sub-block is used only if <i>sage_dmr_species_flag</i> = 1.
<i><species name></i>	Species that DMR cannot remove. Repeat this line for each desired species. Each species must be on a separate line.

Mechanism Tune: *mechanism_tune.in*

The [mechanism tune](#) utility requires the *mechanism_tune.in* input file, which specifies the solver type; the number of reactions to modify in the [reaction mechanism file](#); the name of the directory that contains the genetic algorithm optimizations files; flags to indicate if the utility will be run with the 0D solver, 1D premixed laminar flamespeed solver, or both; and required parameters for setting up the [case.in](#) and [merit.in](#) files.

Chapter 25: Input and Data Files

Chemistry Mechanism Tune: mechanism_tune.in

```
version: 3.1
---

solver_type: NLOPT
nlopt_control:
    model_type: GN_DIRECT_L
    max_num_evaluations: 1000
    rel_tol: 1.0e-2
    abs_tol: 1.0e-2
    main_reaction_control:
        group_dependency_control:
            active: 0
            type: TEMPERATURE
        a_coeff_control:
            active: 1
            reaction_uncertainty_model: MULTIPLIERS
            lower_multiplier: 0.25
            upper_multiplier: 4.00
            sensitive_direction_flag: 0
            initial_guess: [1.0]
        ea_control:
            active: 0
            lower_multiplier: 0.9
            upper_multiplier: 1.1
            initial_guess: [1.0]
    congo_control:
        directory_name: gafolder
        a_coeff_order_of_mag_variation: 1.0
    zero_d_control:
        active: 1
        num_reactions_to_modify: 15
        read_sensitivity_results_flag: 0
    one_d_control:
        active: 1
        num_reactions_to_modify: 7
        read_sensitivity_results_flag: 0
targets_control:
    - target:
        type: IGNITIONDELAY
        values: [9.8792105e-02, 4.4829933e-02, 2.1610038e-02,
1.1022426e-02]
        model: CONSTRAINT
        weights: [1.0, 1.0, 1.0, 1.0]
    - target:
        type: FLAMESPEED
        values: [0.12685, 0.42420, 0.79410, 1.13662]
        model: PERFORMANCE
        weights: [1.0, 1.0, 1.0, 1.0]
        constraint_type: ""
```

Figure 25.182: An example *mechanism_tune.in* file.

Table 25.167: Format of *mechanism_tune.in*.

Parameter	Description	Typical value
<i>solver_type</i>	Solver for tuning mechanism. CONGO = CONVERGE Genetic Optimization (requires the <i>congo_control</i> settings block below), NLOPT = NLopt optimization package (requires the <i>nlopt_control</i> settings block below).	
<i>nlopt_control</i>		

Chapter 25: Input and Data Files

Chemistry Mechanism Tune: mechanism_tune.in

Parameter	Description	Typical value
<i>model_type</i>	NLOpt model (https://nlopt.readthedocs.io/en/latest/NLOpt_Algorithms/). For simplicity, these models are named the same as in NLOpt (with the exception that NLOpt includes <i>NLOPT_</i> at the beginning of all algorithm names). <i>GN_DIRECT</i> = Dividing rectangles algorithm for global optimization, <i>GN_DIRECT_L</i> = Locally-biased DIRECT algorithm for global optimization, <i>LN_COBYLA</i> = Constrained optimization by linear approximations algorithm for local optimization, <i>LN_BOBYQA</i> = Bound optimization by quadratic approximation algorithm for local optimization, <i>LN_NEWUOA_BOUND</i> = Variation of NEWUOA that permits bound constraints for local optimization, <i>LN_NELDERMEAD</i> = Nelder-Mead simplex algorithm for local optimization.	
<i>max_num_evaluations</i>	Maximum number of evaluations for NLOpt to solve to reach convergence.	1000
<i>rel_tol</i>	Relative tolerance. Available range: 1.0e-6 to 1.0.	1.0e-2
<i>abs_tol</i>	Absolute tolerance. Available range: 1.0e-8 to 1.0	1.0e-2
<i>main_reaction_control</i>	This settings sub-block specifies the main reaction controls for the NLOpt solver.	
<i>group_dependency_control</i>		
<i>active</i>	0 = Off, 1 = 1D group dependency active.	
<i>type</i>	Type of group dependency.	TEMPERATURE
<i>a_coeff_control</i>	This settings sub-block specifies the parameters for the reaction A factors for the forward reactions.	
<i>active</i>	0 = Off, 1 = Modify A coefficients in forward reaction rates.	
<i>reaction_uncertainty_model</i>	<i>MULTIPLIERS</i> = Use <i>lower_multiplier</i> and <i>upper_multiplier</i> ,	

Chapter 25: Input and Data Files

Chemistry Mechanism Tune: mechanism_tune.in

Parameter	Description	Typical value
	<i>MECHANISM</i> = Use reaction mechanism file with <i>UNCRT/<uncertainty factor>/</i> keyword for reaction A factors to vary (if <i>UNCRT/<uncertainty factor>/</i> keyword does not exist for a sensitive reaction, NLOpt will use <i>MULTIPLIERS</i> for only that reaction), <i><file name></i> = Provide a file name that contains a list of reaction numbers followed by uncertainties, with each reaction on a separate line (if a sensitive reaction is not found in the file, NLOpt will use <i>MULTIPLIERS</i> for only that reaction).	
<i>lower_multiplier</i>	Lower multiplier.	0.25
<i>upper_multiplier</i>	Upper multiplier.	4.00
<i>sensitive_direction_flag</i> *	Reaction A factors are changed in: 0 = Both the positive and negative directions, 1 = The sensitive direction as determined by the sensitivity analysis.	
<i>initial_guess</i>	Initial guess(es) within the lower and upper multipliers for all 0D and 1D <i>num_reactions_to_modify</i> reaction A factors.	
<i>ea_control</i>	This settings sub-block specifies the parameters for the activation energy for the forward reactions.	
<i>active</i>	0 = Off, 1 = Modify activation energy in forward reaction rates.	
<i>lower_multiplier</i>	Lower multiplier.	
<i>upper_multiplier</i>	Upper multiplier.	
<i>initial_guess</i>	Initial guess within the lower and upper multiplier for all 0D and 1D <i>num_reactions_to_modify</i> activation energy multipliers.	
<i>congo_control</i>		
<i>directory_name</i>	Directory name where CONGO will be run.	
<i>a_coeff_order_of_mag_variation</i>	Specify a factor by which the magnitude of the reaction A factors will vary.	1.0
<i>zero_d_control</i>		

Chapter 25: Input and Data Files

Chemistry Mechanism Tune: mechanism_tune.in

Parameter	Description	Typical value
<i>active</i>	0 = Do not run 0D cases with adjoint sensitivity analysis, 1 = Run 0D cases with adjoint sensitivity analysis.	
<i>num_reactions_to_modify</i>	Number of 0D sensitive reactions in which the <i>A</i> factor will be modified.	10
<i>read_sensitivity_results_flag</i>	0 = Do not read <i>zero_d_adjoint_rank.out</i> , 1 = Read a completed <i>zero_d_adjoint_rank.out</i> file (sensitivity will not be re-run and the mechanism tune utility will set up CONGO input files [<i>mechanism_tune.in</i> > <i>solver_type</i> = CONGO] or begin tuning [<i>mechanism_tune.in</i> > <i>solver_type</i> = NLOPT] based on a completed <i>zero_d_adjoint_rank.out</i> file).	0
<i>one_d_control</i>		
<i>active</i>	0 = Do not run 1D cases with sensitivity analysis, 1 = Run 1D cases with sensitivity analysis.	
<i>num_reactions_to_modify</i>	Number of 1D sensitive reactions in which the <i>A</i> factor will be modified.	10
<i>read_sensitivity_results_flag</i>	0 = Do not read the <i>one_d_sens_rank.out</i> , 1 = Read a completed <i>one_d_sens_rank.out</i> file (sensitivity will not be re-run and the mechanism tune utility will set up CONGO input files [<i>mechanism_tune.in</i> > <i>solver_type</i> = CONGO] or begin tuning [<i>mechanism_tune.in</i> > <i>solver_type</i> = NLOPT] based on a completed <i>one_d_sens_rank.out</i> file).	0
<i>targets_control</i>		
<i>- target</i>		
<i>type</i>	Target type. Must list either <i>IGNITIONDELAY</i> (optionally followed by a number) or <i>FLAMESPEED</i> (optionally followed by a number).	
<i>values</i>	Flow-formatted set of target values.	
<i>model</i>	Model type. Must list either <i>CONSTRAINT</i> or <i>PERFORMANCE</i> .	
<i>weights</i>	Flow-formatted set of either constraint weights or performance weights to the target, depending on model.	
<i>constraint_type</i>	<i>MAXIMUM</i> = Greater than,	

Chapter 25: Input and Data Files

Chemistry Mechanism Tune: mechanism_tune.in

Parameter	Description	Typical value
	<i>MINIMUM</i> = Less than. Read only when <i>model</i> = <i>CONSTRAINT</i> .	

* Note: Blocks or parameters indicated with an asterisk are optional.

Surrogate Blender: *blender.in*

The [surrogate blender](#) can import and export *blender.in* files. A *blender.in* file contains a table with information about component and target fuel properties and the weight given to each property.

```
1          BLENDER_FLAG
1          USE_GLOBAL_NLOPT_FLAG
0.001      NLOPT_EQUALITY_TOLERANCE
0.0001     NLOPT_RELATIVE_TOLERANCE
0.0001     NLOPT_ABSOLUTE_TOLERANCE
0          USE_INITIAL_GUESS_FLAG
4          NUM_COMPONENTS
4          NUM_TARGETS
TABLE      MW           TSI           H/C           DEN@288.15
C9H12      120.196    52            1.33333     867.931
NC12H26    170.341    5.1           2.16667     751.912
IC8H18     114.233    7.7           2.25          686.27
X135C9H12  120.196    61            1.33333     869.436
TARGETS    142.0       21.4          1.96          804.0
WEIGHTS    1.0          1.0           1.0           1.0
TYPE       MOLE         MOLE          ATOM          MOLE
1          DCN_FLAG
0          DCN_MODEL
47.1 1
0          OCTANE_FLAG
0          OCTANE_MODEL
0          DISTILLATION_FLAG
```

Figure 25.183: An example *blender.in* file.

Table 25.168: Format of *blender.in*.

0 = Surrogate blender off,	BLENDER_FLAG
1 = Surrogate blender on.	
0 = Use local nonlinear optimization localization,	USE_GLOBAL_NLOPT_FLAG
1 = Use global nonlinear optimization localization.	
Equality tolerance for nonlinear optimization.	NLOPT_EQUALITY_TOLERANCE
Relative tolerance for nonlinear optimization.	NLOPT_RELATIVE_TOLERANCE
Absolute tolerance for nonlinear optimization.	NLOPT_ABSOLUTE_TOLERANCE
0 = Do not use initial guess,	USE_INITIAL_GUESS_FLAG
1 = Use initial guess for distribution.	

Chapter 25: Input and Data Files

Chemistry Surrogate Blender: blender.in

Number of components.	NUM_COMPONENTS	
Number of property targets.	NUM_TARGETS	
TABLE	Property j	Property $j+1$
Fuel i	Property j for fuel i .	Property $j+1$ for fuel i .
Fuel $i+1$	Property j for fuel $i+1$.	Property $j+1$ for fuel $i+1$.
(add rows for additional fuels)		
Targets	Property j for target fuel.	Property $j+1$ for target fuel.
Weights	Weight of property j . Must be a number between 0 and 999.	Weight of property $j+1$. Must be a number between 0 and 999.
Type	Type of blending of property j for target fuel.	Type of blending of property $j+1$ for target fuel.
0 = Do not determine the derived cetane number (DCN) for the fuel or the fuel mixture. 1 = Determine DCN for the fuel or the fuel mixture.	DCN_FLAG	
0 = Use Jameel et al., 2016 model to determine DCN for the fuel or fuel mixture. 1 = Determine DCN for the fuel or the fuel mixture.	DCN_MODEL	
Target, Weight	Target and weight for the DCN model.	
0 = Do not determine the RON, MON, and octane number for the fuel or the fuel mixture. 1 = Determine the RON, MON, and octane number for the fuel or the fuel mixture.	OCTANE_FLAG	
0 = Use the Jameel et al., 2018 model to determine the RON, MON, and octane number, 1 = Use the Ghosh et al., 2005 model to determine the RON, MON, and octane number.	OCTANE_MODEL	
0 = Do not specify the distillation percent for the fuel, 1 = Specify the distillation percent for the fuel.	DISTILLATION_FLAG	

Chapter 25: Input and Data Files

Chemistry Surrogate Blender: blender.in

Distilled percent, Temperature, Target values for distillation.
Weight

Pathway Flux Analysis: pfa.in

[Pathway flux analysis](#) (PFA) calculates the reaction rates at specific times based on the kinetic mechanism and relevant species concentrations in 0D or 1D solution files. To run the PFA utility, provide a [0D](#) or [1D](#) solution file, [mechanism data file](#), [thermodynamic data file](#), and a *pfa.in* file in the Case Directory.

```
version: 3.1
---

mechanism_filename: mech.dat
thermodynamic_filename: therm.dat
pfa_cases:
  - case:
    case_name: case_1
    solution_filename: zero_d_sol_case00001.out
    start: 0.001
    end: 0.0011
    reaction_rate_multiplier: 1
    take_top_n_reactions: 20
    low_mole_fraction_filter: 1e-20
    pathway_diagram_control:
      element_filter: C
      element_threshold: 6
      reaction_rate_filter: 1e-10
      merge_links: 0
      contribution_percentage_filter: 0.01
  - case:
    case_name: case_2
    solution_filename: zero_d_sol_case00001.out
    start: 4e-4
    end: 4e-4
    reaction_rate_multiplier: 1
    take_top_n_reactions: 20
    low_mole_fraction_filter: 1e-8
    pathway_diagram_control:
      element_filter: C
      element_threshold: 6
      reaction_rate_filter: 1e-10
      merge_links: 0
      contribution_percentage_filter: 0.1
```

Figure 25.184: An example *pfa.in* file.

Table 25.169: Format of *pfa.in*.

Parameter	Description
<i>mechanism_filename</i>	The name of the reaction mechanism data file, e.g., <i>mech.dat</i> .
<i>thermodynamic_filename</i>	The name of the thermodynamic data file, e.g., <i>therm.dat</i> .
<i>pfa_cases</i>	
- <i>case</i>	
<i>case_name</i>	Name for PFA case.
<i>solution_filename</i>	0D or 1D solution file name on which PFA is performed.

Chapter 25: Input and Data Files

Chemistry Pathway Flux Analysis: pfa.in

Parameter	Description
<i>start</i>	The starting point for PFA in time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) for a 0D case or in distance (meters) for a 1D case. If start time exceeds the range in the solution file, the minimum value in the solution file will be assigned and a warning message will appear. If <i>start</i> and <i>end</i> are identical or closer than the local step size, PFA will be performed at $(start+end)/2$.
<i>end</i>	The ending point for PFA in time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) for a 0D case or in distance (meters) for a 1D case. If end time exceeds the range in the solution file, the maximum value in the solution file will be assigned and a warning message will appear. If <i>start</i> and <i>end</i> are identical or closer than the local step size, PFA will be performed at $(start+end)/2$.
<i>reaction_rate_multiplier</i>	Scaling factor of reaction rates to increase or decrease all reaction rates simultaneously. This value must match combust.in > <i>sage_model</i> > <i>ode_solver</i> > <i>reaction_multiplier</i> to obtain accurate results.
<i>take_top_n_reactions</i>	For each species, consider only the top <i>n</i> reactions with the highest contribution percentages.
<i>low_mole_fraction_filter</i>	Consider only the species that have mole fractions higher than this threshold.
<i>pathway_diagram_control</i>	The filters below apply only to the visualization of the PFA and not to the spec_mol_<number>.out or rate_tab_<number>.out files.
<i>element_filter</i>	Consider only the species that have the specified element (e.g., C, H, O, N).
<i>element_threshold</i>	Consider only the species that has more than <i>n</i> atoms of the specified element.
<i>reaction_rate_filter</i>	Consider only reaction rates that are faster than this minimum reaction rate when generating the reaction pathway diagram.
<i>merge_links</i>	0 = Do not combine arrows with identical sources and targets, 1 = When multiple arrows have identical sources and targets, combine into a wider arrow and sum the contributions.
<i>contribution_percentage_filter</i>	Show only reactions that have contribution percentages higher than this threshold.

25.13 Heat Transfer Mapping

This section describes the input files for the [heat transfer mapping utility](#).

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Mapping: htc_map.in

Heat Transfer Mapping: htc_map.in

The *htc_map.in* file is required to run the [heat transfer mapping utility](#).

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Mapping: htc_map.in

```
version: 3.1
---

#-----Geometry Control-----#
2.000000e-03      tolerance
1.0               grow_mult
2                 grow_interp_scheme
2                 num_copies
1.0               scale_xyz
0.0               rot_angle_x
0.0               rot_angle_y
0.0               rot_angle_z
0.0               trans_x
0.0               trans_y
0.0               trans_z
1.0               scale_xyz
0.0               rot_angle_x
0.0               rot_angle_y
0.0               rot_angle_z
0.0               trans_x
0.0               trans_y
0.0               trans_z
-0.2              trans_z
sc-rx-ry-rz-tx-ty-tz   orderofoperations
#-----ABAQUS Control-----#
1                 read_abaqus
1                 num_surface
PISTON_FACE        name
1                 use_elslet
0                 write_pressure_abaqus
#-----HTC Control-----#
1e10              max_htc
1                 use_bulk_fluid_temp
2                 bulk_temp_num_bounds
1                 transferboundID
300.0             Tbulk
2                 transferboundID
400.0             Tbulk
#-----Mapping Control-----#
0                 time_avg_option
1                 temp_map_option
2                 enforce_boundID_match
2                 num_boundaries
1                 boundID
2                 boundID
1                 enforce_num_boundary_groups
3                 num_transfer_boundaries
2                 boundID
4                 boundID
8                 boundID
2                 num_surface_boundaries
1                 boundID
3                 boundID
0                 hit_location_output_files
1                 map_additional_variables
0                 use_convective_ht
#-----Output Control-----#
1                 ensight_outputfiles
1                 ensight_gold_format_option
0                 gmv_outputfiles
0                 fieldview_outputfiles
1                 tecplot_outputfiles
1                 dedicatedboundaryoutputfiles
Piston            name
4                 numboundaries
1                 boundary
2                 boundary
3                 boundary
8                 boundary
#-----Time Control-----#
-440.0            start_time
280.0             end_time
0                 startoption
#-----Moving Boundaries-----#
2                 num_valve_entries
16 13
17 8
```

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Mapping: htc_map.in

Table 25.170: Format of *htc_map.in*.

Parameter	Description	Typical value
<i>tolerance</i>	Distance (<i>m</i>) used by search algorithm to initialize triangles from the corresponding <i>transfer.out</i> data point.	2.0e-3
<i>grow_mult</i>	Internal grid projection growth multiplier. Do not change.	1.0
<i>grow_interp_scheme</i>	Grid projection interpolation scheme. 0 = Arithmetic average, 1 = Inverse distance weighted average, 2 = Inverse distance-squared weighted average.	2
<i>num_copies</i>	Number of copies of the original geometry (<i>i.e.</i> , the geometry used to generate <i>transfer.out</i>) in the target surface file. For example, set <i>num_copies</i> = 2 to map the results of a single-cylinder simulation to a 2-cylinder target surface. Include a separate set of <i>scale_xyz</i> , <i>rot_angle_*</i> , and <i>trans_*</i> parameters for each copy.	1
<i>scale_xyz</i>	Scaling factor to be applied to the x, y, and z coordinates of the target surface file in order to align with the x, y, and z coordinates in the <i>transfer.out</i> file. This factor also can be used to change the length units.	1.00
<i>rot_angle_x</i> , <i>rot_angle_y</i> , <i>rot_angle_z</i>	Rotation values (in <i>degrees</i>) used to rotate about the x, y, and z axes of the target surface.	0
<i>trans_x</i> , <i>trans_y</i> , <i>trans_z</i>	Translation values (in <i>meters</i>) used to translate the x, y, and z coordinates of the target surface.	0
<i>orderofoperations</i>	Order of scale, rotation, and translation operations, delimited with dashes.	sc-rx-ry-rz- tx-ty-tz
<i>read_abaqus</i>	Flag to read an Abaqus file. 0 = Do not read, 1 = Read Abaqus file (add another row providing file name) and use file for HTC mapping, 2 = Read Abaqus file (add another row providing file name) and write a surface file as <i>abaqus.dat</i> .	0
<i>num_surface</i>	Number of surfaces in Abaqus file (applies only when <i>read_abaqus</i> is non-zero above).	
<i>name</i>	Surface name from Abaqus file (applies only when <i>num_surface</i> is greater than 0 above). Repeat this line for each surface.	

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Mapping: htc_map.in

Parameter	Description	Typical value
<i>use_elset</i>	Flag to use elements defined in Abaqus file (applies only when <i>read_abaqus</i> is non-zero above). 0 = Do not use elements defined in Abaqus file, 1 = Use elements defined in Abaqus file.	
<i>write_pressure_abaqus</i>	Flag to write mapped pressure data (applies only when <i>read_abaqus</i> > 0). 0 = Do not write the mapped pressure field to a file, 1 = Write the mapped pressure field to a file called <i>pres_abaqus.inp</i> .	0
<i>max_htc</i>	Maximum allowed heat transfer coefficient. Larger values of heat transfer coefficient will be capped to this value.	1e10
<i>use_bulk_fluid_temp</i>	Flag to use bulk fluid temperature for heat transfer coefficient mapping. 0 = Do not use bulk fluid temperature, 1 = Use bulk fluid temperature for specified boundaries.	0
<i>bulk_temp_num_bounds</i>	Number of boundaries using a bulk fluid temperature (applies only when <i>use_bulk_fluid_temp</i> = 1 above). <i>transferboundID</i> and <i>Tbulk</i> must be provided for each boundary (see below).	
<i>transferboundID</i>	Boundary ID from <i>transfer.out</i> .	
<i>Tbulk</i>	Bulk fluid temperature (K) for the specified boundary.	
<i>time_avg_option</i>	0 = Average heat transfer coefficient over all simulation time, 1 = Average heat transfer coefficient over only the time the triangle was in contact with the fluid.	0
<i>temp_map_option</i>	0 = Use arithmetic average for temperature, 1 = Use HTC-weighted average for temperature.	1
<i>enforce_boundID_match</i>	0 = Do not enforce consistent boundary IDs between <i>transfer.out</i> and the target surface file, 1 = Enforce consistent boundary IDs between <i>transfer.out</i> and the target surface file for all boundaries, 2 = Enforce consistent boundary IDs only for specified boundaries, 3 = Provide groups of boundary IDs from <i>transfer.out</i> and the target surface file for	1

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Mapping: htc_map.in

Parameter	Description	Typical value
	boundary ID matching. For each group, near-wall cell data with the listed transfer boundary IDs can be mapped only to triangles with the listed surface boundary IDs.	
<i>num_boundaries</i>	Number of boundaries for which IDs must be consistent between <i>transfer.out</i> and the target surface file (applies only when <i>enforce_boundID_match</i> = 2). Provide <i>boundID</i> for each boundary.	
<i>boundID</i>	Boundary ID that must be consistent (applies only when <i>enforce_boundID_match</i> = 2).	
<i>enforce_num_boundary_groups</i>	Number of boundary ID groups for boundary ID matching (applies only when <i>enforce_boundID_match</i> = 3).	
<i>num_transfer_boundaries</i>	Number of transfer boundaries in the current group (applies only when <i>enforce_boundID_match</i> = 3). Provide <i>boundID</i> for each transfer boundary.	
<i>boundID</i>	Transfer boundary ID for the current group (applies only when <i>enforce_boundID_match</i> = 3).	
<i>num_surface_boundaries</i>	Number of surface boundaries in the current group (applies only when <i>enforce_boundID_match</i> = 3). Provide <i>boundID</i> for each surface boundary.	
<i>boundID</i>	Surface boundary ID for the current group (applies only when <i>enforce_boundID_match</i> = 3).	
<i>hit_location_outputfiles</i>	Create detailed *.dat files showing locations of data points used to initialize triangles (can be loaded into CONVERGE Studio for viewing) in the directory <i>detailedoutput</i> . There are four different detailed files: one file listing the uninitialized triangles, and separate files listing the triangles initialized by each algorithm (<i>Direct Hits</i> , <i>Neighbors</i> and <i>Grown Points</i>). You can load these files in CONVERGE Studio for diagnostic purposes to monitor the nature of FEA triangles.	0
<i>map_additional_variables</i>	Map additional variables from <i>transfer.out</i> , if present.	0
<i>use_convective_ht</i>	Use convective heat transfer and heat flux rather than total heat transfer (requires <i>map_additional_variables</i> = 1 above).	0
<i>ensight_outputfiles</i>	0 = Do not create output files for geometry and variables that can be loaded into EnSight,	1

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Mapping: htc_map.in

Parameter	Description	Typical value
	1 = Create output files for geometry and variables that can be loaded into EnSight (*.case, *.geo, *.htc, etc.).	
<i>ensight_gold_format_option</i>	0 = Use EnSight6 file format, 1 = Use EnSight Gold file format.	1
<i>gmv_outputfiles</i>	0 = Do not create output files that can be loaded to General Mesh Viewer (GMV), 1 = Create output files that can be loaded to GMV.	1
<i>fieldview_outputfiles</i>	0 = Do not create output files that can be loaded into Fieldview, 1 = Create output files that can be loaded into FieldView.	1
<i>tecplot_outputfiles</i>	0 = Do not create output files that can be loaded into Tecplot, 1 = Create Tecplot files in ASCII format, 2 = Create Tecplot files in binary format.	1
<i>dedicatedboundaryoutputfiles</i>	0 = Do not create dedicated boundary output files, N = Create N dedicated boundary output files. For each file, provide <i>name</i> , <i>numboundaries</i> , and the boundary IDs to include (see below).	0
<i>name</i>	Name of boundary output file (applies only when <i>dedicatedboundaryoutputfiles</i> is greater than 0).	
<i>numboundaries</i>	Number of boundaries for the current file (applies only when <i>dedicatedboundaryoutputfiles</i> is greater than 0).	
<i>boundary</i>	Boundary ID to be included in the current file (applies only when <i>dedicatedboundaryoutputfiles</i> is greater than 0).	
<i>start_time</i>	Start time for averaging of data.	Start time of an engine cycle.
<i>end_time</i>	End time for averaging of data.	End time of an engine cycle.
<i>startoption</i>	0 = Neglect first data dump in <i>transfer.out</i> and use the second data dump for initialization, 1 = Use the first data dump for initialization.	0

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Mapping: htc_map.in

Parameter	Description	Typical value
<code>num_valve_entries</code>	Number of entries needed to account for the mapping of moving valves.	
<code><boundary ID 1> <boundary ID 2></code>	Boundary 1 is typically the valve stem and Boundary 2 is typically the moving part of the valve. The number of entries for this pair depends on the value in <code>num_valve_entries</code> .	

You can use the `htc_map` utility to transform a surface temperature file between coordinate systems. Prepend the transformation parameters from the `htc_map.in` file to the surface temperature file, and run the `htc_map` utility with the `transform` option, as `htc_map transform inputfilename.dat outputfilename.dat`. Figure 25.186 shows a sample `surface_temperature.dat` file with the transformation information included.

```
1.0          scale_xyz
0.0          rot_angle_x
0.0          rot_angle_y
0.0          rot_angle_z
0.0          trans_x
0.0          trans_y
0.0          trans_z
sc-rx-ry-rz-tx-ty-tz  orderofoperations
x           y           z           T
0.000      0.000      0.000      300.0
0.100      0.000      0.000      310.0
0.100      0.100      0.000      320.0
0.000      0.100      0.000      310.0
```

Figure 25.186: An example surface temperature file that includes transformation information.

Heat Transfer Output Control: transfer.in

The `transfer.in` file specifies the fluid WALL boundaries for which CONVERGE will write wall heat transfer data. Enter this file name (typically `transfer.in`) for [inputs.in > output_control > transfer_flag](#). You can also specify extra output variables to include in [transfer.out](#).

```
version: 3.1
---

boundary_definition:
- boundary:
  boundary_id: 1
- boundary:
  boundary_id: 2
  side: FORWARD

variables:
- velocity
- cond
```

Figure 25.187: An example `transfer.in` file.

Chapter 25: Input and Data Files

Heat Transfer Mapping Heat Transfer Output Control: *transfer.in*

Table 25.171: Format of *transfer.in*.

Parameter	Description
<i>boundary_definition</i>	
- <i>boundary</i>	Boundary sub-block. Repeat for each boundary for which CONVERGE will write wall heat transfer data.
<i>boundary_id</i>	Boundary ID.
<i>side</i>	FORWARD or REVERSE (used only if <i>boundary_id</i> corresponds to an INTERFACE boundary).
<i>variables*</i>	
- <i>sie</i> *	Include cell specific internal energy (J/kg) in <i>transfer.out</i> . This variable includes the formation energy of species.
- <i>velocity</i> *	Include all three components of cell velocity (m/s) in <i>transfer.out</i> .
- <i>eps</i> *	Include cell turbulence dissipation rate (m^2/s^3) in <i>transfer.out</i> . This variable is for RANS turbulence models.
- <i>cond</i> *	Include cell thermal conductivity (W/m-K) in <i>transfer.out</i> . This variable includes the turbulent component of thermal conductivity.
- <i>tke</i> *	Include cell turbulent kinetic energy (m^2/s^2) in <i>transfer.out</i> . This variable is for RANS turbulence models.
- <i>sgs_ke</i> *	Include cell sub-grid scale kinetic energy (m^2/s^2) in <i>transfer.out</i> . This variable is for LES turbulence models.
- <i>sgs_eps</i> *	Include cell sub-grid scale dissipation rate (m^2/s^3) in <i>transfer.out</i> . This variable is for LES turbulence models.

* Note: Sub-blocks or parameters indicated with an asterisk are optional.

25.14 CONGO

This section describes the input files for CONVERGE's [optimization and model interrogation utility](#).

GA/DoE Setup: *congo.in*

The *congo.in* file, which is mandatory for all CONGO cases, defines the parameters for the genetic algorithm (GA) or design of experiments (DoE).

Chapter 25: Input and Data Files

CONGO GA/DoE Setup: congo.in

```
0          restart_flag
0          run_individual
0          run_generation
0          delay_between_runs
5          pause_to_check_for_results
20         gen_timeout_minutes
0          monitor_runs_lib_flag
0          new_dir_flag
2          num_copy_templates
*          copy_template
*.*        copy_template
1          input_files
inputs.in   file_name
100        max_gen
1          random_seed_flag
0.00       mutation_frac
0.97       p_conv
0          convergence_flag
1          elite_ind_flag
30         dna_length
1          exp_type
```

Figure 25.188: An example *congo.in* file.

Table 25.172: Format of *congo.in*.

Parameter	Description	Recommended value
<i>restart_flag</i>	Restart the CONGO case from restart.in .	0 or 1
<i>run_individual</i>	Used only to study a GA (ignore).	
<i>run_generation</i>	Used only to study a GA (ignore).	
<i>delay_between_runs</i>	Seconds to pause between CONGO runs to allow the <i>random_seed_flag</i> value to change.	0
<i>pause_to_check_for_results</i>	Seconds for which CONGO pauses between checking for the results file in the run directories.	60 s for CONVERGE simulations, 4 s for mechanism tuning simulations
<i>gen_timeout_minutes</i>	Minutes after which CONGO will move to the next case. This value should be greater than the expected runtime for any case.	Examples: For cases with a 30-second runtime, set <i>gen_timeout_minute</i> to 5 minutes. For cases with a 16-hour runtime, set <i>gen_timeout_minute</i> to 20 hours.
<i>monitor_runs_lib_flag</i>	Provides the option to add a user-defined routine to check for crashed cases. Set to 0 to disable this feature. Contact the Applications team for assistance if you wish to enable this feature.	0

Chapter 25: Input and Data Files

CONGO GA/DoE Setup: congo.in

Parameter	Description	Recommended value
<i>new_dir_flag</i>	0 = Re-use run directories, 1 = Make new directories each generation.	0
<i>num_copy_templates</i>	Number of file copy templates on the following lines (CONGO invokes the <code>cp</code> command). CONGO uses the copy templates to determine which files and folders it will copy for each individual in a CONGO case. The two copy templates listed below should copy all of the files in the CONVERGE Case Directory. This Case Directory is designated in <i>case.in</i> > <i>dir_name</i> , usually named " <i>input_files</i> ". Change these templates only if you do not want to copy all of the files into each individual's folder.	2
<i>copy_template</i>	Template 1. (Repeat this row corresponding to the value you enter in <i>num_copy_templates</i> . Use two rows, as shown here, if you are copying all of the input files in a normal case directory.)	*
<i>copy_template</i>	Template 2.	*.*
<i>input_files</i>	Number of input files on following lines.	1
<i>file_name</i>	Enter <i>default</i> to automatically include all of the standard CONVERGE input file names. Otherwise, enter the name of the CONVERGE input file name to be included in the CONGO case. Repeat this row as needed to correspond with the number you specify in the <i>input_files</i> row above.	<i>default</i>
<i>max_gen</i>	Maximum generations created by the GA.	100
<i>random_seed_flag</i>	0 = Seed the random number generator with a fixed value, 1 = Use the system time as the seed for the random number.	0
<i>mutation_frac</i>	GA mutation fraction. This parameter provides a way to add more randomness to the optimization, but this option typically is not used for micro GA runs.	0
<i>p_conv</i>	GA convergence criteria fraction.	0.97
<i>convergence_flag</i>	0 = Convergence calculated relative to the merit score of the elite (fittest individual or best-so-far), 1 = Convergence calculated relative to the mean merit score.	0

Chapter 25: Input and Data Files

CONGO GA/DoE Setup: congo.in

Parameter	Description	Recommended value
<i>elite_ind_flag</i>	0 = The elite individual may be any individual, 1 = The elite individual is always individual 0. Assigning the elite individual to the 0 position will ensure that the elite individual is not modified during the current generation. It is also convenient to keep the elite individual in the 0 position when analyzing the results of the GA.	1
<i>dna_length</i>	Number of bits for parameter DNA.	30
<i>exp_type</i>	Experiment type. 0 = Design of experiments, 1 = Genetic algorithm, 2 = Design of experiments of the genetic algorithm.	1

Case: case.in

The *case.in* file, which is mandatory for all CONGO cases, defines the parameter ranges for the experiment model inputs and parsing for the CONVERGE input files. The name of this file is defined by the *file_name* parameter in [congo.in](#).

Chapter 25: Input and Data Files

CONGO Case: case.in

```
version: 3.1
---

1           num_modes
1           weight
input_files dir_name
9           pop_size
2           num_param
1           num_common_param
0           num_dependent_param
0           num_dependent_file_names
3           num_dyn_params
0           num_dyn_profiles
1           dyn_lib_flag
0           param_number
NozzleScaling name
real        type
1           common_param
none        marker
0.8         min
1.2         max
1           param_number
SOI         name
real        type
0           common_param
GA_INJECTION_START marker
-12.0       min
-8.0        max
0           dyn_param_num
nozzledia  name
GA_NOZZLEDIA marker
1           dyn_param_num
nozzleRadius name
GA_NOZZLERAD marker
2           dyn_param_num
duration   name
GA_DURATION marker
```

Figure 25.189: An example *case.in* file for a GA case.

Chapter 25: Input and Data Files

CONGO Case: case.in

```
version: 3.1
---

1           num_modes
input_files dir_name
4           num_runs
2           num_param
0           num_dependent_param
0           num_dependent_file_names
3           num_dyn_params
0           num_dyn_profiles
1           dyn_lib_flag
0           param_number
NozzleScaling@ name
real        type
none        marker
0.8         value
1           value
1.2         value
1           value
1           param_number
SOI@        name
real        type
DOE_INJECTION_START marker
-12        value
-12        value
-12        value
-9           value
0           dyn_param_num
nozzledia  name
GA_NOZZLEDIA marker
1           dyn_param_num
nozzleRadius name
GA_NOZZLERAD marker
2           dyn_param_num
duration   name
GA_DURATION marker
```

Figure 25.190: An example *case.in* file for a DoE case.

Table 25.173: Parameters in *case.in*.

Parameter	Description	Recommended value
<i>num_modes</i>	Number of modes (<i>idle</i> , <i>power</i> , <i>speed load</i> , etc.) in the simulation. A single mode simulation is the simplest and most common.	1
<i>weight</i>	Weighting factor for each mode (sum of weights must add to 1). Repeat this <i>weight</i> row as needed to correspond with <i>num_modes</i> value. The order in which you specify the weight of each mode must correspond to the order of the mode directories you specify in the <i>dir_name</i> rows below. Used only for Genetic	1.0

Chapter 25: Input and Data Files

CONGO Case: case.in

Parameter	Description	Recommended value
	Algorithm (GA) CONGO cases. Not used for Design of Experiments (DoE) cases.	
<i>dir_name</i>	Name of the directory for each mode. Include 1 <i>dir_name</i> row for each mode. Create one directory for each mode in the main directory in which all GA or DoE sub-directories will be stored. For a single-mode simulation, create just one directory called <i>input_files</i> and specify this name in one <i>dir_name</i> row.	<i>input_files</i>
<i>pop_size</i>	Population size. GA cases only.	
<i>num_runs</i>	Number of runs. Used only for Design of Experiments (DoE) CONGO cases. Not used for Genetic Algorithm (GA) cases.	
<i>num_param</i>	Number of defined parameters.	
<i>num_common_param</i>	Number of common parameters. Static parameters are always common and must be included. GA cases only.	
<i>num_dependent_param</i>	Number of parameters defined with a gain and offset from a defined or dynamic parameter. For example, you can automatically modify the embedding around the spark plug by making this parameter dependent on the spark timing.	
<i>num_dependent_file_names</i>	Number of dependent file names.	
<i>num_dyn_params</i>	Number of dynamic parameters that will be calculated by the user routine. For example, you can use a dynamic parameter to create	

Chapter 25: Input and Data Files

CONGO Case: case.in

Parameter	Description	Recommended value
	and modify the piston bowl shape.	
<i>num_dyn_profiles</i>	Number of dynamic profiles (e.g., an injection rate-shape) that will be modified by the user routine.	
<i>dyn_lib_flag</i>	Use a user dynamic library for dynamic parameters. For example, you can specify <i>lib.congo.so</i> , which is a UDF library in CONVERGE, to generate the values of the dynamic parameters.	
<i>param_number</i>	Zero-based index of parameters (<i>i.e.</i> , the first parameter name always will be 0).	
<i>name</i>	Parameter name. Enter any string. This string will appear in the <i>param.[run#]-[gen#]</i> output files generated by CONGO, so a descriptive string is helpful.	
<i>type</i>	Parameter type. Options are <i>real</i> , <i>integer</i> , <i>static</i> , <i>list</i> , and <i>real_variation</i> .	
<i>common_param</i>	Parameter type for multiple modes. 0 = Independent, 1 = Common. GA cases only.	
<i>marker</i>	The string of text that you designate as the value of any independent or common parameter (in any of the CONVERGE input files) to be optimized by the GA for the independent or common parameters. The name of the marker specified here must match the name you specify in the relevant input file. Refer to CONGO/CONVERGE	

Chapter 25: Input and Data Files

CONGO Case: case.in

Parameter	Description	Recommended value
Communication for more information.		
<i>value</i>	Enter one row and number for each value in the "design space" for the parameter designated by the <i>name</i> above. DoE cases only. Refer to "Example DoE Design Space" for more information.	
<i>min</i>	Minimum value in the range to be considered in the GA for the parameter designated by the <i>name</i> above. GA cases only.	
<i>max</i>	Maximum value in the range to be considered in the GA for the parameter designated by the <i>name</i> above. GA cases only.	
<i>dyn_param_num</i>	Zero-based index of dynamic parameter.	
<i>name</i>	Dynamic parameter name. Enter any string. This string will appear in the <i>dyn_param_[run#]-[gen#]</i> output files generated by CONGO, so a descriptive string is helpful.	
<i>marker</i>	The string of text that you designate as the value of any dynamic parameter (in any of the CONVERGE input files) to be optimized by the GA for the dynamic parameters. The name of the marker specified here must match the name you specify in the relevant input file. Refer to CONGO/CONVERGE Communication for more information.	
<i>dependent_param_num</i>	Zero-based index of dependent parameters.	

Chapter 25: Input and Data Files

CONGO Case: case.in

Parameter	Description	Recommended value
Repeat each of these rows for each dependent parameter.	<i>name</i> Dependent parameter name. Enter any string. This string will appear in the <i>param</i> . [run#]-[gen#] output files generated by CONGO, so a descriptive string is helpful.	
	<i>dependency_name</i> Name of the base parameter (either defined or dynamic) on which dependent parameter is based. Dependent Parameter Value = $(\text{Gain} * \text{Value of Base Parameter}) + \text{Offset}$	
<i>marker</i>	The string of text that you designate as the value of any dependent parameter (in any of the CONVERGE input files) to be optimized by the GA for the dependent parameters. The name of the marker specified here must match the name you specify in the relevant input file. Refer to CONGO/CONVERGE Communication for more information.	
<i>gain</i>	Gain value. See <i>dependency_name</i> entry above.	1
<i>offset</i>	Offset value. See <i>dependency_name</i> entry above.	

Execution: execute.in

The *execute.in* file, which is mandatory for all CONGO cases, defines how CONVERGE will be executed for a GA or DoE simulation.

Chapter 25: Input and Data Files

CONGO Execution: execute.in

```
0      test_run_flag
0      recollect_data_flag
1      script_ssh_flag
duo1   script_machine
1      start_script
example script_name
0      run_background_flag
0      before_script
0      after_script
0      end_script
8      num_concurrent_cases
0      ssh_flag
mpd.hosts machine_names
1      machines_per_run
0      create_batch_file_flag
example batch_file_name
1      run_batch_file_flag
exe    batch_command
1      num_execute_commands
$path$/stalagmite </dev/null > logfile-$rundir$-$generation$ &
```

Figure 25.191: An example *execute.in* file.

Table 25.174: Format of *execute.in*.

Parameter	Description	Recommended value
<i>test_run_flag</i>	0 = Start CONVERGE simulation after preparing the input files, 1 = Prepare input files only (does not start CONVERGE).	0 or 1
<i>recollect_data_flag</i>	0 = Do not recollect data, 1 = Recollect data in the event of a crash,	0 or 1
<i>script_ssh_flag</i>	0 = Do not run scripts on a remote machine, 1 = Run scripts on a remote machine.	0 or 1
<i>script_machine</i>	Specify the machine name on which to run scripts.	<machine name>
<i>start_script</i>	Number of scripts to run at the start of CONGO run. For each start script, the script name and <i>run_background_flag</i> must follow.	<number of scripts>
<i>script_name</i>	Name of script to run at the start of CONGO run.	<name of script>
<i>run_background_flag</i>	0 = Do not run in the background, 1 = Run in the background.	0 or 1
<i>before_script</i>	Number of scripts to run before CONGO runs. For each before script, the script name and <i>run_background_flag</i> must follow.	<number of scripts>
<i>script_name</i>	Name of script to run before CONGO runs.	<name of script>
<i>run_background_flag</i>	0 = Do not run in the background, 1 = Run in the background.	0 or 1
<i>after_script</i>	Number of scripts to run after CONGO runs. For each after script, the script name and	<number of scripts>

Chapter 25: Input and Data Files

CONGO Execution: execute.in

Parameter	Description	Recommended value
	<i>run_background_flag</i> must follow.	
<i>script_name</i>	Name of script to run after CONGO runs.	<name of script>
<i>run_background_flag</i>	0 = Do not run in the background, 1 = Run in the background.	0 or 1
<i>end_script</i>	Number of scripts to run at the end of CONGO run. For each end script, the script name and <i>run_background_flag</i> must follow.	<number of scripts>
<i>script_name</i>	Name of script to run at the end of the CONGO run.	<name of script>
<i>run_background_flag</i>	0 = Do not run in the background, 1 = Run in the background.	0 or 1
<i>num_concurrent_cases</i>	Number of CONGO cases to run concurrently. Typically equal to (case.in > pop_size - 1) for a GA.	<number of cases>
<i>ssh_flag</i>	0 = Do not run CONGO cases on a remote machine. 1 = Run CONGO cases on a remote machine,	0 or 1
<i>machine_names</i>	File that lists all of the machine names on which to run CONGO.	mpd.hosts
<i>machines_per_run</i>	Number of machines used per run folder for parallel execution (mpd.host files created in each run directory).	
<i>create_batch_file_flag</i>	0 = Do not create a batch file with execute commands 1 = Create a batch file with execute commands,	0 or 1
<i>batch_file_name</i>	Batch file name.	
<i>run_batch_file_flag</i>	0 = Do not run batch file, 1 = Run batch file.	0 or 1
<i>batch_command</i>	Batch command.	
<i>num_execute_commands</i>	Number of the following rows containing execution commands.	
execution commands	Execution commands (one per line).	

