

University of Southampton Research Repository

Copyright © and Moral Rights for this thesis and, where applicable, any accompanying data are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis and the accompanying data cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content of the thesis and accompanying research data (where applicable) must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holder/s.

When referring to this thesis and any accompanying data, full bibliographic details must be given, e.g.

Thesis: Author (Year of Submission) "Full thesis title", University of Southampton, name of the University Faculty or School or Department, PhD Thesis, pagination.

Data: Author (Year) Title. URI [dataset]

UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Engineering
Aerodynamics and Flight Mechanics Group

**A two-dimensional strand/Cartesian
Adaptive Mesh Refinement solver for
automated mesh generation around
hypersonic vehicles**

by

Chay William Charles Atkins

MEng

ORCID: 0000-0001-9243-4903

*A thesis for the degree of
Doctor of Philosophy*

November 2022

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Engineering

Doctor of Philosophy

A two-dimensional strand/Cartesian Adaptive Mesh Refinement solver for automated mesh generation around hypersonic vehicles

by Chay William Charles Atkins

The design of an efficient Thermal Protection System (TPS) for a hypersonic vehicle will often require the use of computational simulations at the expected operating conditions. For accurate heating predictions, the computational mesh must adequately resolve the shock structures and boundary layer. Manually refining and updating the mesh in a through-trajectory simulation can consume a large amount of resources as the freestream conditions change and the body may ablate. As such, there is a need for automated mesh generation techniques that can sufficiently resolve the shock structures and boundary layer region.

In this work a two-dimensional strand/Cartesian Adaptive Mesh Refinement (CAMR) solver has been developed within the AMROC framework. The solver combines two highly-automated meshing methods - the CAMR method in the off-body region and the strand mesh method in the near-body region - using overset grid assembly techniques. A hypersonic fluid model and mapped mesh methods have been implemented to enable hypersonic simulations on strand mesh domains. Implicit and Implicit/Explicit (IMEX) methods, specifically developed for the highly-stretched strand grids, have been added to AMROC and their efficiency assessed. A novel strand meshing technique that gives better mesh quality close to the wall has been developed and coupled to a strand-surface deformation algorithm. Finally, an overset grid assembly library that takes advantage of the specific characteristics of the strand and CAMR meshes has been implemented and tested.

Test cases show that the strand/CAMR method is able to capture dynamic shock structures and give accurate heating results, including in cases where shocks cross the overset boundary and impinge on the surface. Simulations of recessing TPS surfaces and free-body motion demonstrate the automated meshing capabilities of the solver. Novel investigations of unsteady, hypersonic flows are carried out using the new solver. The strand/CAMR method is shown to enable automated simulations of complex hypersonic flows and could reduce the resources required for TPS analysis.

Contents

List of Figures	ix
List of Tables	xvii
List of Additional Material	xix
Declaration of Authorship	xxi
Acknowledgements	xxiii
1 Introduction	1
1.1 Motivation	1
1.2 Automated Meshing in Hypersonic Flow Simulations	5
1.3 Research Scope and Key Achievements	11
1.4 Organisation of Thesis	13
2 Literature Review	15
2.1 Physical Models	15
2.1.1 Thermodynamic Model	18
2.1.2 Energy Transfer Between Modes	25
2.1.3 Chemistry Model	29
2.1.4 Thermodynamic-Chemical Coupling	30
2.1.5 Transport Properties	34
2.2 Strand/Cartesian Adaptive Mesh Refinement Solver Review	37
2.2.1 Cartesian Adaptive Mesh Refinement Solver	37
2.2.2 Strand Mesh Generation	43
2.2.3 Strand Mesh Solution Techniques	47
2.2.4 Overset Grid Assembly	49
2.3 Chapter Summary	55
3 Hypersonic Flow Model and Numerical Solution	57
3.1 Governing Equations	57
3.2 Spatial Integration	64
3.2.1 Inviscid Fluxes	66
3.2.2 Viscous Fluxes	73
3.2.3 Boundary Conditions	75
3.2.4 Axisymmetric Formulation	80
3.2.5 Method of Manufactured Solutions	82