Merit: merit.in

The *merit.in* file, which is mandatory for all CONGO cases, defines the merit function in terms of CONVERGE performance parameters. An explanation of the default merit function is given in the [Merit Function](#) section of [Chapter 6 - Optimization Techniques](#).

Chapter 25: Input and Data Files

CONGO Merit: merit.in

```
version: 3.1
---

ga_output          output_file_name
4                  num_output_vars
GISFC             output_var_name
NOX               output_var_name
PM                output_var_name
PCP               output_var_name
0                  merit_lib_flag
1                  num_performance_vars
GISFC             performance_var_name
minimize           type
120               value
1                  weight
3                  num_constraint_vars
NOX               constraint_var_name
maximum           type
0                  weighting_greater
0                  weighting_less
1                  power_factor
5                  value
PM                constraint_var_name
maximum           type
1                  weighting_greater
0                  weighting_less
1                  power_factor
1                  value
PCP               constraint_var_name
maximum           type
5                  weighting_greater
0                  weighting_less
1                  power_factor
15                value
```

Figure 25.192: An example *merit.in* file.

Table 25.175: Format of *merit.in*.

Parameter	Description	Recommended value
<i>output_file_name</i>	The CONVERGE output file name containing the performance variables. In CONVERGE, you can specify this file name by customizing the <i>user_ga_merit_flag.c</i> user-defined function file. This parameter will tell CONGO what files to search for to find the GA performance variables.	<i>ga_output</i>
<i>num_output_vars</i>	The number of GA or DoE variables CONVERGE writes to the output file (<i>e.g.</i> , <i>ga_output.#-#</i>). This value must match the number of rows of <i>output_var_name</i> immediately below this row.	
<i>output_var_name</i>	Name of a GA or DoE variable that CONVERGE writes to the output file (<i>e.g.</i> , <i>ga_output.#-#</i>). This name must match the name in the CONVERGE output file.	

Chapter 25: Input and Data Files

CONGO Merit: merit.in

Parameter	Description	Recommended value
	Include a separate row for each output variable. Many of these variables you list here will also be performance or constraint parameters, but not necessarily all of them.	
<i>merit_lib_flag</i>	0 = CONGO will use the default merit function as defined by the parameters below, 1 = Use a user defined library for merit calculation. (Variables below this line are then ignored.)	
<i>num_performance_vars</i> (Used only when <i>merit_lib_flag</i> = 0.)	The number of performance variables used in merit calculation. These variables are used to add to the GA or DoE merit score. In the rows below, include rows for the <i>performance_var_name</i> , <i>type</i> , <i>value</i> , and <i>weight</i> of each performance variable.	
<i>performance_var_name</i>	The name of a GA or DoE variable that CONGO will use to add to the merit score. This name must match the name in CONVERGE output file.	
Repeat these <i>type</i> rows for each performance variable. (Used only when <i>merit_lib_flag</i> = 0.)	<p>The type of performance variable for the <i>performance_var_name</i> listed above. Choose from <i>maximize</i>, <i>minimize</i>, <i>error</i>, or <i>normalized_error</i>.</p> <p>For a <i>maximize</i> type of performance variable, CONGO will divide the value of the variable from the CONVERGE run by the target value (listed as <i>value</i> in the row below). CONGO will use this ratio in the merit calculation.</p> <p>Conversely, for a <i>minimize</i> type of performance variable, the target value will be divided by the variable value in the merit function.</p> <p>An <i>error</i> type of performance variable uses the absolute difference between the value of the variable and the target value in the merit function.</p> <p>A <i>normalized_error</i> type of performance variable uses the normalized absolute difference between the value of the variable and the target value in the merit function.</p> <p>An explanation of the default merit function is given in the Merit Function section of Chapter 6 - Optimization Techniques.</p>	
<i>value</i>	The target value for the performance parameter listed above. For multi-mode CONGO simulations, include a <i>value</i> row for each mode being evaluated.	
<i>weight</i>	The weight assigned to the performance parameter listed above. Use the <i>weight</i> to assign relative importance to each performance parameter.	

Chapter 25: Input and Data Files

CONGO Merit: merit.in

Parameter	Description	Recommended value
<i>num_constraint_vars</i> (Used only when <i>merit_lib_flag</i> = 0.)	The number of constraint variables used in merit calculation (default formula). These variables are used to subtract from the GA or DoE merit score. In the rows below, include rows for the <i>constraint_var_name</i> , <i>type</i> , <i>weighting_greater</i> , <i>weighting_less</i> , <i>power_factor</i> , and <i>value</i> of each constraint variable.	
<i>constraint_var_name</i> (Used only when <i>merit_lib_flag</i> = 0.)	The name of a GA or DoE variable that CONGO will use to subtract from the merit score. This name must match the name in CONVERGE output file.	
Repeat these rows for each constraint variable. (Used only when <i>merit_lib_flag</i> = 0.)	<i>type</i> The type of constraint variable for the <i>constraint_var_name</i> listed above. Choose from <i>maximum</i> or <i>minimum</i> . For a <i>maximum</i> type of performance variable, CONGO will subtract from the merit calculation if the calculated value is above the <i>value</i> specified below. Conversely, for a <i>minimum</i> type of performance variable, CONGO will subtract from the merit calculation if the calculated value is below the <i>value</i> specified below. An explanation of the default merit function is given in the Merit Function section of Chapter 6 - Optimization Techniques .	
<i>weighting_greater</i>	If the calculated value of a <i>maximum</i> constraint variable is greater than the target value (specified in the <i>value</i> row below), CONGO will use the value you specify here for the weight of this <i>maximum</i> constraint variable contribution to merit.	
<i>weighting_less</i>	If the calculated value of a <i>minimum</i> constraint variable is less than the target value (specified in the <i>value</i> row below), CONGO will use the value you specify here for the weight of this <i>minimum</i> constraint variable contribution to merit.	
<i>power_factor</i>	The power to which the constraint variable performance is taken in the default merit function. Use this option to make one constraint variable exponentially more important than others.	
<i>value</i>	The target value for the constraint parameter listed above. For multi-mode CONGO simulations, include a <i>value</i> row for each mode being evaluated.	

UDI: udi.in

The *udi.in* file, which is optional for CONGO cases, allows you to enter a user-defined individual (UDI) into the genetic algorithm experiment at any generation. You could use this UDI as a baseline case, for example. You can load more than one individual from a single file if desired.

```
0      loadindividual
8      number
0.170000  x
1.170000  y
0.170000  x
2.170000  y
0.170000  x
3.170000  y
1.170000  x
4.170000  y
0.170000  x
5.170000  y
2.170000  x
6.170000  y
2.170000  x
7.170000  y
3.170000  x
8.170000  y
```

Figure 25.193: An example *udi.in* file that specifies eight unique individuals, with two parameters being varied: one parameter named *x* and one parameter named *y*.

Table 25.176: Format of *udi.in*.

Parameter	Description
<i>loadindividual</i>	0 = Do not load the user-defined individual, 1 = Load individual in next generation (automatically changed to 0 once loaded).
<i>number</i>	Number of individuals to define (listed in subsequent rows).
< <i>parameter name</i> >	The first of these rows must be the name (not the <i>marker</i>) of the first parameter (as it appears in the <i>case.in</i> file) of the first individual. In the previous example file, <i>x</i> is the name of first parameter; <i>y</i> is the second parameter. Both of these parameters are repeated according to the number of individuals specified in the <i>number</i> row. In the example below, 2 parameters are specified for 8 individuals, resulting in 16 < <i>parameter name</i> > rows.

GA to DOE: *gatodoe.in*

The *gatodoe.in* file, which is optional for CONGO cases, allows you to automatically set up a set of cases—on which you can then run a design of experiments—from the genetic algorithm experiment. Typically, cases with high merit in the GA may be re-run to obtain detailed output files or re-run for validation with higher accuracy (higher grid resolution).

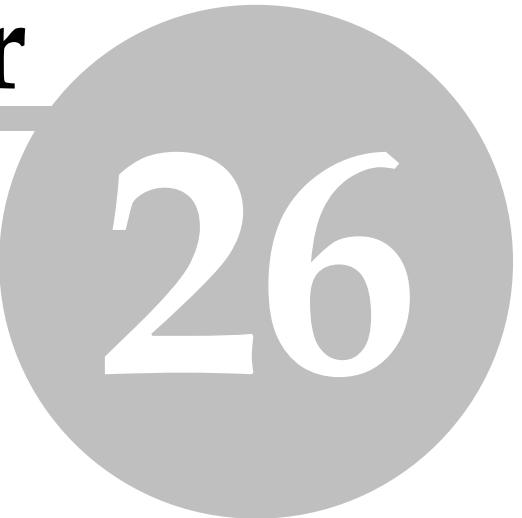
Chapter 25: Input and Data Files

CONGO GA to DOE: gatodoe.in

Table 25.177: Format of *gatodoe.in*.

Parameter	Description	Recommended value
<i>gainputfile</i>	Name of the GA input file to read.	
<i>filetorecreate</i>	Name of the DoE input file to create.	
<i>number of cases</i>	Number of parameter files assembled in current directory to read for case definition.	
<i>filename</i>	Parameter file name from GA run directories.	param.#-#

Chapter



26

Output Files

26 Output Files

This chapter details the output files that are generated when running a 3D CONVERGE simulation or a [utility](#).

26.1 3D Simulation Output Files

During a simulation, CONVERGE generates ASCII-formatted output files that contain average values (or sums) for the entire domain as well as for individual regions. Output file names with data for the entire domain are formatted as *<file name>.out* (e.g., *thermo.out*). Region-specific output file names are formatted as *<file name>_region<region ID>.out* (e.g., *thermo_region0.out*).

If your simulation is [from a restart](#), CONVERGE includes the restart number (*i.e.*, the value of *inputs.in > simulation_control > restart_number*) in the output file names. File names for the entire domain are formatted as *<file name><restart number>.out*. Region-specific output file names are formatted as *<file name><restart number>_region<region ID>.out*.

acoustic_receiver_<ID>_stream_<ID>_pressure.out

CONVERGE generates an *acoustic_receiver_<ID>_stream_<ID>.out* for each FW-H receiver specified in [acoustics.in](#).

Table 26.1: Description of *acoustic_receiver_<ID>_stream_<ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in > simulation_control > crank_flag = 0</i> or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Pressure (Pa)	Pressure at this receiver point.

amr.out

CONVERGE generates *amr.out* when performing [Adaptive Mesh Refinement](#) (AMR) (*i.e.*, when *inputs.in > grid_control > amr_flag = 1*). This file lists sub-grid scale (SGS) or embedding criterion values that CONVERGE calculates when performing (AMR). These values are written to *amr.out* in the order in which they are encountered in [amr.in](#). When a given AMR definition is inactive, CONVERGE writes -1 in place of the corresponding criterion.

Chapter 26: Output Files

3D Simulation Output Files amr.out

Table 26.2: Description of *amr.out*.

Column	Header (<i>units</i>)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
Varies	<AMR type> <variable> <region ID>	<AMR type> = subgrid_scale (for amr_type = SGS), embed_criterion (for amr_type = EMBED ABOVE or EMBED BELOW), or lower_embed_criterion (for amr_type = EMBED BETWEEN, requires upper_embed_criterion in the next column), <variable> = Target variable for AMR, <region ID> = Region in which AMR is performed. This column is repeated for every variable in <i>amr.in</i> .
Varies	upper_embed_criterion <variable> <region ID>	This column is included only when <AMR type> = lower_embed_criterion in the previous column. <variable> = Target variable for AMR, <region ID> = Region in which AMR is performed.

area_avg_flow.out

CONVERGE generates *area_avg_flow.out* for 3D simulations that include [INFLOW](#), [OUTFLOW](#), or [PERIODIC](#) boundaries. This file tabulates mass flow and area-weighted average thermodynamic data at the relevant boundaries.

If there is more than one INFLOW or OUTFLOW boundary, CONVERGE tabulates mass flow and thermodynamic data for the other INFLOW/OUTFLOW boundaries in additional columns. The *area_avg_flow.out* file includes a header that indicates the region and variable contained in each column.

Chapter 26: Output Files

3D Simulation Output Files area_avg_flow.out

Table 26.3: Description of *area_avg_flow.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Cycle_Number or Pseudo_Time	Simulation cycle number (i.e., number of time-steps) if inputs.in > solver_control > steady_solver = 0. The pseudo time-step if steady_solver = 1.
3*	Mass_Flow_Rate (kg/s)	Mass flow rate normal to the INFLOW/OUTFLOW boundary plane. A positive value indicates flow out of the domain while a negative value indicates flow into the domain.
4*	Total_Pres (Pa)	Area-weighted average total pressure.
5*	Static_Pres (Pa)	Area-weighted average static pressure.
6*	Avg_Temp (K)	Area-weighted average temperature.
7*	Avg_Velocity (m/s)	Area-weighted average velocity normal to the INFLOW/OUTFLOW boundary plane. A positive value indicates flow out of the domain while a negative value indicates flow into the domain.
8*	Avg_Density (kg/m ³)	Area-weighted average density.
9*	Avg_Mach (dimensionless)	Area-weighted average Mach number.
10*	Avg_totTemp (K)	Area-weighted average total temperature.
11*	Avg_totEnth (J/kg)	Area-weighted average total enthalpy, calculated from the local total enthalpy given by $H = \int_{T_o}^T c_p dT + \sum_{k=1}^N \Delta H_{f,k}^o Y_k + (1/2) u_i u_i,$ where T_o is the temperature at standard state conditions, T is the local temperature, c_p is the mixture-averaged specific heat capacity at constant pressure, $\Delta H_{f,k}^o$ is the standard enthalpy of formation of the k -th species, Y_k is the mass fraction of species k , N is the total number of species, and u_i is the velocity.

* These columns are repeated for each INFLOW, OUTFLOW, and PERIODIC boundary. If there are two matching PERIODIC boundaries, these columns are written for only one of the matching boundaries.

area_avg_regions_flow.out

CONVERGE generates *area_avg_regions_flow.out* when inter-region flow output is active (i.e., when [inputs.in](#) > output_control > region_flow_flag is non-zero). This file provides mass flow rates and area-weighted averages for the interfaces between adjacent regions, [flow-through INTERFACE](#) boundaries, and/or [INFLOW and OUTFLOW boundaries](#). For [inputs.in](#) > output_control > region_flow_flag = 1, CONVERGE does not write species-specific data. For

Chapter 26: Output Files

3D Simulation Output Files area_avg_regions_flow.out

[inputs.in](#) > output_control > region_flow_flag = 2, CONVERGE writes species-specific data as specified in [regions_flow.in](#).

Table 26.4: Description of area_avg_regions_flow.out.

Column	Header (units)	When CONVERGE includes this column	Description
1	Time (seconds) or Crank (crank angle degrees)	Always	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Cycle_Number or Pseudo_Time	Always	Simulation cycle number (i.e., number of time-steps) if inputs.in > solver_control > steady_solver = 0. The pseudo time-step if steady_solver = 1.
3	Flow_Rate (kg/s) Regions_<i>_to_<j>	Always	Mass flow rate from region <i>i</i> to region <i>j</i> .
4	Tot_Mass (kg) Regions_<i>_to_<j>	Always	Integrated mass flow rate from region <i>i</i> to region <i>j</i> .
5	Velocity (m/s) Regions_<i>_to_<j>	Always	Fluid velocity between regions <i>i</i> and <i>j</i> .
6	Tot_Pres (Pa) Regions_<i>_to_<j>	Always	Area-weighted average total pressure at the interface of regions <i>i</i> and <i>j</i> .
7	Static_Pres (Pa) Regions_<i>_to_<j>	Always	Area-weighted average static pressure at the interface of regions <i>i</i> and <i>j</i> .
8	Avg_Temp (K) Regions_<i>_to_<j>	Always	Area-weighted average temperature at the interface of the regions <i>i</i> and <i>j</i> .
9	Avg_tot_Temp (K) Regions_<i>_to_<j>	Always	Area-weighted average total temperature at the interface of regions <i>i</i> and <i>j</i> .
10	Avg_tot_Enth (J/kg) Regions_<i>_to_<j>	Always	Area-weighted average total enthalpy at the interface of regions <i>i</i> and <i>j</i> , calculated from the local total enthalpy given by $H = \int_{T_o}^T c_p dT + \sum_{k=1}^N \Delta H_{f,k}^o Y_k + (1/2) u_i u_i,$ where T_o is the temperature at standard state conditions, T is the local temperature, c_p is the mixture-averaged specific heat $\Delta H_{f,k}^o$ capacity at constant pressure, $\Delta H_{f,k}^o$ is the standard enthalpy of formation of the <i>k</i> -th species, Y_k is the mass fraction of species <i>k</i> , N is the total number of species, and u_i is the velocity.

Chapter 26: Output Files

3D Simulation Output Files area_avg_regions_flow.out

Column	Header (units)	When CONVERGE includes this column	Description
11	Rate_<species name> (kg/s) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Mass flow rate for the specified species. Columns 14 and 15 are repeated for each species specified in regions_flow.in . Columns 14 and 15 are also written for each INFLOW or OUTFLOW boundary.
12	Tot_<species name> (kg) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Total species mass passing through the interface of regions i and j .
13	Xfrac-<species name> Bound_id_<boundary ID>	When inputs.in > output_control > region_flow_flag = 2	Mole fraction for the specified gas species. This information is written only for each INFLOW and/or OUTFLOW boundary.
14	Rate_<passive name> (kg/s) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Mass flow rate for the specified passive from region i to region j . Columns 17 and 18 are repeated for each passive specified in regions_flow.in . Columns 17 and 18 are also written for each INFLOW or OUTFLOW boundary.
15	Tot_<passive name> (kg) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Total passive mass passing through the interface of regions i and j .
16	Rate_<species name> (kg/s)	When inputs.in > output_control > region_flow_flag = 2	Mass flow rate for the parcel species from region i to region j . Columns 19 and 20 are repeated for each species specified in regions_flow.in . Columns 19 and 20 are also written for each INFLOW or OUTFLOW boundary.
17	Tot_<species name> (kg)	When inputs.in > output_control > region_flow_flag = 2	Total parcel species mass passing through the interface of regions i and j .
18	TKE (m^2/s^2)	When regions_flow.in > output_tke_flag = ON	Area-weighted average turbulent kinetic energy at the interface of regions i and j . This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
19	EPS (m^2/s^3)	When regions_flow.in > output_eps_flag = ON	Area-weighted average turbulent dissipation at the interface of regions i and j .

Chapter 26: Output Files

3D Simulation Output Files area_avg_regions_flow.out

Column	Header (<i>units</i>)	When CONVERGE includes this column	Description
20	OMEGA (s^{-1})	When <i>regions_flow.in</i> > <i>output_omega_flag</i> = ON	Area-weighted average specific dissipation rate at the interface of regions <i>i</i> and <i>j</i> .
21	Ca	When <i>regions_flow.in</i> > <i>output_ca_flag</i> = ON	Area-weighted average liquid area fraction at the interface of regions <i>i</i> and <i>j</i> . Set <i>regions_flow.in</i> > <i>output_ca_flag</i> = ON only when volume of fluid modeling is active.

az_info<dump number>_<dump time>.out

CONVERGE generates *az_info<dump number>_<dump time>.out* when [adaptive zoning](#) output is active (i.e., when *combust.in* > *adaptive_zone_model* > *active* = 1 and *adaptive_zone_model* > *output_flag* = 1). This file provides adaptive zoning information.

Because the dump number appears first in the file name, an alphabetical listing of the directory contents is chronologically consistent. These files are stored in the same directory as the corresponding *post*.h5* files. The *post*.h5* file output location is determined by [stream_setting.in](#) > *post_file_option*.

Chapter 26: Output Files

3D Simulation Output Files az_info<dump number>_<dump time>.out

Table 26.5: Description of *az_info<dump number>_<dump time>.out* for a single-stream simulation.

Column	Header (units)	Description
1	zone_id	Zone identification number.
2	num_cells	Number of cells in the corresponding zone.
3	Var1_bin_size	Interval length in the first variable (e.g., temperature) of the corresponding zone.
4	Var1_bin_avg	Mid-value in the first variable (e.g., temperature) of the bin interval.
5	Var2_bin_size	Interval length in the second variable (e.g., equivalence ratio) of the corresponding zone.
6	Var2_bin_avg	Mid-value in the second variable (e.g., equivalence ratio) of the bin interval.
7	Volume_fraction	Volume fraction of this zone relative to the domain.
8	Mass_fraction	Mass fraction of this zone relative to the domain.
Varies	<species>_massfrac_avg	Mean mass fraction of this species in this zone. This set of columns is repeated for each species in the simulation.
Varies	<species>_massfrac_min	Minimum mass fraction of this species in this zone.
Varies	<species>_massfrac_max	Maximum mass fraction of this species in this zone.

Chapter 26: Output Files

3D Simulation Output Files az_info<dump number>_<dump time>.out

Table 26.5: Description of *az_info<dump number>_<dump time>.out* for a multi-stream simulation.

Column	Header (units)	Description
1	stream_id	Stream identification number.
2	zone_id	Zone identification number.
3	num_cells	Number of cells in the corresponding zone.
4	Var1_bin_size	Interval length in the first variable (e.g., temperature) of the corresponding zone.
5	Var1_bin_avg	Mid-value in the first variable (e.g., temperature) of the bin interval.
6	Var2_bin_size	Interval length in the second variable (e.g., equivalence ratio) of the corresponding zone.
7	Var2_bin_avg	Mid-value in the second variable (e.g., equivalence ratio) of the bin interval.
8	Volume_fraction	Volume fraction of this zone relative to the domain.
9	Mass_fraction	Mass fraction of this zone relative to the domain.
Varies	<species>_massfrac_avg	Mean mass fraction of this species in this zone. This set of columns is repeated for each species in the simulation.
Varies	<species>_massfrac_avg	Minimum mass fraction of this species in this zone.
Varies	<species>_massfrac_avg	Maximum mass fraction of this species in this zone.

battery_num_<number>_stream_<stream ID>.out

CONVERGE generates *battery_num_<number>_stream_<stream ID>.out* when there is a [battery heat source](#) (i.e., when *source.in* > *source* > *equation* = BATTERY). This file lists the state of charge (SOC), voltage, electrical current and temperature of the battery over time.

Table 26.6: Description of *battery_num_<number>_stream_<stream ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (DEG)	Time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
2	SOC (none)	State of charge of the battery.
3	Vt (volts)	Voltage of the battery.
4	current (Amps)	Electrical current of the battery (positive value indicates discharging and negative current indicates charging).
5	Tmean (K)	Mean temperature of the battery (Kelvin).

Chapter 26: Output Files

3D Simulation Output Files beam_<name>_monitor_point_<ID>.out

beam_<name>_monitor_point_<ID>.out

CONVERGE generates *beam_<name>_monitor_point_<ID>.out* when the [FSI beam](#) model is active (*i.e.*, when *beam_objects* are specified in [fsi.in](#)). The file contains the predicted pressures in the various crevice volumes in addition to the total mass in the crevice.

Table 26.7: Description of *beam_<name>_monitor_point_<ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	displacement (m)	Displacement of this beam monitor point.
3	velocity (m/s)	Velocity of this beam monitor point.
4	acceleration (m/s ²)	Acceleration of this beam monitor point.
5 [‡]	load (N/m)	Beam load at this beam monitor point.
Varies [‡]	bending_stress (Pa)	Bending stress at this beam monitor point.
Final [‡]	shear_stress (Pa)	Shear stress at this beam monitor point.

borghi.out

CONVERGE generates *borghi.out* when the Borghi diagram data output option is active (*i.e.*, when [combust.in](#) > borghi_diagram_output_flag = 1). The *borghi.out* file contains data to generate a Borghi diagram.

Table 26.8: Description of *borghi.out*.

Column	Header (units)	Description
1	Time (seconds), Crank (crank angle degrees), or Cycles	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero, or the simulation cycle number (<i>i.e.</i> , number of time-steps).
2	Mass (kg)	Summation of all cell masses.
3	Pressure (MPa)	Volume-averaged pressure.
4	Density (kg/m ³)	Total mass divided by total volume.
5	Cp (J/kg-K)	Specific heat at constant pressure.
6	Conductivity (J/m-K)	Mass-averaged conductivity.
7	Turb_Length (m)	Mass-averaged turbulent length scale.
8	U_Prime (m/s)	Mass-averaged turbulent velocity.
9	SI (m/s)	Mass-averaged laminar flamespeed.
10	U_Prime/Sl	Value for the y axis of the Borghi diagram.
11	deltaL (m)	Estimated average flame thickness.
12	Turb_Length/deltaL	Values for the x axis of the Borghi diagram.

Chapter 26: Output Files

3D Simulation Output Files bound<boundary ID>-wall.out

bound<boundary ID>-wall.out

CONVERGE generates *bound<boundary ID>-wall.out* when the wall output option is active (*i.e.*, when [inputs.in](#) > *output_control* > *wall_output_flag* = 1). This file contains average near-wall quantities and heat transfer data for [WALL boundaries](#), which are defined with a boundary ID in [boundary.in](#).

Table 26.9: Description of *bound<boundary ID>-wall.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Avg_Vel_Mag (m/s)	The average velocity magnitude in the cells adjacent to this boundary.
3	TKE (m^2/s^2)	The average turbulent kinetic energy in the cells adjacent to this boundary. This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
4	EPS (m^2/s^3)	The average turbulent dissipation in the cells adjacent to this boundary.
5	OMEGA (s^{-1})	The average specific dissipation rate in the cells adjacent to this boundary.
6	Turb_Visc (m^2/s)	The average turbulent viscosity in the cells adjacent to this boundary.
7	Near_Wall_Avg_T (K)	The average temperature in the cells adjacent to this boundary.
8	Near_Wall_Max_T (K)	The maximum temperature in the cells adjacent to this boundary.
9	Near_Wall_Min_T (K)	The minimum temperature in the cells adjacent to this boundary.
10	Wall_Avg_T (K)	The average temperature of the WALL boundary.
11	Wall_Max_T (K)	The maximum temperature of the WALL boundary.
12	Wall_Min_T (K)	The minimum temperature of the WALL boundary.
13	Avg_Pres (N/m ²)	The average pressure in the cells adjacent to this boundary.

Chapter 26: Output Files

3D Simulation Output Files bound<boundary ID>-wall.out

Column	Header (units)	Description
14	AvgHTC (W/m ² -K)	The average heat transfer coefficient in the cells adjacent to this boundary. AvgHTC is 0 for solid cells, while for cells subject to the convection boundary condition, it is the value defined in boundary.in . For all other cells, AvgHTC is calculated as
		$\text{avg_HTC} = \frac{1}{A} \left(\frac{\sum_+ (q)}{\sum_+ (\Delta T \frac{dA_+}{A_+})} + \frac{\sum_- (q)}{\sum_- (\Delta T \frac{dA_-}{A_-})} \right).$
15 [‡]	HT_xfer_Rate (J/s)	The integrated heat transfer rate in the cells adjacent to this boundary. A positive value indicates energy flow out of the domain into the boundary, while a negative value indicates energy flow into the domain from the boundary.
16 [‡]	Wall_Area (m ²)	The total area for the WALL boundary.
17 [‡]	TotHT_xfer (J)	The total heat transfer to the boundary. The total heat transfer is the time-integrated heat transfer rate (column 15).
18 [‡]	Pres_Force_X (N)	Total pressure force (x component) exerted on the WALL boundary.
19 [‡]	Pres_Force_Y (N)	Total pressure force (y component) exerted on the WALL boundary.
20 [‡]	Pres_Force_Z (N)	Total pressure force (z component) exerted on the WALL boundary.
21 [‡]	Visc_Force_X (N)	Total shear force (x component) exerted on the WALL boundary.
22 [‡]	Visc_Force_Y (N)	Total shear force (y component) exerted on the WALL boundary.
23 [‡]	Visc_Force_Z (N)	Total shear force (z component) exerted on the WALL boundary.
24 ^{‡†}	Pres_Torque_X (N-m)	Total pressure torque about x axis exerted on the WALL boundary.
25 ^{‡†}	Pres_Torque_Y (N-m)	Total pressure torque about y axis exerted on the WALL boundary.
26 ^{‡†}	Pres_Torque_Z (N-m)	Total pressure torque about z axis exerted on the WALL boundary.
27 ^{‡†}	Visc_Torque_X (N-m)	Total viscous torque about x axis exerted on the WALL boundary.
28 ^{‡†}	Visc_Torque_Y (N-m)	Total viscous torque about y axis exerted on the WALL boundary.

Chapter 26: Output Files

3D Simulation Output Files bound<boundary ID>-wall.out

Column	Header (units)	Description
29 [†]	Visc_Torque_Z (N-m)	Total viscous torque about z axis exerted on the WALL boundary.
30	Yplus_avg	The average y^+ value on the WALL boundary.
31	Yplus_max	The maximum y^+ value on the WALL boundary.
32	Yplus_min	The minimum y^+ value on the WALL boundary.
33	Voltage (Volt)	Area-averaged voltage on the WALL boundary.
34	Electric_current (Ampere)	Integrated current through the WALL boundary.
35	Pres_Rot_Power (W)	Rotational power on this WALL boundary from pressure forces.
36	Vist_Rot_Power (W)	Rotational power on this WALL boundary from viscous forces.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in > output_control > output_multiplier](#).

[†] To calculate torque, CONVERGE calculates the cross product of the position vector from a specified origin to the cell face (\mathbf{r}) and the force on a cell face composing the boundary (\mathbf{F}). Then CONVERGE integrates this product over the entire boundary as follows:

$$\tau = \int_{\text{boundary}} \mathbf{r} \times \mathbf{F}. \quad (26.1)$$

You can use the optional [boundary.in > boundary_conditions > boundary > torque > center](#) parameter to specify the origin for the torque calculation. If you do not specify this parameter, CONVERGE uses the rotation origin ([boundary.in > boundary_conditions > boundary > velocity > origin](#)) for rotating and rotating-translating boundaries; the origin specified in the arbitrary motion file for arbitrary-motion boundaries; or the origin of the coordinate system (0, 0, 0) for other types of WALL boundaries.

cell_count_ranks.out

CONVERGE generates *cell_count_ranks.out* for all 3D simulations. The *cell_count_ranks.out* file written to the *outputs_original* or *outputs_restart** directory lists the total number of cells in the domain. CONVERGE also writes a separate *cell_count_ranks.out* file to each stream-specific output directory (e.g., *stream0*) listing the number of cells in that stream.

The cells-per-processor data should be used to check [load balancing](#). If the load balancing is poor, consider reducing the values of [inputs.in > simulation_control > load_cyc](#) and [simulation_control > imbalance_factor](#).

Table 26.10: Description of *cell_count_ranks.out*.

Column	Header (units)	Description
1 [†]	Cycle_Number	Cycle number (ncyc).

Chapter 26: Output Files

3D Simulation Output Files cell_count_ranks.out

Column	Header (units)	Description
1 [‡]	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Total_Cells	The total number of cells in the domain [†] or in the stream [‡] .
Varies	Rank< <i>i</i> >	The total number of cells on rank (processor) <i>i</i> . One column will be added per rank.

[†] For the file in the *outputs_original* or *outputs_restart** directory.

[‡] For the files in the stream-specific output directories.

cell_count_regions.out

CONVERGE generates *cell_count_regions.out* for each stream in a 3D simulation. This file contains information on the number of cells in each [region](#) in the stream. The cell count data is written out as a function of time.

Table 26.11: Description of *cell_count_regions.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Total_Cells	The total number of cells in the entire domain.
Varies	Region< <i>i</i> >	The total number of cells in region_id = <i>i</i> . One column will be added per region.

cht1d_point<number>_bound<boundary ID>.out

CONVERGE generates *cht1d_point<number>_bound<boundary ID>.out* when both of the following are true:

- You have enabled the [1D CHT model](#) for a WALL boundary.
- You have defined a [monitor point](#) that is associated with the above WALL boundary (that is, the WALL boundary is specified in [monitor_points.in](#) > monitor_points > point > boundary_id).

The *cht1d_point<number>_bound<boundary ID>.out* file contains heat transfer data for the near-wall fluid cell that is closest to the monitor point. This output includes the temperature at the solid wall and at various depths within the solid. If multiple monitor points are associated with the WALL boundary, CONVERGE generates a separate file for each monitor point.

We recommend using monitor points that are located on the WALL boundary, not within the fluid, as shown in the figure below. The output is the same regardless of which variables are monitored.

Chapter 26: Output Files

3D Simulation Output Files cht1d_point<number>_bound<boundary ID>.out

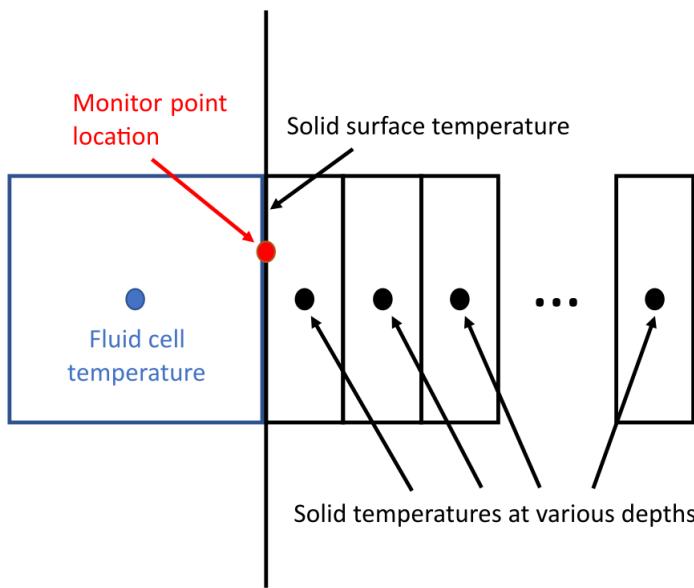


Figure 26.1: Temperatures reported in `cht1d_point<number>_bound<boundary ID>.out`.

Table 26.12: Description of `cht1d_point<number>_bound<boundary ID>.out`.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <code>inputs.in > simulation_control > crank_flag = 0</code> or in crank angle degrees if <code>crank_flag</code> is non-zero.
2	FLUX	Flux (J/s).
3	HTC	Heat transfer coefficient (W/m ² /K).
4	Fluid_Temp	Fluid temperature (K).
5	Surf_Solid_T	Solid temperature at the solid surface (K).
6	AVG_Solid_T	Average solid temperature (K).
Varies	<depth>M	The solid temperature at various depths (corresponding to the sublayers) in the solid layer (K). The number of columns corresponds to the values specified for <code>cht1d.in > boundaries > boundary > solid_layers > solid_layer > num_sub_layers</code> .

crevice.out

CONVERGE generates `crevice.out` when the [crevice model](#) is active (*i.e.*, when `engine.in > crevice_flag = 1`). The file contains the predicted pressures in the various crevice volumes in addition to the total mass in the crevice.

Table 26.13: Description of `crevice.out`.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <code>inputs.in > simulation_control > crank_flag = 0</code> or in crank angle degrees if <code>crank_flag</code> is non-zero.

Chapter 26: Output Files

3D Simulation Output Files crevice.out

Column	Header (units)	Description
2	P2(ring 1) (MPa)	Pressure in section 2 of the crevice region .
3	P3(gap) (MPa)	Pressure in section 3 of the crevice region .
4	P4(ring2) (MPa)	Pressure in section 4 of the crevice region .
5	P(2n-1)(gap)	Pressure in section $2n - 1$ of the crevice region for the n -th ring.
6	P(2n)(ring n)	Pressure in section $2n$ of the crevice region for the n -th ring. Columns 5 and 6 are repeated for each additional ring n .
Varies [‡]	Mass (kg)	Total mass in the crevice region.
Varies [‡]	Species name (e.g., O2)	Mass fraction of the named species in the crevice region, with one column for each gas species in the simulation.
Final [‡]	blowby_rate (kg/s)	Rate of mass blowby into the crankcase.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in > output_control > output_multiplier](#).

crevice_rings.out

CONVERGE generates *crevice_rings.out* when the [crevice model](#) is active (*i.e.*, when [engine.in > crevice_flag = 1](#)). The file contains information about the motion of the two rings.

Table 26.14: Description of *crevice_rings.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	hb (ring 1) (meters)	Ring 1 position relative to the bottom of the ring region .
3	dhb/dt (ring 1) (m/s)	Ring 1 velocity.
4	hb (ring 2) (meters)	Ring 2 position relative to the bottom of the ring region.
5	dhb/dt (ring 2) (m/s)	Ring 2 velocity.
6+	hb (ring n) (meters)	Ring n position relative to the bottom of the ring region.
Varies	dhb/dt (ring n) (m/s)	Ring n velocity.

dmr_mech_info.out

CONVERGE generates *dmr_mech_info.out* when the [SAGE detailed chemical kinetics solver](#) and [dynamic mechanism reduction](#) are active (*i.e.*, when [combust.in > sage_model > active = 1](#) and [sage_model > dmr_flag = 1](#)). This file records the species and reaction statistics (average, minimum, and maximum values) as the simulation progresses.

Table 26.15: Description of *dmr_mech_info.out*.

Chapter 26: Output Files

3D Simulation Output Files dmr_mech_info.out

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Avg_Species	Average number of species per cell in the region.
3	Max_Species	Number of species in the cell with the most species.
4	Min_Species	Number of species in the cell with the least species.
5	Avg_Reactions	Average number of reactions per cell in the region.
6	Max_Reactions	Number of reactions in the cell with the most reactions.
7	Min_Reactions	Number of reactions in the cell with the least reactions.

drill_bit_profile<boundary ID>.out

CONVERGE generates *drill_bit_profile<boundary ID>.out* when a drill bit profile is used for WALL boundary motion (*i.e.*, when [motion_sets.in](#) > motion_object > mechanism = DRILL_BIT). The <boundary ID> corresponds to the boundary ID value in [boundary.in](#) that corresponds to the drill bit.

Table 26.16: Description of the header in *drill_bit_profile<boundary ID>.out*.

Parameter	Description
TEMPORAL	Specifies a temporally varying file.
CYCLIC <period>	Specifies the cyclic period in crank angle degrees.
<Header>	Specifies the headers as shown in Table 26.17.
<i>supplied_position</i>	Specifies the x, y, z coordinates that give the origin of the local coordinate system (motion_sets.in > motion_object > drill_bit_center) followed by two orthogonal vectors to define the orientation of the boundary. The first vector is the drill bit axis (motion_sets.in > motion_object > drill_bit_axis) and the second is orthogonal to the drill bit axis. CONVERGE will use the coordinate system origin as a reference to move the boundary.
<Data>	Motion profile recorded every 0.1 crank angle degrees.

Following the header, the *drill_bit_profile<boundary ID>.out* files contain several columns of data, which are described in the table below.

Table 26.17: Description of *drill_bit_profile<boundary ID>.out*.

Colu mn	Header (units)	Description
1	crank	Time in crank angle degrees.
2	x	Temporally varying origin of the local coordinate system. Calculated based on the rotation of the drill bit center (motion_sets.in > motion_object > drill_bit_center) with respect to the shaft axis.
3	y	

Chapter 26: Output Files

3D Simulation Output Files drill_bit_profile<boundary ID>.out

Column mn	Header (units)	Description
4	z	
5	v1x	Initial direction of the x axis for the local coordinate system. \mathbf{v}_{rot} in Equation 26.2.
6	v1y	
7	v1z	
8	v2x	Initial direction of the y axis for the local coordinate system. This is an arbitrary
9	v2y	axis that is orthogonal to the drill bit axis.
10	v2z	

Drill bit arbitrary motion is determined by

$$\mathbf{v}_{rot} = \mathbf{v} \cos \theta + (\mathbf{u} \times \mathbf{v}) \sin \theta + \mathbf{u}(\mathbf{u} \cdot \mathbf{v})(1 - \cos \theta) \quad (26.2)$$

where \mathbf{v}_{rot} is the rotated axis, \mathbf{v} is the drill bit axis ([motion_sets.in > motion_object > drill_bit_axis](#)), \mathbf{u} is the shaft axis ([motion_object > shaft_axis](#)), and θ is the time in *crank angle degrees*.

To use this file as an [arbitrary motion](#) profile in another simulation, change the file extension from *.out* to *.in* and include it in [boundary.in > boundary_conditions > boundary > velocity > value](#).

dynamic.out

CONVERGE generates *dynamic.out* for all 3D simulations. These files contain swirl ratio, tumble ratio, angular momentum, and vorticity data. The data in *dynamic.out* apply to the entire domain. If a simulation has more than one region, or if [inputs.in > output_control > dynamic_flag = 1](#), CONVERGE generates a *dynamic_region<region ID>.out* file for each region in the domain. For *dynamic_region<region ID>.out*, this *<region ID>* is the value set as [dynamic.in > ang_mom_flux > output > ang_mom_flux_region_to](#).

The output frequency of this file is controlled by [inputs.in > output_control > twrite_files](#).

Table 26.18: Description of *dynamic.out* and *dynamic_region<region ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
-	2* Tumble_Ratio_X	Average tumble ratio about the x axis.
-	3* Tumble_Ratio_X+	Positive tumble ratio about the x axis.
-	4* Tumble_Ratio_X-	Negative tumble ratio about the x axis.
-	5* Tumble_Ratio_Y	Average tumble ratio about the y axis.

Chapter 26: Output Files

3D Simulation Output Files dynamic.out

Column	Header (units)	Description
-	6* Tumble_Ratio_Y+	Positive tumble ratio about the y axis.
-	7* Tumble_Ratio_Y-	Negative tumble ratio about the y axis.
-	8* Swirl_Ratio	Swirl ratio.
	9* Swirl_Ratio_+	Positive swirl ratio about the z axis.
	10* Swirl_Ratio_-	Negative swirl ratio about the z axis.
2†	- Angular_Vel_X (rad/s)	Average angular velocity in the x direction.
3†	- Angular_Vel_X+ (rad/s)	Positive angular velocity in the x direction.
4†	- Angular_Vel_X- (rad/s)	Negative angular velocity in the x direction.
5†	- Angular_Vel_Y (rad/s)	Average angular velocity in y direction.
6†	- Angular_Vel_Y+ (rad/s)	Positive angular velocity in the y direction.
7†	- Angular_Vel_Y- (rad/s)	Negative angular velocity in the y direction.
8†	- Angular_Vel_Z (rad/s)	Angular velocity in the z direction.
9†	- Angular_Vel_Z+ (rad/s)	Positive angular velocity in the z direction.
10†	- Angular_Vel_Z- (rad/s)	Negative angular velocity in the z direction.
11	Ang_Mom_X (kg·m ² /s)	Angular momentum about the x axis.
12	Ang_Mom_Y (kg·m ² /s)	Angular momentum about the y axis.
13	Ang_Mom_Z (kg·m ² /s)	Angular momentum about the z axis.
14	Mag_Vorticity (s ⁻¹)	Magnitude of vorticity.
15	Ang_Mom_Flux_X (kg·m ² /s ²)	Angular momentum flux in the x direction between two specified regions. This column is written only if the <code>ang_mom_flux</code> settings block is included in dynamic.in .
16	Ang_Mom_Flux_Y (kg·m ² /s ²)	Angular momentum flux in the y direction between two specified regions. This column is written only if the <code>ang_mom_flux</code> settings block is included in dynamic.in .
17	Ang_Mom_Flux_Z (kg·m ² /s ²)	Angular momentum flux in the z direction between two specified regions. This column is written only if the <code>ang_mom_flux</code> settings block is included in dynamic.in .
18	Mass_Center_X (meters)	X component of the center of mass that is used in swirl and tumble calculations.
19	Mass_Center_Y (meters)	Y component of the center of mass that is used in swirl and tumble calculations.
20	Mass_Center_Z (meters)	Z component of the center of mass that is used in swirl and tumble calculations.

*[inputs.in](#) > simulation_control > crank_flag is 1.

†[inputs.in](#) > simulation_control > crank_flag is 0.