3.3	Time Integration	89
3.3.1	AMROC Time Integration	89
3.3.2	Implicit Time Integration	92
3.3.3	Implicit Test Cases	104
3.4	Chapter Summary	111
4	Automated Meshing Techniques	113
4.1	AMROC Framework	113
4.2	Strand Mesh Implementation	116
4.2.1	Strand Generation	116
4.2.2	Strand Clipping	122
4.2.3	Ghost Cell Creation	124
4.2.4	Strand Mesh Motion	125
4.3	Overset Domain Assembly	127
4.3.1	Hole Cutting	128
4.3.2	Receptor and Potential Donor Identification	129
4.3.3	Communication Pattern Set-up	130
4.3.4	Receptor Exchange	133
4.3.5	Donor Search and Interpolation	135
4.3.6	Boundary Data Exchange	145
4.3.7	Verification	145
4.3.8	Donor Search Performance	147
4.4	Chapter Summary	151
5	Verification and Validation	153
5.1	Overset Order-of-Accuracy Tests	154
5.2	Vortex Shedding Over a Square Cylinder	156
5.3	Hypersonic Sphere Shock Stand-off	158
5.4	Blunt Body Heat Flux Predictions	162
5.4.1	Cylinder	163
5.4.2	Spherical Capsule	173
5.5	Shock-Shock Interaction Test Case	176
5.6	Automated Surface Mesh Deformation	180
5.6.1	Recessing Plate	180
5.6.2	Recessing Nose-tip	184
5.7	Free-body Cylinder Simulations	189
5.7.1	Force Integration and ALE Flux Verification	189
5.7.2	Cylinder-Ramp Interactions	191
5.8	Chapter Summary	194
6	Unsteady Flow Predictions	195
6.1	Unsteady Double-Wedge Flow	195
6.1.1	Simulation Set-up	196
6.1.2	Results and Discussion	197
6.1.3	Summary	206
6.2	Type IV Shock Interaction with Surface Recession	207
6.2.1	Simulation Set-up	207

6.2.2	Results and Discussion	209
6.2.3	Summary	216
6.3	Influence of Nose Bluntness on Unsteady Double-Cone Flows	217
6.3.1	Double-Cone Flows	217
6.3.2	Simulation Set-Up	220
6.3.3	Results and Discussion	225
6.3.4	Summary	243
6.4	Chapter Summary	244
7	Conclusion	245
7.1	Summary and Contributions	245
7.1.1	Adaptive Mesh Refinement in Object-oriented C++ (AMROC) Ex- tensions	245
7.1.2	Implicit and Implicit/Explicit Time Integration	246
7.1.3	Novel Strand Mesh Generation Algorithm	246
7.1.4	Overset Library with Novel Donor Search Algorithms	247
7.1.5	Overset Heat Flux Predictions in Hypersonic Flows	247
7.1.6	Strand/Cartesian Adaptive Mesh Refinement (CAMR) Surface Deformation	248
7.1.7	Novel Unsteady Flow Simulations	248
7.2	Possible Future Research	249
7.2.1	Fluid-Material Coupling	249
7.2.2	Extension to Three Dimensions	250
7.2.3	Overset Grid Assembly	251
7.2.4	Implicit Time Integration	252
7.2.5	Cut-Cell Comparisons	252
7.2.6	Alternative Thermochemical Models	252
Appendix A		255
Appendix A.1	Thermochemical Data for Air and Nitrogen	255
Appendix A.2	Method of Manufactured Solutions Inputs and Results	257
Appendix A.3	Block Tri-diagonal Solution Method	261
Appendix A.4	Point Hash Implementation	263
Appendix A.5	Blasius Boundary Layer Comparison	264
Appendix A.6	Compressible Flat Plate Comparison	267
Glossary		271
References		275