Chapter 26: Output Files

3D Simulation Output Files dynamic.out

The swirl ratio is defined as the ratio of the angular speed of the flow about the center of mass in the z direction, ω_3 , to the angular speed of the crankshaft, $\omega_{crankshaft}$:

$$swirl_ratio = \frac{\omega_3}{\omega_{crankshaft}}. \quad (26.3)$$

The tumble ratio in x direction is defined as the ratio of the angular speed of the flow about the center of mass in the x direction, ω_1 , to the angular speed of the crankshaft, $\omega_{crankshaft}$:

$$tumble_ratio_x = \frac{\omega_1}{\omega_{crankshaft}}. \quad (26.4)$$

Similarly, the tumble ratio in the y direction is calculated by evaluating the ratio of the angular speed of the flow about the center of mass in the y direction, ω_2 , to the angular speed of the crankshaft, $\omega_{crankshaft}$.

The components of ω_i are calculated from the angular momentum, L_i , and the moment of inertia, I_i , as

$$\omega_i = \frac{L_i}{I_i}. \quad (26.5)$$

The angular momentum, L_k , for a Cartesian system can be expressed as

$$L_k = x_i u_j \epsilon_{ijk}, \quad (26.6)$$

where u_j represents the velocity fields and x_i represents the coordinate system with the axis of rotation at $x_i=0$ and the Levi-Civita symbol ϵ_{ijk} expressed as

$$\epsilon_{ijk} = \begin{cases} 0 & \text{if any two indices are the same} \\ +1 & \text{if } ijk = 123, 231, \text{ or } 312 \\ -1 & \text{if } ijk = 132, 213, \text{ or } 321. \end{cases} \quad (26.7)$$

For a discrete system of cells, the angular momentum about the x, y, and z axis, L_1 , L_2 , and L_3 , respectively, can be calculated as

Chapter 26: Output Files

3D Simulation Output Files dynamic.out

$$\begin{aligned}
 L_1 &= \sum_{n=1}^{\text{numcells}} m_n ((y_n - y_{cm}) w_n - (z_n - z_{cm}) v_n) \\
 L_2 &= \sum_{n=1}^{\text{numcells}} m_n ((z_n - z_{cm}) u_n - (x_n - x_{cm}) w_n) \\
 L_3 &= \sum_{n=1}^{\text{numcells}} m_n ((x_n - x_{cm}) v_n - (y_n - y_{cm}) u_n),
 \end{aligned} \tag{26.8}$$

where m_n is the mass of each cell; x_n , y_n , and z_n are the coordinates of each cell; u_n , v_n , and w_n are components of velocity for each cell; and x_{cm} , y_{cm} , and z_{cm} are the center of mass. CONVERGE uses the center of mass as the rotation center when calculating rotational quantities such as swirl and tumble. For sector cases, $x_{cm} = 0$ and $y_{cm} = 0$ (*i.e.*, the z axis is the axis of rotation).

The moment of inertia about the x, y, and z axis, I_1 , I_2 , and I_3 , respectively, for a system of cells can be expressed as

$$\begin{aligned}
 I_1 &= \sum_{n=0}^{\text{numcells}} m_n [(y_n - y_{cm})^2 + (z_n - z_{cm})^2] \\
 I_2 &= \sum_{n=0}^{\text{numcells}} m_n [(z_n - z_{cm})^2 + (x_n - x_{cm})^2] \\
 I_3 &= \sum_{n=0}^{\text{numcells}} m_n [(x_n - x_{cm})^2 + (y_n - y_{cm})^2].
 \end{aligned} \tag{26.9}$$

The vorticity magnitude quantifies the rotating motion of the fluid in a particular region. Mathematically, the curl of the velocity vector yields the vorticity. The next equation expresses the calculation of vorticity in tensor notation:

$$\omega = \nabla \times \bar{u} = \left| \begin{array}{ccc} \hat{i} & \hat{j} & \hat{k} \\ \frac{\partial}{\partial x_i} & \frac{\partial}{\partial x_j} & \frac{\partial}{\partial x_k} \\ u_i & u_j & u_k \end{array} \right| = \left(\frac{\partial u_k}{\partial x_j} - \frac{\partial u_j}{\partial x_k} \right) \hat{i} - \left(\frac{\partial u_k}{\partial x_i} - \frac{\partial u_i}{\partial x_k} \right) \hat{j} + \left(\frac{\partial u_j}{\partial x_i} - \frac{\partial u_i}{\partial x_j} \right) \hat{k}, \tag{26.10}$$

where u_i , u_j , and u_k represent the x, y, and z components of velocity, respectively. CONVERGE calculates the magnitude of the vorticity as

Chapter 26: Output Files

3D Simulation Output Files dynamic.out

$$|\omega| = \sqrt{\left(\frac{\partial u_k}{\partial x_j} - \frac{\partial u_j}{\partial x_k}\right)^2 + \left(\frac{\partial u_k}{\partial x_i} - \frac{\partial u_i}{\partial x_k}\right)^2 + \left(\frac{\partial u_j}{\partial x_i} - \frac{\partial u_i}{\partial x_j}\right)^2}. \quad (26.11)$$

ecfm.out

CONVERGE generates *ecfm.out* when the [Extended Coherent Flamelet Model](#) (ECFM) is active (*i.e.*, when `combust.in > ecfm_model > active = 1`). The *ecfm.out* file contains information about combustion progress for the ECFM.

Table 26.19: Description of *ecfm.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>crank_flag</i> = 0 in <i>inputs.in</i> or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	mbg (kg)	Mass of burned gas.
3	mbg_sigma (kg)	Mass of burned gas generated by flame propagation.
4	mbgign_done (kg)	Mass of burned gas generated by ISSIM. This value should be close to <i>mbgign_targ</i> .
5	mbgign_targ (kg)	Target mass of burned gases generated by ISSIM.
6	flamesurf (m^2)	Flame surface.
7	surfbgign_done (m^2)	Flame surface generated by ISSIM. This value should be close to <i>surfbgign_targ</i> .
8	surfbgign_targ (m^2)	Target flame surface generated by ISSIM.
9	volbg (m^3)	Volume of burned gas.
10	volbg_sigma (m^3)	Volume of burned gas generated by flame propagation.
11	radbg_sigma (meters)	Radius of a sphere with a volume of <i>volbg</i> .
12	rbgign (m)	Radius of burned gases given by the transport equation of the passive Ψ in the ISSIM (ISSIM_RBGIGN), averaged over the flame surface.
13	surfbg_sigma (m^2)	Surface of a sphere with a volume of <i>volbg</i> .
14	alphaign	ISSIM transition factor (ISSIM_ALPHA), averaged over the flame surface (1 = ignition phase, 0 = end of ignition phase).
15	flspeed (m/s)	Laminar flamespeed averaged over the flame surface.
16	flthick (meters)	Laminar flame thickness averaged over the flame surface.

Chapter 26: Output Files

3D Simulation Output Files ecfm.out

Column	Header (units)	Description
17	flspeed_cor	Laminar flamespeed correction factor due to heat loss at walls, averaged over the flame surface.
18	fuelair_eqratio	Fuel/air equivalence ratio averaged over the flame surface, $\langle \overline{\Phi} \rangle = \frac{\int \Sigma \overline{\Phi} dV}{\int \Sigma dV}$.
19	tempfg (K)	Fresh gas temperature averaged over the flame surface, $\langle \widetilde{T}_u \rangle = \frac{\int \Sigma \widetilde{T}_u dV}{\int \Sigma dV}$.
20	at_sgs (s^{-1})	Turbulent strain term in flame surface density equation, $\langle At_{sgs} \rangle = \frac{\int At_{sgs} \Sigma dV}{\int \Sigma dV}$ averaged over the flame surface:
21	at_res (s^{-1})	Dilatation source term in flame surface density equation, $\langle At_{res} \rangle = \frac{\int At_{res} \Sigma dV}{\int \Sigma dV}$ averaged over the flame surface:
22	curv_sgs (s^{-1})	Curvature source term in flame surface density equation, $\langle Curv_{sgs} \rangle = \frac{\int Curv_{sgs} \Sigma dV}{\int \Sigma dV}$ averaged over the flame surface:
23	destrflsurf (s^{-1})	Destruction term in flame surface density equation, averaged $\langle D \rangle = \frac{\int D \Sigma dV}{\int \Sigma dV}$ over the flame surface:
24	time_stretch_cor	Time stretch correction C_t averaged over the flame surface: $\langle C_t \rangle = \frac{\int C_t (At_{sgs} + Curv_{sgs}) \Sigma dV}{(\langle At_{sgs} \rangle + \langle Curv_{sgs} \rangle) \int \Sigma dV}$
25	wall_stretch_cor	Wall stretch correction C_w averaged over the flame surface: $\langle C_w \rangle = \frac{\int C_w (At_{sgs} + Curv_{sgs}) \Sigma dV}{(\langle At_{sgs} \rangle + \langle Curv_{sgs} \rangle) \int \Sigma dV}$
26	lt (meters)	Integral length scale averaged over the flame surface.

Chapter 26: Output Files

3D Simulation Output Files ecfm.out

Column	Header (units)	Description
27	uprime (m/s)	Turbulent velocity fluctuation, $\sqrt{\frac{2}{3}k}$, averaged over the flame surface.
28	lt_over_flthick	Ratio of integral scale to laminar flame thickness averaged over the flame surface.
29	uprim_over_flsp	Ratio of turbulent velocity fluctuation to laminar flamespeed averaged over the flame surface.
30	reynolds	Turbulent Reynolds number based on the density and viscosity of unburned gases.
31	karlovitz	Karlovitz number, $Ka = \frac{u_\eta}{\eta} \frac{\delta_l}{S_l}$, averaged over the flame surface.
32	damkohler	Damköhler number, $Ka = \sqrt{Re_t}$, averaged over the flame surface.
33	resolv_flamesurf (m^2)	Resolved flame surface, $S_{res} = \int \nabla \bar{c} dV$.
34	ki_eq	Equilibrium flame wrinkling obtained via KPP analysis, $\Xi_{eq} = 1 + \frac{2\sqrt{3v_t a_t}}{S_l}$, averaged over the flame surface, $\langle \Xi_{eq} \rangle = \frac{\int \Xi_{eq} \nabla \bar{c} dV}{S_{res}}$.
35	ki_eff	Effective wrinkling, defined as the ratio of the total flame surface to the resolved flame surface. Because the resolved flame surface does not correctly represent the mean flame surface of the flame kernel during ignition, ki_eff is not correctly defined during ignition. The maximum of S_{res} and surfbg_sigma is used to avoid unphysical values of ki_eff during ignition, $\Xi_{eff} = \frac{S_{tot}}{\max(S_{res}, surfbg_sigma)}$.
36	ctilde_max	Maximum value of global progress variable \tilde{c} . Can be used to monitor combustion progress.
37	ctilde_sigma_max	Maximum value of global progress variable due to flame propagation.

Chapter 26: Output Files

3D Simulation Output Files ecfm.out

Column	Header (units)	Description
38	knock_index	Ratio of $\dot{\omega}_{TKI}/(\dot{\omega}_{TKI} + \dot{\omega}_\Sigma)$.
39	integrated_knock_index	Value of knock_index integrated over time.
40	curv_res (s^{-1})	Resolved stretch by curvature, averaged over the flame surface (applies only to LES).
41	stretch_ign (s^{-1})	Stretch source term in flame surface density equation imposed by ISSIM for LES.

ecfm3z.out

CONVERGE generates *ecfm3z.out* when the [3-Zone Extended Coherent Flame Model](#) (ECFM3Z) is active (*i.e.*, when *combust.in* > *ecfm3z_model* > *active* = 1). The *ecfm3z.out* file contains information about combustion progress of the ECFM3Z.

Table 26.20: Description of *ecfm3z.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>crank_flag</i> = 0 in <i>inputs.in</i> or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	mbg (kg)	Mass of burned gas.
3	mbg_sigma (kg)	Mass of burned gas generated by flame propagation.
4	mbgign_done (kg)	Mass of burned gas generated by ISSIM. This value should be close to <i>mbgign_targ</i> .
5	mbgign_targ (kg)	Target mass of burned gases generated by ISSIM.
6	flamesurf (m^2)	Flame surface.
7	surfbgign_done (m^2)	Flame surface generated by ISSIM. This value should be close to <i>surfbgign_targ</i> .
8	surfbgign_targ (m^2)	Target flame surface generated by ISSIM.
9	volbg (m^3)	Volume of burned gas.
10	volbg_sigma (m^3)	Volume of burned gas generated by flame propagation.
11	radbg_sigma (m)	Radius of a sphere with a volume of <i>volbg</i> .
12	rbgign (m)	Radius of burned gases given by the transport equation of the passive Ψ in the ISSIM (<i>ISSIM_RBGIGN</i>), averaged over the flame surface.
13	surfbg_sigma (m^2)	Surface of a sphere with a volume of <i>volbg</i> .
14	alphaign	ISSIM transition factor (<i>ISSIM_ALPHA</i>), averaged over the flame surface (1 = ignition phase, 0 = end of ignition phase).
15	flspeed (m/s)	Laminar flamespeed averaged over the flame surface.
16	flthick (m)	Laminar flame thickness averaged over the flame surface.

Chapter 26: Output Files

3D Simulation Output Files ecfm3z.out

Column	Header (units)	Description
17	flspeed_cor	<u>Laminar flamespeed correction factor</u> due to heat loss at walls, averaged over the flame surface.
18	fuelair_eqratio	Fuel/air equivalence ratio averaged over the flame surface, $\langle \bar{\Phi} \rangle = \frac{\int \bar{\Sigma} \Phi dV}{\int \bar{\Sigma} dV}$.
19	tempfg (K)	Fresh gas temperature averaged over the flame surface, $\langle \tilde{T}_u \rangle = \frac{\int \bar{\Sigma} \tilde{T}_u dV}{\int \bar{\Sigma} dV}$.
20	at_sgs (s^{-1})	Turbulent strain term in flame surface density equation, $\langle At_{sgs} \rangle = \frac{\int At_{sgs} \Sigma dV}{\int \Sigma dV}$ averaged over the flame surface:
21	at_res (s^{-1})	Dilatation source term in flame surface density equation, $\langle At_{res} \rangle = \frac{\int At_{res} \Sigma dV}{\int \Sigma dV}$ averaged over the flame surface:
22	curv_sgs (s^{-1})	Curvature source term in flame surface density equation, $\langle Curv_{sgs} \rangle = \frac{\int Curv_{sgs} \Sigma dV}{\int \Sigma dV}$ averaged over the flame surface:
23	destrflsurf (s^{-1})	Destruction term in flame surface density equation, averaged $\langle D \rangle = \frac{\int D \Sigma dV}{\int \Sigma dV}$ over the flame surface:
24	time_stretch_cor	<u>Time stretch correction</u> C_t , averaged over the flame surface: $\langle C_t \rangle = \frac{\int C_t (At_{sgs} + Curv_{sgs}) \Sigma dV}{(\langle At_{sgs} \rangle + \langle Curv_{sgs} \rangle) \int \Sigma dV}$
25	wall_stretch_cor	<u>Wall stretch correction</u> C_w , averaged over the flame surface: $\langle C_w \rangle = \frac{\int C_w (At_{sgs} + Curv_{sgs}) \Sigma dV}{(\langle At_{sgs} \rangle + \langle Curv_{sgs} \rangle) \int \Sigma dV}$
26	lt (m)	Integral length scale averaged over the flame surface.

Chapter 26: Output Files

3D Simulation Output Files ecfm3z.out

Column	Header (units)	Description
27	uprime (m/s)	$\sqrt{\frac{2}{3}k}$ Turbulent velocity fluctuation, $\sqrt{\frac{2}{3}k}$, averaged over the flame surface.
28	lt_over_flthick	Ratio of integral scale to laminar flame thickness averaged over the flame surface.
29	uprim_over_flsp	Ratio of turbulent velocity fluctuation to laminar flamespeed averaged over the flame surface.
30	reynolds	Turbulent Reynolds number based on the density and viscosity $Re_t = \frac{ul}{\nu_u}$ of unburned gases, $\frac{ul}{\nu_u}$, averaged over the flame surface.
31	karlovitz	$Ka = \frac{u_\eta}{\eta} \frac{\delta_l}{S_l}$ Karlovitz number, $Ka = \frac{u_\eta}{\eta} \frac{\delta_l}{S_l}$, averaged over the flame surface.
32	damkohler	$\sqrt{Re_t}$ Damköhler number, $\sqrt{Re_t}$, averaged over the flame surface.
33	resolv_flamesurf (m ²)	Resolved flame surface, $S_{res} = \int \nabla c dV$.
34	ki_eq	Equilibrium flame wrinkling obtained via KPP analysis, $\Xi_{eq} = 1 + \frac{2\sqrt{3\nu_t a_t}}{S_l}$, averaged over the flame surface, $\langle \Xi_{eq} \rangle = \frac{\int \Xi_{eq} \nabla c dV}{S_{res}}$.
35	ki_eff	Effective wrinkling, defined as the ratio of the total flame surface to the resolved flame surface. Because the resolved flame surface does not correctly represent the mean flame surface of the flame kernel during ignition, ki_eff is not correctly defined during ignition. The maximum of S _{res} and surfbg_sigma is used to avoid unphysical values of ki_eff during ignition, $\Xi_{eff} = \frac{S_{tot}}{\max(S_{res}, surfbg_sigma)}$.
36	ctilde_max	Maximum value of global progress variable \tilde{c} . Can be used to monitor combustion progress.

Chapter 26: Output Files

3D Simulation Output Files ecfm3z.out

Column	Header (units)	Description
37	ctilde_sigma_max	Maximum value of global progress variable due to flame propagation.
38	knock_index	Ratio of $\dot{\omega}_{TKI}/(\dot{\omega}_{TKI} + \dot{\omega}_\Sigma)$.
39	integrated_knock_index	Value of knock_index integrated over time.
40	curv_res (s^{-1})	Resolved stretch by curvature, averaged over the flame surface (applies only to LES).
41	stretch_ign (s^{-1})	Stretch source term in flame surface density equation imposed by ISSIM for LES.

emissions.out

CONVERGE generates *emissions.out* when [emissions modeling](#) is active (i.e., [combust.in](#) > *emissions_flag* = 1). The *emissions.out* file contains emissions-related masses, and the frequency with which these data are written is controlled by [inputs.in](#) > *output_control* > *twrite_files*.

Table 26.21: Description of *emissions.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2 [‡]	Hiroy_Soot (kg)	Predicted soot in region from the Hiroyasu soot model.
3* [‡]	NOx (kg)	Total mass of NOx in the region (from the Zel'dovich model).
4	Molefrac_NOx	Mole fraction of NOx in the region.
5* [‡]	HC (kg)	Total mass of hydrocarbon species in the region.
6	Molefrac_HC	Mole fraction of hydrocarbon species in the region.
7* [‡]	CO (kg)	Total mass of CO in region.
8	Molefrac_CO	Mole fraction of CO in the region.
9* [‡]	CO2 (kg)	Total mass of CO2 in region.
12	Molefrac_CO2	Mole fraction of CO2 in the region.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > *output_control* > *output_multiplier*.

equiv_ratio_bin.out

CONVERGE generates *equiv_ratio_bin.out* when the mixing output option is active (i.e., when [inputs.in](#) > *output_control* > *mixing_output_flag* = 1). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the fraction of the total mass which has an equivalence ratio value within the specified range of each bin. The mass within the domain is binned by equivalence ratio into twenty intervals, starting at 0.0-0.1 and ending at 1.9-2.0.

Chapter 26: Output Files

3D Simulation Output Files equiv_ratio_bin.out

Table 26.22: Description of *equiv_ratio_bin.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	0.0 - 0.1	Mass fraction that has a equivalence ratio from 0.0 to 0.1.
3	0.1 - 0.2	Mass fraction that has an equivalence ratio from 0.1 to 0.2.
4	0.2 - 0.3	Mass fraction that has an equivalence ratio from 0.2 to 0.3.
5	0.3 - 0.4	Mass fraction that has an equivalence ratio from 0.3 to 0.4.
6	0.4 - 0.5	Mass fraction that has an equivalence ratio from 0.4 to 0.5.
7	0.5 - 0.6	Mass fraction that has an equivalence ratio from 0.5 to 0.6.
8	0.6 - 0.7	Mass fraction that has an equivalence ratio from 0.6 to 0.7.
9	0.7 - 0.8	Mass fraction that has an equivalence ratio from 0.7 to 0.8.
10	0.8 - 0.9	Mass fraction that has an equivalence ratio from 0.8 to 0.9.
11	0.9 - 1.0	Mass fraction that has an equivalence ratio from 0.9 to 1.0.
12	1.0 - 1.1	Mass fraction that has an equivalence ratio from 1.0 to 1.1.
13	1.1 - 1.2	Mass fraction that has an equivalence ratio from 1.1 to 1.2.
14	1.2 - 1.3	Mass fraction that has an equivalence ratio from 1.2 to 1.3.
15	1.3 - 1.4	Mass fraction that has an equivalence ratio from 1.3 to 1.4.
16	1.4 - 1.5	Mass fraction that has an equivalence ratio from 1.4 to 1.5.
17	1.5 - 1.6	Mass fraction that has an equivalence ratio from 1.5 to 1.6.
18	1.6 - 1.7	Mass fraction that has an equivalence ratio from 1.6 to 1.7.
19	1.7 - 1.8	Mass fraction that has an equivalence ratio from 1.7 to 1.8.
20	1.8 - 1.9	Mass fraction that has an equivalence ratio from 1.8 to 1.9.
21	1.9 - 2.0	Mass fraction that has an equivalence ratio from 1.9 to 2.0.

erosion_boundary_<boundary ID>.out

When *inputs.in* > *feature_control* > *erosion_flag* = 1, CONVERGE generates an *erosion_boundary_<boundary ID>.out* file for each boundary listed in *erosion.in*. The *erosion_boundary_<boundary ID>.out* file shows the amount of mass that has impinged on and eroded from the boundary, as well as the erosion ratio, the erosion rate, and the mass erosion rate.

Chapter 26: Output Files

3D Simulation Output Files erosion_boundary_<boundary ID>.out

Table 26.23: Description of *erosion_boundary_<boundary ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Mass_Eroded (kg)	Total mass that has eroded from the boundary (m_{er}).
3	Mass_Impinged (kg)	Total mass that has impinged on the boundary (m_{im}).
4	Erosion_Ratio (-)	Erosion ratio (m_{er}/m_{im}).
5	Erosion_Rate (m/s)	Erosion rate $\dot{m}_{er}/(\rho_s A)$, where ρ_s and A are the density and area of the surface, respectively.
6	Mass_Erosion_Rate (kg/m ² -s)	Mass erosion rate \dot{m}_{er}/A .

film_urea.out

CONVERGE generates *film_urea.out* when the [detailed decomposition urea approach](#) and [wall film model](#) are active (i.e., when [urea.in > urea_model = 3](#) and [parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > wall_model = FILM](#)). This file contains film composition data.

Table 26.24: Description of *film_urea.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	H2O_all (kg)	Mass of water in film.
3	Urea_all (kg)	Mass of urea (includes aqueous urea, molten solid urea, and all deposition products) in film.
4	Urea_solid_all (kg)	Mass of solid urea in film.
6	Biuret_all (kg)	Mass of biuret in film.
7	CYA_all (kg)	Mass of cyanuric acid in film.
8	Ammelide_all (kg)	Mass of ammelide in film.
9	NCO_-_all (kg)	Mass of NCO- in film.
10	NH4+_all (kg)	Mass of NH4+ in film.
11	H+_all (kg)	Mass of H+ in film.

fsi_object_<name>.out

CONVERGE generates *fsi_object_<name>.out* when [fluid-structure interaction modeling](#) is active (i.e., when [inputs.in > feature_control > fsi_flag = 1](#)). CONVERGE writes data for each

Chapter 26: Output Files

3D Simulation Output Files fsi_object_<name>.out

FSI object to an *fsi_object_<name>.out* file, where *<name>* is the *object_id* of the FSI object in [*fsi.in*](#). For example, the output file might be *fsi_object_anchor.out* or *fsi_object_3.out* (the *object_id* is a string, not a number).

The format of *fsi_object_<name>.out* depends on the value of [*fsi.in* > rigid_fsi_objects > object > 1dof_constraint_flag](#). The following tables describe the possible formats of *fsi_object_<name>.out*.

Table 26.25: Description of *fsi_object_<name>.out* when *1dof_constraint_flag* = 0.

Column	Header (units)	Description
1	time (seconds)	Time.
2	Pos_x (meters)	Position in the x direction.
3	Pos_y (meters)	Position in the y direction.
4	Pos_z (meters)	Position in the z direction.
5	Vel_x (m/s)	Velocity in the x direction.
6	Vel_y (m/s)	Velocity in the y direction.
7	Vel_z (m/s)	Velocity in the z direction.
8	Acc_x (m/s ²)	Acceleration in the x direction.
9	Acc_y (m/s ²)	Acceleration in the y direction.
10	Acc_z (m/s ²)	Acceleration in the z direction.
11	Angle_x (rad)	Angle in the x direction.
12	Angle_y (rad)	Angle in the y direction.
13	Angle_z (rad)	Angle in the z direction.
14	Rot_Vel_x (rad/s)	Rotational velocity in the x direction.
15	Rot_Vel_y (rad/s)	Rotational velocity in the y direction.
16	Rot_Vel_z (rad/s)	Rotational velocity in the z direction.
17	Rot_Acc_x (rad/s ²)	Rotational acceleration in the x direction.
18	Rot_Acc_y (rad/s ²)	Rotational acceleration in the y direction.
19	Rot_Acc_z (rad/s ²)	Rotational acceleration in the z direction.
20	Force_x (N)	Force in the x direction.
21	Force_y (N)	Force in the y direction.
22	Force_z (N)	Force in the z direction.
23	Moment_x (N-m)	Moment in the x direction.
24	Moment_y (N-m)	Moment in the y direction.
25	Moment_z (N-m)	Moment in the z direction.

Chapter 26: Output Files

3D Simulation Output Files fsi_object_<name>.out

Column	Header (units)	Description
26	React_Force_x (N)	Reaction force in the x direction. This column is written only when fsi.in > rigid_fsi_objects > object general_constrained_motion > active = 1.
27	React_Force_y (N)	Reaction force in the y direction. This column is written only when fsi.in > rigid_fsi_objects > object general_constrained_motion > active = 1.
28	React_Force_z (N)	Reaction force in the z direction. This column is written only when fsi.in > rigid_fsi_objects > object general_constrained_motion > active = 1.
29	React_Moment_x (N-m)	Reaction moment in the x direction. This column is written only when fsi.in > rigid_fsi_objects > object general_constrained_motion > active = 1.
30	React_Moment_y (N-m)	Reaction moment in the y direction. This column is written only when fsi.in > rigid_fsi_objects > object general_constrained_motion > active = 1.
31	React_Moment_z (N-m)	Reaction moment in the z direction. This column is written only when fsi.in > rigid_fsi_objects > object general_constrained_motion > active = 1.

Table 26.26: Description of *fsi_object_<name>.out* when *1dof_constraint_flag = 1*.

Column	Header (units)	Description
1	time (seconds)	Time.
2	Position (meters)	Position.
4	Vel (m/s)	Velocity.
5	Acc (m/s ²)	Acceleration.
6	Force (N)	Force.

Table 26.27: Description of *fsi_object_<name>.out* when *1dof_constraint_flag = 2*.

Column	Header (units)	Description
1	time (seconds)	Time.
2	Angle (degrees)	Angle.
3	Angle (radians)	Angle.
4	Rot_Vel (rad/s)	Rotational velocity.
5	Rot_Acc (rad/s ²)	Rotational acceleration.
6	Moment (N-m)	Moment.

Chapter 26: Output Files

3D Simulation Output Files fsi_spring_<index>.out

fsi_spring_<index>.out

CONVERGE generates *fsi_spring_<index>.out* when [fluid-structure interaction \(FSI\) modeling](#) is active (*i.e.*, when [inputs.in > feature_control > fsi_flag = 1](#)) and [fsi.in > spring_flag = 1](#). CONVERGE writes data for each spring to an *fsi_spring_<index>.out* file, where <index> is the index assigned to the spring (*e.g.*, 0001).

Table 26.28: Description of *fsi_spring_<index>.out*.

Column	Header (units)	Description
1	Time (seconds)	Simulation time.
2	Disp_x (meters)	Displacement in the x direction of the spring end attached to the FSI object.
3	Disp_y (meters)	Displacement in the y direction of the spring end attached to the FSI object.
4	Disp_z (meters)	Displacement in the z direction of the spring end attached to the FSI object.
5	Defl_x (meters)	Deflection of the spring in the x direction.
6	Defl_y (meters)	Deflection of the spring in the y direction.
7	Defl_z (meters)	Deflection of the spring in the z direction.
8	Force_x (N)	Spring force in the x direction.
9	Force_y (N)	Spring force in the y direction.
10	Force_z (N)	Spring force in the z direction.
11	Torque_x (N-m)	Torque about the x axis exerted by the spring.
12	Torque_y (N-m)	Torque about the y axis exerted by the spring.
13	Torque_z (N-m)	Torque about the z axis exerted by the spring.

g_eqn.out

CONVERGE generates *g_eqn.out* when the G-Equation combustion model is active (*i.e.*, when [combust.in > g_eqn_model > active = 1](#)). The *g_eqn.out* file contains information about combustion progress for this combustion model.

Table 26.29: Description of *g_eqn.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	MEAN_TEMP_FOR_G<0 (K)	Density-weighted average of temperature when the value of <i>G_EQN_TRANSPORT</i> is less than zero.
3	MEAN_UNBURNED_TEMP_FOR_G<0 (K)	Density-weighted average of unburned temperature when the value of <i>G_EQN_TRANSPORT</i> is less than zero.

Chapter 26: Output Files

3D Simulation Output Files gti_interface.out

gti_interface.out

CONVERGE generates *gti_interface.out* when co-simulation is active (*i.e.*, when [inputs.in > feature_control > cosimulation_flag = 1](#)), [cosimulation.in](#) contains the GT-SUITE settings block, and GT-SUITE boundaries are specified in [boundary.in](#). The *gti_interface.out* file lists the averaged data for each coupling interface between CONVERGE and GT-SUITE. The output frequency is controlled by [inputs.in > output_control > twrite_transfer](#).

Table 26.30: Description of *gti_interface.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Cycle_Number	Cycle number.
3*	Pressure (Pa)	Area-weighted average static pressure.
4*	Temperature (K)	Area-weighted average temperature.
5*	Avg_Density (kg/m ³)	Area-weighted average density.
6*	Avg_Velocity (m/s)	Area-weighted average velocity normal to the wall boundary.
7*	Mass_Flow (kg/s)	Mass flow rate normal to the boundary plane.

* If more than one WALL boundary is coupled with GT-SUITE, these columns will repeat for each such boundary.

issim_ignition_<number>.out

CONVERGE generates *issim_ignition_<number>.out* when the [ISSIM spark model](#) is active (*i.e.*, when [combust.in > ecfm_model > spark > active = 1](#)). CONVERGE writes an *issim_ignition_<number>.out* file for each ignition event (*e.g.*, *issim_ignition_0.out* for ignition 0). These files contain data related to the [ISSIM](#), including voltages, the spark ignition state, and the critical energy.

Table 26.31: Description of *issim_ignition_<number>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Ampere (Ampere)	Electric current from secondary circuit.
3	volt_spark (V)	Spark voltage.
4	volt_gas (V)	Gas column voltage.
5	volt_bd (V)	Breakdown voltage.
6	sec_energ (J)	Secondary circuit energy.
7	joule_energ (J)	Energy loss in the secondary circuit.

Chapter 26: Output Files

3D Simulation Output Files issim_ignition_<number>.out

Column	Header (units)	Description
8	ign_energ (J)	Energy transferred to the gas phase.
9	length_spk (meters)	Spark kernel length.
10	xi_spk	Spark length factor.
11	length_spk_total (meters)	Total spark length.
12	ibrkdw	Spark ignition state: -1 = Between ignition events (for spark plugs with cyclical ignition), 0 = No spark ignition, 1 = Pre-discharge, 2 = Breakdown, 3 = Arc/glow.
13	ign_fraction	Ignition fraction for each ignition event. For single ignition, <i>ign_fraction</i> = 1.
14	critical_energ (J)	Critical energy.
15	mbgign_targ (kg)	Total burned gas to deposit on the spark plug.
16	dmbgign (kg)	Burned gas to deposit on the spark plug.
17	dmbgign_done (kg)	Total burned gas that has been deposited on the spark plug.
18	mbgign_done (m ²)	Integrated burned gas that been deposited on the spark plug for this ignition event.
19	surfbgign_targ (m ²)	Flame surface density that has been deposited on the spark plug.
20	dsurfbgign_done (m ²)	Integrated flame surface density that has been deposited on the spark plug.
21	surfbgign	Progress variable.
22	coeff_mbign	Mass progress variable coefficient.

lhv_info.out

CONVERGE generates *lhv_info.out* when LHV are specified (*i.e.*, when [inputs.in](#) > *property_control* > *lhv_flag* = 1). This file lists the calculated lower heating value (LHV) corrector (a_6) values used by CONVERGE.

Table 26.32: Description of *lhv_info.out*.

Column	Header (units)	Description
1	LHV_species	LHV species listed in lhv.in .
2	ori_low_a6	Original value of a_6 , the LHV corrector, before applying values from lhv.in .
3	new_low_a6	Calculated value of a_6

Chapter 26: Output Files

3D Simulation Output Files lhv_info.out

Column	Header (units)	Description
		$a_6 = x_1 MW_1 / R$ where x_1 is the required change in enthalpy per kilogram such that the effective heating value is as specified in lhv.in , MW_1 is the molecular weight, and R is the ideal gas constant.
4	fractional_change	Fractional change between original and specified a_6 .

For each LHV in [lhv.in](#), the columns listed in the table above are followed by three rows: the LHV specified by the user via [lhv.in](#), the LHV calculated by CONVERGE for that species, and the resultant LHV fractional correction.

liquid_parcel.out

CONVERGE generates *liquid_parcel.out* when liquid [parcel modeling](#) is active (*i.e.*, when [inputs.in](#) > *feature_control* > *parcel_mode* > *liquid_parcel* = 1). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the injection results and the spray penetration for each nozzle.

Note that the column numbers listed in the table are representative of output from a simulation with only one injector and one nozzle. If a simulation includes more injectors or nozzles, this file will contain additional columns. If a simulation has more than one region, CONVERGE writes region-specific spray information to *liquid_parcel_region<region ID>.out*. These region-specific output files include only columns 1 and 3-9.

Table 26.33: Description of *liquid_parcel.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	tot_parcels (drop+film)	Total number of spray parcels, including parcels in the wall film.
3	spray_parcels (drop)	Number of spray parcels, excluding those in the wall film.
4‡	liq_spray_mass (kg)	Total liquid spray mass in the domain (not including mass in the wall film).
5‡	Fuel species name, (e.g., C7H16 (kg))	Mass of the fuel species. If there is more than one fuel species in the simulation, there will be one column per species.
6	smd (m)	Sauter mean diameter of spray (not including parcels in the wall film).
7	DV10 (m)	The spray parcel diameter representing the tenth percentile by volume (not including parcels in the wall film).
8	DV50 (m)	The volume median particle size (not including parcels in the wall film).

Chapter 26: Output Files

3D Simulation Output Files liquid_parcel.out

Column	Header (units)	Description
9	DV90 (m)	The spray parcel diameter representing the ninetieth percentile by volume (not including parcels in the wall film).
10	Inj_No	Number of the current injector.
11 ^{‡*}	Inj_Mass (kg)	Amount of injected mass for the current injector.
12 [‡]	Tot_Inj_Mass (kg)	Mass that has been injected by the specified injector since the beginning of the simulation.
- 13 [^]	Inj_Vel (m/s)	Injection velocity of the current injector.
13 [*] -	Inj_Vel_Old (m/s)	Injection velocity for the current injector before the contraction coefficient is applied.
14 [*] -	Inj_Vel_New (m/s)	Injection velocity for the current injector after the contraction coefficient is applied.
15 [*] -	C_a	Contraction coefficient for the current injector.
16 [*] 14 [^]	Inj_Press (MPa)	Injection pressure for the current injector.
17 [*] 15 [^]	Noz_No	Number of the current nozzle.
18 [*] 16 [^]	Spray_Penet (m)	Liquid penetration length for the current nozzle.
19 [*] 17 [^]	Eulerian_Lagrangian_m ass_transitions (kg)	Amount of mass that transitions from Eulerian to Lagrangian.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in > output_control > output_multiplier](#).

* [parcel_introduction.in > injections > injection > injection_control > discharge_coeff_flag = 1 or 2](#)

[^] [parcel_introduction.in > injections > injection > injection_control > discharge_coeff_flag = 0](#)

The injection pressure written to the *liquid_parcel.out* file is given by

$$P_{inj} = \frac{1}{2} \rho_l \left(\frac{V}{C_d} \right)^2, \quad (26.12)$$

where ρ_l is the liquid density, C_d is the discharge coefficient and V is the liquid velocity based on the geometric hole diameter (*i.e.*, the velocity before a contraction coefficient is applied).

CONVERGE calculates the Sauter mean diameter, d_{32} , of the spray as follows:

Chapter 26: Output Files

3D Simulation Output Files liquid_parcel.out

$$d_{32} = \frac{\sum_{i=1}^{N_{tot}} N_i d_i^3}{\sum_{i=1}^{N_{tot}} N_i d_i^2}, \quad (26.13)$$

where N_{tot} is the total number of parcels, N_i is the number of drops of parcel i , and d_i is the diameter of parcel i .

liquid_parcel_ecn.out

CONVERGE generates *liquid_parcel_ecn.out* when liquid [parcel modeling](#) is active (*i.e.*, when [*inputs.in*](#) > *feature_control* > *parcel_mode* > *liquid_parcel* = 1). The output frequency for this file is controlled by [*inputs.in*](#) > *output_control* > *twrite_files*. This file lists the liquid penetration based on various percentages of the liquid mass and includes vapor penetration information based on the ECN (Engine Combustion Network) definition. The output frequency for this file is controlled by [*inputs.in*](#) > *output_control* > *twrite_files*. Columns 2-9 are repeated for each nozzle on each injector.

Table 26.34: Description of *liquid_parcel_ecn.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Inj_No	Number of the current injector.
3	Noz_No	Number of the current nozzle.
5	Spray_Penet90 (m)	Penetration distance from the nozzle exit for the current nozzle based on 90% of the liquid mass.
6	Spray_Penet95 (m)	Penetration distance from the nozzle exit for the current nozzle based on 95% of the liquid mass.
7	Spray_Penet97 (m)	Penetration distance from the nozzle exit for the current nozzle based on 97% of the liquid mass.
8	Spray_Penet99 (m)	Penetration distance from the nozzle exit for the current nozzle based on 99% of the liquid mass.
9	Vapor_Penet (m)	Penetration distance from the nozzle exit for the current nozzle based on <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>penet_control</i> > <i>vapor_frac</i> .

liquid_parcel_film.out

CONVERGE generates *liquid_parcel_film.out* when the [wall film model](#) is active (*i.e.*, when [*parcel_wall_interaction.in*](#) > *liquid_wall_interaction_control* > *default_wall_control* > *wall_model* = *FILM*). The output frequency of data in this file is controlled by [*inputs.in*](#) > *output_control* > *twrite_files*. The *liquid_parcel_film.out* file summarizes the wall film data. Note that the film

Chapter 26: Output Files

3D Simulation Output Files liquid_parcel_film.out

data is not written out in a region-by-region manner (*i.e.*, the output corresponds to the entire domain).

Table 26.35: Description of *liquid_parcel_film.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	film_parcels	Number of film parcels in the domain.
3*	film_mass (kg)	Total liquid film mass on all walls.
Varies*	bound_id_<boundary ID>	Mass of the liquid film on the specified boundary. The number of columns will depend on the number of wall boundaries in the simulation.

* The value in this column is the product of the simulated quantity and the value of [inputs.in > output_control > output_multiplier](#).

liquid_parcel_film_accum.out, liquid_parcel_film_accum_net.out

CONVERGE generates *liquid_parcel_film_accum.out* and *liquid_parcel_film_accum_net.out* when the [wall film model](#) is active (*i.e.*, when [parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > wall_model = FILM](#)). The *liquid_parcel_film_accum.out* file lists the liquid parcel mass that has made contact with a boundary. The *liquid_parcel_film_accum_net.out* file gives the liquid parcel mass that has reached a boundary and remained on it (*i.e.*, mass that has not splashed or rebounded).

Table 26.36: Description of *liquid_parcel_film_accum.out* and *liquid_parcel_film_accum_net.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2*	total_hit (kg)	Total liquid film mass that has interacted with the boundary since the beginning of the simulation.
Varies*	bound_id_<boundary ID>	Mass of the liquid film on the specified boundary. The number of columns will depend on the number of wall boundaries in the simulation.

* The value in this column is the product of the simulated quantity and the value of [inputs.in > output_control > output_multiplier](#).

liquid_parcel_film_scrape.out

CONVERGE generates *liquid_parcel_film_scrape.out* when the [wall film model](#) is active (*i.e.*, [parcel_wall_interaction.in > liquid_wall_interaction_control > default_wall_control > wall_model = FILM](#)). This file provides information on the liquid parcel mass that has been scraped from the boundary.

Chapter 26: Output Files

3D Simulation Output Files liquid_parcel_film_scrape.out

Table 26.37: Description of *liquid_parcel_film_scrape.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2*	total_scraped (kg)	Total liquid film mass that has been scraped by the boundary since the beginning of the simulation.
Varies*	bound_id_<boundary ID>	Mass of the liquid film on the specified boundary. The number of columns will depend on the number of wall boundaries in the simulation.

* The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > output_control > output_multiplier.

map_<time>.h5

CONVERGE generates *map_<time>.h5* at the end of each 3D simulation. The *<time>* corresponds to the time in seconds (if [inputs.in](#) > simulation_control > crank_flag = 0) or in crank angle degrees (if crank_flag is non-zero). CONVERGE writes additional *map_<time>.h5* file(s) as specified in [write_map.in](#).

You can use a *map_<time>.h5* file to initialize variables in a new simulation via [mapping](#).

Table 26.38: Description of *map_<time>.h5*.

Column	Header (units)	Description
1	X (meters)	X coordinate of the cell center.
2	Y (meters)	Y coordinate of the cell center.
3	Z (meters)	Z coordinate of the cell center.
4	U (m/s)	U velocity component of the cell.
5	V (m/s)	V velocity component of the cell.
6	W (m/s)	W velocity component of the cell.
7	TEMP (K)	Temperature information in the cell.
8	PRES (Pa)	Pressure information in the cell.
9	Species name (e.g., C7H16)	Species mass fractions in the cell.
10	Passive name (e.g., soot)	Passive value in the cell.
11	TKE (m^2/s^2)	Turbulent kinetic energy value in the cell. This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
12	EPS (m^2/s^3)	Turbulent dissipation value in the cell.
13	OMEGA (s^{-1})	Specific dissipation rate value in the cell.

Chapter 26: Output Files

3D Simulation Output Files map_<time>.h5

Column	Header (units)	Description
14	REGION ID	Region ID.
15	STREAM ID	Stream ID.

map_bound<boundary ID>_<time>.out

CONVERGE generates *map_bound<boundary ID>_<time>.out* for each [INFLOW](#) or [OUTFLOW boundary](#) (i.e., when [boundary.in](#) > *boundary_conditions* > *boundary* > *type* = INFLOW or OUTFLOW) at the end of a 3D simulation. CONVERGE writes additional *map_bound<boundary ID>_<time>.out* files throughout the simulation as specified in [write_map.in](#). You can use a *map_bound<boundary ID>_<time>.out* file to specify [boundary conditions](#) for another simulation.

Table 26.39: Description of the header in *map_bound<boundary ID>_<time>.out*.

Parameter	Description
<i>scale_xyz</i>	Scaling to be applied to the x, y, and z coordinates.
<i>trans_x</i>	Translation to be applied to x coordinates.
<i>trans_y</i>	Translation to be applied to y coordinates.
<i>trans_z</i>	Translation to be applied to z coordinates.
<i>rot_axis</i>	Axis about which the coordinates and velocity of the data below will be rotated. Typically the rotation is applied after scaling and translating. For velocity, however, the rotation is applied before scaling and translating.
<i>rot_angle</i>	Rotation angle (in degrees) about the <i>rot_axis</i> of the data below. Use the right hand rule to determine the direction of rotation about the <i>rot_axis</i> .
0.000 second	Present for file formatting reasons. Note that this row does not specify the time at which this file was written.

Table 26.40: Description of *map_bound<boundary ID>_<time>.out* (after the header).