List of Figures

1.1	A comparison of simulations using nonequilibrium and ideal gas model fluid models.	3
1.2	The influence of different weather particles on the nose-tip of a hypersonic vehicle (reproduced from Ref. [197] with the author's permission).	4
1.3	Illustrations of h -refinement (top) and r -refinement (bottom).	6
1.4	Comparison of a body-conforming mesh and a CAMR mesh in the stagnation region of a cylinder flow.	6
1.5	An enhanced view of an overset mesh with overlapping near-body and off-body domains.	10
2.1	A velocity-altitude map which gives guidance on the flight regimes where nonequilibrium effects should be considered (redrawn from Ref. [272]).	17
2.2	The vibrational-electronic energy of O_2 and O between 800 K and 20,000 K.	24
2.3	The vibrational-electronic energy of N_2 and N between 800 K and 20,000 K.	24
2.4	A Tree-based Adaptive Mesh Refinement (TAMR) quad-tree structure for a two-dimensional mesh.	39
2.5	A block-Structured Adaptive Mesh Refinement (SAMR) refinement hierarchy with structured rectangular refinement blocks (© Ralf Deiterding).	40
2.6	The velocity field over a double wedge when using a body conforming grid and a Cartesian grid with embedded boundaries.	42
2.7	An example of the original strand mesh generation procedure.	43
2.8	The nodes of a two-dimensional strand mesh before clipping (left) and after clipping (right).	43
2.9	An illustration of the effect of strand mesh smoothing for a surface with convex corners.	44
2.10	An illustration of the difference in growth vectors for a smoothed strand mesh (left) and a multi-strand mesh (right) at a convex part of the surface.	45
2.11	An illustration of a multi-level strand mesh.	46
2.12	An illustration of the multi-level iso-surface and spring analogy smoothing technique.	47
2.13	An illustration showing the structured, high aspect ratio strand mesh and how this influences the time integration methods.	48
2.14	An example of a body-fitted domain overset onto a background Cartesian domain. The dark red cells in the Cartesian domain show the hole-cut.	50
2.15	Ghost cell locations for the overset boundaries, where the dark red points are the receptor centres.	52
2.16	An auxiliary, Cartesian mesh (dark red) is used to group cells from a mapped/unstructured mesh (blue) into spatial buckets.	53

2.17	Stencil walk illustration on structured and unstructured grids. On a structured grid, both i and j indices can be traversed in one step	54
3.1	The stencil for calculating the ξ derivative (left) and η derivative (right) at the blue point (the $i - 1/2, j$ face).	75
3.2	The stencil for calculating the derivatives at the centre of a cell on a mapped mesh.	82
3.3	The concentration, c , through the domains in the diffusion Method of Manufactured Solutions (MMS) case on the mapped mesh.	85
3.4	The normalised errors for the Cartesian and mapped MMS diffusion test case, compared with the theoretical errors.	86
3.5	The sinusoidal mapping and an example density field for the Euler and Navier-Stokes MMS test cases.	87
3.6	The convergence of the density residuals in the subsonic and supersonic MMS test cases for the Harten-Lax-van Leer-Contact (HLLC) flux scheme.	87
3.7	The normalised error convergence for different MMS cases, compared with the theoretical convergence for first- and second-order-accuracy.	88
3.8	A depiction of the index mapping and Left Hand Side (LHS) matrix.	96
3.9	The complex-step and finite-difference L_2 norm error for various step sizes.	97
3.10	A depiction of the index mapping and linear system when using a line-implicit method, where the lines are constructed in the η direction.	101
3.11	A depiction of the additive Schwarz method for η implicit time integration.	103
3.12	The mesh used for the flat plate simulation, stretched in both the wall-normal and wall-aligned directions.	105
3.13	The density residual convergence using different linear solver termination criteria (the curves overlap).	106
3.14	A comparison with the analytic solution for the subsonic flat plate test case.	107
3.15	The density residual convergence plotted against the wall time (left) and iteration count (right) when using implicit and explicit time integration.	110
3.16	The density residual convergence plotted against the wall time when using implicit and explicit time integration.	110
4.1	Visualisations of the growth vectors and different strand types.	117
4.2	A strand mesh generated around a square with different amounts of strand growth vector smoothing.	119
4.3	Visualisations of the vectors used to create the arc.	120
4.4	Visualisations of the arc vectors and geometric parameters and the placement of nodes along the arc.	121
4.5	A comparison of a strand mesh and arcing strand mesh in the convex region of a surface.	122
4.6	A double wedge with strand clipping applied and different amounts of smoothing.	123
4.7	Ghost cell centres for a periodic and non-periodic surface.	124
4.8	Example surface motion vectors for a recessing surface.	126
4.9	A flow chart illustrating the overset grid assembly operations carried out on every processor.	128
4.10	The overset mesh and domain decomposition used in the overset grid assembly description.	129

4.11	The Oriented Bounding Box (OBB) parameters for a two-dimensional OBB.	131
4.12	The overlap algorithm projects the corners of each box onto each axis to determine whether the boxes overlap.	132
4.13	A visualisation of the receptor spatial buckets on the background processors (left) and near-body processors (right) in the cylinder example. .	134
4.14	The receptor buckets that are exchanged following overlap tests between the near-body processors and background processors.	134
4.15	The cell in physical (x, y) space is mapped to (ξ, η) space.	136
4.16	Examples of one-dimensional ranges transformed to two-dimensional points.	138
4.17	An example of an intersection search for a point and a set of one-dimensional ranges.	138
4.18	An illustration of the <i>kd</i> -tree domain decomposition, overlaid with a range search window.	139
4.19	An example of a bucket array overlaid with a range search window. . . .	140
4.20	Examples of OBB (left) and Axis-Aligned Bounding Box (AABB) (right) inverse maps.	142
4.21	The duplicate receptors created in the cylinder example when using a large number of ghost cells.	143
4.22	The cylindrical overset mesh (left) and scalar field (right) used in the interpolation order-of-accuracy tests.	146
4.23	The sinusoidal overset mesh (left) and scalar field (right) used in the interpolation order-of-accuracy tests.	146
4.24	The total memory requirements across both processors for the different search methods.	147
4.25	The set-up times for the different methods.	148
4.26	The set-up times for the different methods with the optimised inverse map creation.	149
4.27	The search times for the different methods.	149
4.28	Overset strand meshes around a re-entry capsule, recessed cone and double wedge.	150
4.29	The donor set-up and search times for a range of mesh geometries. . .	150
5.1	The overset domain used for the order-of-accuracy tests, showing the advected Gaussian density bump.	155
5.2	The results from the overset order-of-accuracy tests.	156
5.3	The near-body mesh with smoothing residuals of 1×10^{-5} (left) and 1×10^{-8} (right).	157
5.4	The strand/CAMR mesh for the square cylinder simulation (left) and the y -velocity profile showing the vortex shedding (right).	157
5.5	The pressure field and strand/CAMR mesh around the sphere in the Mach 11.2 case.	159
5.6	A comparison of the normalised shock stand-off data from the experiments and the simulations.	160
5.7	A comparison of the stagnation line temperature profiles (left) and the mass fractions (right) in the Mach 8.4 case.	161
5.8	A comparison of the temperature field for the ideal gas model (left) and the nonequilibrium air model (right) for the Mach 16.1 case.	161

5.9	A comparison of the stagnation line temperature profiles (left) and the mass fractions in the nonequilibrium simulation (right) for the Mach 16.1 case.	162
5.10	The results from the mesh convergence study.	164
5.11	Visualisations of the thermochemical nonequilibrium in the cylinder test case.	164
5.12	A comparison of the experimental and simulated surface results in the HEG cylinder test case when using different time-integration methods.	165
5.13	The body-fitted and strand/CAMR mesh in the stagnation region.	167
5.14	A comparison of the surface heat flux in the HEG cylinder simulation when using single and overset domains.	167
5.15	The strong scaling results for the overset solver.	169
5.16	An illustration of the synchronisation in an balanced overset SAMR simulation.	170
5.17	The proportion of the overset time relative to the total simulation time when using different overset parallel decompositions.	170
5.18	An illustration of the single-synchronisation method.	171
5.19	The proportion of the overset time relative to the total simulation time when using the single-synchronisation method.	172
5.20	The spherical capsule geometry and flow field.	174
5.21	A comparison of the experimental and simulated surface measurements in the spherical capsule test case.	175
5.22	A schematic of the Edney Type IV shock structure.	177
5.23	The results of the Type IV shock interaction spatial convergence study.	178
5.24	The small (left) and large (right) strand meshes used in the Type IV interaction test case.	178
5.25	The x -velocity field for the large strand mesh. The entire shock interaction is contained within the strand domain.	179
5.26	A comparison of the surface heat flux (left) and pressure (right) results from the simulations and the experiment [286].	180
5.27	The y -velocity field in the nozzle and test section of the arc-heater surface recession test case.	181
5.28	The automated mesh motion for in the arc-heater wedge test case.	182
5.29	The pressure field and mesh refinement levels for the unablated and ablated surfaces.	183
5.30	The surface heat flux and temperature field for the arc-heater wedge test case.	183
5.31	An ablated graphite nose-tip, in a transitional flow regime (redrawn from Ref. [293]).	184
5.32	The x -velocity field in the Deutsches Zentrum für Luft- und Raumfahrt (DLR) L3K simulation when using a cylindrical test piece.	185
5.33	The surface heat flux results for the cylindrical test piece in the DLR L3K arc heater.	186
5.34	The parameters used to define the ablated nose-tip shape.	186
5.35	The upper half of the ablated nose-tip image redrawn from Ref. [293] (left) and the nose-tip surfaces used in the simulations (right).	187
5.36	The automated mesh motion for the ablating nose-tip.	188
5.37	The automated mesh adaptation captures the pulsating shock structure.	188