Column	Header (units)	Description
1	X (meters)	X coordinate of the cell center.
2	Y (meters)	Y coordinate of the cell center.
3	Z (meters)	Z coordinate of the cell center.
4	U (m/s)	U velocity component of the cell.
5	V (m/s)	V velocity component of the cell.
6	W (m/s)	W velocity component of the cell.
7	TEMPERATURE (K)	Temperature information in the cell.
8	PRESSURE (Pa)	Pressure information in the cell.

Chapter 26: Output Files

3D Simulation Output Files map_bound<boundary ID>_<time>.out

Column	Header (units)	Description
9	Species name (e.g., C7H16)	Species mass fraction in the cell.
10	Passive name (e.g., soot)	Passive value in the cell.
11	TKE (m^2/s^2)	Turbulent kinetic energy value in the cell. This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
12	EPS (m^2/s^3)	Turbulent dissipation value in the cell.
13	OMEGA (s^{-1})	Specific dissipation rate value in the cell.
14	REGION ID	Region ID.
15	STREAM ID	Stream ID.

map_parcel_<time>.h5

CONVERGE generates *map_parcel_<time>.h5* when [discrete phase modeling](#) is active (*i.e.*, when *inputs.in* > *feature_control* > *parcel_mode* > *liquid_parcel*, *solid_parcel*, or *gas_parcel* = 1). The <time> corresponds to the time in *seconds* (if *inputs.in* > *simulation_control* > *crank_flag* = 0) or in *crank angle degrees* (if *crank_flag* is non-zero). CONVERGE writes additional *map_parcel_<time>.h5* file(s) as designated in [write_map.in](#).

You can use a *map_parcel_<time>.h5* file to initialize variables in a new simulation via [mapping](#).

Table 26.41: Description of *map_parcel_<time>.h5*.

Column	Header (units)	Description
1	X (<i>meters</i>)	X coordinate of the parcel.
2	Y (<i>meters</i>)	Y coordinate of the parcel.
3	Z (<i>meters</i>)	Z coordinate of the parcel.
4	U_VEL (<i>m/s</i>)	X component of velocity of the parcel.
5	V_VEL (<i>m/s</i>)	Y component of velocity of the parcel.
6	W_VEL (<i>m/s</i>)	Z component of velocity of the parcel.
7	TEMP (K)	Temperature of the parcel.
8	NUM	Number of drops in the parcel.
9	RADIUS (<i>meters</i>)	Radius of the drops in the parcel.
10	REGION_ID	Region ID of the parcel.
11	BOUND_ID	Boundary ID of the parcel.

Chapter 26: Output Files

3D Simulation Output Files map.Parcel_<time>.h5

Column	Header (units)	Description
Varies	<species name>	Liquid mass fraction of <species name> in each drop in the parcel. This file will contain one column per species in the parcel.
Varies	FILM	The value of <i>film_flag</i> for the parcel. 0 = The parcel is not in the wall film, 1 = The parcel is in the wall film.
Varies	FROM_INJECTOR	Injector from which the parcel originated.
Varies	FROM_NOZZLE	Nozzle from which the parcel originated.

mass_avg_flow.out

CONVERGE generates *mass_avg_flow.out* when a 3D simulation includes an [INFLOW or OUTFLOW boundary](#) (*i.e.*, when *boundary.in* > *boundary_conditions* > *boundary* > *type* = INFLOW or OUTFLOW). This file provides mass flow and mass flow-weighted average data at each INFLOW and OUTFLOW boundary.

The first two columns list the cycle number and time, respectively. The next set of columns (Mass_Flow_Rate through Tot_Spray) tabulate the boundary mass flow and thermodynamic data, and CONVERGE repeats these columns for each INFLOW or OUTFLOW boundary. CONVERGE includes the boundary ID in each column heading for easy identification of the data.

Table 26.42: Description of *mass_avg_flow.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Cycle_Number or Pseudo_Time	Simulation cycle number (<i>i.e.</i> , number of time-steps) if <i>inputs.in</i> > <i>solver_control</i> > <i>steady_solver</i> = 0. The pseudo time-step if <i>steady_solver</i> = 1.
3	Mass_Flow_Rate (kg/s)	Mass flow rate normal to the INFLOW/OUTFLOW boundary plane. A positive value indicates flow out of the domain while a negative value indicates flow into the domain.
4	Total_Pres (Pa)	Mass flow-weighted average total pressure.
5	Static_Pres (Pa)	Mass flow-weighted average static pressure.
6	Avg_Temp (K)	Mass flow-weighted average temperature.
7	Avg_Velocity (m/s)	Mass flow-weighted average velocity normal to the INFLOW/OUTFLOW boundary plane.
8	Avg_Density (kg/m ³)	Mass flow-weighted average density.
9	Avg_Mach	Mass flow-weighted average Mach number.
10	Avg_totTemp (K)	Mass flow-weighted average total temperature.

Chapter 26: Output Files

3D Simulation Output Files mass_avg_flow.out

Column	Header (units)	Description
11	Avg_totEnth (J/kg)	<p>Mass flow-weighted average total enthalpy, calculated from the local total enthalpy given by</p> $H = \int_{T_o}^T c_p dT + \sum_{k=1}^N \Delta H_{f,k}^o Y_k + (1/2) u_i u_i,$ <p>where T_o is the temperature at standard state conditions, T is the local temperature, c_p is the mixture-averaged specific heat capacity at constant pressure, $\Delta H_{f,k}^o$ is the standard enthalpy of formation of the k-th species, Y_k is the mass fraction of species k, N is the total number of species, and u_i is the velocity.</p>

mass_avg_regions_flow.out

CONVERGE generates *mass_avg_regions_flow.out* when inter-region flow output is active (*i.e.*, when [inputs.in](#) > *output_control* > *region_flow_flag* is non-zero). This file provides mass flow rates and mass-flow-weighted averages for the interfaces between adjacent regions, [flow-through INTERFACE](#) boundaries, and/or [INFLOW and OUTFLOW boundaries](#). For [inputs.in](#) > *output_control* > *region_flow_flag* = 1, CONVERGE does not write species-specific data. For [inputs.in](#) > *output_control* > *region_flow_flag* = 2, CONVERGE writes species-specific data as specified in [regions_flow.in](#).

For a general variable ϕ , CONVERGE calculates the mass-flow-weighted averages according to

$$\bar{\phi} = \beta \frac{\sum_{i \in +} F_i^+ \phi_i}{\sum_{i \in +} F_i^+} + (1 - \beta) \frac{\sum_{i \in -} F_i^- \phi_i}{\sum_{i \in -} F_i^-}, \quad (26.14)$$

where $F_i^{+/-}$ indicates a positive/negative mass flow rate across cell face i and the sums in the second and third terms are over the faces with positive and negative mass flow rates, respectively. The factor β is given by

$$\beta = \frac{\sum_{i \in +} A_i}{\sum_i A_i}, \quad (26.15)$$

where A_i is the area of face i . The averaging procedure in Equation 26.14 prevents unrealistic fluctuations in $\bar{\phi}$ when an interface contains cell faces with both positive and negative mass flow rates.

Table 26.43: Description of *mass_avg_regions_flow.out*.

Chapter 26: Output Files

3D Simulation Output Files mass_avg_regions_flow.out

Column	Header (<i>units</i>)	When CONVERGE includes this column	Description
1	Time (seconds) or Crank (crank angle degrees)	Always	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Cycle_Number or Pseudo_Time	Always	Simulation cycle number (<i>i.e.</i> , number of time-steps) if inputs.in > solver_control > steady_solver = 0. The pseudo time-step if steady_solver = 1.
3	Flow_Rate (kg/s) Regions_<i>_to_<j>	Always	Mass flow rate from region <i>i</i> to region <i>j</i> .
4	Tot_Mass (kg) Regions_<i>_to_<j>	Always	Integrated mass flow rate from region <i>i</i> to region <i>j</i> .
5	Velocity (m/s) Regions_<i>_to_<j>	Always	Fluid velocity between regions <i>i</i> and <i>j</i> .
6	Tot_Pres (Pa) Regions_<i>_to_<j>	Always	Mass-flow-weighted average total pressure at the interface of regions <i>i</i> and <i>j</i> .
7	Static_Pres (Pa) Regions_<i>_to_<j>	Always	Mass-flow-weighted average static pressure at the interface of regions <i>i</i> and <i>j</i> .
8	Avg_Temp (K) Regions_<i>_to_<j>	Always	Mass-flow-weighted average temperature at the interface of the regions <i>i</i> and <i>j</i> .
9	Avg_tot_Temp (K) Regions_<i>_to_<j>	Always	Mass-flow-weighted average total temperature at the interface of regions <i>i</i> and <i>j</i> .
10	Avg_tot_Enth (J/kg) Regions_<i>_to_<j>	Always	Mass-flow-weighted average total enthalpy at the interface of regions <i>i</i> and <i>j</i> , calculated from the local total enthalpy given by $H = \int_{T_o}^T c_p dT + \sum_{k=1}^N \Delta H_{f,k}^o Y_k + (1/2) u_i u_i,$ where T_o is the temperature at standard state conditions, T is the local temperature, c_p is the mixture-averaged specific heat capacity at constant pressure, $\Delta H_{f,k}^o$ is the standard enthalpy of formation of the <i>k</i> -th species, Y_k is the mass fraction of species <i>k</i> , N is the total number of species, and u_i is the velocity.
11	Rate_<species name> (kg/s) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Mass flow rate for the specified species. Columns 14 and 15 are repeated for each species specified in regions_flow.in . Columns

Chapter 26: Output Files

3D Simulation Output Files mass_avg_regions_flow.out

Column	Header (units)	When CONVERGE includes this column	Description
			14 and 15 are also written for each INFLOW or OUTFLOW boundary.
12	Tot_<species name> (kg) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Total species mass passing through the interface of regions <i>i</i> and <i>j</i> .
13	Xfrac-<species name> Bound_id_<boundary ID>	When inputs.in > output_control > region_flow_flag = 2	Mole fraction for the specified gas species. This information is written only for each INFLOW and/or OUTFLOW boundary.
14	Rate_<passive name> (kg/s) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Mass flow rate for the specified passive from region <i>i</i> to region <i>j</i> . Columns 17 and 18 are repeated for each passive specified in regions_flow.in . Columns 17 and 18 are also written for each INFLOW or OUTFLOW boundary.
15	Tot_<passive name> (kg) Regions_<i>_to_<j>	When inputs.in > output_control > region_flow_flag = 2	Total passive mass passing through the interface of regions <i>i</i> and <i>j</i> .
16	Rate_<species name> (kg/s)	When inputs.in > output_control > region_flow_flag = 2	Mass flow rate for the parcel species from region <i>i</i> to region <i>j</i> . Columns 19 and 20 are repeated for each species specified in regions_flow.in . Columns 19 and 20 are also written for each INFLOW or OUTFLOW boundary.
17	Tot_<species name> (kg)	When inputs.in > output_control > region_flow_flag = 2	Total parcel species mass passing through the interface of regions <i>i</i> and <i>j</i> .
18	TKE (m^2/s^2)	When regions_flow.in > output_tke_flag = ON	Mass-flow-weighted average turbulent kinetic energy at the interface of regions <i>i</i> and <i>j</i> . This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
19	EPS (m^2/s^3)	When regions_flow.in > output_eps_flag = ON	Mass-flow-weighted average turbulent dissipation at the interface of regions <i>i</i> and <i>j</i> .
20	OMEGA (s^{-1})	When regions_flow.in >	Mass-flow-weighted average specific dissipation rate at the interface of regions <i>i</i> and <i>j</i> .

Chapter 26: Output Files

3D Simulation Output Files mass_avg_regions_flow.out

Column	Header (units)	When CONVERGE includes this column	Description
		<i>output_omega_flag = ON</i>	
21	Ca	When <i>regions_flow.in</i> > <i>output_ca_flag</i> = ON	Mass-flow-weighted average liquid area fraction at the interface of regions <i>i</i> and <i>j</i> . Set <i>regions_flow.in</i> > <i>output_ca_flag</i> = ON only when <u>volume of fluid modeling</u> is active.

mass_balance.out

CONVERGE generates *mass_balance.out* when a 3D simulation includes an INFLOW or OUTFLOW boundary (*i.e.*, when *boundary.in* > *boundary_conditions* > *boundary* > *type* = INFLOW or OUTFLOW). This file provides the mass flow rates for each INFLOW and OUTFLOW boundary, as well as the net mass flow rate for the domain. When *inputs.in* > *feature_control* > *parcel_mode* > *liquid_parcel* = 1, this file includes contributions from the parcel mass flow at each INFLOW and OUTFLOW boundary, nozzle injector, and boundary injector.

Positive values indicate flow out of the domain, while negative values indicate flow into the domain.

Table 26.44: Description of *mass_balance.out*.

Column	Header (units)	Description
1	Cycles (none)	Simulation cycle number (ncyc).
2	Cycle_Number (none) or Pseudo_Time (seconds)	Simulation cycle number (<i>i.e.</i> , number of time-steps) if <i>inputs.in</i> > <i>solver_control</i> > <i>steady_solver</i> = 0. The pseudo time-step if <i>steady_solver</i> = 1.
3*	Mass_Flow_Rate (kg/s)	Mass flow rate normal to the INFLOW/OUTFLOW boundary plane.
4†	Parcel_Mass_Flow_Rate (kg/s)	Parcel mass flow rate normal to the INFLOW/OUTFLOW boundary plane.
Varies†‡	Inj_Mass_Flow_Rate (kg/s)	Parcel mass flow rate from the injector.
Varies	Net_Mass_Flow_Rate_In (kg/s)	Net mass flow rate into the domain, including all INFLOW boundaries and injectors.
Varies	Net_Mass_Flow_Rate_Out (kg/s)	Net mass flow rate out of the domain, including all OUTFLOW boundaries.
Varies	Net_Mass_Flow_Rate (kg/s)	Net mass flow rate throughout the domain, including all INFLOW and OUTFLOW boundaries and all injectors.

* Repeated for each INFLOW and OUTFLOW boundary.

Chapter 26: Output Files

3D Simulation Output Files mass_balance.out

† Included only when [inputs.in](#) > feature_control > parcel_mode > liquid_parcel = 1.

‡ Repeated for each nozzle injector and boundary injector.

memory_usage.out

CONVERGE generates *memory_usage.out* for all 3D simulations. The output frequency is controlled by [inputs.in](#) > output_control > twrite_files. This file reports the total memory usage and memory usage per processor (or rank) at each cycle of a CONVERGE simulation. The *memory_usage.out* file records all memory usage data in megabytes (MB).

Starting with column 3, CONVERGE writes the memory usage of each processor used in the simulation. For example, if you run a simulation on three processors, CONVERGE will label columns 3 to 5 as *Rank0 (MB)*, *Rank1 (MB)*, and *Rank2 (MB)*, respectively.

Table 26.45: Description of *memory_usage.out*.

Column	Header (units)	Description
1	Cycle_Number (none)	Cycle number (ncyc).
2	Total_MEM (MB)	Total memory used by all processors.
3	Rank <i><i></i> (MB)	Memory used by CONVERGE on rank (processor) <i>i</i> .

mixing.out

CONVERGE generates *mixing.out* when the mixing output option is active (*i.e.*, when [inputs.in](#) > output_control > mixing_output_flag = 1). This file contains information on the equivalence ratio, relative air-fuel ratio, and reaction ratio. The equivalence ratio is given as

$$\phi = \frac{2 \sum_i N_i \eta_{C,i} + \frac{1}{2} \sum_i N_i \eta_{H,i}}{\sum_i N_i \eta_{O,i}}, \quad (26.16)$$

where N_i is the number of moles of species i and $\eta_{C,i}$, $\eta_{H,i}$ and $\eta_{O,i}$ are the number of carbon (C), hydrogen (H) and oxygen (O) atoms, respectively, for species i . The standard deviation for equivalence ratio is given by

$$\phi_{STD} = \sqrt{\frac{\sum_{cell} m_{cell} (\phi_{cell} - \phi_{mean})^2}{m_{total}}}, \quad (26.17)$$

where the subscript *cell* indicates the cell value, *mean* represents the mean value, *total* indicates the total value, *STD* indicates the standard deviation and the parameter *m* is the mass. The *mean* value is calculated by the following expression:

Chapter 26: Output Files

3D Simulation Output Files mixing.out

$$\phi = \frac{\sum_j 2 \sum_i N_i \eta_{C,i} + \frac{1}{2} \sum_i N_i \eta_{H,i}}{\sum_j \sum_i N_i \eta_{O,i}}, \quad (26.18)$$

where j is the total number of cells in the domain.

The relative air-fuel ratio is given as

$$\lambda = \phi^{-1}. \quad (26.19)$$

The standard deviation for air-fuel ratio is given by

$$\lambda_{STD} = \sqrt{\frac{\sum_{cell} m_{cell} (\lambda_{cell} - \lambda_{mean})^2}{m_{total}}}. \quad (26.20)$$

The reaction equivalence ratio is given as

$$\phi_R = \frac{2 \sum_i N_i \eta_{C,i} + \frac{1}{2} \sum_i N_i \eta_{H,i}}{\sum_i N_i \eta_{O,i}}, \quad (26.21)$$

where i is all species except H₂O and CO₂. The standard deviation for reaction equivalence ratio is given by

$$\phi_{R\ STD} = \sqrt{\frac{\sum_{cell} m_{cell} (\phi_{R\ cell} - \phi_{R\ mean})^2}{m_{total}}}. \quad (26.22)$$

The reaction relative air-fuel ratio is given as

$$\lambda_R = \phi_R^{-1}. \quad (26.23)$$

The standard deviation for reaction air-fuel ratio is given by

Chapter 26: Output Files

3D Simulation Output Files mixing.out

$$\lambda_{R\ STD} = \sqrt{\frac{\sum_{cell} m_{cell} (\lambda_{R\ cell} - \lambda_{R\ mean})^2}{m_{total}}}. \quad (26.24)$$

Note that ϕ_{mean} , λ_{mean} , and $\lambda_{R\ mean}$ are calculated by summing over all the cells as shown in Equation 26.18. These calculations are unchanged if an oxygenated fuel is used, and thus the results in *mixing.out* are unreliable for oxygenated fuels. Contact the Convergent Science Applications team if you would like to perform these calculations for oxygenated fuels.

Table 26.46: Description of *mixing.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Lambda_Mean	The average lambda (relative air-fuel ratio) value for the domain.
3	Lambda_StdDev	The standard deviation of lambda for the domain.
4	Phi_Mean	The average equivalence ratio value for the domain. Note that this quantity includes only gases.
5	Phi_StdDev	The standard deviation of equivalence ratio for the domain.
6	Rct_Lmda_Mean	The average lambda (relative air fuel ratio) value for the domain, which does not include CO ₂ and H ₂ O in the calculation.
7	Rct_Lmda_StdDev	The standard deviation of lambda for the domain, which does not include CO ₂ and H ₂ O in the calculation.
8	Rct_Phi_Mean	The average equivalence ratio value for the domain, which does not include CO ₂ and H ₂ O in the calculation.
9	Rct_Phi_StdDev	The standard deviation of equivalence ratio for the domain, which does not include CO ₂ and H ₂ O in the calculation.
10	Overall_Phi	The overall equivalence ratio for the domain. This quantity includes liquids (fuel, spray parcels, etc.) as well as gases.
11	Overall_Phi_StdDev	The standard deviation of the equivalence ratio for the domain. This quantity includes liquids (fuel, spray parcels, etc.) as well as gases.

monitor_line_<number>_<average type>_avg.out

CONVERGE generates *monitor_line_<number>_mass_avg.out* when the monitor point/line feature is active (i.e., when *inputs.in* > *output_control* > *monitor_points_flag* = 1) and the *monitor_points.in* file includes monitor lines (as opposed to monitor points). CONVERGE generates one *monitor_line_<number>_mass_avg.out* file for each monitor line that is designated as *mass_avg* in *monitor_points.in*.

Chapter 26: Output Files

3D Simulation Output Files monitor_line_<number>_<average type>_avg.out

CONVERGE generates *monitor_line_<number>_volume_avg.out* when the monitor point/line feature is active (*i.e.*, when [inputs.in](#) > *output_control* > *monitor_points_flag* = 1) and the [monitor_points.in](#) file includes monitor line (as opposed to monitor points). CONVERGE generates one *monitor_line_<number>_volume_avg.out* file for each monitor line that is designated as *vol_avg* in [monitor_points.in](#).

CONVERGE writes this file to the stream-specific output directory for the stream specified in [monitor_points.in](#) > *monitor_lines* > *line* > *end_points* > *point* > *stream_id*.

Table 26.47: Description of *monitor_line_<number>_<average type>_avg.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
Varies	Monitored variable	In <i>monitor_line_<number>_mass_avg.out</i> , the mass-averaged value of the monitored variable at the monitor point. In <i>monitor_line_<number>_volume_avg.out</i> , the volume-averaged value of the monitored variable at the monitor point. These lines repeat for each point on the monitor line.

monitor_line_<number>_coordinates.out

CONVERGE generates *monitor_line_<number>_coordinates.out* when the monitor point/line feature is active (*i.e.*, when [inputs.in](#) > *output_control* > *monitor_points_flag* = 1) and the [monitor_points.in](#) file includes monitor lines (as opposed to monitor points). CONVERGE generates one *monitor_line_<number>_coordinates.out* file for each monitor line.

CONVERGE writes this file to the stream-specific output directory for the stream specified in [monitor_points.in](#) > *monitor_lines* > *line* > *end_points* > *point* > *stream_id*.

Table 26.48: Description of *monitor_line_<number>_coordinates.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Point	Point number.
3-5	Point	The x, y, and z coordinate of this point on the monitor line.

monitor_point_<number>_<average type>_avg.out

CONVERGE generates *monitor_point_<number>_mass_avg.out* when the monitor point/line feature is active (*i.e.*, when [inputs.in](#) > *output_control* > *monitor_points_flag* = 1) and the [monitor_points.in](#) file includes monitor points (as opposed to monitor lines). CONVERGE generates one *monitor_point_<number>_mass_avg.out* file for each monitor point designated as *mass_avg* in [monitor_points.in](#).

Chapter 26: Output Files

3D Simulation Output Files monitor_point_<number>_<average type>_avg.out

CONVERGE generates *monitor_point_<number>_volume_avg.out* when the monitor point/line feature is active (*i.e.*, when *inputs.in* > *output_control* > *monitor_points_flag* = 1) and the *monitor_points.in* file includes monitor points (as opposed to monitor lines). CONVERGE generates one *monitor_point_<number>_volume_avg.out* file for each monitor point designated as *vol_avg* in *monitor_points.in*.

CONVERGE writes this file to the stream-specific output directory for the stream specified in *monitor_points.in* > *monitor_points* > *point* > *stream_id*.

Table 26.49: Description of *monitor_point_<number>_<average type>_avg.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
Varies	Monitored variable	In <i>monitor_point_<number>_mass_avg.out</i> , the mass-averaged value of the monitored variable at the monitor point. In <i>monitor_point_<number>_volume_avg.out</i> , the volume-averaged value of the monitored variable at the monitor point.

numerics.out

CONVERGE generates *numerics.out* when *inputs.in* > *output_control* > *numerics_output_flag* = 1 (default value). This file shows the total number of iterations per cycle (*ncyc*) for each equation.

Table 26.50: Description of *numerics.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Cycle_Number	Simulation cycle number (<i>ncyc</i>).
3*	PISO_Iter	Number of PISO iterations.
3†	SIMPLE_Iter	Number of SIMPLE iterations.
4	Pres_Iter	Number iterations for the pressure equation.
5	Ustar_Iter	Number iterations for the momentum equations.
6	Energy_Iter	Number of iterations for the energy equation.
7	Density_Iter	Number of iterations for the density equation.
8	Turbulence_Iter	Number of iterations for the turbulence equation(s).
9	Species_Iter	Number of iterations for the species equation(s).
10	Passive_Iter	Number of iterations for the passive equation(s).

* If *solver.in* > *ns_solver_scheme* = PISO.

† If *solver.in* > *ns_solver_scheme* = SIMPLE.

Chapter 26: Output Files

3D Simulation Output Files parcel_flow.out

parcel_flow.out

CONVERGE generates *parcel_flow.out* when a simulation includes discrete phase modeling. This file provides parcel flow rates and area-weighted averages for the interfaces between adjacent regions, [flow-through INTERFACE](#) boundaries, and/or [INFLOW and OUTFLOW boundaries](#).

Table 26.51: Description of *parcel_flow.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Cycle_Number or Pseudo_Time	Simulation cycle number (<i>i.e.</i> , number of time-steps) if <i>inputs.in</i> > <i>solver_control</i> > <i>steady_solver</i> = 0. The pseudo time-step if <i>steady_solver</i> = 1.
Varies	Rate_Liq_LIQPARCEL_0 (kg/s) Regions_<i>_to_<j> or <boundary name>	Total parcel mass flow rate through this boundary or between these regions.
Varies	Mass_Liq_LIQPARCEL_0 (kg) Regions_<i>_to_<j> or <boundary name>	Integrated total parcel mass flow through this boundary or between these regions.
Varies	NumP_Liq_LIQPARCEL_0 (number) Regions_<i>_to_<j> or <boundary name>	Integrated total number of parcels that have flowed through this boundary or between these regions.
Varies	NumD_Liq_LIQPARCEL_0 (number) Regions_<i>_to_<j> or <boundary name>	Integrated total number of drops that have flowed through this boundary or between these regions.
Varies	Rate_Liq_LIQPARCEL_<species name> (kg/s) Regions_<i>_to_<j> or <boundary name>	Parcel mass flow rate of this species through this boundary or between these regions.
Varies	Mass_Liq_LIQPARCEL_<species name> (kg) Regions_<i>_to_<j> or <boundary name>	Integrated parcel mass flow of this species through this boundary or between these regions.
Varies	NumP_Liq_LIQPARCEL_<species name> (number) Regions_<i>_to_<j> or <boundary name>	Integrated number of parcels of this species that have flowed through this boundary or between these regions.
Varies	NumD_Liq_LIQPARCEL_<species name> (number)	Integrated number of drops of this species that have flowed through this boundary or

Chapter 26: Output Files

3D Simulation Output Files parcel_flow.out

Column	Header (units)	Description
	Regions_<i>_to_<j> or <boundary name>	between these regions.

passive.out

CONVERGE generates *passive.out* for all 3D simulations. The output frequency is controlled by *inputs.in* > *output_control* > *twrite_files*. This file specifies the mass, mean, and standard deviation of each [passive](#) and non-transport passive. The first column in the output file gives the simulation time while the remaining columns (except for the last column) give the statistics of each passive and non-transport passive defined in [species.in](#). The final column lists the total mass in the region.

Table 26.52: Description of *passive.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
Varies [‡]	<passive name> (<i>varies</i>)	Total value of the passive.
Varies	<passive name>_Mean (<i>varies</i>)	Mean passive value in the domain.
Varies	<passive name>_StdDev (<i>varies</i>)	Standard deviation of the passive value in the domain.
Final [‡]	Total Mass (kg)	Total mass in the region.

[‡] The value in this column is the product of the simulated quantity and the value of *inputs.in* > *output_control* > *output_multiplier*.

phenom_soot_model.out

CONVERGE generates *phenom_soot_model.out* when a phenomenological soot model ([Gokul](#), [Dalian](#), or [Waseda](#)) is active (*i.e.*, if *emissions.in* > *phenom_soot_model* > *active* is non-zero). The file contains soot output data averaged over the computational domain for the time indicated.

Table 26.53: Description of *phenom_soot_model.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2 [‡]	Soot_Mass (kg)	Predicted soot in region from the phenomenological soot model.
3 [‡]	NumDensity (parts/m ³)	Soot number density, averaged in the computational domain.
4 [‡]	Incept_Mass (kg)	Soot mass due to inception, averaged in the computational domain.

Chapter 26: Output Files

3D Simulation Output Files phenom_soot_model.out

Column	Header (units)	Description
5 [‡]	SurfaGrowth_Mass (kg)	Soot mass due to surface growth, averaged in the computational domain.
6 [‡]	Oxid_Mass (kg)	Soot mass due to oxidation, averaged in the computational domain.
7 [‡]	Coag_Mass (kg)	Soot mass due to coagulation, averaged in the computational domain.

[‡] The value in this column is the product of the simulated quantity and the value of [*inputs.in*](#) > *output_control* > *output_multiplier*.

piston_profile<boundary ID>.out

CONVERGE generates *piston_profile<boundary ID>.out* when the option to write the [*piston position profile*](#) is active (*i.e.*, when [*boundary.in*](#) > *boundary_conditions* > *boundary* > *velocity* > *echo_profile* = 1 for the piston's velocity boundary condition). The <boundary ID> corresponds to the boundary ID value in [*boundary.in*](#) that corresponds to the piston.

Note that CONVERGE measures the piston displacement relative to the original location in the [*surface geometry file*](#).

Table 26.54: Description of *piston_profile<boundary ID>.out*.

Column	Header (units)	Description
1	Time (crank angle degrees)	Time in crank angle degrees.
2	X (meters)	X coordinate of the piston position.
3	Y (meters)	Y coordinate of the piston position.
4	Z (meters)	Z coordinate of the piston position.

planes_flow.out

CONVERGE generates *planes_flow.out* when the uniformity index output is active (*i.e.*, when [*inputs.in*](#) > *output_control* > *uniformity_index_output_flag* = 1). This file contains mean, minimum, maximum, and uniformity index information calculated across planes. Set up input parameters for the uniformity index output in [*uniformity_index.in*](#).

Table 26.55: Description of *planes_flow.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time (in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero) if <i>inputs.in</i> > <i>solver_control</i> > <i>steady_solver</i> = 0, Cycle number if <i>inputs.in</i> > <i>solver_control</i> > <i>steady_solver</i> = 1.
2	TIME	Cycle number.

Chapter 26: Output Files

3D Simulation Output Files planes_flow.out

Column	Header (units)	Description
Varies	<Norm type>_<variable name>; <slice name>	Value of the selected norm for this variable on this slice.
Varies	MEAN_<variable name>; <slice name>	Mean value for this variable on this slice.
Varies	MAX_<variable name>; <slice name>	Maximum value for this variable on this slice.
Varies	MIN_<variable name>; <slice name>	Minimum value for this variable on this slice.

progressive_cavity_rotor_profile<boundary ID>.out

CONVERGE generates *progressive_cavity_rotor_profile<boundary ID>.out* when a progressive cavity rotor profile is used for WALL boundary motion (*i.e.*, when [*motion_sets.in*](#) > *motion_object* > *mechanism* = PROGRESSIVE_CAVITY). The <boundary ID> corresponds to the boundary ID value in [*boundary.in*](#) that corresponds to the progressive cavity rotor.

Table 26.56: Description of the header in *progressive_cavity_rotor_profile<boundary ID>.out*.

Parameter	Description
TEMPORAL	Specifies a temporally varying file.
CYCLIC <period>	Specifies the cyclic period in <i>crank angle degrees</i> .
<Header>	Specifies the headers as shown in the following table.
<i>supplied_position</i>	Specifies the x, y, z coordinates that give the origin of the local coordinate system followed by two orthogonal vectors in a plane perpendicular to the rotor axis (<i>motion_sets.in</i> > <i>motion_object</i> > <i>rotor_axis</i>) to define the orientation of the boundary. CONVERGE will use the coordinate system origin as a reference to move the boundary.
<Data>	Motion profile recorded every 0.1 <i>crank angle degrees</i> .

Table 26.57: Description of *progressive_cavity_rotor_profile<boundary ID>.out* (after the header).

Column	Header (units)	Description
1	crank	Time in <i>crank angle degrees</i> .
2	x	Temporally varying origin of the local coordinate system. x and y are calculated in Equation 26.25 and z remains constant.
3	y	
4	z	
5	v1x	Initial direction of the x axis for the local coordinate system. Updated based on a rotation about the rotor axis.
6	v1y	
7	v1z	

Chapter 26: Output Files

3D Simulation Output Files progressive_cavity_rotor_profile<boundary ID>.out

Column	Header (units)	Description
8	v2x	Initial direction of the y axis for the local coordinate system. Updated based on a rotation about the rotor axis.
9	v2y	
10	v2z	

Using the initial *supplied_position*, CONVERGE calculates the progressive cavity hypocycloidal motion by

$$\begin{aligned}x(\theta) &= r(L-1)\cos\theta + r\cos((L-1)\theta) \\y(\theta) &= r(L-1)\sin\theta - r\sin((L-1)\theta)\end{aligned}\quad (26.25)$$

where r is the rotor radius (*motion_sets.in* > *motion_object* > *rotor_radius*), L is the number of lobes (*motion_object* > *number_of_lobes*), and θ is the time in *crank angle degrees*. Equation 26.25 is used to update the temporally varying origin of the local coordinate system to generate the *progressive_cavity_rotor_profile<boundary ID>.out* file.

To use this file as an *arbitrary motion* profile in another simulation, change the file extension from *.out* to *.in* and include it in *boundary.in* > *boundary_conditions* > *boundary* > *velocity* > *value*.

radius_pdf.out

CONVERGE generates *radius_pdf.out* when the parcel radius distribution output is active (i.e., when *inputs.in* > *output_control* > *smd_pdf_flag* = 3). The *radius_pdf.out* file contains information about the parcel radius distribution in the region or box specified in *radius_pdf.in*.

Table 26.58: Description of *radius_pdf.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
Varies	(radius range in <i>m</i>) num_parcels	Number of parcels with radii in the range that defines this bin.

react_ratio_bin.out

CONVERGE generates *react_ratio_bin.out* when the mixing output option is active (i.e., when *inputs.in* > *output_control* > *mixing_output_flag* = 1). The output frequency is controlled by *inputs.in* > *output_control* > *twrite_files*. This file summarizes the fraction of the total mass which has a reaction ratio value within the specified range of each bin. The mass within the domain is binned by reaction ratio into twenty intervals, starting at 0.0-0.1 and ending at 1.9-2.0.

Chapter 26: Output Files

3D Simulation Output Files react_ratio_bin.out

Table 26.59: Description of *react_ratio_bin.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	0.0 - 0.1	Mass fraction that has a reaction ratio from 0.0 to 0.1.
3	0.1 - 0.2	Mass fraction that has a reaction ratio from 0.1 to 0.2.
4	0.2 - 0.3	Mass fraction that has a reaction ratio from 0.2 to 0.3.
5	0.3 - 0.4	Mass fraction that has a reaction ratio from 0.3 to 0.4.
6	0.4 - 0.5	Mass fraction that has a reaction ratio from 0.4 to 0.5.
7	0.5 - 0.6	Mass fraction that has a reaction ratio from 0.5 to 0.6.
8	0.6 - 0.7	Mass fraction that has a reaction ratio from 0.6 to 0.7.
9	0.7 - 0.8	Mass fraction that has a reaction ratio from 0.7 to 0.8.
10	0.8 - 0.9	Mass fraction that has a reaction ratio from 0.8 to 0.9.
11	0.9 - 1.0	Mass fraction that has a reaction ratio from 0.9 to 1.0.
12	1.0 - 1.1	Mass fraction that has a reaction ratio from 1.0 to 1.1.
13	1.1 - 1.2	Mass fraction that has a reaction ratio from 1.1 to 1.2.
14	1.2 - 1.3	Mass fraction that has a reaction ratio from 1.2 to 1.3.
15	1.3 - 1.4	Mass fraction that has a reaction ratio from 1.3 to 1.4.
16	1.4 - 1.5	Mass fraction that has a reaction ratio from 1.4 to 1.5.
17	1.5 - 1.6	Mass fraction that has a reaction ratio from 1.5 to 1.6.
18	1.6 - 1.7	Mass fraction that has a reaction ratio from 1.6 to 1.7.
19	1.7 - 1.8	Mass fraction that has a reaction ratio from 1.7 to 1.8.
20	1.8 - 1.9	Mass fraction that has a reaction ratio from 1.8 to 1.9.
21	1.9 - 2.0	Mass fraction that has a reaction ratio from 1.9 to 2.0.

residuals.out

CONVERGE generates residuals.out when *solver.in* > *Steady_state_solver* > *steady_residual_output_flag* = 1. This file contains the relative residuals for the mass, momentum, energy, and species equations at each solver cycle.

Table 26.60: Description of *residuals.out*.

Column	Header (units)	Description
1	Cycles (None)	Cycle number.
2	Mass (None)	Relative residual for the mass transport equation.

Chapter 26: Output Files

3D Simulation Output Files residuals.out

Column	Header (units)	Description
3	X-Momentum (None)	Relative residual for the transport equation for the x component of momentum.
4	Y-Momentum (None)	Relative residual for the transport equation for the y component of momentum.
5	Z-Momentum (None)	Relative residual for the transport equation for the z component of momentum.
6	Energy	Relative residual for the energy transport equation.
Varies	<Species name> (None)	Relative residual for the transport equation for the specified species.
Varies	<Turbulence variable> (None)	Relative residual for the transport equation for this turbulence transport equation.

rot_object<number>-wall_power_torque.out

CONVERGE generates *rot_object<number>-wall_power_torque.out* when wall power torque is specified for the wall output option (*i.e.*, when the *write_wall_power_torque* settings block is included in [wall_output.in](#)). A separate *rot_object<number>-wall_power_torque.out* file (where the <number> corresponds to the rotation object's position in the list of rotation objects in [wall_output.in](#)) is written for each rotation object specified in the *write_wall_power_torque* settings block of *wall_output.in*.

For rotating or arbitrary-motion WALL boundaries, torques are calculated about the rotation axis. For all other WALL boundaries, torques are calculated about the origin.

Table 26.61: Description of *rot_object<number>-wall_power_torque.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2‡	Pres_Torque_X (N-m)	Total pressure torque about x axis exerted on the rotation object.
3‡	Pres_Torque_Y (N-m)	Total pressure torque about y axis exerted on the rotation object.
4‡	Pres_Torque_Z (N-m)	Total pressure torque about z axis exerted on the rotation object.
5‡	Visc_Torque_X (N-m)	Total viscous torque about x axis exerted on the rotation object.
6‡	Visc_Torque_Y (N-m)	Total viscous torque about y axis exerted on the rotation object.
7‡	Visc_Torque_Z (N-m)	Total viscous torque about z axis exerted on the rotation object.

Chapter 26: Output Files

3D Simulation Output Files rot_object<number>-wall_power_torque.out

Column	Header (units)	Description
8 [‡]	Pres_Rot_Power (W)	Rotational power due to pressure torque. Calculated as the dot product of the angular velocity (identical direction to the boundary rotation axis) and torque.
9 [‡]	Visc_Rot_Power (W)	Rotational power due to viscous torque. Calculated as the dot product of the angular velocity (identical direction to the boundary rotation axis) and torque.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > *output_control* > *output_multiplier*.

scalar_diss_rate.out

CONVERGE generates *scalar_diss_rate.out* when the [RIF model](#) is active (*i.e.*, when [combust.in](#) > *rif_model* > *active* = 1). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. The file contains the scalar dissipation rate for each flamelet.

Table 26.62: Description of *scalar_diss_rate.out*.

Column	Header (units)	Description
1	Time (seconds or crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
Varies	Flamelet<flamelet ID> (s ¹)	Scalar dissipation rate. This column is repeated for each flamelet.

skip_species.out

CONVERGE generates *skip_species.out* when [skip species](#) is active (*i.e.*, when [inputs.in](#) > *feature_control* > *skip_species_flag* = 1). This file contains information about each iteration of the skip species process. The first line contains the simulation time and the skip species start and end times. This information comes directly from the [skip_species.in](#) file and is included so that you can verify the skip species start and end times. On the next line, CONVERGE writes the number of gas and liquid species that are included in the chemistry computations (*i.e.*, the number of species that were not skipped) and the total number of fluid species in the simulation. The final line lists the species that were not skipped.

After the above information, CONVERGE prints a block of text when the skip species process ends. The first line of this block contains the simulation time and repeats the skip species species start and end times. The next line contains a message that the skip species computations are ending and that CONVERGE will return to solving all fluid species.

If skip species is CYCLIC, you will see multiple blocks of information.

Chapter 26: Output Files

3D Simulation Output Files skip_species.out

```
Simulation Time: -4.0999100e+02; Skip species start time: 1.3500000e+02, end time  
7.0000000e+02 (DEG)  
Starting skip species session. Computing 5 gas species, and 0  
liquid species out of total 48 fluid species  
Here is the list of the species:  
IC8H18 O2 N2 CO2 H2O  
  
Simulation Time: -1.9566372e+01; Skip species reduction start time: 1.3500000e+02, end  
time 7.0000000e+02 (DEG)  
Ending skip species session. Computing all species, i.e. 48 total fluid  
species  
.
```

Figure 26.2: An example *skip_species.out* file.

smd_pdf.out

CONVERGE generates *smd_pdf.out* when the Sauter Mean Diameter (SMD) distribution output is active (*i.e.*, when [inputs.in](#) > *output_control* > *smd_pdf_flag* = 1). This file contains spray information calculated across a plane. Set up input parameters for the SMD distribution output in [smd_pdf.in](#).

Table 26.63: Description of *smd_pdf.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	If inputs.in > <i>solver_control</i> > <i>steady_solver</i> = 0, time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero. If inputs.in > <i>solver_control</i> > <i>steady_solver</i> = 1, cycle number.
2	Pressure (Pa)	Average pressure across the calculation plane.
3	Max_Pres (Pa)	Maximum pressure across the calculation plane.
4	Min_Pres (Pa)	Minimum pressure across the calculation plane.
5	Volume (m ³)	Volume of the calculation region.
6	tke (m ² /s ²)	Turbulent kinetic energy of the calculation region. This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
7	SMD_Plane (m)	SMD for all of the parcels in the calculation region.
8	Total_mass (kg)	Total parcel mass in the calculation region.
Varies	(size) Mass_fraction	Fraction of the total parcel mass which is contained in this bin.

solid_parcel.out

CONVERGE generates *solid_parcel.out* when solid [parcel_modeling](#) is active (*i.e.*, when [inputs.in](#) > *feature_control* > *parcel_mode* > *solid_parcel* = 1). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the injection results and the spray penetration for each nozzle.

Chapter 26: Output Files

3D Simulation Output Files solid.Parcel.out

Table 26.64: Description of solid.Parcel.out.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	solid_parcel (drop)	Number of solid parcels.
3 [‡]	solid.Parcel_Mass (kg)	Total solid parcel mass in the domain.
4 [‡]	<parcel_species> (kg)	Mass of the solid species. If there is more than one solid species in the simulation, there will be one column per species.
5	smd (m)	Sauter mean diameter of spray.
6	DV10 (m)	The solid parcel diameter representing the tenth percentile by volume.
7	DV50 (m)	The solid parcel diameter representing the fiftieth percentile by volume.
8	DV90 (m)	The solid parcel diameter representing the ninetieth percentile by volume.
9	Inj_No	Number of the current injector.
10 [‡]	Inj_Mass (kg)	Amount of injected mass for the current injector.
11 ^{‡*}	Tot_Inj_Mass (kg)	Mass that has been injected by the specified injector since the beginning of the simulation.
12 [^]	Inj_Vel (m/s)	Injection velocity of the current injector.
12*	Inj_Vel_Old (m/s)	Injection velocity for the current injector before the contraction coefficient is applied.
13*	- Inj_Vel_New (m/s)	Injection velocity for the current injector after the contraction coefficient is applied.
14*	- C_a	Contraction coefficient for the current injector.
15*	13 [^] Inj_Press (MPa)	Injection pressure for the current injector.
16*	14 [^] Noz_No	Number of the current nozzle.
17*	15 [^] Spray_Penet (m)	Solid penetration length for the current nozzle.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > output_control > output_multiplier.

* [parcel_introduction.in](#) > injections > injection > injection_control > discharge_coeff_flag = 1 or 2.

[^] [parcel_introduction.in](#) > injections > injection > injection_control > discharge_coeff_flag = 0.

solid.Parcel_ecn.out

CONVERGE generates solid.Parcel_ecn.out when solid [parcel modeling](#) is active (*i.e.*, when [inputs.in](#) > feature_control > parcel_mode > solid.Parcel = 1). The output frequency for this file is controlled by [inputs.in](#) > output_control > twrite_files. This file lists the solid penetration based

Chapter 26: Output Files

3D Simulation Output Files solid.Parcel_ecn.out

on various percentages of the solid mass and includes vapor penetration information based on the ECN (Engine Combustion Network) definition. The output frequency for this file is controlled by [*inputs.in*](#) > *output_control* > *twrite_files*. Columns 2-8 are repeated for each nozzle on each injector.