5.38	The motion of the rectangular body and strand mesh through the background Cartesian domain.	190
5.39	A comparison between the analytic and computed position and body angle for the free-body force integration test.	190
5.40	A comparison of results obtained when using an Eulerian or Langrangian reference frame.	191
5.41	The temperature fields (left) and the mesh close to the cylinder (right) for the free-body simulation after 0 ms, 2 ms and 4.5 ms of motion.	192
5.42	The temperature fields (left) and the mesh close to the elliptical cylinder (right) for the free-body simulation after 0 ms, 2 ms and 4 ms of motion.	193
6.1	The double-wedge geometry.	196
6.2	The convergence study results for the double-wedge simulation at $120 \mu\text{s}$. The grey line shows the intersection of the cones.	197
6.3	The pressure field (left) and strand/CAMR mesh (right) for the double-wedge simulation at $75 \mu\text{s}$	198
6.4	The translational-rotational temperature field (left) and the temperature field over-layed with the strand/CAMR mesh (right) at the double-wedge intersection.	198
6.5	The pressure field (left) and the x -velocity field (right) at the corner of the double-wedge.	199
6.6	The shock structure in the double-wedge simulation at $75 \mu\text{s}$	199
6.7	The temperature field at the corner of the double-wedge after $90 \mu\text{s}$ (left) and $130 \mu\text{s}$ (right).	200
6.8	The time-averaged heat flux for the double-wedge simulation.	201
6.9	A comparison of the heat flux results at different simulation times with single domain results from Ref. [115]. The grey line shows the intersection of the cones.	202
6.10	The numerical Schlieren at $240 \mu\text{s}$ from AMROC compared with that in Ref. [115].	204
6.11	A comparison of the heat flux results at different simulation times using different time integration methods. The grey line shows the intersection of the cones.	205
6.12	A comparison of the heat flux results using the Advection Upstream Splitting Method (AUSM) and HLLC flux schemes.	206
6.13	The temperature field in the baseline, smooth cylinder case and the three recessed surfaces, with recession depths of $0.025R$, $0.05R$ and $0.1R$	208
6.14	The strand/CAMR mesh close to the recessed regions in the shock interaction test case.	209
6.15	An example of the stand/CAMR mesh in the $0.1R$ case	209
6.16	The temperature field and streamlines for one cycle of the oscillations seen in the $0.025R$ recession case. The time between the images is $8 \mu\text{s}$	210
6.17	The pressure-time map for the $0.025R$ case. The black line indicates the centre of the recessed region.	211
6.18	The temperature field and streamlines for one cycle of the large scale oscillations seen in the $0.05R$ recession case.	212
6.19	The pressure-time map for the $0.05R$ case. The black line indicates the centre of the recessed region.	213