Table 26.65: Description of *solid.Parcel_ecn.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Inj_No	Number of the current injector.
3	Noz_No	Number of the current nozzle.
4	Spray_Penet90 (m)	Penetration distance from the nozzle exit for the current nozzle based on 90% of the solid mass.
5	Spray_Penet95 (m)	Penetration distance from the nozzle exit for the current nozzle based on 95% of the solid mass.
6	Spray_Penet97 (m)	Penetration distance from the nozzle exit for the current nozzle based on 97% of the solid mass.
7	Spray_Penet99 (m)	Penetration distance from the nozzle exit for the current nozzle based on 99% of the solid mass.
8	Vapor_Penet (m)	Penetration distance from the nozzle exit for the current nozzle based on <i>parcel_introduction.in</i> > <i>injections</i> > <i>injection</i> > <i>penet_control</i> > <i>vapor_frac</i> .

soot_hiroy.out

CONVERGE generates *soot_hiroy.out* when the [*Hiroyasu-NSC soot model*](#) is active (*i.e.*, when [*emissions.in*](#) > *hiroy_soot_model* > *active* = 1). This file contains the Hiroyasu-NSC soot as a function of time as well as the formation and oxidation terms.

Table 26.66: Description of *soot_hiroy.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2 [‡]	Hiroy_Soot (kg)	Predicted soot in region from the Hiroyasu soot model.
3 [‡]	Hiroy_Form (kg)	Predicted time-integrated formation of soot in region from the Hiroyasu soot model.
4 [‡]	Hiroy_Oxid (kg)	Predicted time-integrated oxidation of soot in region from the Hiroyasu soot model.

[‡] The value in this column is the product of the simulated quantity and the value of [*inputs.in*](#) > *output_control* > *output_multiplier*.

Chapter 26: Output Files

3D Simulation Output Files soot_pm_model.out

soot_pm_model.out

CONVERGE generates *soot_pm_model.out* when the [Particulate Mimic \(PM\) model](#) is active (*i.e.*, when [*emissions.in*](#) > *detailed_soot_model* > *active* = PM). The file contains PM model soot data that is averaged over the computational domain.

The column numbers in this file vary depending on the value of [*emissions.in*](#) > *detailed_soot_model* > *pm_num_moments*. The column numbers in the table below assume that *pm_num_moments* = 2.

Table 26.67: Description of *soot_pm_model.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2 [‡]	Soot_Mass (kg)	Predicted soot mass from the PM soot model, calculated as the sum of soot mass due to each of the soot formation steps (columns 9-14). Soot_Mass = PiMass + SgMass - FrMass - OxMass + ConMass + CoagMass.
3	NumDensity (parts/m ³)	Soot number density, averaged in the computational domain.
4	VolFrac (m ³ [soot]/m ³ [gas])	The fraction of the volume occupied by soot (m ³) in the gas (m ³)
5	TotSurf (m ² [soot]/m ³ [gas])	Soot surface area per unit gas volume, averaged in the computational domain.
6	Ave_Dia (m)	Mean soot diameter in the computational domain.
7	Dispersion (m/m)	Width of the soot particle size distribution, averaged in the computational domain. (Available only when the second moment is calculated.)
8	Variance	Variance of the soot particle size distribution function, averaged in the computational domain. (Available only when the second moment is calculated.)
9 [‡]	PiMass (kg)	Soot mass due to particle inception (nucleation), averaged in the computational domain.
10 [‡]	SgMass (kg)	Soot mass due to surface growth, averaged in the computational domain.
11 [‡]	FrMass (kg)	Soot mass due to fragmentation, averaged in the computational domain.
12 [‡]	OxMass (kg)	Soot mass due to oxidation, averaged in the computational domain.
13 [‡]	ConMass (kg)	Soot mass due to condensation, averaged in the computational domain.

Chapter 26: Output Files

3D Simulation Output Files soot_pm_model.out

Column	Header (units)	Description
14 [‡]	CoagMass (kg)	Soot mass due to coagulation, averaged in the computational domain.
15	MSoot0 (mole/kg)	Zeroth soot moment (<i>mole</i> of soot per <i>kg</i> of gas), averaged in the computational domain.
16	MSoot1 (mole/kg)	First soot moment (<i>mole</i> of soot per <i>kg</i> of gas), averaged in the computational domain.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > *output_control* > *output_multiplier*.

soot_psm_model.out

CONVERGE writes *soot_psm_model.out* when the [Particulate Size Mimic \(PSM\) model](#) is active (*i.e.*, when [emissions.in](#) > *detailed_soot_model* > *active* = PSM). The file contains PSM model soot data that is averaged over the computational domain.

The column numbers in this file vary depending on the value of [emissions.in](#) > *detailed_soot_model* > *psm_num_sections*.

Table 26.68: Description of soot_psm_model.out.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2 [‡]	Soot_Mass (kg)	Predicted soot mass from the PSM soot model, calculated as the sum of soot mass due to each of the soot formation steps (columns 9-14).
3	NumDensity (parts/m ³)	Soot number density, averaged in the computational domain.
4	VolFrac (m ³ [soot]/m ³ [gas])	Soot volume fraction, averaged in the computational domain.
5 [‡]	PiMass (kg)	Soot mass due to particle inception (nucleation), averaged in the computational domain.
6 [‡]	SgMass (kg)	Soot mass due to surface growth, averaged in the computational domain.
7 [‡]	FrMass (kg)	Soot mass due to fragmentation, averaged in the computational domain.
8 [‡]	OxMass (kg)	Soot mass due to oxidation, averaged in the computational domain.
9 [‡]	ConMass (kg)	Soot mass due to condensation, averaged in the computational domain.
10 [‡]	CoagMass (kg)	Soot mass due to coagulation, averaged in the computational domain.

Chapter 26: Output Files

3D Simulation Output Files soot_psm_model.out

Column	Header (units)	Description
11	YSoot#	Soot mass fraction, averaged in each section (#) the computational domain.
12	PSDF# (m^{-3})	Soot particle size distribution function in each section (#) of the computational domain, calculated as $d(NumDensity)/d \log(\text{soot particle diameter})$.
13	SSize# (nm)	Soot particle diameter in each section (#) of the computational domain.

[‡]The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > *output_control* > *output_multiplier*.

source_energy.out

CONVERGE writes *source_energy.out* when [source modeling](#) is active (*i.e.*, when [inputs.in](#) > *property_control* > *source_flag* = 1). The file contains heat release data for each source using the name provided in [source.in](#) > *source* > *description*. The <number> in the column name represents the source index (*i.e.*, the order in which the sources are listed in *source.in*).

Table 26.69: Description of *source_energy.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time (in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero).
2*	<source description>_<number> Integrated_HR (J)	Integrated heat release data for the source.
3*	<source description>_<number> HR_Rate (J/time)	Heat release rate data for the source.

* This column is repeated for each source in [source.in](#).

Chapter 26: Output Files

3D Simulation Output Files species_mass.out

species_mass.out

CONVERGE generates *species_mass.out* when the option to write species total mass information is active (*i.e.*, when [inputs.in](#) > *output_control* > *species_output_flag* is non-zero). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the total mass of each species specified in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

The order of columns 2+ is the same as the order in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

Table 26.70: Description of *species_mass.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
Varies [‡]	Species name (kg)	Total species mass. This column is repeated for every species as defined by inputs.in > <i>output_control</i> > <i>species_output_flag</i> .

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > *output_control* > *output_multiplier*.

species_mass_frac.out

CONVERGE generates *species_mass_frac.out* when the option to write species mass fraction information is active (*i.e.*, when [inputs.in](#) > *output_control* > *species_output_flag* = 2, 3, or 4). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the mass fraction of each species specified in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

The order of columns 2+ is the same as the order in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

Table 26.71: Description of *species_mass_frac.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Species name (dimensionless: kg of this species per kg of all species)	Mass fraction of the species. This column is repeated for every species as defined by inputs.in > <i>output_control</i> > <i>species_output_flag</i> .

Chapter 26: Output Files

3D Simulation Output Files species_mass_src.out

species_mass_src.out

CONVERGE generates *species_mass_src.out* when the option to write species volume-integrated source terms is active (*i.e.*, when [inputs.in](#) > *output_control* > *species_output_flag* = 5). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the total mass of each species specified in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#). The species mass is calculated based on the volume integral of the species source term, which can stabilize much faster than species flux at the boundary for case setups with long flow through times.

The order of columns 2+ is the same as the order in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

Table 26.72: Description of *species_mass_src.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
Varies [‡]	Species name (kg)	Total species mass. This column is repeated for every species as defined by inputs.in > <i>output_control</i> > <i>species_output_flag</i> .

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > *output_control* > *output_multiplier*.

species_mole_frac.out

CONVERGE generates *species_mole_frac.out* when the option to write species mole fraction information is active (*i.e.*, when [inputs.in](#) > *output_control* > *species_output_flag* = 4). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the mole fraction of each species specified in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

The order of columns 2+ is the same as the order in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

Table 26.73: Description of *species_mole_frac.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Species name (dimensionless: mole of this species per mole of all species)	Mole fraction of the species. This column is repeated for every species as defined by inputs.in > <i>output_control</i> > <i>species_output_flag</i> .

Chapter 26: Output Files

3D Simulation Output Files species_std_masfrac.out

species_std_masfrac.out

CONVERGE generates *species_std_masfrac.out* when the option to write species standard deviation of mass fraction is active (*i.e.*, when [inputs.in](#) > *output_control* > *species_output_flag* = 3 or 4). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. This file summarizes the standard deviation of the mass fraction of each species specified in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

The order of columns 2+ is the same as the order in the *species* section of [mech.dat](#), [species.in](#), and [species_output.in](#).

Table 26.74: Description of *species_std_masfrac.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Species name	Standard deviation of the mass fraction of the species. This column is repeated for every species as defined by inputs.in > <i>output_control</i> > <i>species_output_flag</i> .

species_vol.out

CONVERGE generates *species_vol.out* and *species_vol_region<region ID>.out* when [volume of fluid \(VOF\) modeling](#) is active (*i.e.*, when [inputs.in](#) > *feature_control* > *vof_flag* = 1) and when species mass information is written (*i.e.*, when [inputs.in](#) > *output_control* > *species_output_flag* is non-zero or [inputs.in](#) > *output_control* > *species_output_flag* is set to [species_output.in](#) and then the *species_output.in* file includes <*total_vol*>). The output frequency is controlled by [inputs.in](#) > *output_control* > *twrite_files*. The *species_vol.out* file contains species volume data for a VOF simulation. The *species_vol_region<region ID>.out* files contain region-specific species volume data for a VOF simulation.

Table 26.75: Description of *species_vol.out* and *species_vol_region<region ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2 [‡]	Total Volume (m^3)	Total volume of fluid.
Varies [‡]	<Species name> (m^3)	Total volume of the specified species calculated according to Dalton's Law.
Varies [‡]	<Species name>_partial (m^3)	Total volume of the specified species calculated according to Amagat's Law.

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > *output_control* > *output_multiplier*.

Chapter 26: Output Files

3D Simulation Output Files spray_rate_inj<injector ID>.out

spray_rate_inj<injector ID>.out

CONVERGE generates *spray_rate_inj<injector ID>.out* at the start of a simulation when at least one injector is defined in [parcel_introduction.in](#). A separate *spray_rate_inj<injector ID>.out* file is written for each injector. You can use this file to verify the rate-shape information.

Table 26.76: Description of *spray_rate_inj<injector ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Index	Index corresponds to rate-shape entries defined in parcel_introduction.in .
3	Mass_Inject (kg)	Amount of mass injected through the injector.
- 4 [†]	Inj_Vel (m/s)	Injection velocity for the current injector.
- 5 [†]	Inj_Pres (MPa)	Injection pressure for current injector.
4*	- Inj_Vel_Old (m/s)	Injection velocity for the current injector before the contraction coefficient is applied.
5*	- Inj_Vel_New (m/s)	Injection velocity for the current injector after the contraction coefficient is applied.
6*	- Inj_Pres (MPa)	Injection pressure for current injector.
7*	- C_a	Contraction coefficient for current injector.

* [parcel_introduction.in](#) > *injections* > *injection* > *injection_control* > *discharge_coeff_flag* = 1 or 2
† [parcel_introduction.in](#) > *injections* > *injection* > *injection_control* > *discharge_coeff_flag* = 0

The total area of the nozzles for an injector is given by the product of the number of nozzles and the area of each nozzle, as follows:

$$A_{nozzles} = num_noz \cdot \frac{\pi}{4} diam_noz^2, \quad (26.26)$$

where [parcel_introduction.in](#) > *injections* > *injection* > *nozzles* > *nozzle* > *geometry* > *diameter* gives the diameter of each nozzle.

If the rate-shape entries are given by $\psi_i^{rateshape}$ at each interval, then the mass of injection during each interval ($dm_i^{rateshape}$) is given by

Chapter 26: Output Files

3D Simulation Output Files spray_rate_inj<injector ID>.out

$$dm_i^{rateshape} = \left(\frac{\psi_i^{rateshape} + \psi_{i-1}^{rateshape}}{2} \right) \cdot dt_i^{rateshape} \cdot A_{nozzles} \cdot \rho_l, \quad (26.27)$$

where $dt_i^{rateshape}$ is the time interval between two rate-shape entries and ρ_l is the density of the liquid spray.

Note that this calculation assumes that the units of $\psi_i^{rateshape}$ are velocity (m/s). However, only the shape of the input rate-shape (not the magnitude of each entry) is important. To convert them to true velocities, CONVERGE calculates the scaling factor $vel_{scale_rateshape}$ for the rate-shape entries:

$$vel_{scale_rateshape} = \frac{tot_mass}{\sum_{i=1}^{numvel_inject} dm_i^{rateshape}}, \quad (26.28)$$

where the total mass of injection is given by [parcel_introduction.in > injections > injection > injection_control > tot_mass](#). The velocity in *spray_rate_inj<injector ID>.out* is represented by *inj_vel_old*. CONVERGE converts the old velocities to true velocities, as follows:

$$inj_vel_old = vel_{scale_rateshape} \cdot \psi_i^{rateshape}. \quad (26.29)$$

CONVERGE calculates the injection pressure (*inj_pres*) as

$$inj_pres = \frac{1}{2} \rho_l \left(\frac{inj_vel_old}{C_d} \right)^2, \quad (26.30)$$

where C_d is the discharge coefficient. CONVERGE evaluates the new injection velocity (*inj_vel_new*) as

Chapter 26: Output Files

3D Simulation Output Files spray_rate_inj<injector ID>.out

$$inj_vel_new = \frac{inj_vel_old}{C_a}, \quad (26.31)$$

where C_a is the contraction coefficient.

spray_urea_file.out

CONVERGE generates *spray_urea_file.out* when the [detailed decomposition urea approach](#) is active (*i.e.*, when *urea.in* > *urea_model* = 3).

Table 26.77: Description of *spray_urea_file.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	Water (kg)	Mass of water in spray.
3	Urea_tot (kg)	Total mass of urea in spray.
4	Urea_sol (kg)	Mass of solid urea in spray.
5	Urea_aq (kg)	Mass of aqueous urea in spray.
6	Biuret (kg)	Mass of biuret in spray.
7	CYA (kg)	Mass of cyanuric acid in spray.
8	Ammelide (kg)	Mass of ammelide in spray.
9	NH4plus (kg)	Mass of NH4+ in spray.
10	NCOminus (kg)	Mass of NCO- in spray.
11	Hplus (kg)	Mass of H+ in spray.

steady_state.out

CONVERGE generates *steady_state.out* when the [steady-state monitor](#) is active (*i.e.*, when *inputs.in* > *solver_control* > *monitor_steady_state* = 1).

Columns 3 through 5 are repeated for each variable specified in [monitor_steady_state.in](#). Note that CONVERGE writes the last six rows of *steady_state.out* only if you are using automatic solution monitoring (*solver.in* > *Steady_state_solver* > *steady_auto_flag* = 1).

Chapter 26: Output Files

3D Simulation Output Files steady_state.out

Table 26.78: Description of *steady_state.out*.

Column	Header	Description
1	Time or Cycles	If inputs.in > solver_control > steady_solver = 0, time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero. If inputs.in > solver_control > steady_solver = 1, cycle number.
2	Cycle Number or Pseudo_Time	Simulation cycle number (<i>i.e.</i> , number of time-steps) if inputs.in > solver_control > steady_solver = 0. The pseudo-time elapsed if steady_solver = 1.
Varies	Current Value <variable name specified in monitor_steady_state.in>	Value of the specified variable at this time-step.
Varies	Mean <variable name specified in monitor_steady_state.in>	Mean of the second sample set . CONVERGE writes 0 until the first and second sample sets have been populated with data.
Varies	Std <variable name specified in monitor_steady_state.in>	Standard deviation of the second sample set . CONVERGE writes 0 until the first and second sample sets have been populated with data.
Varies	Diff_mean <variable name specified in monitor_steady_state.in>	Difference between the means of the first and second sample set .
Varies	Std_dev <variable name specified in monitor_steady_state.in>	Difference between the standard deviations of the first and second sample set .
Varies	grid_scale	Current value of grid scale.
Varies	piso_tol or simple_tol	Current value of the iterative algorithm (<i>e.g.</i> , PISO) tolerance.
Varies	tol_scale	Current value of tol_scale.
Varies	max_cfl_u	Current value of the convective CFL number.
Varies	omega_presrat	Current value of the pressure over-relaxation ratio.
Varies	pres_solver_type	Current value of the pressure solver type.

supercycle_point<number>.out

CONVERGE generates *supercycle_point<number>.out* when [super-cycling](#) is active (*i.e.*, when [inputs.in](#) > feature_control > cht_supercycle_flag = 1 and the supercycle_points settings block is included in [supercycle.in](#)). CONVERGE writes solid temperature data for each monitor point to *supercycle_point<number>.out*, where <number> is the monitor point number as specified in [supercycle.in](#).

Table 26.79: Description of *supercycle_point<number>.out*.

Chapter 26: Output Files

3D Simulation Output Files supercycle_point<number>.out

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Temperature (K)	Solid temperature at monitor point.

supercycle_stream<number>_balance.out

CONVERGE generates *supercycle_stream<number>_balance.out* when [super-cycling](#) is active (*i.e.*, when [inputs.in](#) > feature_control > cht_supercycle_flag = 1). This file is written at the conclusion of each super-cycle and contains energy balance data. CONVERGE creates one file for each solid stream.

The first and last columns are always time and the sum of the fluxes, respectively. The other columns will vary depending on the system.

Table 26.80: Description of *supercycle_stream<number>_balance.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
for a WALL boundary or a solid-solid INTERFACE boundary		
Varies	bound_id	Boundary ID (of a WALL boundary).
Varies	flux (J/s)	Flux.
for a fluid-solid INTERFACE boundary		
Varies	bound_id	Boundary ID (of an INTERFACE boundary).
Varies	flux (J/s)	Flux.
Varies	surf_temp_inner (K)	Surface temperature of the boundary assigned to the specified stream (one side of the INTERFACE).
Varies	surf_temp_outer (K)	Surface temperature of the boundary on the other side of the INTERFACE.
Final total (J/s)		
Final	total (J/s)	Sum of all fluxes.

surface_species_cov.out

CONVERGE generates *surface_species_cov.out* when [surface chemistry modeling](#) is active (*i.e.*, when [inputs.in](#) > feature_control > surface_chemistry_flag = 1). This file summarizes the species coverages of each surface species listed in [surface_mech.dat](#).

If the number of porous regions specified in [surface_chemistry.in](#) is greater than 1, CONVERGE will instead write a separate *surface_species_cov_region<region ID>.out* file for each specified region number. CONVERGE will not generate an averaged *surface_species_cov.out* if multiple regions exist.

Chapter 26: Output Files

3D Simulation Output Files surface_species_cov.out

Table 26.81: Description of *surface_species_cov.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
Varies	Surface species name (dimensionless: mole/cm ² of this species per mole/cm ² of all species)	Coverage of the surface species. This column is repeated for every surface species and lists the species in the same order as in the SITE section of <i>surface_mech.dat</i> .

temperature.out

CONVERGE generates *temperature.out* for all 3D simulations. The output frequency is controlled by *inputs.in* > output_control > twrite_files. This file summarizes the fraction of the domain above a specified temperature. There are four fixed temperatures used as criterion for these files: 2500 K, 2600 K, 2700 K, and 2800 K.

Table 26.82: Description of *temperature.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Mass_Frac > 2500 K	Mass fraction of the region above 2500 K.
3	Mass_Frac > 2600 K	Mass fraction of the region above 2600 K.
4	Mass_Frac > 2700 K	Mass fraction of the region above 2700 K.
5	Mass_Frac > 2800 K	Mass fraction of the region above 2800 K.

thermo.out

CONVERGE generates *thermo.out* for all 3D simulations. The format of this file is described in [a later section](#).

time.out

CONVERGE generates *time.out* for all 3D simulations. CONVERGE writes information to *time.out* at each time-step (*i.e.*, the value of *inputs.in* > output_control > twrite_files has no effect on *time.out*). This file contains some of the time-related information that is also written to the [log file](#).

Because it is possible to have different time-step sizes in different streams, CONVERGE writes a separate *time.out* file for each stream to the output directory for that stream (*e.g.*, */outputs_original/stream0/*).

Table 26.83: Description of *time.out*.

Chapter 26: Output Files

3D Simulation Output Files time.out

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero.
2	Cycle_Number or Pseudo_Time	Simulation cycle number (<i>i.e.</i> , number of time-steps) if inputs.in > solver_control > steady_solver = 0. The pseudo time-step if steady_solver = 1.
3	dt (seconds)	Time-step size.
4	Max_CFL	Maximum convective CFL number for this time-step.
5	Max_Visc_CFL	Maximum viscous CFL number for this time-step.
6	Max_Cond_CFL	Maximum conductive CFL number for this time-step.
7	Max_Diff_CFL	Maximum diffusive CFL number for this time-step.
8	Max_Mach_CFL	Maximum Mach CFL number for this time-step.
9	WallTime (seconds)	Computational time to solve this time-step.
10	Num_Recovers	The number of recoveries for this time-step.
11	dt_limiter	The parameter that limited the size of this time-step. If CONVERGE writes dt_grow , then the previous time-step was small enough to resolve the relevant physics and preserve stability, and thus CONVERGE will increase the time-step size. If a time-step had one or more recoveries (see previous column), then the limiter will be followed by the parameter(s) that caused the recoveries. This column is blank for the first time-step because the first time-step is determined by inputs.in > temporal_control > dt_start.

transfer.out

CONVERGE generates *transfer.out* when the FEA heat transfer output is active (*i.e.*, when [inputs.in](#) > output_control > transfer_flag = 1). This file contains wall heat transfer data for each cell adjacent to a WALL boundary and is used for coupling with finite element analysis. If you set [inputs.in](#) > output_control > transfer_flag to a file name (*e.g.*, [transfer.in](#)) and supply the corresponding file, CONVERGE will write data to *transfer.out* only for specific boundaries.

The output frequency is controlled by [inputs.in](#) > output_control > twrite_transfer. At each write time, CONVERGE appends summary information (data write time, average gas temperature and pressure, total wall surface area, and surface average heat flux and heat transfer coefficient) and then wall heat transfer data to *transfer.out*.

Table 26.84: Description of *transfer.out*.

Chapter 26: Output Files

3D Simulation Output Files transfer.out

Column	Header (units)	Description
1	Number	Data point number.
2	Bound_id	Wall boundary ID.
3	X (m)	Current x coordinate of the boundary cell.
4	Y (m)	Current y coordinate of the boundary cell.
5	Z (m)	Current z coordinate of the boundary cell.
6	F_TEMP (K)	Temperature of the fluid in the boundary cell.
7	FLUX (W/m ²)	Heat flux at the boundary cell.
8	AREA (m ²)	Area of the boundary cell.
9	HTC (W/m ² /K)	Heat transfer coefficient at the boundary cell. HTC is 0 for solid cells, the value defined in boundary.in for cells subject to the convection boundary condition, and given by the heat flux divided by the temperature difference in all other cases.
10	B_TEMP (K)	Temperature of the boundary adjacent to the cell.
11	Y_PLUS	Dimensionless wall distance at the boundary cell.
12	X_ORIG (m)	Original x coordinate of the boundary cell that comes from the surface file (e.g., <i>surface.dat</i>).
13	Y_ORIG (m)	Original y coordinate of the boundary cell that comes from the surface file (e.g., <i>surface.dat</i>).
14	Z_ORIG (m)	Original z coordinate of the boundary cell that comes from the surface file (e.g., <i>surface.dat</i>).
15	PRES (Pa)	Pressure at the boundary cell.
16	FLUX_CONV (W/m ²)	Convective heat flux at the boundary cell.
17	HTC_CONV (W/m ² /K)	Convective heat transfer at the boundary cell.
18	VEL_MAG (m/s)	Magnitude of velocity at the boundary cell.
19	MAG_WALL_STRESS (N/m ²)	Magnitude of shear stress at the wall.
Optional output that is included if listed in <i>transfer.in</i> .		
20	SIE (J/kg)	Cell specific internal energy including the formation energy of species.
21	VEL_X (m/s)	Cell velocity in the x direction.
22	VEL_Y (m/s)	Cell velocity in the y direction.
23	VEL_Z (m/s)	Cell velocity in the z direction.
24	EPS (m ² /s ³)	Cell turbulence dissipation rate.
25	COND (W/m-K)	Cell thermal conductivity including the turbulent component of thermal conductivity.

Chapter 26: Output Files

3D Simulation Output Files transfer.out

Column	Header (units)	Description
26	TKE (m^2/s^3)	Cell turbulent kinetic energy. This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
27	FILM_TEMP (K)	Average temperature of liquid film on wall (if a film is present), otherwise temperature of the fluid in the boundary cell.

turbulence.out

CONVERGE generates *turbulence.out* when [turbulence modeling](#) is active (*i.e.*, when [*inputs.in*](#) > *solver_control* > *turbulence_solver* = 1). The output frequency is controlled by [*inputs.in*](#) > *output_control* > *twrite_files*. This file summarizes the turbulence quantities. The first column lists the simulation time while the remaining columns give the mass-averaged turbulence data at that time.

Table 26.85: Description of *turbulence.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2	TKE (m^2/s^2)	Mass-averaged turbulent kinetic energy (tke). This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
3	TKE_StdDev (m^2/s^2)	Standard deviation of turbulent kinetic energy. This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
4	EPS (m^2/s^3)	Mass-averaged turbulent dissipation (eps).
5	EPS_StdDev (m^2/s^3)	Standard deviation of turbulent dissipation (eps).
6	OMEGA (s^{-1})	Mass-average specific dissipation rate (omega).
7	OMEGA_StdDev (s^{-1})	Standard deviation of specific dissipation rate.
8	Turb_Kin_Visc (m^2/s)	Mass-averaged turbulent viscosity.
9	Kin_Visc_StdDev(m^2/s)	Standard deviation of turbulent viscosity.
10	Lengthscale (m)	Mass-averaged turbulent length scale.
11	Lengthscale_StdDev (m)	Standard deviation of turbulent length scale.
12	UPrime (m/s)	Mass-averaged turbulent velocity.
13	UPrime_StdDev (m/s)	Standard deviation of turbulent velocity.
14	Visc_Ratio (none)	Ratio of molecular and turbulent viscosity.
15	Visc_RatStdDev (none)	Standard deviation of ratio of molecular and turbulent viscosity.

Chapter 26: Output Files

3D Simulation Output Files turbulence.out

Column	Header (units)	Description
16	Turb_Dyn_Visc ($N \cdot s/m^2$)	Dynamic turbulent viscosity.
17	Dyn_Visc_StdDev ($N \cdot s/m^2$)	Standard deviation of turbulent dynamic viscosity.
18	V2 (m^2/s^2)	Velocity variance normal to the streamline.
19	V2_StdDev (m^2/s^2)	Standard deviation of velocity variance normal to the streamline.
20	ZETA (none)	Velocity scales ratio.
21	ZETA_StdDev (none)	Standard deviation of velocity scales ratio.
22	F (s^{-1})	Elliptic relaxation function.
23	F_StdDev (s^{-1})	Standard deviation of elliptic relaxation function.
24	R11 (m^2/s^2)	Component 11 of the Reynolds stress tensor.
25	R11_StdDev (m^2/s^2)	Standard deviation of component 11 of the Reynolds stress tensor.
26	R22 (m^2/s^2)	Component 22 of the Reynolds stress tensor.
27	R22_StdDev (m^2/s^2)	Standard deviation of component 22 of the Reynolds stress tensor.
28	R33 (m^2/s^2)	Component 33 of the Reynolds stress tensor.
29	R33_StdDev (m^2/s^2)	Standard deviation of component 33 of the Reynolds stress tensor.
30	R12 (m^2/s^2)	Component 12 of the Reynolds stress tensor.
31	R12_StdDev (m^2/s^2)	Standard deviation of component 12 of the Reynolds stress tensor.
32	R13 (m^2/s^2)	Component 13 of the Reynolds stress tensor.
33	R13_StdDev (m^2/s^2)	Standard deviation of component 13 of the Reynolds stress tensor.
34	R23 (m^2/s^2)	Component 23 of the Reynolds stress tensor.
35	R23_StdDev (m^2/s^2)	Standard deviation of component 23 of the Reynolds stress tensor.

Recall that the dynamic viscosity is the absolute viscosity, given by $\mu_t = \rho c_\mu \frac{k^2}{\varepsilon}$, and that the kinematic viscosity (m^2/s) is the dynamic viscosity divided by density. The parameter c_μ is a turbulence model constant, k is the turbulent kinetic energy, and ε is the turbulent dissipation. For simulations using a RANS turbulence model, the turbulent length scale is given by

Chapter 26: Output Files

3D Simulation Output Files turbulence.out

$$le = c_{\mu}^{3/4} \frac{k^{3/2}}{\varepsilon}, \quad (26.32)$$

while for LES calculations the turbulent length scale is equal to the [grid filter width](#) (the cube root of the cell volume). The turbulent velocity is given by

$$u' = \sqrt{\frac{2}{3}} k. \quad (26.33)$$

The standard deviations for turbulent kinetic energy, turbulent dissipation rate, turbulent kinematic viscosity, length scale, turbulent velocity, viscosity ratios, and turbulent dynamic viscosity are evaluated as

$$\psi_{STD} = \sqrt{\frac{\sum_{cell} m_{cell} (\psi_{cell} - \psi_{mean})^2}{m_{total}}}, \quad (26.34)$$

where ψ represents any of these turbulence variables and m is the mass. The subscripts are as follows: *cell* indicates the cell value, *mean* represents the mean value, *total* indicates the total value, and *STD* indicates the standard deviation.

user_pdpa_<time>.out

CONVERGE generates *user_pdpa_<time>.out* when the Phase Doppler Particle Analyzer (PDPA) distribution output is activated (*i.e.*, when [*inputs.in* > *output_control* > *smd_pdf_flag* = 2\). The *<time>* corresponds to the time in seconds \(if \[*inputs.in* > *simulation_control* > *crank_flag* = 0\\) or in *crank angle degrees* \\(if *crank_flag* is non-zero\\). These files contain spray information calculated across a set of planes. Set up input parameters for the PDPA distribution output, including the frequency with which CONVERGE writes these files, in \\[*pdpa.in*\\]\\(#\\).\]\(#\)](#)

26.35: Description of *user_pdpa_<time>.out*.

Column	Header (units)	Description
1	X (m)	X location of this plane.
2	Y (m)	Y location of this plane.
3	Z (m)	Z location of this plane.
4	Num_parcels	Number of parcels which passed through this plane.
5	Num_drops	Number of particles which passed through this plane.
6	D10 (m)	10% of the total parcel mass was made up by parcels smaller than this diameter.

Chapter 26: Output Files

3D Simulation Output Files user_pdpa_<time>.out

Column	Header (units)	Description
7	SMD (m)	Average Sauter mean diameter of parcels which passed through this plane.
8	Volume_flux (m/s)	Volume flux through this plane.
9	Mass_flux (kg/m ² -s)	Mass flux through this plane.
10	Avg_U (m/s)	Average U velocity through this plane.
11	Avg_V (m/s)	Average V velocity through this plane.
12	Avg_W (m/s)	Average W velocity through this plane.
13	Dv10 (m)	The spray parcel diameter representing the tenth percentile by volume.
14	Dv50 (m)	The volume median particle size.
15	Dv90 (m)	The spray parcel diameter representing the ninetieth percentile by volume.

vof_spray.out

CONVERGE generates *vof_spray.out* when [one-way coupling between VOF and Lagrangian spray modeling](#) is active (*i.e.*, when *vof.in* > *vof_spray_flag* = 1). This file contains position, velocity, turbulence, temperature, and cell size information for the liquid parcels in the VOF simulation. CONVERGE writes this data to *vof_spray.out* at the interval specified for *vof_spray.in* > *twrite_vof_spray*. The header provides general information about the nozzle, the two regions between which the data apply, and liquid parcel data from the VOF simulation. The injector and nozzle IDs correspond to those specified in [*vof_spray.in*](#). The column output contains specific data for the liquid parcels.

Table 26.86: Description of *vof_spray.out*.

Column	Header (units)	Description
1	X (m)	Current x coordinate.
2	Y (m)	Current y coordinate.
3	Z (m)	Current z coordinate.
4	U (m/s)	The x component of velocity.
5	V (m/s)	The y component of velocity.
6	W (m/s)	The z component of velocity.
7	Liquid VOF	Volume fraction of liquid at the given position.
8	Total Mass Liquid (kg)	Mass of liquid at the given position.

Chapter 26: Output Files

3D Simulation Output Files vof_spray.out

Column	Header (units)	Description
9	TKE (m^2/s^2)	Mass-averaged turbulent kinetic energy (tke). This is the transported tke for RANS models that explicitly transport this quantity. Otherwise, it is derived from sub-grid velocity (LES) or estimated from eddy viscosity and eps (Spalart-Allmaras).
10	EPS (m^2/s^3)	Mass-averaged turbulent dissipation (eps).
11	TEMP (K)	Temperature at the given position.
12	dx (m)	Cell size in the x direction.
13	dy (m)	Cell size in the y direction.
14	dz (m)	Cell size in the z direction.

volumes.out

CONVERGE generates *volumes.out* and *volumes_region<region ID>.out* when [volume of fluid \(VOF\) modeling](#) is active (i.e., when *inputs.in* > *feature_control* > *vof_flag* = 1). The output frequency is controlled by *inputs.in* > *output_control* > *twrite_files*. The *volumes.out* file contains liquid and gas volume data for a VOF simulation. The *volumes_region<region ID>.out* files contain region-specific liquid and gas volume data for a VOF simulation.

Table 26.87: Description of *volumes.out* and *volumes_region<region ID>.out*.

Column	Header (units)	Description
1	Time (seconds) or Crank (crank angle degrees)	Time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero.
2 [‡]	Total Volume (m^3)	Total volume of fluid.
3 [‡]	Species name (m^3)	Total volume of the first species.
4 [‡]	Species name (m^3)	Total volume of the second species.

[‡] The value in this column is the product of the simulated quantity and the value of *inputs.in* > *output_control* > *output_multiplier*.

wall_stress<number>_<time>.out

CONVERGE generates *wall_stress<number>_<time>.out* when the wall output option is active (i.e., when *inputs.in* > *output_control* > *wall_output_flag* = 1). This file contain the force, stress, and pressure acting on WALL boundaries. The file name includes the output number and output time.

Table 26.88: Description of *wall_stress<number>_<time>.out*.

Column	Header (units)	Description
1	Bound_id	Wall boundary ID.
2	X (m)	Wall coordinate in the x direction.
3	Y (m)	Wall coordinate in the y direction.

Chapter 26: Output Files

3D Simulation Output Files wall_stress<number>_<time>.out

Column	Header (units)	Description
4	Z (m)	Wall coordinate in the z direction.
5	Mag Wall_Stress (N/m ²)	Vector sum of the wall stresses in the x, y, and z directions.
6	Wall_Stress_X (N/m ²)	Wall stress in the x direction.
7	Wall_Stress_Y (N/m ²)	Wall stress in the y direction.
8	Wall_Stress_Z (N/m ²)	Wall stress in the z direction.
9	Force_X (N)	Force on the wall in x direction.
10	Force_Y (N)	Force on the wall in y direction.
11	Force_Z (N)	Force on the wall in z direction.
12	Pres (N/m ²)	Pressure at the wall.

walltime.out

CONVERGE generates *walltime.out* when wall time information output is active (*i.e.*, when [inputs.in](#) > *output_control* > *walltime_output_flag* = 1). CONVERGE dynamically writes *walltime.out*, which specifies wall-clock time per time-step in column format. If [inputs.in](#) > *output_control* > *log_level* = 3, the [CONVERGE log file](#) will also contain this detailed wall-clock time per time-step information.

The first column in *walltime.out* contains time-step information (*i.e.*, time in *seconds* if [inputs.in](#) > *simulation_control* > *crank_flag* = 0 or in *crank angle degrees* if *crank_flag* is non-zero). The subsequent columns list the times CONVERGE spent performing simulation tasks. The list of columns (simulation tasks) depends on your case settings. The last section of the CONVERGE log file (*Total Simulation Run Time*) lists the columns that you can expect to see in *walltime.out*. Figure 26.3 shows the last section of an example *converge.log* file, and Table 26.89 lists the columns in a corresponding *walltime.out* file.

Note that the timing sections and subsections are approximate, and the sum of all subsections will not exactly equal the reported total simulation time. This is especially true if you are running a simulation in parallel. CONVERGE reports the subsection values only so that you can determine what aspects of your simulation are relatively more or less costly.

Total Simulation Run Time:	7.732 seconds
CFD Simulation Setup	= 1.393 seconds (18.02%)
Load Balance	= 0.001 seconds (0.01%)
Move Surface and Update Grid	= 0.129 seconds (1.67%)
Solving Transport	= 0.236 seconds (3.05%)
Spray	= 5.272 seconds (68.19%)
Updating Boundary Conditions	= 0.012 seconds (0.16%)
Writing Output	= 0.539 seconds (6.97%)
.out files	= 0.128 seconds (1.66%)
Post	= 0.246 seconds (3.18%)
Restart	= 0.196 seconds (2.54%)
Program used	7.731614 seconds.

Figure 26.3: Extract from an example *converge.log* file.

Chapter 26: Output Files

3D Simulation Output Files walltime.out

Table 26.89: Columns in an example *walltime.out* file.

Column	Header (units)
1	Time (s)
2	Load_Balance (s)
3	Move_Surface_and_Update_Grid (s)
4	Solving_Transport (s)
5	Spray (s)
6	Updating_Boundary_Conditions (s)
7	Writing_Output (s)
8	Writing_Output > .out_files (s)
9	Writing_Output > Post (s)
10	Writing_Output > Restart (s)

`zero_d_cases_rank<number>.out`

CONVERGE generates `zero_d_cases_rank<number>.out` when performing a [SAGE combustion simulation](#) in which the CVODE error output option is active (*i.e.*, when `combust.in > sage_model > active = 1` and `sage_model > ccode_error_output_flag = 1`). CONVERGE generates `zero_d_cases_rank<number>.out` for any cells that have CVODE errors and a temperature of less than 5000 K. The `<number>` corresponds to the processor number. This file has the same format as [`zero_d_cases.in`](#). When `combust.in > sage_model > solve_temp = 1`, this file is saved within your Case Directory in a subdirectory named either `ccode_constant_volume` or `ccode_constant_pressure` (depending on whether `combust.in > sage_model > option = CONSTANT_VOLUME` or `CONSTANT_PRESSURE`). When `sage_model > solve_temp = 0` and the change in temperature for a case does not exceed `combust.in > sage_model > delta_temp` (*i.e.*, temperature is not re-solved), the case is written to `zero_d_cases_rank<number>.out` in the `ccode_constant_temperature_pressure` subdirectory. When `sage_model > solve_temp = 0` and the change in temperature for a case exceeds `sage_model > delta_temp` (*i.e.*, temperature is re-solved), the case is written to `zero_d_cases_rank<number>.out` in the `ccode_constant_volume` or `ccode_constant_pressure` subdirectory based on whether `sage_model > option = CONSTANT_VOLUME` or `CONSTANT_PRESSURE`, respectively.

CONVERGE generates `zero_d_cases_rank<number>.out` so that you can use it to perform a [zero-dimensional constant volume or constant pressure autoignition simulation](#) on the cells that had CVODE errors. In order to use one of these files for a 0D simulation, change the file name to `zero_d_cases.in`. Typically you do not need to edit the contents of this file before running a 0D simulation.

`zero_d_solver.out`

CONVERGE generates `zero_d_solver.out` when performing a [SAGE combustion simulation](#) in which the CVODE error output option is active (*i.e.*, when `combust.in > sage_model > active = 1` and `sage_model > ccode_error_output_flag = 1`). This file has the same format as

Chapter 26: Output Files

3D Simulation Output Files zero_d_solver.out

[*zero_d_solver.in*](#) and is saved within your Case Directory in a subdirectory called *cvoode_constant_volume*, *cvoode_constant_pressure*, and/or *cvoode_constant_temperature_pressure*.

This file is generated so that you can use it to perform a [zero-dimensional constant volume or constant pressure autoignition simulation](#). In order to use this file for a 0D simulation, change its name to *zero_d_solver.in*.

If CVODE errors do not occur, this file will have a *zero_d_input_control > case_type = CONSTANT_VOLUME* (if [*combust.in > sage_model > option = CONSTANT_VOLUME*](#)), *CONSTANT_PRESSURE* (if *sage_model > option = CONSTANT_PRESSURE*), or *CONSTANT_TEMPERATURE_PRESSURE* (if *sage_model > solve_temp = 0*).

If CVODE errors do occur when [*combust.in > sage_model > solve_temp = 1*](#), *zero_d_solver.out* will have a *zero_d_input_control > case_type = CVODE_CONSTANT_VOLUME* (if [*combust.in > sage_model > option = CONSTANT_VOLUME*](#)) or *CVODE_CONSTANT_PRESSURE* (if *sage_model > option = CONSTANT_PRESSURE*) and be saved in a subdirectory named either *cvoode_constant_volume* or *cvoode_constant_pressure*, respectively.

If CVODE errors do occur when [*combust.in > sage_model > solve_temp = 0*](#), the temperature is re-solved only when the change in temperature exceeds [*combust.in > sage_model > delta_temp*](#), therefore two *zero_d_solver.out* files are created in two different subdirectories: one for the scenario in which temperature is not re-solved (*i.e.*, change in temperature does not exceed *sage_model > delta_temp*), and one for the scenario in which temperature is re-solved (*i.e.*, change in temperature exceeds *sage_model > delta_temp*). For the scenario in which temperature is not re-solved, *zero_d_input_control > case_type = CVODE_CONSTANT_TEMPERATURE_PRESSURE* in *zero_d_solver.out* and the file is saved in a subdirectory named *cvoode_constant_temperature_pressure*. For the scenario in which temperature is re-solved, *zero_d_input_control > case_type = CVODE_CONSTANT_VOLUME* (if *sage_model > option = CONSTANT_VOLUME*) or *CVODE_CONSTANT_PRESSURE* (if *sage_model > option = CONSTANT_PRESSURE*) in *zero_d_solver.out* and the file is saved in a subdirectory named *cvoode_constant_volume* or *cvoode_constant_pressure*, respectively.

26.2 Thermodynamic Output File

This section describes the output file that CONVERGE writes for all 3D simulations and for one-dimensional laminar flamespeed calculations that use the PISO solver.

thermo.out

CONVERGE generates *thermo.out* for all 3D simulations and for [1D laminar flamespeed calculations that use the PISO solver](#) (*i.e.*, when [*one_d_solver.in > one_d_general_control > solver_control > solver_type = PISO*](#) or *HYBRID*). The output frequency is controlled by [*inputs.in > output_control > twrite_files*](#). This file contains thermodynamic and combustion data.

Chapter 26: Output Files

Thermodynamic Output File thermo.out

Table 26.90: Description of *thermo.out*.

Column	Header (units)	Description
1	Time (seconds), Crank (crank angle degrees), or Cycles	Time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero, or the simulation cycle number (i.e., number of time-steps).
2	Pressure (MPa)	Volume-averaged pressure.
3	Max_Pres (MPa)	Maximum pressure.
4	Min_Pres (MPa)	Minimum pressure.
5	Mean_Temp (K)	Mass-averaged temperature.
6	Max_Temp (K)	Maximum temperature.
7	Min_Temp (K)	Minimum temperature.
8 [‡]	Volume (m^3)	Summation of all cell volumes.
9 [‡]	Mass (kg)	Summation of all cell masses.
10	Density (kg/m^3)	Total mass divided by total volume.
11 [‡]	Integrated_HR (J)	Total heat release (summed over time) due to chemical heat release and any added energy sources.
12 [‡]	HR_Rate (J/time)	Heat release rate due to chemical heat release and any added energy sources.
13	C_p (J/kg-K)	Specific heat at constant pressure.
14	C_v (J/kg-K)	Specific heat at constant volume.
15	Gamma (C_p/C_v)	Ratio of specific heats.
16	Kin_Visc (m^2/s)	Molecular kinematic viscosity.
17	Dyn_Visc ($N\cdot s/m^2$)	Molecular dynamic viscosity.
18	Specific_Entropy (J/kg-K)	Specific entropy. CONVERGE calculates this value for compressible gas streams only, and writes 0 for other streams. This column is written if and only if <i>specific_entropy</i> is specified in post.in .

[‡] The value in this column is the product of the simulated quantity and the value of [inputs.in](#) > output_control > output_multiplier.

26.3 Mechanism Check Output Files

This section describes the output files that CONVERGE writes after checking the mechanism data files for errors.

mech_check.out

The *mech_check.out* file is generated for all simulations in which CONVERGE reads a [reaction mechanism data file](#) (*mech.dat*).

Chapter 26: Output Files

Mechanism Check Output Files mech_check.out

The *mech_check.out* file is in two parts. The first portion describes the species in *mech.dat*.

Table 26.91: Description of the first part of *mech_check.out*.

Column	Header (units)	Description
1	#	Species number.
2	SPECIES	Species listed in the reaction mechanism file.
3	PHASE	Phase of the species listed in the reaction. Typically G for gas phase.
4	MW	Molecular weight of the species.
5	LOW	Low temperature limit.
Varies	MID	Intermediate temperature limit. This column is repeated for as many MID temperatures as listed for the NASA 9 thermodynamic data file format, if the species listed above is specified in the NASA 9 format.
Varies	HIGH	High temperature limit.
Varies	<ELEMENT>	Number of <ELEMENT> atoms in the species. This column is repeated for each element listed in the mechanism data file.

The second part of *mech_check.out* lists all the reactions read by CONVERGE in the reaction mechanism file. Each reaction is assigned a three-digit reaction number, which is followed by the reaction coefficients. Special conditions for the reaction (such as enhancements) are printed offset below the reaction. The following table describes the format of this portion of *mech_check.out*.

Table 26.92: Description of the second part of *mech_check.out*.

Column	Header (units)	Description
1	#	Reaction number.
2	REACTION	Reaction listed in the reaction mechanism file.
3	A	Pre-exponential factor for the reaction.
4	b	Temperature dependence of exponential factor for the reaction.
5	E	Activation energy for the reaction (<i>cal/mole</i>).

surface_mech_check.out

The *surface_mech_check.out* file is generated for all simulations in which CONVERGE reads a surface reaction mechanism file ([surface_mech.dat](#)).

The *surface_mech_check.out* file is in two parts. The first portion describes the surface species in [surface_mech.dat](#).

Chapter 26: Output Files

Mechanism Check Output Files surface_mech_check.out

Table 26.93: Description of the first part of *surface_mech_check.out*.

Column	Header (units)	Description
1	Num	Surface species number.
2	SPECIES	Surface species listed in <i>surface_mech.dat</i> .
3	SITES	Site occupancy for the surface species.
4	TYPE	Type of surface species. I = Surface species, S = Catalytic open site species.
5	MW	Molecular weight of the species.
6	LOW	Low temperature limit.
Varies	MID	Intermediate temperature limit. This column is repeated for as many MID temperatures as listed for the NASA 9 thermodynamic data file format, if the species is specified in the NASA 9 format.
Varies	HIGH	High temperature limit.
Varies	<ELEMENT>	Number of <ELEMENT> atoms in the species. This column is repeated for each element.

The second part of *surface_mech_check.out* lists all the surface reactions read by CONVERGE and assigns a reaction number to each one.

Table 26.94: Description of the second part of *surface_mech_check.out*.

Column	Header (units)	Description
1	Num	Surface reaction number.
2	REACTION	Surface reaction listed in <i>surface_mech.dat</i> .
3	A	Pre-exponential factor for the reaction.
4	b	Temperature dependence of exponential factor for the reaction.
5	E	Activation energy for the reaction (<i>cal/mole</i>).

26.4 HTC Mapping Output Files

This section describes the output files that CONVERGE writes after running the [heat transfer mapping \(htc_map\) utility](#).

htc_triangles.map

CONVERGE generates *htc_triangles.map* after running the [heat transfer mapping \(htc_map\) utility](#). This file contains time-averaged data for each triangle in the finite element analysis

Chapter 26: Output Files

HTC Mapping Output Files htc_triangles.map

(FEA) surface file. CONVERGE will automatically include additional variables (*e.g.*, pressure) if it finds additional columns in [*transfer.out*](#).

Table 26.95: Description of *htc_triangles.map*.

Column	Header (<i>units</i>)	Description
1	tri_num	Triangle number.
2	x (<i>m</i>)	X coordinate of the triangle.
3	y (<i>m</i>)	Y coordinate of the triangle.
4	z (<i>m</i>)	Z coordinate of the triangle.
5	TriArea (<i>m</i> ²)	Area of the triangle.
6	BoundaryTemp (<i>K</i>)	Wall temperature.
7	Flux (<i>W/m</i> ²)	Heat flux.
8	Htc (<i>W/m</i> ² / <i>K</i>)	Heat transfer coefficient (HTC).
9	FluidTemp (<i>K</i>)	Fluid temperature.
10	Yplus	Dimensionless wall distance (<i>y+</i>).
11	timewithhits (<i>crank angle degrees</i>)	Time over which the triangle is mapped. This is less than the total time only if the boundary is not exposed for the entire engine cycle.
12	totaltime (<i>crank angle degrees</i>)	Total time.

htc_vertices.map

CONVERGE generates *htc_vertices.map* after running the [*heat transfer mapping \(htc map\) utility*](#). This file contains time-averaged data for each vertex in the finite element analysis (FEA) surface file. CONVERGE will automatically include additional variables (*e.g.*, pressure) if it finds additional columns in [*transfer.out*](#).

CONVERGE uses the average of all the values at the center of each neighboring triangle to calculate the vertex value of each quantity.

Chapter 26: Output Files

HTC Mapping Output Files htc_vertices.map

Table 26.96: Description of *htc_vertices.map*.

Column	Header (units)	Description
1	vert_num	Vertex number.
2	x (m)	X coordinate of the vertex.
3	y (m)	Y coordinate of the vertex.
4	z (m)	Z coordinate of the vertex.
5	BoundaryTemp (K)	Wall temperature.
6	Flux (W/m ²)	Heat flux.
7	Htc (W/m ² /K)	Heat transfer coefficient (HTC).
8	FluidTemp (K)	Fluid temperature.
9	Yplus	Dimensionless wall distance (y^+).
10	timewithhits (crank angle degrees)	Time over which the vertex is mapped. This is less than the total time only if the boundary is not exposed for the entire engine cycle.
11	totaltime (crank angle degrees)	Total time.
12*	SumAngle (deg)	The sum of the angles sharing the vertex. This sum may be less than 360 <i>degrees</i> if the vertex is located at the edge of a geometry or at one or more boundaries. This sum may also vary for vertices at contoured intersections of triangles.

*The number of this column will be different if additional variables are included from [transfer.out](#).

26.5 Proper Orthogonal Decomposition Output File

This section describes the output file that CONVERGE writes after performing a [proper orthogonal decomposition](#) (POD).

energy_fraction.out

CONVERGE generates *energy_fraction.out* after performing a [proper orthogonal decomposition](#) (POD) with the *pod* utility. This file lists the fraction of each variable accounted for by each POD mode.

Table 26.97: Description of *energy_fraction.out*.

Column	Header (units)	Description
1	Mode	POD mode of this row.
Varies	<Variable Name>	Fraction of this cell variable's content accounted for by this mode.

Chapter 26: Output Files

0D Simulation Output Files

26.6 0D Simulation Output Files

This section describes the output files written by the [0D chemistry utilities](#).

extinction_limit.out

CONVERGE generates *extinction_limit.out* when the option to estimate the extinction or ignition limits in a [well-stirred reactor](#) is active (*i.e.*, when `zero_d_solver.in > well_stirred_reactor_control > turning_points = EXTINCTION` or `EXTINCTION_AND_IGNITION`). CONVERGE alters inlet elemental equivalent ratios to determine the residence time.

Table 26.98: Description of *extinction_limit.out*.

Column	Header (units)	Description
1	NUM (none)	Well-stirred reactor simulation number.
2	Pressure (Pa)	The mass flow at the specified location.
3	Inlet_temp (K)	The temperature at the inlet.
4	Inlet_phi_elem (none)	Elemental equivalence ratio at the inlet.
5	Extinction_residence_time (seconds)	Calculated extinction residence time.
6	Ignition_residence_time (seconds)	Calculated ignition residence time.

ignition_det.out

CONVERGE generates *ignition_det.out* for all [zero-dimensional simulations](#) except the well-stirred reactor. This file contains different information depending on the 0D case type.

For [constant volume, constant pressure, constant temperature and pressure, CEQ, variable volume](#), and [PFR](#) case types (*i.e.*, for `zero_d_solver.in > zero_d_input_control > case_type = CONSTANT_VOLUME, CONSTANT_PRESSURE, CONSTANT_TEMPERATURE_PRESSURE, CEQ_CONSTANT_ENTHALPY_PRESSURE, CEQ_CONSTANT_TEMPERATURE_PRESSURE, VARIABLE_VOLUME, or CONSTANT_PRESSURE_PLUG_FLOW_REACTOR`), *ignition_det.out* contains the ignition delay data for each simulation.

Chapter 26: Output Files

0D Simulation Output Files ignition_det.out

Table 26.99: Description of *ignition_det.out* when *zero_d_input_control > case_type = CONSTANT_VOLUME, CONSTANT_PRESSURE, CONSTANT_TEMPERATURE_PRESSURE, CEQ_CONSTANT_ENTHALPY_PRESSURE, CEQ_CONSTANT_TEMPERATURE_PRESSURE, VARIABLE_VOLUME, or CONSTANT_PRESSURE_PLUG_FLOW_REACTOR*.

Column	Header (units)	Description
1	NUM	Case number.
2	Pressure (bar)	Pressure.
3	Temperature (K)	Temperature.
4	Phi_fuel_oxid	Equivalence ratio.
5	EGR_ratio	EGR ratio.
6	Phi_elem	Elemental equivalence ratio.
7	Time1 (seconds)	Ignition delay time or end time (s).
8*	Time2 (seconds)	Double ignition delay time (s).

*The last column is written only when double ignition delay is active (i.e., when [zero_d_solver.in > zero_d_output_control > double_ignition_delay_flag = 1](#)).

The following table describes the format of *ignition_det.out* for an [HCCI engine case type](#) (i.e., for [zero_d_solver.in > zero_d_input_control > case_type = ENGINE](#)).

Table 26.100: Description of *ignition_det.out* when *zero_d_input_control > case_type = ENGINE*.

Column	Header (units)	Description
1	NUM	Case number.
2	Pressure (bar)	Pressure.
3	Temperature (K)	Temperature.
4	Phi_fuel_oxid	Equivalence ratio.
5	EGR_ratio	EGR ratio.
6	Phi_elem	Elemental equivalence ratio.
7	Compression_ratio	Compression ratio.
8	Crank (deg)	Ignition delay time.

The following table describes the format of *ignition_det.out* for an [engine RON/MON case type](#) (i.e., for [zero_d_solver.in > zero_d_input_control > case_type = ENGINE_RON](#) or [ENGINE_MON](#)).

Chapter 26: Output Files

0D Simulation Output Files ignition_det.out

Table 26.101: Description of *ignition_det.out* when *zero_d_input_control > case_type = ENGINE_RON or ENGINE_MON*.

Column Header (units)	Description
1 NUM	Case number.
2 Pressure (bar)	Pressure.
3 Temperature (K)	Temperature.
4 Phi_fuel_oxid	Equivalence ratio.
5 EGR_ratio	EGR ratio.
6 Phi_elem	Elemental equivalence ratio.
7 Crank (deg)	Ignition delay time.
8 RON_CCR or MON_CCR	Critical compression ratio of the fuel.
9 RON or MON	Research octane number or motor octane number for the fuel.

The following table describes the format of *ignition_det.out* for an [engine RON and MON case type](#) (*i.e.*, for [*zero_d_solver.in* > *zero_d_input_control* > *case_type* = *ENGINE_RON_MON*](#)).

Table 26.102: Description of *ignition_det.out* when *zero_d_input_control > case_type = ENGINE_RON_MON*.

Column Header (units)	Description
1 NUM	Case number.
2 Pressure (bar)	Pressure.
3 Phi_fuel_oxid	Equivalence ratio.
4 EGR_ratio	EGR ratio.
5 Phi_elem	Elemental equivalence ratio.
6 RON_CCR	Critical compression ratio of the fuel.
7 RON	Research octane number for the fuel. This value is the PRF of the corresponding reference CCR.
8 MON_CCR	Critical compression ratio of the fuel.
9 MON	Motor octane number for the fuel.

CONVERGE does not generate *ignition_det.out* for a well-stirred reactor case type (*i.e.*, for [*zero_d_solver.in* > *zero_d_input_control* > *case_type* = *WELL_STIRRED_REACTOR*](#)).

sens<case ID>.out

CONVERGE generates *sens<case ID>.out* when you include [forward sensitivity analysis](#) in a zero-dimensional simulation (*i.e.*, when [*zero_d_solver.in* > *zero_d_sensitivity_control* >](#)

Chapter 26: Output Files

0D Simulation Output Files sens<case ID>.out

sensitivity_flag = FORWARD). This file includes the sensitivity coefficient matrix at each time-step. The <case ID> corresponds to the case ordering in [zero_d_cases.in](#) (e.g., the first case in [zero_d_cases.in](#) would yield *sens1.out*). You can post-process this file with the [sens_convert](#) utility to create a folder containing [three additional output files](#), which contain sensitivity coefficients for only the variables (species and/or temperature) in which you are interested.

sens<case ID>_var<number>_<name>*.out

CONVERGE writes [sens<case ID>.out](#) when you include [forward sensitivity analysis](#) in a zero-dimensional simulation (set [zero_d_solver.in](#) > [zero_d_sensitivity_control](#) > *sensitivity_flag* = FORWARD). By post-processing this file with the [sens_convert](#) utility, you can create a folder for each case (named *output<case ID>*, where <case ID> corresponds to the case ordering in [zero_d_cases.in](#)) containing three files:

sens<case ID>_var<number>_<name>.out,
sens<case ID>_var<number>_<name>_neg.out, and
sens<case ID>_var<number>_<name>_pos.out.

These files are named such that <case ID> in the output folder name and the file name represents the order of the 0D cases in [zero_d_cases.in](#), <number> represents the variable number (either the species number from [species_info.dat](#) or [1 + the maximum number of species in [species_info.dat](#)] for temperature), and <name> is the variable name (either the species name or TEMP for temperature). For example, Figure 26.4 below shows a file named *sens0_var38_TEMP.out*. In this example, the case ID is 0, the variable number is 38, and the variable name is TEMP (for temperature).

The *sens<case ID>_var<number>_<name>.out* file contains columns for the time (in seconds), the variable quantity (either species concentration or temperature), and the sensitivity coefficient for each reaction (listed by the reaction numbers in *mech_check.out*) at each time-step. The sensitivity coefficient columns lists the reaction number, maximum sensitivity coefficient value (positive or negative), the ranking based on the absolute value of the previous number (positive coefficient values have positive rankings and negative coefficient values have negative rankings), and the sensitivity coefficients for the reaction, in this order. CONVERGE finds the maximum absolute value of the sensitivity coefficient for each reaction and ranks them. CONVERGE lists the value and ranking after the reaction number in the second and third row, respectively, in *sens<case ID>_var<number>_<name>.out*.

Table 26.103: Description of *sens<case ID>_var<number>_<name>.out*.

Column Header (units)	Description
1	Time (seconds)
2	Variable Quantity (Conc in mol/cm ³ or Temp in K)
Varies	Reaction Sensitivity Coefficients

Chapter 26: Output Files

0D Simulation Output Files sens<case ID>_var<number>_<name>*.out

Column Header (units)	Description
	absolute value of the previous number (positive coefficient values have positive rankings and negative coefficient values have negative rankings), and the sensitivity coefficients for the reaction, in this order.

```
#      Time      Temp      Reaction      Reaction      Reaction
#      sec       K        1          2          3
#                  6.136757e+00 -1.838705e-01  1.912526e-02
#                  3           -15          19
# 2.030059e-17 1.268100e+03 1.569349e-14 -4.749065e-23 2.042330e-33
# 2.030262e-13 1.268100e+03 1.569554e-10 -4.749540e-15 2.043835e-21
...     ...     ...     ...     ...     ...
```

Figure 26.4: Excerpt of *sens0_var38_TEMP.out*.

The *sens<case ID>_var<number>_<name>_pos.out* and *sens<case ID>_var<number>_<name>_neg.out* files have a column for time (in seconds) and a column of all of the positive or negative sensitivity coefficients, respectively. CONVERGE ranks each reaction in a case according to the maximum absolute value of the sensitivity coefficient of each reaction. The output files list the columns of sensitivity coefficients in ascending order by rank, as shown below in Figure 26.5.

#	Total Neg	Place	Place	Place	Place
#	28	1	2	3	4
#	Time	Reaction	Reaction	Reaction	Reaction
#	sec	1	35	6	37
2.030059e-17	-5.742926e-13	-5.563760e-53	-1.189573e-27	1.787541e-46	
2.030262e-13	-5.743501e-09	-6.759515e-33	-1.189663e-19	1.781131e-30	
...

Figure 26.5: Excerpt of *sens0_var1_IC8H18_neg.out*.

species_info.dat

CONVERGE generates *species_info.dat* when you include [forward sensitivity analysis](#) in a zero-dimensional simulation (*i.e.*, when *zero_d_solver.in* > *zero_d_sensitivity_control* > *sensitivity_flag* = FORWARD). The *species_info.dat* file contains a numbered list of all the species in the [reaction mechanism file](#) (*e.g.*, *mech.dat*).

zero_d_adjoint.out

CONVERGE generates *zero_d_adjoint.out* for a [mechanism tuning](#) simulation with [adjoint sensitivity analysis](#) (*i.e.*, when *zero_d_solver.in* > *zero_d_sensitivity_control* > *sensitivity_flag* = ADJOINT).

The *zero_d_adjoint.out* file consolidates information from the *zero_d_adjoint_case<case ID>.out* files. It contains the reactions from the reaction mechanism file sorted in decreasing order of sensitivity, as shown below in Figure 26.6. The rank of the reaction is the first column, and the subsequent columns list the reaction number from each case.

Chapter 26: Output Files

0D Simulation Output Files zero_d_adjoint.out

# column	1	2	3	4
#	Place	Case_1_Reac_Num	Case_2_Reac_Num	Case_3_Reac_Num
#	(none)	(none)	(none)	(none)
#				
1		16	16	16
2		-15	-27	-27
3		-27	-15	-13

Figure 26.6: Excerpt of *zero_d_adjoint.out*.

Table 26.104: Description of *zero_d_adjoint.out*.

Column	Header (units)	Description
1	Place (none)	The order of sensitivity for reactions from the reaction mechanism file.
2*	Case_<case ID>_Reac_Num (none)	<p>The reaction number for this reaction in <case ID>, where <case ID> corresponds to the case ordering in <i>zero_d_cases.in</i>. A positive reaction number indicates that the flamespeed increases as the pre-exponential factor increases, and vice versa. A negative reaction number indicates that the flamespeed decreases as the pre-exponential factor increases, and vice versa.</p> <p>This column is repeated for each case.</p>

zero_d_adjoint_case<case ID>.out

CONVERGE generates *zero_d_adjoint_case<case ID>.out* for a zero-dimensional simulation with [adjoint sensitivity analysis](#) (i.e., when *zero_d_solver.in* > *zero_d_sensitivity_control* > *sensitivity_flag* = ADJOINT).

The *zero_d_adjoint_case<case ID>.out* files are named by the ordering of the 0D cases in *zero_d_cases.in* (e.g., the first case in *zero_d_cases.in* would yield *zero_d_adjoint_case00001.out*).

Note that the header of the second column is the name of the variable for which you chose to perform adjoint sensitivity analysis. For each such variable, a column will be generated after the second.

Table 26.105: Description of *zero_d_adjoint_case<case ID>.out*.

Column	Header (units)	Description
1	Reaction (none)	Reaction number. The valence of the reaction number (+ or -) specifies if the normalized sensitivity coefficient was positive or negative. This value will be the same as the reaction number specified in <i>mech_check.out</i> .
2*	<Variable name> (none)	Sensitivity coefficient normalized by the specified variable at the ignition delay or at end time. Adjoint sensitivity analysis is typically performed for temperature.

Chapter 26: Output Files

0D Simulation Output Files zero_d_adjoint_rank.out

zero_d_adjoint_rank.out

CONVERGE generates *zero_d_adjoint_rank.out* for a [mechanism tuning](#) simulation with [adjoint sensitivity analysis](#) (*i.e.*, when *zero_d_solver.in* > *zero_d_sensitivity_control* > *sensitivity_flag* = ADJOINT).

The *zero_d_adjoint_rank.out* file lists the combined ranking from all the 0D cases into one file, and ranks the reactions in decreasing order of sensitivity from all the cases.

Table 26.106: Description of *zero_d_adjoint_rank.out*.

Column Header (units)	Description	
1	Place (none)	Sensitivity rank of the reaction.
2	Reac_Num (none)	Reaction number in the new reaction mechanism file.

zero_d_end_time_species.out

CONVERGE generates *zero_d_end_time_species.out* when a list of species is specified in *zero_d_solver.in* > *zero_d_output_control* > *end_time_species*.

Table 26.107: Description of *zero_d_end_time_species.out*.

Column	Header (units)	Description
1	NUM (none)	Case number.
2	Time (seconds)	Simulation end time.
3	Temperature (K)	Temperature at end time.
4	Pressure (MPa)	Pressure at end time.
5+	<Species Name> (mass-fraction or mole-fraction)	Species mass fraction (if <i>zero_d_solver.in</i> > <i>zero_d_output_control</i> > <i>species_fraction_output_type</i> = MASS) or mole fraction (if <i>zero_d_solver.in</i> > <i>zero_d_output_control</i> > <i>species_fraction_output_type</i> = MOLE) at end time.

zero_d_sol_case<case ID>.out

CONVERGE generates *zero_d_sol_case<case ID>.out* for constant volume, constant pressure, constant temperature and pressure, or variable volume [zero-dimensional chemistry](#) simulations in which outputting additional output files is active (*i.e.*, for *zero_d_solver.in* > *zero_d_input_control* > *case_type* = CONSTANT_VOLUME, CONSTANT_PRESSURE, CONSTANT_TEMPERATURE_PRESSURE, or VARIABLE_VOLUME and *zero_d_solver.in* > *zero_d_output_control* > *output_file_flag* = 1). The <case ID> corresponds to the case ordering in *zero_d_cases.in* (*e.g.*, the first case in *zero_d_cases.in* would yield *zero_d_sol_case00001.out*).

Chapter 26: Output Files

0D Simulation Output Files zero_d_sol_case<case ID>.out

This file contains a header with the initial pressure, temperature, equivalence ratio, and EGR ratio, followed by temperature, pressure and species concentrations at each time-step. For species concentrations, CONVERGE writes one column for each species in the simulation.

Table 26.108: Description of zero_d_sol_case<case ID>.out.

Column	Header (units)	Description
1	Time (seconds)	The distance the flame front traveled.
2	Temperature (K)	The temperature at the specified location.
3	Pressure (Pa)	The mass flow at the specified location.
Varies	<Species Name> (mole/mass-fraction)	The species mass or mole fraction (based on the value of zero_d_solver.in > zero_d_output_control > species_fraction_output_type) at the specified time.

Engine Case Types

CONVERGE generates *zero_d_sol_case<case ID>.out* for [HCCI engine](#) and [engine RON and/or MON](#) case types in which the option to generate additional output files is active (*i.e.*, for [zero_d_solver.in](#) > zero_d_input_control > case_type = ENGINE, ENGINE_RON, ENGINE_MON, or ENGINE_RON_MON and [zero_d_solver.in](#) > zero_d_output_control > output_file_flag = 1). The <case ID> corresponds to the case ordering in [zero_d_cases.in](#) (*e.g.*, the first case in *zero_d_cases.in* would yield *zero_d_sol_case00001.out*). The following table describes the format of *zero_d_sol_case<case ID>.out* for these cases. A header records the initial pressure, temperature, equivalence ratio, and EGR ratio. Note that CONVERGE writes the species concentrations in mass and mole fraction.

Table 26.109: Description of zero_d_sol_case<case ID>.out for engine cases.

Column	Header (units)	Description
1	Crank (degrees)	Time in crank angle degrees.
2	Temperature (K)	The temperature at the specified piston location.
3	Pressure (Pa)	The mass flow at the specified piston location.
4	Volume (m ³)	Volume of the engine cylinder.
5	Heat flux (W/m ²)	Heat flux in the engine cylinder.
Varies	<Species Name>	The species mass or mole fraction (based on the value of zero_d_solver.in > zero_d_output_control > species_fraction_output_type) at the specified time.

Plug Flow Reactor

CONVERGE generates *zero_d_sol_case<case ID>.out* for [plug flow reactor simulations](#) in which the option to generate additional output files is active (*i.e.*, for simulations with [zero_d_solver.in](#) > zero_d_input_control > case_type =

Chapter 26: Output Files

0D Simulation Output Files zero_d_sol_case<case ID>.out

CONSTANT_PRESSURE_PLUG_FLOW.REACTOR and [zero_d_solver.in](#) >
`zero_d_output_control > output_file_flag = 1`). The <case ID> corresponds to the case ordering in [zero_d_cases.in](#) (e.g., the first case in `zero_d_cases.in` would yield `zero_d_sol_case00001.out`). The following table describes the format of `zero_d_sol_case<case ID>.out` for PFR cases. A header records initial pressure, temperature, equivalence ratio, and EGR.

Table 26.110: Description of `zero_d_sol_case<case ID>.out` for PFR cases.

Column	Header (units)	Description
1	Location (m)	Domain length.
2	Temperature (K)	Temperature in the reactor.
3	Velocity (m/s)	Velocity of the flow at the specified location.
Varies	<Species Name> (mole-fraction)	Mole fraction of the species at the specified location.

Well-Stirred Reactor

CONVERGE generates `zero_d_sol_case<case ID>.out` for [well-stirred reactor simulations](#) that do not include finding the ignition and/or extinction limits (i.e., for simulations with `zero_d_solver.in > zero_d_input_control > case_type = WELL_STIRRED_REACTOR` and `zero_d_solver.in > well_stirred_reactor_control > find_turning = OFF`). The following table describes the format of `zero_d_sol_case<case ID>.out` for WSR cases. A header records pressure, temperature, and inlet elemental equivalence ratio.

Table 26.111: Description of `zero_d_sol_case<case ID>.out` for WSR cases.

Column	Header (units)	Description
1	Time (seconds)	Time in seconds.
2	Residence_Time (seconds)	Residence time scale in the WSR.
3	Temperature (K)	Temperature in the WSR.
4	Pressure (MPa)	Pressure in the WSR.
5	Mass (kg)	Mass in the WSR.
Varies	<Species Name> (mole-fraction)	The species mole fraction.

CEQ

CONVERGE generates `zero_d_sol_case<case ID>.out` for all [chemical equilibrium simulations](#) (i.e., for all simulations with `zero_d_solver.in > zero_d_input_control > case_type = CEQ_CONSTANT_ENTHALPY_PRESSURE` or `CEQ_CONSTANT_TEMPERATURE_PRESSURE`). The following table describes the format of `zero_d_sol_case<case ID>.out` for cases with CEQ. A header records equivalence ratio,

Chapter 26: Output Files

0D Simulation Output Files zero_d_sol_case<case ID>.out

pressure, EGR, and initial and equilibrium temperature. Note that CONVERGE writes the species concentrations in mass and mole fraction.

Table 26.112: Description of zero_d_sol_case<case ID>.out for cases with CEQ.

Column	Header (units)	Description
1	Num	Variable number.
2	Variable	Variable name (<i>i.e.</i> , the species name from <i>mech.dat</i>).
3	Initial	The mass fraction value of the variable at the beginning of the simulation.
4	Equilibrium	The mass fraction equilibrium value of the variable.
5	Initial	The mole fraction value of the variable at the beginning of the simulation.
6	Equilibrium	The mole fraction equilibrium value of the variable.

CONVERGE does not generate *zero_d_sol_case<case ID>.out* for the engine RON/MON table case type (*i.e.*, for *zero_d_solver.in* > *zero_d_input_control* > *case_type* = *ENGINE_RON_MON_TABLE*).

26.7 1D Simulation Output Files

This section describes the output files written by the [1D chemistry utilities](#).

flamespeed.out

CONVERGE generates *flamespeed.out* for a [1D PISO or hybrid chemistry](#) simulation (*i.e.*, when *one_d_solver.in* > *one_d_general_control* > *solver_control* > *solver_type* = *PISO* or *HYBRID*). This file contains an entry at each time-step so that you can monitor several solution variables. In general, you do not need to consult this file.

Table 26.113: Description of flamespeed.out.

Column	Header (units)	Description
1	Time (seconds)	Time elapsed in the 1D simulation.
2	Flame_location (m)	Distance traveled by the flame front.
3	flamespeed_1 (m/s)	Flamespeed calculated via the finite derivatives method.
4	flamespeed_2 (m/s)	Flamespeed calculated via the progress variable.

one_d_flamespeed.out

CONVERGE generates *one_d_flamespeed.out* for all 1D [premixed laminar freely propagating flamespeed](#) simulations (*i.e.*, when *one_d_solver.in* > *one_d_general_control* > *case_type* = *LAMINAR_FREELY_PROPAGATING_FLAME*). This file contains the initial conditions for each flamespeed case along with the resultant flamespeed.

Chapter 26: Output Files

1D Simulation Output Files one_d_flamespeed.out

Table 26.114: Description of *one_d_flamespeed.out*.

Column	Header (units)	Description
1	Case Number	Case number of simulation in the order the cases appear in <i>one_d_cases.in</i> .
2	Temperature (K)	The initial temperature specified in <i>one_d_cases.in</i> .
3	Pressure (MPa)	The initial pressure specified in <i>one_d_cases.in</i> .
4	Phi_fuel_oxid	The equivalence ratio specified in <i>one_d_cases.in</i> .
5	EGR_ratio	The exhaust gas recirculation ratio specified in <i>one_d_cases.in</i> .
6	Phi_elem	Elemental equivalence ratio.
7	Flame_speed (m/s)	Calculated flamespeed. When <i>one_d_solver.in</i> > <i>one_d_general_control</i> > <i>solver_control</i> > <i>solver_type</i> = PISO or HYBRID, this is the flamespeed calculated via the progress variable.
8	Flame_thickness (m)	Calculated flame thickness.

one_d_end_distance_species.out

CONVERGE generates *one_d_end_distance_species.out* when a list of species is specified in [*one_d_solver.in*](#) > [*one_d_general_control*](#) > [*output_control*](#) > [*end_distance_species*](#).

Table 26.115: Description of *one_d_end_distance_species.out*.

Column	Header (units)	Description
1	NUM (none)	Case number.
2	Distance (cm)	Domain length specified in <i>one_d_solver.in</i> > <i>solver_control</i> > <i>domain_length</i> .
3	Temperature (K)	Temperature at the specified distance.
4	Density (g/cm ³)	Density at the specified distance.
5+	<Species Name> (mass-fraction or mole-fraction)	Species mass fraction (if <i>one_d_solver.in</i> > <i>one_d_general_control</i> > <i>output_control</i> > <i>species_fraction_output_type</i> = MASS) or mole fraction (if <i>one_d_solver.in</i> > <i>one_d_general_control</i> > <i>output_control</i> > <i>species_fraction_output_type</i> = MOLE) at the specified distance.

one_d_sens.out

CONVERGE generates *one_d_sens.out* for a [*mechanism tuning*](#) simulation with one-dimensional sensitivity (*i.e.*, when [*one_d_solver.in*](#) > [*one_d_newton_control*](#) > [*output_control*](#) > [*sensitivity_flag*](#) = 1).

Chapter 26: Output Files

1D Simulation Output Files one_d_sens.out

The *one_d_sens.out* file consolidates information from the [*one_d_sens_case<case ID>.out*](#) files. It contains the reactions from the reaction mechanism file sorted in decreasing order of sensitivity, as shown below in Figure 26.7. The rank of the reaction is the first column, and the subsequent columns list the reaction number from each case.

# column	1	2	3	4
# Place	Case_1_Reac_Num	Case_2_Reac_Num	Case_3_Reac_Num	
# (none)	(none)	(none)	(none)	(none)
#				
1	16	16	16	
2	-15	-27	-27	
3	-27	-15	-13	

Figure 26.7: Excerpt of *one_d_sens.out*.

Table 26.116: Description of *one_d_sens.out*.

Column	Header (units)	Description
1	Place (none)	The order of sensitivity for reactions from the reaction mechanism file.
2*	Case_<case ID>_Reac_Num (none)	The reaction number for this reaction in <case ID>, where <case ID> corresponds to the case ordering in <i>one_d_cases.in</i> . A positive reaction number indicates that the flamespeed increases as the pre-exponential factor increases, and vice versa. A negative reaction number indicates that the flamespeed decreases as the pre-exponential factor increases, and vice versa. This column is repeated for each case.

one_d_sens_case<caseID>.out

CONVERGE generates *one_d_sens_case<case ID>.out* for a [1D simulation with sensitivity analysis](#) (i.e., when [*one_d_solver.in*](#) > *one_d_newton_control* > *output_control* > *sensitivity_flag* = 1). In these files, a header records the pressure, unburned temperature, equivalence ratio, and EGR ratio by volume. The following table describes the format of the rest of *one_d_sens_case<case ID>.out*, where <case ID> corresponds to the case ordering in [*one_d_cases.in*](#) (e.g., the first case in *one_d_cases.in* would yield *one_d_sens_case00001.out*).

Table 26.117: Description of *one_d_sens_case<case ID>.out*.

Column	Header (units)	Description
1	Place (none)	Sensitivity ranking for the reaction in this row based on the absolute value of the sensitivity coefficient.
2	Reaction (none)	Reaction number.
3	Sensitivity (none)	Maximum sensitivity coefficient value (positive or negative).
4	Reaction_Type (none)	The chemical equation of the reaction.

Chapter 26: Output Files

1D Simulation Output Files one_d_sens_case<caseID>.out

This output file contains a list of the species in the mechanism and the sensitivity factor of each, sorted by largest absolute value to smallest absolute value. The case details are written as comments at the top of the file.

```
# CONVERGE 3.1
# column      1          2          3          4
#       Place     Reaction   Sensitivity Reaction_Type
#       (none)    (none)     (none)     (none)
#
# One-D premixed flame at pressure = 1.000000e+00 bar, unburned temperature = 3.000000e+02
K, phi = 1.000000e+00, EGR ratio by volume = 0.000000e+00
# Grid Points: 148   Species: 53   Reactions: 325   Flame Speed: 3.8767397e+01
1           38      5.503337e-01  # H+O2=O+OH
2           52      -1.648043e-01 # H+CH3 (+M)=CH4 (+M)
3           99      1.191751e-01  # OH+CO=H+CO2
4           97      7.664641e-02  # OH+CH3=CH2 (S)+H2O
5          119      7.428686e-02  # HO2+CH3=OH+CH3O
6           35      -7.211104e-02 # H+O2+H2O=HO2+H2O
...
```

Figure 26.8: Example *one_d_sens_case<case ID>.out* file.

one_d_sens_rank.out

CONVERGE generates *one_d_sens_rank.out* for a [mechanism tuning](#) simulation with one-dimensional sensitivity (*i.e.*, when *one_d_solver.in* > *one_d_newton_control* > *output_control* > *sensitivity_flag* = 1). This file lists the combined ranking from all the 1D cases into one file, and ranks the reactions in decreasing order of sensitivity from all the 1D cases.

Table 26.118: Description of *one_d_sens_rank.out*.

Column Header (units)	Description
1	Place (none)
2	Reac_Num (none)

one_d_sol_case<case ID>.out

CONVERGE generates *one_d_sol_case<case ID>.out* for [1D premixed laminar freely propagating flamespeed](#) simulations in which outputting additional output files is active (*i.e.*, when *one_d_solver.in* > *one_d_general_control* > *output_control* > *output_file_flag* = 1). The <case ID> corresponds to the case ordering in *one_d_cases.in* (*e.g.*, the first case in *one_d_cases.in* would yield *one_d_sol_case00001.out*). This file contains a header with the initial pressure, temperature, and equivalence ratio, followed by domain-specific values of temperature and mass flow.

Chapter 26: Output Files

1D Simulation Output Files one_d_sol_case<case ID>.out

Table 26.119: Description of *one_d_sol_case<case ID>.out*.

Column	Header (units)	Description
1	Distance (cm)	The distance the flame front traveled.
2	Temperature (K)	The temperature at the specified location.
3	Mass_Flow (g/s-cm ²)	The mass flow at the specified location.
4	Density (g/cm ³)	The density at the specified location.
5	Velocity (cm/s)	The velocity at the specified location.
Varies	<Species Name>	The species mass fraction at the specified location. Repeated for each species.

one_d_strain_rate.out

CONVERGE generates *one_d_strain_rate.out* for 1D [laminar counterflow calculations](#) (*i.e.*, when *one_d_solver.in* > *one_d_general_control* > *case_type* = *LAMINAR_COUNTERFLOW_FLAME*).

Table 26.120: Description of *one_d_strain_rate.out*

Column	Header (units)	Description
1	Case	Case number of simulation in the order the cases appear in <i>one_d_cases.in</i> .
2	Pressure (MPa)	The initial pressure specified in <i>one_d_cases.in</i> . The simulation is approximately a constant pressure simulation.
3	Maximum_temperature (K)	The maximum temperature reached by the counterflow flame.
4	Global_strain_rate (1/s)	The strain rate of the fuel under the specified conditions.

schmidt_species_case<number>.dat

CONVERGE generates *schmidt_species_case<number>.out* for a 1D laminar flamespeed calculation with Newton solver in which the Schmidt species are calculated (*i.e.*, when *one_d_solver.in* > *one_d_newton_control* > *output_control* > *schmidt_species_flag* = 1). This file has the same format as [*schmidt_species.dat*](#). To use this file in a CONVERGE simulation, rename it to *schmidt_species.dat* and include it in your case setup.

tlf_table_failed_cases.out

CONVERGE generates *tlf_table_failed_cases.out* when [generating a TLF table](#) if a case fails to converge before the case time limit is reached or if the solver is terminated before a case has been calculated. This file contains the initial conditions for each of the 1D premixed laminar flamespeed cases that failed to converge or run.

Table 26.121: Description of *tlf_table_failed_cases.out*.

Chapter 26: Output Files

1D Simulation Output Files tlf_table_failed_cases.out

Column	Header (units)	Description
1	Case Number	Case number of simulation in the order in which the cases are generated from the values and ranges in <i>one_d_template.in</i> . The case number matches the case number in <i>one_d_flamespeed.out</i> .
2	Temperature (K)	The initial temperature of the case.
3	Pressure (MPa)	The initial pressure of the case.
4	Phi_elem	The elemental equivalence ratio of the case.
5	EGR_ratio	The exhaust gas recirculation ratio of the case.
6	Fuel_frac	The fuel mass fraction of the case.

26.8 Mechanism Reduction Output Files

This section describes the output files that CONVERGE writes after running the [mechanism reduction utility](#).

ignition_det.out

CONVERGE generates *ignition_det.out* after it generates the reduced mechanism using the zero-dimensional [mechanism reduction utility](#) (*i.e.*, when *mechanism_reduction.in* > *drgep_control* > *zero_d_control* > *active* = 1 and/or when *mechanism_reduction.in* > *sensitivity_analysis_control* > *zero_d_control* > *active* = 1). This file has the same format as described in an [earlier section](#) for constant volume, constant pressure, constant temperature and pressure, CEQ, variable volume, and PFR case types.

ignition_ske.out

CONVERGE generates *ignition_ske.out* after it generates the reduced mechanism using the zero-dimensional [mechanism reduction utility](#) (*i.e.*, when *mechanism_reduction.in* > *drgep_control* > *zero_d_control* > *active* = 1 and/or when *mechanism_reduction.in* > *sensitivity_analysis_control* > *zero_d_control* > *active* = 1). This file contains the ignition delay data for the reduced mechanism. This file is for informational purposes only (*i.e.*, the utility does not use this file during the mechanism reduction process). Note that if lumping is activated, the utility will also write *ignition_lump.out*. The *ignition_lump.out* file has the same structure as the *ignition_ske.out* file.

Chapter 26: Output Files

Mechanism Reduction Output Files ignition_ske.out

Table 26.122: Description of *ignition_ske.out* and *ignition_lump.out*.

Column	Header (units)	Description
1	NUM	Case number.
2	Pressure (MPa)	Pressure.
3	Temperature (K)	Temperature.
4	Phi_fuel_oxid (<i>none</i>)	Equivalence ratio.
5	EGR_ratio (<i>none</i>)	EGR ratio.
6	Phi_elem (<i>none</i>)	Elemental equivalence ratio.
7	Time1 (sec)	Ignition delay time.

mech_lump.dat

CONVERGE generates *mech_lump.dat* after running the [mechanism reduction utility](#) with isomer lumping activated (i.e., when *mechanism_reduction.in* > *isomer_lumping_control* > *zero_d_control* > *active* = 1). The mechanism reduction tool extracts the relevant species and reaction information for the skeletal mechanism from the original reaction mechanism file and writes this information to the *mech_ske.dat* file. The structure of the *mech_lump.dat* file is identical to the [reaction mechanism](#) file. The 0D utility uses this file to calculate the ignition delay times for the skeletal mechanism.

mech_ske.dat

CONVERGE generates *mech_ske.dat* after running the [mechanism reduction utility](#). The mechanism reduction tool extracts the relevant species and reaction information for the skeletal mechanism from the original reaction mechanism file and writes this information to the *mech_ske.dat* file. The structure of the *mech_ske.dat* file is identical to the [reaction mechanism](#) file. The 0D utility uses this file to calculate the ignition delay times and laminar flamespeed times for the skeletal mechanism.

one_d_flamespeed_det.out

CONVERGE generates *one_d_flamespeed_det.out* after it generates the reduced mechanism using the one-dimensional [mechanism reduction utility](#) (i.e., when *mechanism_reduction.in* > *drgep_control* > *one_d_control* > *active* = 1 and/or when *mechanism_reduction.in* > *sensitivity_analysis_control* > *one_d_control* > *active* = 1). This file contains the laminar flamespeed data for the comprehensive mechanism and has the same format as [one_d_flamespeed.out](#).

one_d_flamespeed_ske.out

CONVERGE generates *one_d_flamespeed_ske.out* after it generates the reduced mechanism using the one-dimensional [mechanism reduction utility](#) (i.e., when *mechanism_reduction.in* > *drgep_control* > *one_d_control* > *active* = 1 and/or when *mechanism_reduction.in* > *sensitivity_analysis_control* > *one_d_control* > *active* = 1). This file contains the laminar

Chapter 26: Output Files

Mechanism Reduction Output Files one_d_flamespeed_ske.out

flamespeed data for the skeletal mechanisms. This file is for informational purposes only (*i.e.*, the utility does not use this file during the mechanism reduction process).

Table 26.123: Description of *one_d_flamespeed_ske.out*.

Column	Header (units)	Description
1	Case (<i>none</i>)	Case number.
2	Temperature (K)	Temperature.
3	Pressure (MPa)	Pressure.
4	Phi_fuel_oxid (<i>none</i>)	Equivalence ratio.
5	EGR_ratio (<i>none</i>)	EGR ratio.
6	Phi_elem (<i>none</i>)	Elemental equivalence ratio.
7	Flame_speed (m/s)	Laminar flamespeed.
8	Flame_thickness (m)	Laminar flame thickness.

26.9 Mechanism Tune Output Files

This section describes the output files that CONVERGE writes after running the [mechanism tune utility](#).

mechanism_tune.out

CONVERGE generates *mechanism_tune.out* after running the [mechanism tune utility](#).

Chapter 26: Output Files

Mechanism Tune Output Files mechanism_tune.out

# column	1	2
#	Order (none)	Reaction (none)
1	67	
2	52	
3	78	
4	76	
5	103	
6	99	
7	77	
8	64	
9	98	
10	91	
11	59	
12	60	
13	107	
14	13	
15	95	
16	66	
17	20	
18	23	
19	24	
20	33	
21	34	
22	63	
23	79	
24	65	
25	84	

Figure 26.9: An example *mechanism_tune.out* file.

nominal.out

CONVERGE generates *nominal.out* after running the [mechanism tune utility](#).