6.20	The temperature field and streamlines for one cycle of the oscillations seen in the $0.1R$ recession case. The time between the images is $10\ \mu s$	214
6.21	The pressure-time map for the $0.1R$ case. The black line indicates the centre of the recessed region.	214
6.22	The parameters used to define the ablated nose-tip shape.	218
6.23	An example of the Type IV shock interaction commonly seen in double-cone flows.	219
6.24	The sharp cone surface shown alongside the constant D (ablated) geometries (left) and constant L_1 geometries (right). Images are to scale.	221
6.25	The pressure-time maps for the Mid (left) and Fine (right) grids in the grid convergence study for the $R_{n,2}$ constant D geometry.	222
6.26	The inviscid axial force coefficient plotted against time (left) and power spectrum (right) of the axial force coefficient in the grid convergence study.	222
6.27	Numerical Schlieren comparison.	223
6.28	The Adaptive Mesh Refinement (AMR) mesh at two different times in the pulsation cycle.	223
6.29	The automated mesh motion for an ablating double-cone geometry.	224
6.30	The numerical Schlieren and streamlines for approximately one unsteady oscillation for the sharp cone geometry.	226
6.31	The pressure-time map for the sharp cone surface. The black line represents the double-cone intersection.	227
6.32	The inviscid axial force coefficient (left) and Power Spectral Density (PSD) (right) of the axial force coefficient for the sharp cone case.	228
6.33	The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,1}$, constant D geometry.	229
6.34	The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,1}$ constant L_1 geometry.	230
6.35	The pressure-time map for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,1}$	231
6.36	The axial force coefficient plotted against time for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,1}$	232
6.37	The axial force PSD plot for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,1}$	232
6.38	The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,2}$, constant D geometry.	234
6.39	The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,2}$ constant L_1 geometry.	235
6.40	The pressure-time map for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,2}$	236
6.41	The axial force coefficient plotted against time for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,2}$	236
6.42	The axial force PSD plot for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,2}$	236
6.43	The numerical Schlieren and streamlines for the steady case, $R_{n,3}$, constant D geometry.	237
6.44	The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,2}$ constant L_1 geometry.	238

6.45 The pressure-time map for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,3}$	239
6.46 The axial force coefficient plotted against time for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,3}$	239
6.47 The axial force coefficient PSD plot for the constant L_1 geometry, with $R_n = R_{n,3}$	239
6.48 The pressure (left) and shear stress (right) on cones with different nose radii.	240
6.49 The entropy field for the sharp cone (left) and $R_{n,1}$ cone (right). Calculated using $S = [\gamma/(\gamma - 1)] \ln(T/T_\infty) - \ln(p/p_\infty)$	241
6.50 The C_A power Root-Mean-Square (RMS) as a function of R_n/D (left) and R_n/L_1 (right).	243
Appendix A.1 The normalised error convergence for the subsonic AUSM fluxes.	258
Appendix A.2 The normalised error convergence for the supersonic AUSM fluxes.	258
Appendix A.3 The normalised error convergence for the subsonic HLLC fluxes.	258
Appendix A.4 The normalised error convergence for the supersonic HLLC fluxes.	259
Appendix A.5 The normalised error convergence for the subsonic AUSM+ fluxes.	259
Appendix A.6 The normalised error convergence for the supersonic AUSM+ fluxes.	259
Appendix A.7 The normalised error convergence for the supersonic van Leer fluxes.	260
Appendix A.8 The overset strand mesh used in the Blasius boundary layer simulations.	265
Appendix A.9 A comparison between the AMROC strand mesh boundary layer profile and the Blasius boundary layer profile for three different blowing wall conditions.	266
Appendix A.10 A comparison of the boundary layer velocity profile using a strand and Cartesian domain.	268
Appendix A.11 A comparison of the boundary layer velocity profiles over an adiabatic flat plate, where $M_\infty = 3.0$	269
Appendix A.12 A comparison of the boundary layer temperature profile over an adiabatic flat plate, where $M_\infty = 3.0$	269
Appendix A.13 A comparison of the boundary layer velocity profile over an isothermal flat plate, where $M_\infty = 3.0$	270
Appendix A.14 A comparison of the boundary layer temperature profile over an isothermal flat plate, where $M_\infty = 3.0$	270