Table 26.124: Description of *nominal.out*.

Column	Header	Description
1	Gen#	Generation number.
2	Ind#	Individual number.
Varies	A_coeff_<ID>	Value of this coefficient for this generation and individual. Repeat for each metric.
Varies	fitness	Fitness of this coefficient for this generation and individual.
Varies	merit	Merit of this coefficient for this generation and individual.

nominal_perform.out

CONVERGE generates *nominal_perform.out* after running the [mechanism tune utility](#).

Table 26.125: Description of *nominal_perform.out*.

Column	Header	Description
1	Gen#	Generation number.
2	Ind#	Individual number.

Chapter 26: Output Files

Mechanism Tune Output Files nominal_perform.out

Column Header	Description
Varies <metric name>	Value of this metric for this generation and individual. Repeat for each metric.
Varies fitness	Fitness of this coefficient for this generation and individual.
Varies merit	Merit of this coefficient for this generation and individual.

tune_group_reac_sens.out

CONVERGE generates *tune_group_reac_sens.out* after running the [mechanism tune utility](#) with NLOpt with 1D group dependency active (*i.e.*, *mechanism_tune.in* > *nlopt_control* > *main_reaction_control* > *group_dependency_control* > *active* = 1). After listing the reaction numbers in each group, CONVERGE adds -1 where necessary to ensure that all columns in the file contain the same number of items.

Table 26.126: Description of *tune_group_reac_sens.out*.

Column Header	Description
1 Group_A	List of reaction numbers in group A.
2 Group_B	List of reaction numbers in group B.
3 Group_C	List of reaction numbers in group C.
4 Group_D	List of reaction numbers in group D.
5 Group_E	List of reaction numbers in group E.

tune_perform.out

CONVERGE generates *tune_perform.out* after running the [mechanism tune utility](#) with NLOpt (*i.e.*, *mechanism_tune.in* > *solver_control* = *NLOPT*).

Table 26.127: Description of *tune_perform.out*.

Column Header (units)	Description
1 Function (<i>none</i>)	Number of times the tuning algorithm has repeated (<i>i.e.</i> , 0 refers to the original mechanism).
2 Error (<i>none</i>)	Error between the flamespeed or ignition delay in column 3+ and the target flamespeed or ignition delay.
3+ flamespeed<number> (<i>m/s</i>)/ignitiondelay<number> (<i>sec</i>)	Flamespeed or ignition delay. This row is repeated for each case in <i>zero_d_cases.in</i> or <i>one_d_cases.in</i> .

tune_reaction.out

CONVERGE generates *tune_reaction.out* after running the [mechanism tune utility](#) with NLOpt (*i.e.*, *mechanism_tune.in* > *solver_control* = *NLOPT*).

Chapter 26: Output Files

Mechanism Tune Output Files tune_reaction.out

Table 26.128: Description of *tune_reaction.out*.

Column	Header (units)	Description
1	Function (<i>none</i>)	Number of times the tuning algorithm has repeated (<i>i.e.</i> , 0 refers to the original mechanism).
2+	A_reaction_<reaction number> or E_reaction_<reaction number>	Reaction A factor or E_a value. This row is repeated for each sensitive reaction, where <reaction number> refers to the reaction number in <i>mech_check.out</i> .

tune_reaction_factors.out

CONVERGE generates *tune_reaction_factors.out* after running the [mechanism tune utility](#) with NLOpt (*i.e.*, [*mechanism_tune.in*](#) > *solver_control* = NLOPT).

Table 26.129: Description of *tune_reaction_factors.out*.

Column	Header (units)	Description
1	Function (<i>none</i>)	Number of times the tuning algorithm has repeated (<i>i.e.</i> , 0 refers to the original mechanism).
2+	A_reaction_<reaction number>, E_reaction_<reaction number>, or Group_<letter>	Multiplier used for the A factor, E_a value, or group. <reaction number> refers to the reaction number from <i>mech_check.out</i> and <letter> refers to the group listed in <i>tune_group_reac_sens.out</i> .

tune_reaction_results.out

CONVERGE generates *tune_reaction_results.out* after running the [mechanism tune utility](#) with NLOpt (*i.e.*, [*mechanism_tune.in*](#) > *solver_control* = NLOPT).

Table 26.130: Description of *tune_reaction_results.out*.

Column	Header (units)	Description
1	Reaction	Reaction number from <i>mech_check.out</i> .
Varies	A_ori	Original A factor. This column is written only if <i>mechanism_tune.in</i> > <i>nlopt_control</i> > <i>main_reaction_control</i> > <i>a_coeff_control</i> > <i>active</i> = 1.
Varies	A_new	New (tuned) A factor. This column is written only if <i>mechanism_tune.in</i> > <i>nlopt_control</i> > <i>main_reaction_control</i> > <i>a_coeff_control</i> > <i>active</i> = 1.
Varies	E_ori	Original E_a value. This column is written only if <i>mechanism_tune.in</i> > <i>nlopt_control</i> > <i>main_reaction_control</i> > <i>ea_control</i> > <i>active</i> = 1.
Varies	E_new	New (tuned) E_a value. This column is written only if <i>mechanism_tune.in</i> > <i>nlopt_control</i> > <i>main_reaction_control</i> > <i>ea_control</i> > <i>active</i> = 1.

Chapter 26: Output Files

Mechanism Tune Output Files tune_target_results.out

tune_target_results.out

CONVERGE generates *tune_target_results.out* after running the [mechanism tune utility](#) with NLOpt (*i.e.*, [mechanism tune.in](#) > solver_control = NLOPT).

Table 26.131: Description of *tune_target_results.out*.

Column	Header	Description
1	Case	Case number from <i>zero_d_cases.in</i> or <i>one_d_cases.in</i> .
2	Target	Target value from <i>mechanism_tune.in</i> .
3	Untuned	Initial value from untuned mechanism.
4	Tuned	Value from tuned mechanism.
5	Type	Target type, either <i>flamespeed</i> or <i>ignitiondelay</i> followed by a number.

26.10 Surrogate Blender Output File

This section describes the output file associated with CONVERGE Studio's [surrogate blender](#) tool.

blender.out

When you run CONVERGE Studio's [surrogate blender](#) tool, CONVERGE Studio generates a dialog box that lists the optimal mixture composition, target fuel properties, and surrogate fuel properties. This dialog box has an option to generate a *blender.out* file that includes these data.

Remember that you must convert the mole fractions in *blender.out* to mass fractions for use in a CONVERGE CFD simulation.

Table 26.132: Description of *blender.out*. The two types of rows (fuel and property/target/surrogate) are repeated as necessary.

#	mole_fraction	
Composition		
Fuel <i>i</i>	Mole fraction of fuel <i>i</i> in optimal surrogate mixture.	
#	Property	Target
Property <i>j</i>	Property <i>j</i> for the target fuel.	Property <i>j</i> for the surrogate mixture.

Chapter 26: Output Files

Surrogate Blender Output File blender.out

```
# Composition (mole-fraction)
C9H12          2.368421e-01
NC12H26        3.947368e-01
IC8H18          2.894737e-01
X135C9H12      7.894737e-02

# Property      Target      Surrogate
MW             1.420000e+02   1.340029e+02
TSI            2.140000e+01   2.406250e+01
H/C            1.960000e+00   1.896104e+00
DEN@288.15    8.040000e+02   7.750001e+02
DCN            4.710000e+01   4.149964e+01
```

Figure 26.10: Sample *blender.out* file.

26.11 Pathway Flux Analysis Output Files

This section describes the output files associated with [pathway flux analysis](#) (PFA).

graphviz_input_<number>.out

CONVERGE generates *graphviz_input_<number>.out* after performing a [pathway flux analysis](#) (PFA). The <number> corresponds to the *i*-th case in [*pfa.in*](#). The *graphviz_input_<number>.out* file contains information about the reactions that satisfy the filter conditions set in *pfa.in* > *pathway_diagram_control* and the line width, color, and label. CONVERGE Studio can create a pathway diagram based on the information in this file.

rate_tab_<number>.out

CONVERGE generates *rate_tab_<number>.out* after performing [pathway flux analysis](#) (PFA). The <number> corresponds to the *i*-th case in [*pfa.in*](#). The *rate_tab_<number>.out* file contains information about rates of the reactions. Use CONVERGE Studio to visualize the output of this file.

Table 26.133: Description of *rate_tab_<number>.out*.

Column	Header (units)	Description
1	Time (seconds or crank angle degrees) or Distance (meters)	For a 0D simulation, time in seconds if <i>inputs.in</i> > <i>simulation_control</i> > <i>crank_flag</i> = 0 or in crank angle degrees if <i>crank_flag</i> is non-zero. For a 1D simulation, distance in meters.
Varies	<Species Name>	Reaction number in which the species appears followed by the reaction rate at the given time-step.

spec_mol_<number>.out

CONVERGE generates *spec_mol_<number>.out* after performing [pathway flux analysis](#) (PFA). The <number> corresponds to the *i*-th case in [*pfa.in*](#). The *spec_mol_<number>.out* file contains information about the mole fraction of the species.

Table 26.134: Description of *spec_mol_<number>.out*.

Chapter 26: Output Files

Pathway Flux Analysis Output Files spec_mol_<number>.out

Column	Header (units)	Description
1	Time (seconds or crank angle degrees) or Distance (meters)	For a 0D simulation, time in seconds if inputs.in > simulation_control > crank_flag = 0 or in crank angle degrees if crank_flag is non-zero. For a 1D simulation, distance in meters.
2	Temperature (K)	Temperature of the reaction.
3	Pressure (MPa)	Pressure of the reaction.
Varies	<Species Name>	The mole fraction of the species.

26.12 CONGO Output Files

[CONGO](#) generates output in the [individual run directories](#) of the CONVERGE simulations and in the [main CONGO folder](#). CONGO uses the data written to the individual run directories to create the output files in the main CONGO folder.

Individual Run Directory Output Files

The output files [CONGO](#) generates in the individual run directories contain information regarding the values of the parameters selected by the GA or the DoE, as well as the performance (fitness and merit) of the individuals.

param.<run number>-<generation number>

This file contains the parameter values CONGO selects for each run. You can specify the parameters that will be varied by CONGO in the [case.in](#) file.

ga_output.<run number>-<generation number>

This file contains the values of the response variables (both *performance* and *constraint*) for each run. CONGO uses a UDF to generate this file. The name of this file will be consistent with [merit.in](#) > *output_file_name*. We recommend that you enter *ga_output* for consistency.

<output file name>-<run number>-<generation number>.out

CONGO creates a uniquely named instance of all CONVERGE output files (e.g., *liquid_parcel_ecn-6-47.out*) for each run in each case directory.

Main CONGO Folder Output Files

In the main [CONGO](#) folder, CONGO writes ASCII output files that summarize the results of the CONGO GA or DoE, based on information it reads from the *ga_output* and *param* output files it writes to the individual run directories.

Output files from CONGO record the results of the GA or DoE. These output files are saved in the main CONGO directory and serve to indicate the progress and eventual conclusions of the CONGO run. Some of these files include information regarding the parameter values chosen by CONGO (*bestcases.out*, *convergance.out*, *output.out*, *output_converged.out*, *bestcases.out*), while others contain information on response (performance and constraint) variable results for individual runs (*perform.out*, *bestcases_perform.out*). Many of these files

Chapter 26: Output Files

CONGO Output Files Main CONGO Folder Output Files

also contain the fitness and merit calculated for each individual. Others contain only the fitness and merit of select individual runs (*congo_max.out*, *congo_micro.out*).

Most of these files apply only to GA cases. The files *output.out*, *perform.out*, and *output_converged.out* are written for both GA and DoE cases.

- *congo_max.out* lists the maximum merit (merit for the best-so-far individual) of each generation.
- *congo_micro.out* lists the generations in which the population achieved micro-convergence, and the best-so-far merit at those generations.

You can plot the *congo_max.out* and *congo_micro.out* files together to show the maximum merit vs. the generation number, with markers indicating micro-convergence events. Figure 26.11 below shows an example plot of *congo_max.out* and *congo_micro.out*.

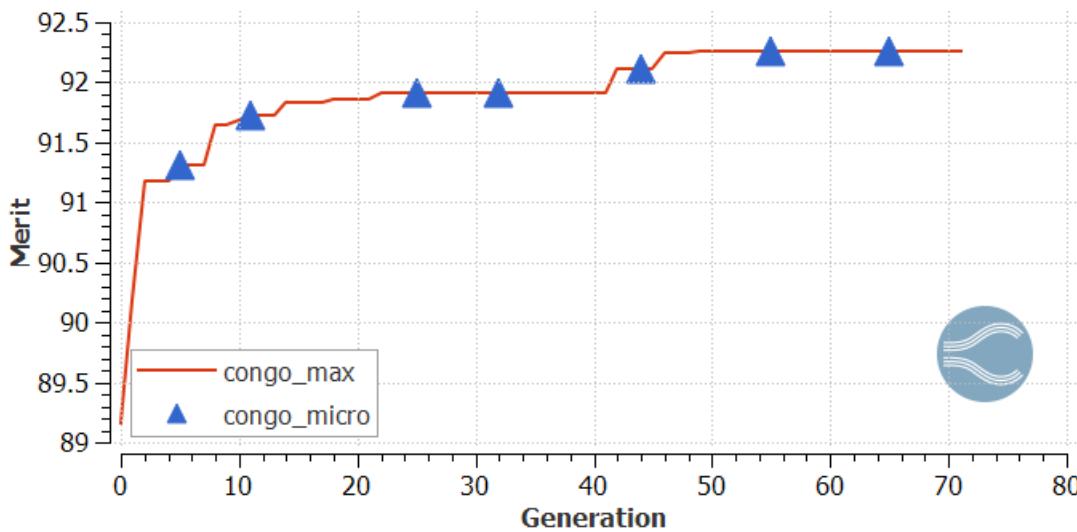


Figure 26.11: Results from *congo_max.out* and *congo_micro.out*.

- *output.out* lists model parameters selected by CONGO for all individuals in all generations in the GA or DoE. This file also lists the merit (and fitness, for multi-mode CONGO cases) for each individual.
- *perform.out* lists the values CONGO calculated for the response variables (performance and constraint) for all individuals in all generations in the GA or DoE. This file also lists the merit (and fitness, for multi-mode CONGO cases) for each individual.
- *bestcases.out* lists model parameters selected by CONGO for each best-so-far run in the GA. This file also lists the merit (and fitness, for multi-mode CONGO cases) for each best-so-far individual.

Chapter 26: Output Files

CONGO Output Files Main CONGO Folder Output Files

- *bestcases_perform.out* lists the values CONGO calculated for the response variables (performance and constraint) for each best-so-far run in the GA. This file also lists the merit (and fitness, for multi-mode CONGO cases) for each best-so-far individual.
- *convergance.out* lists the convergence criteria for each of the model parameters for each generation in the GA. These are not the actual values for the model parameters selected by CONGO, they are simply indicators of how close a generation came to a micro convergence event. The last column in this file also indicates if the generation reached a micro convergence event. The generations marked as "1" in the *Converged* column are the generations included in the *congo_micro.out* file.
- *output_converged.out* lists the actual values of the model parameters selected by CONGO for each individual of each generation that reached micro-convergence.
- *congo_dna.out* lists the binary string of each individual's DNA (for every generation), typically 30 bits for each parameter. The DNA of each individual is followed by the merit and DNA of the best-so-far individual. The end of this file contains sequencing information to define the random number generator used by CONGO. You can use this file to [restart the GA any any generation](#). Alternatively, you can use the *restart.in* file generated automatically by CONGO. *restart.in* contains information for only the most recent generation.
- *genetics.out* lists detailed information about the GA. It shows the process of the tournament and the shuffling of parents and children.

Chapter



27

References

27 References

- Abraham, J., Bracco, F.V., and Reitz, R.D., "Comparisons of Computed and Measured Premixed Charge Engine Combustion," *Combustion and Flame*, 60(3), 309-322, 1985. DOI: 10.1016/0010-2180(85)90036-7
- Abramzon, B. and Sirignano, W.A., "Droplet Vaporization Model for Spray Combustion Calculations," *International Journal of Heat and Mass Transfer*, 32(9), 1605-1618, 1989. DOI: 10.1016/0017-9310(89)90043-4
- Adachi, M., Tanaka, D., Hojyo, Y., Al-Roub, M., Senda, J., and Fujimoto, H., "Measurement of fuel vapor concentration in flash boiling spray by infrared extinction/scattering technique," *JSME Review*, 17, 231-237, 1996. DOI: 10.1016/0389-4304(96)00025-2
- Adelman, H.G., "A Time Dependent Theory of Spark Ignition," *Symposium (International) on Combustion*, 18(1), 1333-1342, 1981. DOI: 10.1016/S0082-0784(81)80137-3
- Alexander, F.J. and Garcia, A.J., "The Direct Simulation Monte Carlo Method," *Computers in Physics*, 11(6), 588-593, 1997. DOI: 10.1063/1.168619
- Amsden, A.A., O'Rourke, P.J., and Butler, T.D., "KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays," Los Alamos National Laboratory Technical Report LA-11560-MS, 1989.
- Amsden, A.A., "KIVA-3V: A Block Structured KIVA Program for Engines with Vertical or Canted Valves," Los Alamos National Laboratory Technical Report LA-13313-MS, 1997.
- Angelberger, C., Poinsot, T., and Delhaye, B., "Improving Near-Wall Combustion and Wall Heat Transfer Modeling in SI Engines Computations," SAE Paper 972881, 1997. DOI: 10.4271/972881
- Annand, W.J.D., "Heat Transfer in the Cylinders of Reciprocating Internal Combustion Engines," *Proceedings of the Institute of Mechanical Engineers*, 177, 973-996, 1963.
- Appel, J., Bockhorn, H., and Frenklach, M., "Kinetic Modeling of Soot Formation with Detailed Chemistry and Physics: Laminar Premixed Flames of C₂ Hydrocarbons," *Combustion and Flame*, 121, 122-136, 2000. DOI: 10.1016/S0010-2180(99)00135-2
- Arabnejad, H., Mansouri, A., Shirazi, S.A., and McLaury, B.S., "Development of mechanistic erosion equation for solid particles," *Wear*, 332-333, 1044-1050, 2015. DOI: 10.1016/j.wear.2015.01.031

- Arcoumanis, C. and Gavaises, M., "Linking Nozzle Flow with Spray Characteristics in a Diesel Fuel Injection System," *Atomization and Sprays*, 8(3), 307-347, 1998. DOI: 10.1615/AtomizSpr.v8.i3.50
- Ashgriz, N. and Poo, J.Y., "Coalescence and Separation in Binary Collisions of Liquid Drops," *Journal of Fluid Mechanics*, 221, 183-204, 1990. DOI: 10.1017/S0022112090003536
- Aubagnac-Karkar D., "Sectional soot modeling for Diesel RANS simulations," Ph.D. thesis, Centrale Supelec, Gif-sur-Yvette, France, 2014.
- Aubagnac-Karkar, D., El Bakali, A., and Desgroux, P., "Soot Particles Inception and PAH Condensation Modelling Applied in a Soot Model Utilizing a Sectional Method," *Combustion and Flame*, 189, 190-206, 2018. DOI: 10.1016/j.combustflame.2017.10.027
- Aulisa, E., Manservisi, R., Scardovelli, R., and Zaleski, S., "Interface Reconstruction with Least-Squares Fit and Split Advection in Three-Dimensional Cartesian Geometry," *Journal of Computational Physics*, 225(2), 2301-2319, 2007. DOI: 10.1016/j.jcp.2007.03.015
- Babajimopoulos, A., Assanis, D.N., Flowers, D.L., Aceves, S.M., and Hessel, R.P., "A Fully Coupled Computational Fluid Dynamics and Multi-Zone Model with Detailed Chemical Kinetics for the Simulation of Premixed Charge Compression Ignition Engines," *International Journal of Engine Research*, 6(5), 497-512, 2005. DOI: 10.1243/146808705X30503
- Bai, C. and Gosman, A. "Development of Methodology for Spray Impingement Simulation," SAE Paper 950283, 1995. DOI: 10.4271/950283
- Balthasar, M., Mauss, F., and Wang, H., "A Computational Study of the Thermal Ionization of Soot Particles and its Effect on their Growth in Laminar Premixed Flames," *Combustion and Flame*, 129(1-2), 204-216, 2002. DOI: 10.1016/S0010-2180(02)00344-9
- Barth, T.J. and Jespersen, D.C., "The Design and Application of Upwind Schemes on Unstructured Meshes," AIAA Paper 1989-0366, 1989. DOI: 10.2514/6.1989-366
- Barths, H., Antoni, C., and Peters, N., "Three-Dimensional Simulation of Pollutant Formation in a DI Diesel Engine Using Multiple Interactive Flamelets," SAE Paper 982459, 1998. DOI: 10.4271/982459
- Beale, J.C., "Modeling Fuel Injection using the Kelvin-Helmholtz/Rayleigh-Taylor Hybrid Atomization Model in KIVA-3V," M.S. Thesis, University of Wisconsin-Madison, Madison, WI, United States, 1999.
- Bedford, K.W. and Yeo, W.K., "Conjunctive Filtering Procedures in Surface Water Flow and Transport," *Large Eddy Simulation of Complex Engineering and Geophysical Flows*, eds. Galperin, B., and Orszag, S., Cambridge University Press, 1993.

- Berni, F., Cicalese, G., Fontanesi, S., "A Modified Thermal Wall Function for the Estimation of Gas-to-Wall Heat Fluxes in CFD In-Cylinder Simulations of High Performance Spark-Ignition Engines," *Applied Thermal Engineering*, 115, 1045-1062, 2017. DOI: 10.1016/j.applthermaleng.2017.01.055
- Bianchi, G.M. and Pelloni, P., "A Cavitation-Induced Atomization Model for High-Pressure Diesel Spray Simulations," *32nd International Symposium on Automotive Technology and Automation*, Vienna, Austria, 1999.
- Bilicki, Z. and Kestin, J., "Physical Aspects of the Relaxation Model in Two-Phase Flow," *Proceedings of the Royal Society of London A*, 428(1875), 379-397, 1990. DOI: 10.1098/rspa.1990.0040
- Birkhold, F., "Selektive Katalytische Reduktion von Stickoxiden in Kraftfahrzeugen: Untersuchung der Einspritzung von Harnstoffwasserlösung," Ph.D. Thesis, University of Karlsruhe, Karlsruhe, Baden-Württemberg, Germany, 2007.
- Blint, R.J., "The Relationship of the Laminar Flame Width to Flame Speed", *Combustion Science and Technology*, 49(1-2), 79-92, 1986. DOI: 10.1080/00102208608923903
- Bodony, D.J., "Analysis of Sponge Zones for Computational Fluid Mechanics", *Journal of Computational Physics*, 212(2), 681-702, 2006. DOI: 10.1016/j.jcp.2005.07.014
- Bougrine, S., Richard, S., Colin, O., Veynante, D., "Fuel Composition Effects on Flame Stretch in Turbulent Premixed Combustion: Numerical Analysis of Flame-Vortex Interaction and Formulation of a New Efficiency Function," *Flow, Turbulence and Combustion*, 93, 259-281, 2014. DOI: 10.1007/s10494-014-9546-4
- Brackbill, J.U., Kothe, D.B., and Zemach, C., "A Continuum Method for Modeling Surface Tension", *Journal of Computational Physics*, 100(2), 335-354, 1992. DOI: 10.1016/0021-9991(92)90240-Y
- Brennen, C.E., *Cavitation and Bubble Dynamics*, Oxford University Press, 1995.
- Brentner, K.S., "An efficient and robust method for predicting helicopter high-speed impulsive noise," *Journal of Sound and Vibration*, 203, 87-100, 1997. DOI: 10.1006/jsvi.1996.0834
- Briggs, W.L., Henson, V.E., and McCormick, S.F., *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, 2000.
- Bruneaux, G., Poinsot, T., and Ferziger, J.H., "Premixed flame-wall interaction in a turbulent channel flow: Budget for the flame surface density evolution equation and modelling," *Journal of Fluid Mechanics*, 349, 191-219, 1997.

- Cant, R.S. and Bray, K.N.C., "Strained Laminar Flamelet Calculations of Premixed Turbulent Combustion in a Closed Vessel," *Symposium (International) on Combustion*, 22(1), 791-799, 1989. DOI: 10.1016/S0082-0784(89)80088-8
- Casalino, D., "An advanced time approach for acoustic analogy predictions," *Journal of Sound and Vibration*, 261, 583-612, 2003. DOI: 10.1016/S0022-460X(02)00986-0
- Cebeci, T. and Cousteix, J., *Modeling and Computation of Boundary-Layer Flows*, Horizons Publishing Inc., 2005.
- Chang, J., Guralp, O., Filipi, Z., Assanis, D., Kuo, T.-W., Najt, P., and Rask, R., "New Heat Transfer Correlation for an HCCI Engine Derived from Measurements of Instantaneous Surface Heat Flux," SAE Paper 2004-01-2996, 2004. DOI: 10.4271/2004-01-2996
- Charlette, F., Meneveau, C., and Veynante, D., "A power-law flame wrinkling model for LES of premixed turbulent combustion Part I: non-dynamic formulation and initial tests", *Combustion and Flame*, 131(1-2), 159-180, 2002. DOI: 10.1016/S0010-2180(02)00400-5
- Chatterjee, D., "Detaillierte Modellierung von Abgaskatalysatoren," Ph.D. Thesis, Ruprecht-Karls-Universität Heidelberg, Heidelberg, Germany, 2001.
- Chen, H., Reuss, D.L., Hung, D.L.S., Sick, V., "A Practical Guide for Using Proper Orthogonal Decomposition in Engine Research," *International Journal of Engine Research*, 14(4), 307-319, 2013. DOI: 10.1177/1468087412455748
- Chiang, C.H., Raju, M.S., and Sirignano, W.A., "Numerical Analysis of a Convecting, Vaporizing Fuel Droplet with Variable Properties," *International Journal of Heat and Mass Transfer*, 35(5), 1307-1324, 1992. DOI: 10.1016/0017-9310(92)90186-V
- Cleary, M.J., "CMC (Conditional Moment Closure) Modelling of Enclosure Fires," Ph.D. thesis, The University of Sydney, Sydney, Australia, 2004.
- Clift, R., Grace, J.R., and Weber, M.E., *Bubbles, Drops, and Particles*, Academic Press, New York, 1978.
- Coffee, T.P. and Heimerl, J.M., "Transport Algorithms for Premixed, Laminar Steady-State Flames," *Combustion and Flame*, 43, p. 273-289, 1981. DOI: 10.1016/0010-2180(81)90027-4
- Colin, O. and Benkenida, A., "The 3-Zones Extended Coherent Flame Model (ECFM3Z) for Computing Premixed/Diffusion Combustion," *Oil & Gas Science and Technology - Revue d'IFP Energies Nouvelles*, 59(6), 593-609, 2004. DOI: 10.2516/ogst:2004043
- Colin, O., Benkenida, A., and Angelberger, C., "3D Modeling of Mixing, Ignition and Combustion Phenomena in Highly Stratified Gasoline Engines," *Oil & Gas Science and Technology - Revue d'IFP Energies Nouvelles*, 58(1), 47-62, 2003. DOI: 10.2516/ogst:2003004

Colin, O., Chevillard, S., Bohbot, J., Senecal, P.K., Pomraning, E., and Wang, M., "Development of a Species-Based Extended Coherent Flamelet Model (SB-ECFM) for Gasoline Direct Injection Engine (GDI) Simulations," *ASME 2018 Internal Combustion Engine Division Fall Technical Conference*, ICEF2018-9684, San Diego, CA, United States, Nov 4-7, 2018. DOI: 10.1115/ICEF2018-9684

Colin, O., da Cruz, A.P., and Jay, S., "Detailed chemistry-based auto-ignition model including low temperature phenomena applied to 3-D engine calculations," *Proceedings of the Combustion Institute*, 30, 2649-2656, 2005. DOI: 10.1016/j.proci.2004.08.058

Colin, O., and Ducros, F., "A thickened flame model for large eddy simulations of turbulent premixed combustion," *Physics of Fluids* 12(7), 1843, 2000. DOI: 10.1063/1.870436

Colin, O. and Truffin, K. "A Spark Ignition Model for Large Eddy Simulation Based on an FSD Transport Equation (ISSIM-LES)," *Proceedings of the Combustion Institute*, 33(2), 3097-3104, 2011. DOI: 10.1016/j.proci.2010.07.023

Craft, T.J., Gerasimov, A.V., Iacovides, H., and Launder, B.E., "Progress in the generalization of wall-function treatments," *International Journal of Heat and Fluid Flow*, 23(2), 148-160, 2002.

Craft, T.J. and Launder, B.E., "New Wall-Reflection Model Applied to the Turbulent Impinging Jet," *AIAA Journal* 30(12), 2970-2972, 1992. DOI: 10.2514/3.48980

Croaker, P., Skvortsov, A., and Kessissoglou, N., "A Simple Approach to Estimate Flow-Induced Noise from Steady State CFD Data," *Australian Acoustical Society Conference 2011*, 54, Gold Coast, Australia, November 2-4, 2011.

Curle, N., "The influence of solid boundaries upon aerodynamic sound," *Proceedings of the Royal Society of London*, 231, 505-514, 1955.

Dahms, R., Fansler, T.D., Drake, M.C., Kuo, T.-W., Lippert, A.M., and Peters, N., "Modeling Ignition Phenomena in a Spray-Guided Spark-Ignited Engine," *Proceedings of the Combustion Institute*, 32(2), 2743-2750, 2009. DOI: 10.1016/j.proci.2008.05.052

Dassault Systemes, "Abaqus Unified FEA - SIMULIA", <https://www.3ds.com/products-services/simulia/products/abaqus/>, 2019, accessed on Apr 2, 2019.

Davidson, L., and Billson, M., "Hybrid LES-RANS using synthesized turbulent fluctuations for forcing in the interface region," *International Journal of Heat and Fluid Flow*, 27, 1028-1042, 2006. DOI: 10.1016/j.ijheatfluidflow.2006.02.025

De Soete, G.G., "Overall Reaction Rates of NO and N₂ Formation from Fuel Nitrogen," *Symposium (International) on Combustion*, 15(1), 1093-1102, 1975. DOI: 10.1016/S0082-0784(75)80374-2

- Deardorff, J.W., "A Numerical Study of Three-Dimensional Turbulent Flow Channel Flow at Large Reynolds Numbers," *Journal of Fluid Mechanics*, 41(2), 453-480, 1970. DOI: 10.1017/S0022112070000691.
- Deutschmann, O., Tischer, S., Correa, C., Chatterjee, D., Kleditzsch, S., Janardhanan, V., Mladenov, N., Minh, H.D., Karadeniz, H., Hettel, M., "DETCHEM User Manual (Version 2.5)," <http://www.detchem.com>, 2014.
- Dixon, A. G. and Cresswell, D. L., "Theoretical prediction of effective heat transfer parameters in packed beds," *AIChE Journal*, 25, 663-676, 1979.
- Dombrowski, N. and Hooper, P.C., "The Effect of Ambient Density on Drop Formation in Sprays," *Chemical Engineering Science*, 17(4), 291-305, 1962. DOI: 10.1016/0009-2509(62)85008-8
- Dombrowski, N. and Johns, W.R., "The Aerodynamic Instability and Disintegration of Viscous Liquid Sheets," *Chemical Engineering Science*, 18(3), 203-214, 1963. DOI: 10.1016/0009-2509(63)85005-8
- Duclos, J.-M. and Colin, O., "Arc and Kernel Tracking Ignition Model for 3D Spark-Ignition Engine Calculations," *Fifth Internal Symposium on Diagnostics and Modeling of Combustion in Internal Combustion Engines*, Nagoya, Japan, 2001.
- Dupont, V., Porkashanian, M., Williams, A., and Woolley, R., "Reduction of NOx Formation in Natural Gas Burner Flames", *Fuel*, 72(4), 497-503, 1993.
- Ebrahimian, V., Nicolle, A., and Habchi, C., "Detailed Modeling of the Evaporation and Thermal Decomposition of Urea-Water Solution in SCR Systems," *AIChE Journal*, 58(7), 1998-2009, 2012. DOI: 10.1002/aic.12736
- Edwards, D.K. and Matavosian, R., "Scaling Rules for Total Absorptivity and Emissivity of Gases," *Journal of Heat Transfer*, 106(4), 684-689, 1984. DOI: 10.1115/1.3246739
- Ewald, J. and Peters, N., "A Level Set Based Flamelet Model for the Prediction of Combustion in Spark Ignition Engines," *15th International Multidimensional Engine Modeling Users Group Meeting*, Detroit, MI, United States, 2005.
- Faeth, G.M., "Current Status of Droplet and Liquid Combustion," *Progress in Energy and Combustion Science*, 3(4), 191-224, 1977. DOI: 10.1016/0360-1285(77)90012-0
- Farassat, F., and Succi, G.P., "The prediction of helicopter discrete frequency noise," *Vertica*, 7(4), 309-320, 1983.

Fenimore, C.P., "Formation of Nitric Oxide in Premixed Hydrocarbon Flames," *Symposium (International) on Combustion Proceedings*, 13(1), 373-380, 1971. DOI: 10.1016/S0082-0784(71)80040-1

Ferguson, C.R., and Kirkpatrick, A.T., "Internal Combustion Engines: Applied Thermosciences," John Wiley & Sons, May 11, 2015.

Ffowcs Williams, J.E., and Hawkings, D.L., "Sound generated by turbulence and surfaces in arbitrary motion," *Philosophical Transactions of the Royal Society A*, 264, 321-342, 1969. DOI: 10.1098/rsta.1969.0031

Finnie, I., "Erosion of Surfaces by Solid Particles," *Wear*, 3, 87-103, 1960.

Fletcher, R., "Conjugate gradient methods for indefinite systems," *Numerical Analysis, Proceedings of the Dundee Conference on Numerical Analysis*, 506, 73-89, 1976. DOI: 10.1007/BFb0080109

Frenklach, M. and Wang, H., "Detailed Modeling of Soot Particle Nucleation and Growth," *Proceedings of the Combustion Institute*, 23(1), 1559-1566, 1991, DOI: 10.1016/S0082-0784(06)80426-1

Gaskell, P.H. and Lau, A.K.C., "Curvature-Compensated Convective Transport: SMART, a New Boundedness-Preserving Transport Algorithm," *International Journal for Numerical Methods in Fluids*, 8(6), 617-641, 1988. DOI: 10.1002/fld.1650080602

Germano, M., Piomelli, U., Moin, P., and Cabot, W.H., "A Dynamic Subgrid-Scale Eddy Viscosity Model," *Physics of Fluids A*, 3(7), 1760-1765, 1991. DOI: 10.1063/1.857955

Gersum, S.V. and Roth, P., "Soot Oxidation in High Temperature N₂O/Ar and NO/Ar," *Symposium (International) on Combustion*, 24(1), 999-1006, 1992. DOI: 10.1016/S0082-0784(06)80118-9

Ghosh, P., Hickey, K.J., and Jaffe, S.B., "Development of a Detailed Gasoline Composition-Based Octane Model," *Industrial & Engineering Chemistry Research*, 45(1), 337-345, 2005. DOI: 10.1021/ie050811h

Gibson, M.M. and Launder, B.E., "Ground effects on pressure fluctuations in the atmospheric boundary layer," *Journal of Fluid Mechanics*, 86, 491-511, 1975.

Gilbert, R.G., Luther, K., and Troe, J., "Theory of Thermal Unimolecular Reactions in the Fall-Off Range. II. Weak Collision Rate Constants," *Berichte der Bunsengesellschaft fur Physikalische Chemie*, 87(2), 169-177, 1983. DOI: 10.1002/bbpc.19830870218

Goldsmith, C.F., Harding, L.B., Georgievskii, Y., Miller, J.A., and Klippenstein, S.J., "Temperature and Pressure-Dependent Rate Coefficients for the Reaction of Vinyl Radical

with Molecular Oxygen," *J. Phys. Chem. A*, 119(28): 7766-7779, 2015. DOI: 10.1021/acs.jpca.5b01088

Golovitchev, V.I., Montorsi, L., Denbratt, I., Corcione, F.E., and Coppola, S., "Numerical Evaluation of Direct Injection of Urea as NOx Reduction Method for Heavy Duty Diesel Engines," SAE Paper 2007-01-0909, 2007. DOI: 10.4271/2007-01-0909

Gonzalez D., M.A., Borman, G., and Reitz, R., "A Study of Diesel Cold Starting using both Cycle Analysis and Multidimensional Calculations," SAE Paper 910180, 1991. DOI: 10.4271/910180

Gordon, S., and McBride, B.J., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications: I. Analysis", *NASA Reference Publication 1311*, October 1994.

Goodwin, D.G., Moffat, H.K., and Speth, R.L., "Cantera: An Object-Oriented Software Toolkit for Chemical Kinetics, Thermodynamics and Transport Processes, v2.0.1.", <http://cantera.github.com/docs/sphinx/html/cti/reactions.html>, 2012.

Gritskevich, M.S., Garbaruk, A.V., Schütze, J., and Menter, F.R., "Developement of DDES and IDDES Formulations for the k- ω Shear Stress Transport Model," *Flow, Turbulence and Combustion*, 88(3): 431-449, 2012. doi:10.1007/s10494-011-9378-4

Gueyffier, D., Li, J., Nadim, A., Scardovelli, R., and Zaleski, S., "Volume-of-Fluid Interface Tracking with Smoothed Surface Stress Methods for Three-Dimensional Flows," *Journal of Computational Physics*, 152(2), 423-456, 1999. DOI: 10.1006/jcph.1998.6168

Gulder, O.L., "Correlations of Laminar Combustion Data for Alternative S.I. Engine Fuels," SAE Paper 841000, 1984. DOI: 10.4271/841000

Habchi, C., Nicolle, A., and Gillet, N., "Numerical Study of Urea-Water Solution Injection and Deposits Formation in an SCR System," *ICLASS 2015, 13th Triennial International Conference on Liquid Atomization and Spray Systems*, Tainian, Taiwan, August 23-27, 2015.

Haider, A. and Levenspiel, O., "Drag Coefficient and Terminal Velocity of Spherical and Non-Spherical Particles," *Powder Technology* 58(1), 63-70, 1989.

Halstead, M.P., Kirsh, L.J., and Quinn, C.P., "The Autoignition of Hydrocarbon Fuels at High Temperatures and Pressures – Fitting of a Mathematical Model," *Combustion and Flame*, 30, 45-60, 1977. DOI: 10.1016/0010-2180(77)90050-5

Han, Z. and Reitz, R.D., "Turbulence Modeling of Internal Combustion Engines Using RNG k- ϵ Models," *Combustion Science and Technology*, 106(4-6), 267-295, 1995. DOI: 10.1080/00102209508907782

Han, Z. and Reitz, R.D., "A Temperature Wall Function Formulation for Variable Density Turbulence Flow with Application to Engine Convective Heat Transfer Modeling," *International Journal of Heat and Mass Transfer*, 40(3), 613-625, 1997. DOI: 10.1016/0017-9310(96)00117-2

Hanjalic, K. and Launder, B., *Modelling Turbulence in Engineering and the Environment*, Cambridge University Press, 2011.

Hanjalic, K., Popovac, M., and Hadziabdic, M., "A robust near-wall elliptic-relaxation eddy-viscosity model for CFD," *International Journal of Heat and Fluid Flow*, 25, 1047-1051, 2004. DOI: 10.1016/j.ijheatfluidflow.2004.07.005

Hanson, R. K. and Salimian, S., "Survey of rate constants in the N/H/O system," *Combustion Chemistry*, 361-421, 1984.

Hatchard, T.D., MacNeil, D.D., Basu, A., Dahn, J.R., "Thermal Model of Cylindrical and Prismatic Lithium-Ion Cells," *Journal of The Electrochemical Society*, 148(7), A755-A761, 2001. DOI: 10.1149/1.1377592

Hestenes, M.R., and Stiefel, E., "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, 49(6), 409-436, 1952. DOI: 10.6028/jres.049.044

Heywood, J.B., *Internal Combustion Engine Fundamentals*, McGraw Hill, Inc., 1988.

Hilber, H.M., Hughes, T.J., and Taylor, R.L., "Improved numerical dissipation for time integration algorithms in structural dynamics," *Earthquake Engineering & Structural Dynamics*, 5(3), 283-292, 1977.

Hiroyasu, H. and Kadota, T., "Models for Combustion and Formation of Nitric Oxide and Soot in DI Diesel Engines," SAE Paper 760129, 1976. DOI: 10.4271/760129

Hirschfelder, J.O., Curtiss, C.F., and Bird, R.B., *Molecular Theory of Gases and Liquids*, John Wiley & Sons, Inc., 1954.

Hohenberg, G.F., "Advanced Approaches for Heat Transfer Calculations," SAE Paper 790825, 1979.

Hou, S., "Investigation of the Interaction Mechanisms Between Closely Spaced Sprays From Micro-Hole Nozzles," Ph.D. Thesis, Dept. of Mechanical and Industrial Engineering, University of Massachusetts Amherst, Amherst, MA, United States, 2005.

Hwang, G. J., Wu, C. C., and Chao, C. H., "Investigation of non-Darcian forced convection in an asymmetrically heated sintered porous channel," *Journal of Heat Transfer*, 111, 725-32, 1995.

Huh, K.Y. and Gosman, A.D., "A Phenomenological Model of Diesel Spray Atomization," *Proceedings of the International Conference of Multiphase Flows*, Sep. 24-27, Tsukuba, Japan, 1991.

Incopera, F.P. and De Witt, D.P., *Introduction to Heat Transfer, Second Edition*, John Wiley & Sons, Inc., 1990.

Issa, R.I., "Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting," *Journal of Computational Physics*, 62(1), 40-65, 1986. DOI: 10.1016/0021-9991(86)90099-9.

Jacobsen, N., Fuhrmann, D.R., and Fredsøe, J., "A wave generation toolbox for the open-source CFD library: OpenFOAM®," *International Journal for Numerical Methods in Fluids*, 70(9), 1073-1088, 2012. DOI: 10.1002/fld.2726

Jameel, A.G.A., Naser, N., Emwas, A.H., Dooley, S., and Sarathy, S.M., "Predicting Fuel Ignition Quality using H NMR Spectroscopy and Multiple Linear Regression Energy Fuels," *Energy Fuels*, 30(11), 9819-9835, 2016. DOI: 10.1021/acs.energyfuels.6b01690

Jameel, A.G.A., Oudenhoven, V.V., Emwas, A.H., and Sarathy, S.M., "Predicting Octane Number Using Nuclear Magnetic Resonance Spectroscopy and Artificial Neural Networks," *Energy Fuels*, 32(5), 6309-6329, 2018. DOI: 10.1021/acs.energyfuels.8b00556

Jaravel, T., "Prediction of Pollutants in Gas Turbines Using Large Eddy Simulation," Ph.D. Thesis, Université de Toulouse, Toulouse, France, 2016.

Jayatilleke, C.L., "The Influence of Prandtl Number and Surface Roughness on the Resistance of the Laminar Sub-layer to Momentum and Heat Transfer," *Progress in Heat and Mass Transfer*, 193-330, 1969.

Jia, M., Peng, Z., and Xie, M., "Numerical Investigation of Soot Reduction Potentials with Diesel Homogeneous Charge Compression Ignition Combustion by an Improved Phenomenological Soot Model," *Journal of Automobile Engineering*, 223(3), 395-412, 2009. DOI: 10.1243/09544070JAUTO993

Johnson, S.G., "The NLopt nonlinear-optimization package", <http://github.com/stevengj/nlopt> and <https://nlopt.readthedocs.io/en/latest/>, 2020, accessed on Jan 31, 2020.

Joseph, D.D., Belanger J., and Beavers G.S., "Breakup of a Liquid Drop Suddenly Exposed to a High-Speed Airstream," *International Journal of Multiphase Flow*, 25(6-7), 1263-1303, 1999. DOI: 10.1016/S0301-9322(99)00043-9

Kader, B.A., "Temperature and Concentration Profiles in Fully Turbulent Boundary Layers," *International Journal of Heat and Mass Transfer*, 24(9), 1541-1544, 1981. DOI: 10.1016/0017-9310(81)90220-9

Kalitzin, G., Medic, G., Iaccarino, G., and Durbin, P., "Near-wall behavior of RANS turbulence models and implications for wall functions," *Journal of Computational Physics* 204(1), 265-291, 2005.

Kaminaga, T., Kusaka, and J., Ishii, Y., "A Three-dimensional Numerical Study on Exhaust Gas Emissions from a Medium-duty Diesel Engine using a Phenomenological Soot Particle Formation Model Combined with Detailed Chemistry," *International Journal of Engine Research*, 9(4), 283-295, 2008. DOI: 10.1243/14680874JER00908

Karimi, S., Zhang, J., Shirazi, S.A., McLaury, B.S., "Evaluation of the Effect of Particle Size on Erosion Calculations Utilizing CFD and Comparison with Submerged Slurry Jet Experiments," *Proceedings of the ASME-JSME-KSME 8th Joint Fluids Engineering Conference*, Jul 28-Aug 1, San Francisco, CA, United States, 2019.

Karypis, G., "METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, Version 5.0", <http://glaros.dtc.umn.edu/gkhome/fetch/sw/metis/manual.pdf>, 2011.

Kazakov, A. and Foster, D., "Modeling of Soot Formation during DI Diesel Combustion using a Multi-Step Phenomenological Model," SAE Paper 982463, 1998. DOI: 10.4271/982463.

Kazakov, A. and Frenklach, M., "Dynamic Modeling of Soot Particle Coagulation and Aggregation: Implementation With the Method of Moments and Application to High-Pressure Laminar Premixed Flames," *Combustion and Flame*, 114(3-4), 484-501, 1998. DOI: 10.1016/S0010-2180(97)00322-2

Kazakov, A., Wang, H., and Frenklach, M., "Detailed Modeling of Soot Formation in Laminar Premixed Ethylene Flames at a Pressure of 10 Bar," *Combustion and Flame*, 100(1-2), 111-120, 1995. DOI: 10.1016/0010-2180(94)00086-8

Kee, R.J., Miller, J.A., and Evans, G.H., Dixon-Lewis, G., "A Computational Model of the Structure and Extinction of Strained, Opposed Flow, Premixed Methane-Air Flames," *Twenty-Second Symposium (International) on Combustion*, 22(1), 1479-1494, 1989. DOI: 10.1016/S0082-0784(89)80158-4

Kee, R.J., Rupley, F.M., and Miller, J.A., "Chemkin-II: A Fortran Chemical Kinetics Package for the Analysis of Gas-Phase Chemical Kinetics," Sandia National Laboratories Technical Report SAND89-8009, 1989.

- Kee, R.J., Coltrin, M.E., Glarborg, P., and Zhu, H., "Chemically Reacting Flow: Theory and Practice, Second Edition," John Wiley and Sons, 2017.
- Kermani, M.J., Gerber, A.G., and Stockie, J.M., "Thermodynamically Based Moisture Prediction Using Roe's Scheme," *The 4th Conference of Iranian AeroSpace Society*, Tehran, Iran, 2003.
- Kim, G.H., Pesaran, A., Spotnitz, R., "A Three-Dimensional Thermal Abuse Model for Lithium-Ion Cells," *Journal of Power Sources*, 170, 476-489, 2007. DOI: 10.1016/j.jpowsour.2007.04.018
- Kim, J. and Anderson, R.W., "Spark Anemometry of Bulk Gas Velocity at the Plug Gap of Firing Engine," SAE Paper 952459, 1995. DOI: 10.4271/952459
- Kim, S.-E., and Choudhury, D., "A Near-Wall Treatment Using Wall Functions Sensitized to Pressure Gradient," *Separated and Complex Flows*, 217, 273-280, 1995.
- Klein, M., Sadiki, A., and Janicka, J., "A Digital Filter Based Generation of Inflow Data for Spatially Developing Direct Numerical or Large Eddy Simulations," *Journal of Computational Physics*, 186(2), 652-665, 2003. DOI: 10.1016/S0021-9991(03)00090-1
- Kong, S.-C. and Reitz, R.D., "Multidimensional Modeling of Diesel Ignition and Combustion using a Multistep Kinetics Model," *Journal of Engineering for Gas Turbines and Power*, 115(4), 781-789, 1993. DOI: 10.1115/1.2906775
- Kong, S.-C., Han, Z., and Reitz, R.D., "The Development and Application of a Diesel Ignition and Combustion Model for Multidimensional Engine Simulation," SAE Paper 950278, 1995. DOI: 10.4271/950278
- Koohandaz, A., Khavasi, E., Eyvazian, A., and Yousefi, H., "Prediction of particles deposition in a dilute quasi-steady gravity current by Lagrangian markers: Effect of shear-induced lift force," *Scientific Reports*, 10, 16673, 2020.
- Koren, B., "A Robust Upwind Discretisation Method for Advection, Diffusion and Source Terms," *Numerical Methods for Advection - Diffusion Problems*, eds. Vreugdenhil, C.B., and Koren, B., Friedrich Vieweg & Sohn, 1993.
- Kuhnke, D., "Spray/Wall-interaction Modelling by Dimensionless Data Analysis," Ph.D. Thesis, Shaker Verlag, 2004, ISBN 3-8322-3539.
- Kumar, S. and Ramkrishna, D., "On the Solution of Population Balance Equations by Discretization—II. A Moving Pivot Technique," *Chemical Engineering Science*, 51(8), 1333-1342, 1996. DOI: 10.1016/0009-2509(95)00355-X

Kuwahara, F., Shirota, M., and Nakayama, A., "A numerical study of interfacial convective heat transfer coefficient in two-energy model for convection in porous media," *International Journal of Heat and Mass Transfer*, 44, 1153-1159, 2001.

Lamb, H., *Hydrodynamics, Sixth Edition*, Dover Publications, New York, 1945.

Launder, B. and Spalding, D., "The Numerical Computation of Turbulent Flows," *Computer Methods in Applied Mechanics and Engineering*, 3(2), 269-289, 1974. DOI: 10.1016/0045-7825(74)90029-2

Lawrence Livermore National Laboratory, HYPRE (2.9.0b Reference Manual), https://computation-rnd.llnl.gov/linear_solvers/download/hypre-2.9.0b_ref_manual.pdf, 2012.

Lawrence Livermore National Laboratory, SUNDIALS, <https://computation.llnl.gov/casc/sundials/main.html>, accessed July 2015.

Lawrence Livermore National Laboratory, SuperLU Users' Guide, <http://crd.lbl.gov/~xiaoye/SuperLU/>, accessed April 2017.

Legier, J.P., Poinsot, T., and Veynante, D., "Dynamically thickened flame LES model for premixed and non-premixed turbulent combustion," *Proceedings of the Summer Program*, Center for Turbulence Research, 2000.

Liakou, A., Denoël, V., and Detournay, E., "Fast In-Plane Dynamics of a Beam with Unilateral Constraints," *Journal of Engineering Mechanics*, 143(2), 2017. DOI: 10.1061/(ASCE)EM.1943-7889.0001175

Liang, L., Stevens, J.G., and Farrell, J.T., "A Dynamic Multi-Zone Partitioning Scheme for Solving Detailed Chemical Kinetics in Reactive Flow Computations," *Combustion Science and Technology*, 181(11), 1345-1371, 2009. DOI: 10.1080/00102200903190836

Lien, F.S. and Kalitzin, G., "Computations of transonic flow with the v²-f turbulence model," *International Journal of Heat and Fluid Flow*, 22, 53-61, 2002. DOI: 10.1016/S0142-727X(00)00073-4

Lien, F.S. and Leschziner, M.A., "Upstream Monotonic Interpolation for Scalar Transport with Application to Complex Turbulent Flows," *International Journal for Numerical Methods in Fluids*, 19(6), 527-548, 1994. DOI: 10.1002/fld.1650190606

Lien, F.S. and Leschziner, M.A., "Low-Reynolds-Number Eddy-Viscosity Modelling Based on Non-Linear Stress-Strain/Vorticity Relations," *Proceedings of the 3rd Symposium on Engineering Turbulence Modelling and Measurements*, Crete, Greece, 1996.

Chapter 27: References

- Lienhard, J.H. IV and Lienhard, J.H. V, *A Heat Transfer Textbook, Fourth Edition*, Phlogiston Press, 2017.
- Lighthill, M.J., "On sound generated aerodynamically I. General theory," *Proceedings of the Royal Society London*, 211:564-587. DOI: 10.1098/rspa.1952.0060
- Lighthill, M.J., "On sound generated aerodynamically I. Turbulence as a source of sound," *Proceedings of the Royal Society London*, 222:1-32. DOI: 10.1098/rspa.1954.0049
- Lilly, D.K., "The Representation of Small-Scale Turbulence in Numerical Simulation Experiments," *Proceedings of the IBM Scientific Computing Symposium on Environmental Sciences*, IBM, 1967.
- Lilly, D.K., "A Proposed Modification of the Germano Subgrid-Scale Closure Method," *Physics of Fluids A*, 4(3), 633-635, 1992. DOI: 10.1063/1.858280
- Lin, X., Perez, H.E., Mohan, S., Siegel, J., Stefanopoulou, A.G., Ding, Y., Castanier, M., "A lumped-parameter electro-thermal model for cylindrical batteries," *Journal of Power Sources*, 257, 1-11, 2014. DOI: 10.1016/j.jpowsour.2014.01.097
- Lindemann, F.A., Arrhenius, S., Langmuir, I., Dhar, N.R., Perrin, J., and Lewis, W.C.McC., "Discussion on 'The Radiation Theory of Chemical Action'," *Transactions of the Faraday Society*, 17, 598-606, 1922. DOI: 10.1039/TF9221700598
- Liu, A.B., Mather, D.K., and Reitz, R.D., "Modeling the Effects of Drop Drag and Breakup on Fuel Sprays," SAE Paper 930072, 1993. DOI: 10.4271/930072.
- Lu, H., Rutland C.J., and Smith L.M., "A Priori Tests of One-Equation LES Modeling of Rotating Turbulence," *Journal of Turbulence*, 8, N37, 2007. DOI: 10.1080/14685240701493947
- Lu, T.F. and Law, C.K., "A Directed Relation Graph Method for Mechanism Reduction," *Proceedings of the Combustion Institute*, 30(1), 1333-1341, 2005. DOI: 10.1016/j.proci.2004.08.145
- Lu, T.F. and Law, C.K., "Strategies for Mechanism Reduction for Large Hydrocarbons: n-Heptane," *Combustion and Flame*, 154(1-2), 153-163, 2008. DOI: 10.1016/j.combustflame.2007.11.013
- Lutz, A.E., Kee, R.J., and Miller, J.A., "SENKIN: A Fortran Program for Predicting Homogeneous Gas Phase Chemical Kinetics with Sensitivity Analysis," Sandia National Laboratories Technical Report SAND87-8248, 1995.
- MacCormack, R.W., *Numerical Computation of Compressible and Viscous Flow*, American Institute of Aeronautics and Astronautics, 2014.

Maciejewski, D., Sukheswalla, P., Wang, C., Drennan, S.A., and Chai, X., "Accelerating Accurate Urea/SCR Film Temperature Simulations to Time-Scales Needed for Urea Deposit Predictions," SAE Paper 2019-01-0982, 2019. DOI: 10.4271/2019-01-0982

Magnussen, B.F., and Hjertager, B.H., "On Mathematical Modeling of Turbulent Combustion with Special Emphasis on Soot Formation and Combustion," *Symposium (International) on Combustion*, 16(1), 719-729. 1977. DOI: 10.1016/S0082-0784(77)80366-4

Manninen, Mikko, Taivassalo, Veikko, and Kallio, Sirpa, "On the Mixture Model for Multiphase Flow," VTT Technical Research Centre of Finland, VTT Publications No. 288, 1996.

Mansouri, A., "A Combined CFD-Experimental Method for Developing an Erosion Equation for Both Gas-Sand and Liquid-Sand Flows," Ph.D. thesis, University of Tulsa, Tulsa, OK, United States, 2016.

Marble, F.E., and Broadwell, J.E., "The Coherent Flame Model for Turbulent Chemical Reactions," *Project SQUID Technical Report TRW-9-PU*, , 1977.

Marchal, C., "Modelisation de la Formation et de l'Oxydation des Suies dans un Moteur Automobile," Ph.D. thesis, Universite d'Orleans, Orleans, France 2008.

Márquez Damián, Santiago, "An Extended Mixture Model for the Simultaneous Treatment of Short and Long Scale Interfaces," Ph.D. thesis, Universidad Nacional del Litoral, Santa Fe, Argentina, 2013.

Mauss, F., "Entwicklung eines Kinetischen Modells der Russbildung mit Schneller Polymerisation," Ph.D. thesis, RWTH Aachen, Aachen, Germany, 1998.

Mauss, F., Trilken, B., Breitbach, H., Peters, N., "Soot Formation in Partially Premixed Diffusion Flames at Atmospheric Pressure," *Soot Formation in Combustion*, Bockhorn, H., Springer, Berlin, Heidelberg, 1994.

Mazumder, S. and Sengupta, D. "Sub-Grid Scale Modeling of Heterogeneous Chemical Reactions and Transport in Full-Scale Catalytic Converters," *Combustion and Flame*, 131(1-2), 85-97, 2002. DOI: 10.1016/S0010-2180(02)00392-9

McBride, B.J., and Gordon, S., "Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications: II. Users Manual and Program Description," NASA Reference Publication 1311, 1996.

McLaury, B.S., Wang, J., Shirazi, S.A., Shadley, J.R., and Rybicki, E.F., "Solid Particle Erosion in Long Radius Elbows and Straight Pipes," *Society of Petroleum Engineers Annual Technical Conference and Exhibition*, Oct 5-8, San Antonio, TX, United States, 1997.

- Mehl, M., Pitz, W.J., Westbrook, C.K., and Curran, H.J., "Kinetic Modeling of Gasoline Surrogate Components and Mixtures under Engine Conditions," *Proceedings of the Combustion Institute*, 33, 193-200, 2011.
- Meneveau, C., "Statistics of Turbulence Subgrid-Scale Stresses: Necessary Conditions and Experimental Tests," *Physics of Fluids*, 6(2), 815-833, 1994. DOI: 10.1063/1.868320
- Meneveau, C. and Poinsot, T. "Stretching and Quenching of Flamelets in Premixed Turbulent Combustion," *Combustion and Flame*, 86(4), 311-332, 1991. DOI: 10.1016/0010-2180(91)90126-V
- Menon, S., Yeung, P.K., and Kim, W.V., "Effect of Subgrid Models on the Computed Interscale Energy Transfer in Isotropic Turbulence," *Computer and Fluids*, 25(2), 165-180, 1996. DOI: 10.1016/0045-7930(95)00036-4
- Menter, F.R., Kuntz, M., and Langtry, R., "Ten Years of Industrial Experience with the SST Turbulence Model," *Turbulence, Heat and Mass Transfer 4*, eds: Hanjalic, K., Nagano, Y., and Tummers, M., Begell House, Inc., 2003.
- Metghalchi, M. and Keck, J.C., "Burning Velocities of Mixtures of Air and Methanol, Isooctane and Indolene at High Pressures and Temperatures," *Combustion and Flame*, 48, 191-210, 1982. DOI: 10.1016/0010-2180(82)90127-4
- Modest, M.F., *Radiative Heat Transfer*, Academic Press, 2003.
- Motz, H. and Wise, H., "Diffusion and Heterogeneous Reaction. III. Atom Recombination at Catalytic Boundary," *Journal of Chemical Physics*, 32(6), 1893-1894, 1960. DOI: 10.1063/1.1731060
- Mundo, Chr., Sommerfeld, M., and Tropea, C., "Droplet-Wall Collisions: Experimental Studies of the Deformation and Breakup Process," *International Journal of Multiphase Flow*, 21(2), 151-173, 1995. DOI: 10.1016/0301-9322(94)00069-V
- Naber, J. and Reitz, R.D., "Modeling Engine Spray/Wall Impingement," SAE Paper 880107, 1988. DOI: 10.4271/880107
- Nagle, J. and Strickland-Constable, R.F., "Oxidation of Carbon Between 1000-2000 °C," *Proceedings of the Fifth Carbon Conference, Volume 1*, Pergamon Press, 1962.
- Namazian, M. and Heywood, J.B., "Flow in the Piston-Cylinder-Ring Crevices of a Spark-Ignition Engine: Effect on Hydrocarbon Emissions, Efficiency and Power," SAE Paper 820088, 1982. DOI: 10.4271/820088

- Neoh, K.G., Howard, J.B., and Sarofim, A.F., "Effect of Oxidation on the Physical Structure of Soot," *Symposium (International) on Combustion*, 20(1), 951-957, 1985. DOI: 10.1016/S0082-0784(85)80584-1
- Netzell, K., "Development and Application of Detailed Kinetic Models for the Soot Particle Size Distribution Function," Ph.D. thesis, Lund University, Lund, Sweden, 2007.
- Nicoud, F., Toda, H.B., Cabrit, O., Bose, S., and Lee, J., "Using singular values to build a subgrid-scale model for large eddy simulations," *Physics of Fluids*, 23(8), 085106, 2011. DOI: 10.1063/1.3623274
- Oka, Y.I., Okamura, K., and Yoshida, T., "Practical estimation of erosion damage caused by solid particle impact, Part 1: Effects of impact parameters on a predictive equation," *Wear*, 259, 95-101, 2005. DOI: 10.1016/j.wear.2005.01.039
- Okong'o, N., and Bellan, J., "Consistent Boundary Conditions for Multicomponent Real Gas Mixtures Based on Characteristic Waves," *Journal of Computational Physics*, 176, 330-344, 2002. DOI: 10.1006/jcph.2002.6990
- O'Rourke, P.J., "Collective Drop Effects on Vaporizing Liquid Sprays," Ph.D. Thesis, Princeton University, Princeton, NJ, United States, 1981.
- O'Rourke, P.J. and Amsden, A.A., "The TAB Method for Numerical Calculation of Spray Droplet Breakup," SAE Paper 872089, 1987. DOI: 10.4271/872089.
- O'Rourke, P.J. and Amsden, A.A., "A Spray/Wall Interaction Submodel for the KIVA-3 Wall Film Model," SAE Paper 2000-01-0271, 2000. DOI: 10.4271/2000-01-0271
- Pandal Blanco, A., "Implementation and Development of an Eulerian Spray Model for CFD Simulations of Diesel Sprays," Ph.D. Thesis, Universitat Politècnica de València, València, Spain, 2016.
- Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*, CRC Press, 1980.
- Patterson, M.A., "Modeling the Effects of Fuel Injection Characteristics on Diesel Combustion and Emissions," Ph.D. Thesis, University of Wisconsin-Madison, Madison, WI, 1997.
- Peters, N., "Laminar Diffusion Flamelet Models in Non-Premixed Turbulent Combustion," *Progress in Energy and Combustion Science*, 10(3), 319-339 1984. DOI: 10.1016/0360-1285(84)90114-X
- Peters, N., *Turbulent Combustion*, Cambridge University Press, 2000.

Piscagila, F., Montorfano, A., and Onorati, A., "Towards the LES Simulation of IC Engines with Parallel Topologically Changing Meshes," *SAE Int. J. Engines*, 6(2), 926-940, 2013. DOI: 10.4271/2013-01-1096

Pitsch, H., "A G-equation Formulation for Large-Eddy Simulation of Premixed Turbulent Combustion," *Center for Turbulence Research Annual Research Briefs*, Stanford University Center for Turbulence Research, 2002.

Pitsch, H., and Duchamp de Lageneste, L., "Large-Eddy Simulation of Premixed Turbulent Combustion Using a Level-Set Approach", *Proceedings of the Combustion Institute*, 29, 2001-2008, 2002.

Poinset, T.J., and Lele, S.K., "Boundary Conditions for Direct Simulations of Compressible Turbulent Flows," *Journal of Computational Physics*, 101(1), 103-129 (1992). DOI: 10.1016/0021-9991(92)90046-2

Pomraning, E., "Development of Large Eddy Simulation Turbulence Models," Ph.D. Thesis, University of Wisconsin-Madison, Madison, WI, 2000.

Pope, S.B., *Turbulent Flows*, Cambridge University Press, 2000.

Pope, S.B., "The Computation of Constrained and Unconstrained Equilibrium Compositions of Ideal Gas Mixtures using Gibbs Function Continuation," Cornell University Report FDA 03-02, 2003.

Pope, S.B., "CEQ: A Fortran Library to Compute Equilibrium Compositions using Gibbs Function Continuation," <http://eccentric.mae.cornell.edu/~pope/CEQ/>, 2003, accessed on 12 Oct 2016.

Popovac, M., and Hanjalic, K., "Compound Wall Treatment for RANS Computation of Complex Turbulent Flows and Heat Transfer," *Flow, Turbulence, and Combustion*, 78, 177-202, 2007. DOI: 10.1007/s10494-006-9067-x

Post, S.L. and Abraham, J., "Modeling the Outcome of Drop-Drop Collisions in Diesel Sprays," *International Journal of Multiphase Flow*, 28(6), 997-1019, 2002. DOI: 10.1016/S0301-9322(02)00007-1

Price, C., Hamzehloo, A., Aleiferis, P., and Richardson, R., "An Approach to Modeling Flash-Boiling Fuel Sprays for Direct-Injection Spark-Ignition Engines," *Atomization and Sprays*, 26(12), 1197-1239, 2016. DOI: 10.1615/AtomizSpr.2016015807

Proudman, I., "The generation of noise by isotropic turbulence," *Proceedings of the Royal Society of London*, 214, 119-132, 1952.

- Ra, Y. and Reitz, R.D, "A Vaporization Model for Discrete Multi-Component Fuel Sprays," *International Journal of Multiphase Flow*, 35(2), 101-117, 2009. DOI: 10.1016/j.ijmultiphaseflow.2008.10.006
- Rahim, F., Elia, M., Ulinski, M., and Metghalchi, M., "Burning velocity measurements of methane-oxygen-argon mixtures and an application to extend methane-air burning velocity measurements," *International Journal of Engine Research*, 3(2), 81-92, 2002. DOI: 10.1243/14680870260127873
- Raj, A., Sander, M., Janardhanan, V., and Kraft, M., "A Study on the Coagulation of Polycyclic Aromatic Hydrocarbon Clusters to Determine their Collision Efficiency," *Combustion and Flame*, 157(3), 523-534, 2010. DOI: 10.1016/j.combustflame.2009.10.003
- Raju, M., Wang, M., Dai, M., Piggot, W., Flowers, D., "Acceleration of Detailed Chemical Kinetics Using Multi-zone Modeling for CFD in Internal Combustion Engine Simulations," *SAE Paper 2012-01-0135*, 2012. DOI: 10.4271/2012-01-0135
- Raju, M., Wang, M., Senecal, P.K., Som, S., and Longman, D.E., "A Reduced Diesel Surrogate Mechanism for Compression Ignition Engine Applications," *ASME 2012 Internal Combustion Engine Division Fall Technical Conference*, Vancouver, Canada, 2012. DOI: 10.1115/ICEF2012-92045
- Reinmann, R., "Theoretical and Experimental Studies of the Formation of Ionized Gases in Spark Ignition Engines," Ph.D. Thesis, Lund Institute of Technology, Lund, Sweden, 1998.
- Reitz, R.D. and Bracco, F.V., "Mechanisms of Breakup of Round Liquid Jets," *Encyclopedia of Fluid Mechanics*, Gulf Publishing Company, 1986.
- Reitz, R.D. and Kuo, T., "Modeling of HC Emissions Due to Crevice Flows in Premixed-Charge Engines," *SAE Paper 892085*, 1989. DOI: 10.4271/892085
- Reitz, R.D., "Modeling Atomization Processes in High-Pressure Vaporizing Sprays," *Atomisation and Spray Technology*, 3(4), 309-337, 1987.
- Reitz, R.D. and Diwakar, R., "Structure of High-Pressure Fuel Sprays," *SAE Paper 870598*, 1987. DOI: 10.4271/870598
- Ren, D., Liu, X., Feng, X., Lu, L., Ouyang, M., Li, J., He, X., "Model-Based Thermal Runaway Prediction of Lithium-Ion Batteries from Kinetics Analysis of Cell Components," *Applied Energy*, 228, 633-644, 2018. DOI: 10.1016/j.apenergy.2018.06.126
- Rhie, C.M., and Chow, W.L., "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation," *AIAA Journal*, 21(11), 1525-1532, 1983. DOI: 10.2514/3.8284

Ricart, L.M., Xin, J., Bower, G.R., and Reitz, R.D., "In-Cylinder Measurement and Modeling of Liquid Fuel Spray Penetration in a Heavy-Duty Diesel Engine," SAE Paper 971591, 1997. DOI: 10.4271/971591

Ricart, L.M., "An Experimental and Computational Study of Fuel Injection, Mixing and Combustion in Diesel Engines," Ph.D. Thesis, University of Wisconsin-Madison, Madison, WI, United States, 1998.

Richard, S., Colin, O., Vermorel, O., Benkenida, A., Angelberger, C., and Veynante, D., "Towards large eddy simulation of combustion in spark ignition engines," *Proceedings of the Combustion Institute* 31, 3059-3066, 2007.

Richards, K.J., "Multidimensional Intake Flow Modeling of HSDI Diesel Engines," M.S. Thesis, University of Wisconsin-Madison, Madison, WI, 1999.

Rider, W.J. and Kothe, D.B., "Reconstructing Volume Tracking," *Journal of Computational Physics*, 141(2), 112-152, 1998. DOI: 10.1006/jcph.1998.5906

Robert, A., Richard, S., Colin, O., Martinez, L., and De Francqueville, L., "LES Prediction and Analysis of Knocking Combustion in a Spark Ignition Engine," *Proceedings of the Combustion Institute*, 35(3), 2941-2948, 2015. DOI: 10.1016/j.proci.2014.05.154

Roe, P.L., "Characteristic-Based Schemes for the Euler Equations," *Annual Review of Fluid Mechanics*, 18, 337-365, 1986. DOI: 10.1146/annurev.fl.18.010186.002005

Rohsenow, W.M., "A Method of Correlating Heat Transfer Data for Surface Boiling of Liquids," *Transactions of ASME*, 74, 969-976, 1952.

Rotta, J., "Statistische Theorie nichthomogener Turbulenz," *Zeitschrift für Physik*, 129(6), 547-572, 1951.

Saad, Y., *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2003.

Sarkar, S., and Hussaini, M.Y., "Computation of the Sound Generated by Isotropic Turbulence," NASA Contractor Report 191543, 1993.

Sazhin, S.S., "Advanced Models of Fuel Droplet Heating and Evaporation," *Progress in Energy and Combustion Science*, 32(2), 162-214, 2006. DOI: 10.1016/j.pecs.2005.11.001

Sazhin, S.S., Krutitskii, P.A., Gusev, I.G., Heikal, M.R., "Transient Heating of an Evaporating Droplet with Presumed Time Evolution of its Radius," *International Journal of Heat and Mass Transfer*, 54(5-6), 1278-1288, 2011. DOI: 10.1016/j.ijheatmasstransfer.2010.10.018

- Scardovelli, R. and Zaleski, S., "Direct Numerical Simulation of Free-Surface and Interfacial Flow," *Annual Review of Fluid Mechanics*, 31, 567-603, 1999. DOI: 10.1146/annurev.fluid.31.1.567
- Schapertons, H. and Lee, W., "Multidimensional Modeling of Knocking Combustion in SI Engines," SAE Paper 850502, 1985. DOI: 10.4271/850502
- Schmidt, D.P., Nouar, I., Senecal, P.K., Hoffman, J., Rutland, C.J., Martin, J., and Reitz, R.D., "Pressure-Swirl Atomization in the Near Field," SAE Paper 1999-01-0496, 1999. DOI: 10.4271/1999-01-0496
- Schmidt, D.P. and Rutland, C.J., "A New Droplet Collision Algorithm," *Journal of Computational Physics*, 164(1), 62-80, 2000. DOI: 10.1006/jcph.2000.6568
- Senecal, P.K., Schmidt, D.P., Nouar, I., Rutland, C.J., Reitz, R.D., and Corradini, M.L., "Modeling High-Speed Viscous Liquid Sheet Atomization," *International Journal of Multiphase Flow*, 25(6-7), 1073-1097, 1999. DOI: 10.1016/S0301-9322(99)00057-9
- Senecal, P.K., "Development of a Methodology for Internal Combustion Engine Design Using Multi-Dimensional Modeling with Validation Through Experiments," Ph.D. Thesis, University of Wisconsin-Madison, Madison, WI, 2000.
- Senecal, P.K., Pomraning, E., and Richards, K.J., "Multi-Dimensional Modeling of Direct-Injection Diesel Spray Liquid Length and Flame Lift-off Length using CFD and Parallel Detailed Chemistry," SAE Paper 2003-01-1043, 2003. DOI: 10.4271/2003-01-1043.
- Senecal, P.K., Richards, K.J., Pomraning, E., Yang, T., Dai, M.Z., McDavid, R.M., Patterson, M.A., Hou, S., and Shethaji, T., "A New Parallel Cut-Cell Cartesian CFD Code for Rapid Grid Generation Applied to In-Cylinder Diesel Engine Simulations," SAE Paper 2007-01-0159, 2007. DOI: 10.4271/2007-01-0159
- Sheng, C., and Allen, C.B., "Efficient Mesh Deformation Using Radial Basis Functions on Unstructured Meshes," *AIAA Journal*, 51(3), 2013. DOI: 10.2514/1.J052126
- Shields, B., Neroorkar, K., and Schmidt, D.P., "Cavitation as Rapid Flash Boiling," *ILASS-Americas 23rd Annual Conference on Liquid Atomization and Spray Systems*, Ventura, CA, United States, 2011.
- Shih, T.-H., Liou, W.W., Shabbir, A., Yang, Z., and Zhu, J., "A New $k-\epsilon$ Eddy Viscosity Model for High Reynolds Number Turbulent Flows—Model Development and Validation," *Computers & Fluids*, 24(3), 227-238, 1995. DOI: 10.1016/0045-7930(94)00032-T
- Siow, Y.-K., "Reynolds-stress turbulence model in the KIVA code for engine simulation," M.S. thesis, Michigan Technological University, Houghton, MI, 2003.

Shir, C.C., "A Preliminary Numerical Study of Atmospheric Turbulent Flows in the Idealized Planetary Boundary Layer," *Journal of the Atmospheric Sciences*, 30, 1327-1339, 1973.

Shur, M.L., Spalart, P.R., Strelets, M.Kh., and Travin, A.K., "A Hybrid RANS-LES Approach with Delayed-DES and Wall-Modeled LES Capabilities," *International Journal of Heat and Fluid Flow*, 26(6):1638-1649, 2008. DOI: 10.1016/j.ijheatfluidflow.2008.07.001

Smagorinsky, J., "General Circulation Experiments with the Primitive Equations," *Monthly Weather Review*, 91(3), 99-164, 1963. DOI: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2

Smith, H., Zochbauer, M., and Lauer, T. "Advanced Spray Impingement Modelling for an Improved Prediction Accuracy of the Ammonia Homogenisation in SCR Systems," SAE Paper 2015-01-1054, 2015. DOI: 10.4271/2015-01-1054

Smith, T.F., Shen, Z.F., and Friedman, J.N., "Evaluation of Coefficients for the Weighted Sum of Gray Gases Model," *Journal of Heat Transfer*, 104(4), 602-608, 1982. DOI: 10.1115/1.3245174

Smoluchowski, M.V., "Versuch einer Mathematischen Theorie der Koagulationskinetik Kolloider Losungen," *Zeitschrift fur Physikalische Chemie*, 92, p. 129-168, 1917.

Som, S. and Aggarwal, S.K., "Effects of Primary Breakup Modeling on Spray and Combustion Characteristics of Compression Ignition Engines," *Combustion and Flame*, 157(6), 1179-1193, 2010. DOI: 10.1016/j.combustflame.2010.02.018

Som, S., Longman, D.E., Aithal, S.M., Bair, R., Garcia, M., Quan, S., Richards, K.J., Senecal, P.K., Shethaji, T., and Weber, M., "A Numerical Investigation on Scalability and Grid Convergence of Internal Combustion Engine Simulations," SAE Paper 2013-01-1095, 2013. DOI: 10.4271/2013-01-1095

Spalart, P.R., and Allmaras, S.R., "A One-Equation Turbulence Model For Aerodynamic Flows," AIAA Paper 1992-0439, 1992. DOI: 10.2514/6.1992-439

Spalart, P.R., and Allmaras, S.R., "A One-Equation Turbulence Model For Aerodynamic Flows," *La Recherche Aérospatiale*, 1, p. 5-21, 1994.

Spalart, P.R., Jou, W-H., Strelets, M., and Allmaras, S.R., "Comments on the Feasibility of LES for Wings and on a Hybrid RANS/LES Approach," Advances in DNS/LES, 1st AFOSR International Conference on DNS/LES, Ruston, LA, USA, Aug 04-08, 1997.

Spalart, P.R., Deck, S., Shur, M.L., Squires, K.D., Strelets, M.Kh., and Travin, A.K., "A New Version of Detached-Eddy Simulation, Resistant to Ambiguous Grid Densities," *Theoretical and Computational Fluid Dynamics*, 20:181-195, 2006. DOI: 10.1007/s00162-006-0015-0

- Spalding, D.B., "A note on mean residence-times in steady flows of arbitrary complexity," *Chemical Engineering Science*, 9, 74-77, 1958.
- Speziale, C.G., "Turbulence Modeling for Time-Dependent RANS and VLES: A Review," *AIAA Journal*, 36(2), 173-184, 1998. DOI: 10.2514/2.7499
- Speziale, C.G., Sarkar, S., and Gatski, T.B., "Modelling the pressure-strain correlation of turbulence: an invariant dynamical systems approach," *Journal of Fluid Mechanics*, 227, 245-272, 1991. DOI: 10.1017/S0022112091000101
- Strelets, M., "Detached Eddy Simulation of Massively Separated Flows," AIAA Paper 2001-0879, 2001. DOI: 10.2514/6.2001-879
- Suga, K., Craft, T.J., and Iacovides, H., "An analytical wall-function for turbulent flows and heat transfer over rough walls," *International Journal of Heat and Fluid Flow*, 27(5), 852-866, 2006.
- Tan, Z. and Reitz, R., "Modeling Ignition and Combustion in Spark Ignition Engines Using a Level Set Method," SAE Paper 2003-01-0722, 2003. DOI: 10.4271/2003-01-0722
- Tan, Z. and Reitz, R., "An ignition and combustion model based on the level-set method for spark ignition engine multidimensional modeling," *Combustion and Flame*, 145, 1-15, 2006. DOI: 10.1016/j.combustflame.2005.12.007
- Tang, M., Pei, Y., Zhang, Y., Tzanetakis, T., Traver, M., Cleary, D., Quan, S., Naber, J., and Lee, S.-Y., "Development of a Transient Spray Cone Angle Correlation for CFD Simulations at Diesel Engine Conditions," SAE Paper 2018-01-0304, 2018. DOI: 10.4271/2018-01-0304
- Taylor, G.I., *The Scientific Papers of Sir Geoffrey Ingram Taylor, Volume 3*, ed. Batchelor, G.K., Cambridge University Press, 1963.
- Theobald, M.A., and Cheng, W.K., "A Numerical Study of Diesel Ignition," ASME Energy-Source Technology Conference and Exhibition, Dallas, TX, United States, 1987.
- Thompson, K.W., "Time Dependent Boundary Conditions for Hyperbolic Systems". *Journal of Computational Physics*, 68, 1-24, 1987. DOI: 10.1016/0021-9991(87)90041-6
- Thompson, K.W., "Time Dependent Boundary Conditions for Hyperbolic Systems II". *Journal of Computational Physics*, 89, 439-461, 1990. DOI: 10.1016/0021-9991(90)90152-Q
- Thompson, P.A., *Compressible-fluid dynamics*, McGraw-Hill, Inc., 1988.

- Tryggvason, G. and Scardovelli, R., *Direct Numerical Simulations of Gas-Liquid Multiphase Flows*, Cambridge University Press, 2011.
- Turns, S.R., *An Introduction to Combustion*, McGraw-Hill, Inc., 1996.
- van Albada, G.D., van Leer, B., and Roberts, W.W., "A Comparative Study of Computational Methods in Cosmic Gas Dynamics," *Astronomy and Astrophysics*, 108(1), 76-84, 1982.
- Van der Vorst, H.A., "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems," *SIAM Journal of Scientific Computing*, 13(2), 631-644, 1992. DOI: doi:10.1137/0913035
- van Leer, B., "Towards the Ultimate Conservative Difference Scheme II. Monotonicity and Conservation Combined in a Second Order Scheme," *Journal of Computational Physics*, 14(4), 361-370, 1974. DOI: 10.1016/0021-9991(74)90019-9
- van Leer, B., "Towards the Ultimate Conservative Difference Scheme III. Upstream-Centered Finite-Difference Schemes for Ideal Compressible Flow," *Journal of Computational Physics*, 23(3), 263-275, 1977. DOI: 10.1016/0021-9991(77)90094-8
- van Leer, B., "Towards the Ultimate Conservative Difference Scheme V. A Second Order Sequel to Godunov's Method," *Journal of Computational Physics*, 32(1), 101-136, 1979. DOI: 10.1016/0021-9991(79)90145-1
- van Oijen, J.A. and de Goey, L.P.H., "Modelling of Premixed Laminar Flames using Flamelet-Generated Manifolds," *Combustion Science and Technology*, 161(1), 113-173, 2000. DOI: 10.1080/00102200008935814
- Vargaftik, N.B., *Handbook of Physical Properties of Liquids and Gases: Pure Substances and Mixtures*, Hemisphere Publishing Corp., 1983.
- Venkatakrishnan, V., "On the Accuracy of Limiters and Convergence to Steady State Solutions," *31st Aerospace Sciences Meeting*, Reno, NV, United States, 1993. DOI: 10.2514/6.1993-880
- Verhoeven, D., "Spark Heat Transfer Measurements in Flowing Gases," *Oil & Gas Science and Technology - Revue de l'Institut Français du Pétrole*, 52(4), 453-464, 1997. DOI: 10.2516/ogst:1997053
- Vishwanathan, G. and Reitz, R.D., "Development of a Practical Soot Modeling Approach and its Application to Low-Temperature Diesel Combustion," *Combustion Science and Technology*, 182(8), 1050-1082, 2010. DOI: 10.1080/00102200903548124

- Wachters, L.H.J. and Westerling, N.A., "The Heat Transfer from a Hot Wall to Impinging Water Drops in a Spheroidal State," *Chemical Engineering Science*, 21(11), 1047-1056, 1966. DOI: 10.1016/0009-2509(66)85100-X
- Waclawczyk, T. and Koronowicz, T., "Modeling of the Wave Breaking with CICSAM and HRIC High-Resolution Scheme," *ECCOMAS CFD*, Egmond an Zee, The Netherlands, 2006.
- Wakao, N., Kaguei, S., and Funazkri, T., "Effect of fluid dispersion coefficients on particle-to-fluid heat transfer coefficients in packed beds," *Chemical Engineering Science*, 34(3), 325-336, 1979.
- Walatka, P.P., Buning, P.G., Pierce, L., and Elson, P.A., "PLOT3D User's Manual," NASA Technical Memorandum 101067, 1990.
- Wang, B., Miles, P., Reitz, R., and Han, Z., "Assessment of RNG Turbulence Modeling and the Development of a Generalized RNG Closure Model," SAE Paper 2011-01-0829, 2011. DOI: 10.4271/2011-01-0829
- Wang, F., Reitz, R., Pera, C., Wang, Z., and Wang, J., "Application of Generalized RNG Turbulence Model to Flow in Motored Single-Cylinder PFI Engine," *Engineering Applications of Computational Fluid Mechanics*, 7(4), 486-495, 2013. DOI: 10.1080/19942060.2013.11015487
- Warnatz, J., "NOX formation in High Temperature Processes," University of Stuttgart, Stuttgart, Germany, 2001.
- Warsi, Z.U.A., *Fluid Dynamics: Theoretical and Computational Approaches*, CRC Press, Inc., 1993.
- Waterson, N.P. and Deconinck, H., "A Unified Approach to the Design and Application of Bounded Higher-Order Convection Schemes," *Numerical Methods in Laminar and Turbulent Flow, Volume 9*, eds. Taylor, C. and Durbetaki, P., Pineridge Press, 1995.
- Weber, C., "Zum Zerfall eines Flüssigkeitsstrahles," *ZAMM-Zeitschrift für Angewandte Mathematik und Mechanik*, 11(2), 136-154, 1931. DOI: 10.1002/zamm.19310110207
- Wen, J.Z., Park, M.J.S.H., Rogak, S.N., and Lightstone, M.F., "Study of Soot Growth in a Plug Flow Reactor using a Moving Sectional Model," *Proceedings of the Combustion Institute*, 30(1), 1477-1484, 2005.
- Werner, H. and Wengle, H., "Large Eddy Simulation of Turbulent Flow Over and Around a Cube in a Plane Channel," *Proceedings of the Eighth Symposium on Turbulent Shear Flows, Volume 2*, 1991.

- Whitaker, S., "Forced Convection Heat Transfer Correlations for Flow in Pipes, Past Flat Plates, Single Cylinders, Single Spheres, and Flow in Packed Beds and Tube Bundles," *AICHE Journal*, 18(2), 361-371, 1972.
- White, F.M. and Christoph, G.H., "A Simple Theory for the Two-Dimensional Compressible Turbulent Boundary Layer," *Journal of Basic Engineering*, 94(3), 636-642, 1972. DOI: 10.1115/1.3425519
- Wilcox, D.C., *Turbulence Modeling for CFD, Second Edition*, DCW Industries, Inc., 1998.
- Wilcox, D.C., *Turbulence Modeling for CFD, Third Edition*, DCW Industries, Inc., 2006.
- Wilcox, D.C., "Formulation of the k-w Turbulence Model Revisited," *AIAA Journal*, 46(11), 2823-2838, 2008. DOI: 10.2514/1.36541
- Wolfshtein, M., "The Velocity and Temperature Distribution of One-Dimensional Flow with Turbulence Augmentation and Pressure Gradient," *International Journal of Heat and Mass Transfer*, 12(3), 301-318, 1969.
- Woschni, G., "A Universally Applicable Equation for the Instantaneous Heat Transfer Coefficient in the Internal Combustion Engine," SAE Paper 670931, 1967.
- Wruck, N.M. and Renz, U., "Transient Phase-Change of Droplets Impacting on a Hot Wall," Wiley-VCH Verlag GmbH, ISBN 978-3-527-27149-8.
- Xin, J., Montgomery, D.T., Han, Z., and Reitz, R.D., "Multidimensional Modeling of Combustion for a Six-Mode Emissions Test Cycle on a DI Diesel Engine," *Journal of Engineering for Gas Turbines and Power*, 119(3), 683-691, 1997. DOI: 10.1115/1.2817041
- Xin, J., Ricart, L., and Reitz, R.D., "Computer Modeling of Diesel Spray Atomization and Combustion," *Combustion Science and Technology*, 137(1-6), 171-194, 1998. DOI: 10.1080/00102209808952050
- Xu, Y. and Hao, C., "A Novel and Efficient Hybrid RSILU Preconditioner for the Parallel GMRES Solution of the Coarse Mesh Finite Difference Equations for Practical Reactor Simulations," *Journal of Nuclear Science and Engineering*, 194(2), 104-119, 2020. DOI: 10.1080/00295639.2019.1657322
- Yoshizawa, A. and Horiuti, K., "A Statistically-Derived Subgrid-Scale Kinetic Energy Model for Large Eddy Simulation of Turbulent Flows," *Journal of the Physical Society of Japan*, 54(8), 2834-2839, 1985. DOI: 10.1143/JPSJ.54.2834
- Zalesak, S.T., "Fully Multi-Dimensional Flux-Corrected Transport Algorithms for Fluids," *Journal of Computational Physics*, 31, 335-362, 1979.

Zhang, S., Zhao, X., Bayyuk, S., "Generalized formulations for the Rhie-Chow interpolation," *Journal of Computational Physics*, 258, 880-914, 2014.

Zhang, Y., Reuterfors, E.P., McLaury, B.S., Shirazi, S.A., and Rybicki, E.F., "Comparison of computed and measured particle velocities and erosion in water and air flows," *Wear*, 263, 330-338, 2007. DOI: 10.1016/j.wear.2006.12.048

Zhou, G., "Numerical Simulations of Physical Discontinuities in Single and Multi-Fluid Flows for Arbitrary Mach Numbers," Ph.D. Thesis, Chalmers University of Technology, Goteborg, Sweden, 1995.

Zimont, V., Polifke, W., Bettelini, M., Weisenstein, W., "An Efficient Computational Model for Premixed Turbulent Combustion at High Reynolds Numbers Based on a Turbulent Flame Speed Closure," *Journal of Engineering for Gas Turbine and Powers*, 120(3), 526-532, 1998. DOI: 10.1115/1.2818178