List of Tables

3.1	The constants used in the diffusion MMS test case.	84
3.2	Results from the diffusion MMS case on the Cartesian mesh.	84
3.3	Results from the diffusion MMS case on the mapped mesh.	85
3.4	Linear solver data for the subsonic flat plate test case.	107
3.5	Linear solver data for the supersonic flat plate test case.	108
3.6	Linear solver data for the cylinder test case.	108
3.7	The total time spent in each part of the implicit update for the three steady-state test cases.	111
4.1	The reduction in searches when using Locality Sensitive Hashing (LSH). Data is from one near-body processor, using the receptors shown in Fig. 4.21.144	
4.2	The results from the order-of-accuracy test on the cylindrical mesh. . . .	146
4.3	The results from the order-of-accuracy test on the sinusoidal mesh. . . .	146
5.1	The Strouhal number as a function of the smoothing residual for the square cylinder simulation.	158
5.2	The difference between the simulated and experimental shock stand-off distances, with and without the experimental error estimates.	160
5.3	Freestream conditions for the HEG cylinder simulation.	163
5.4	The synchronisation times for the HEG test case using different overset parallel decompositions and the multi-synchronisation method.	171
5.5	A comparison of the synchronisation times for the HEG test case using different overset parallel decompositions.	172
5.6	Freestream conditions for the spherical capsule simulation.	173
5.7	The freestream conditions for the Type IV shock-interaction case [303]. .	177
5.8	The parameters used to define the ablated nose-tip shape.	187
6.1	The freestream conditions for the double-wedge simulation.	197
6.2	The wall times for the different time integration methods.	205
6.3	Freestream conditions for the ideal gas double-cone simulations.	220
6.4	The parameters used to define the blunted double-cones.	220
6.5	The computational meshes used in the mesh refinement study.	221
6.6	The non-dimensional length parameters and unsteadiness metrics for the blunted double-cone geometries.	242
Appendix A.1	The species constants used in this work.	255
Appendix A.2	The constants and rate-controlling temperatures used in the Park two-temperature chemical model.	255
Appendix A.3	The Millikan and White constants used relaxation model. . .	256

Appendix A.4 The species constants used in the Blottner viscosity model. . .	256
Appendix A.5 The constants and functions used to create the solutions for the supersonic Euler MMS test cases.	257
Appendix A.6 The constants and functions used to create the solutions for the subsonic Euler MMS test cases.	257
Appendix A.7 The constants and functions used to create the solutions for the subsonic Navier-Stokes MMS test case.	257
Appendix A.8 Freestream conditions for the flat plate test case.	267

List of Additional Material

Thesis Data: <https://doi.org/10.5258/SOTON/D2433>

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. None of this work has been published before submission

Signed:.....

Date:.....

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Ralf Deiterding, for providing the foundations of this research and for his insightful guidance throughout my studies. His deep understanding of both the theoretical and practical aspects of computational simulations has been key to this project's completion. I would like to thank my internal examiner, Dr. Kim Mink-Wan, for the constructive comments he has provided at each review stage, which have helped shape the direction of this research. Many of the test cases used in this work relied upon easy access to a well maintained super-computing system, so I would like to gratefully acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton.

There are a number of colleagues I wish to thank for the support they have provided throughout this project. I would like to thank Keith and Martin for obtaining the funding for and initiating this research. I am deeply indebted to the people who provided me with the time and independence to fully immerse myself in this work, particularly the Programme Manager, Joanne, the Project Technical Authorities, Mike and James, the Project Managers, Julie and David, and my Team Leaders, Simon, John and Carl. I would like to thank Duncan for his careful proof reading of my thesis, and various other manuscripts, and the helpful comments he provided. Without the support of all these people this project would not have been possible.

Finally, I would like to thank my family for encouraging and supporting me throughout my education, and my partner for her unwavering belief in me during this process.

Chapter 1

Introduction

1.1 Motivation

Over the past 20 years there has been an increasing amount of investment and interest in manned space flight. This has led to a range of new hypersonic vehicle concepts being designed and tested by organisations such as SpaceX, Blue Origin, Boeing, the Sierra Nevada Corporation and the European Space Agency (ESA). In addition to these vehicles, there is continued investment in hypersonic cruise and glide vehicles that remain within the atmosphere. Both types of hypersonic vehicle experience an extremely harsh aerothermal environment during atmospheric flight. The post-shock temperatures can rise to thousands of Kelvin [9] so a robust Thermal Protection System (TPS) is needed to protect the vehicle's payload. Failure of the TPS is likely to result in the destruction of the payload, with potential loss of life, so it is crucial that the TPS is able to withstand the expected flight conditions.

Creating an efficient TPS to shield the payload from the extreme aerothermal environment is one of the main engineering challenges hypersonic vehicle designers face. Using a thicker TPS, or denser materials, can increase the level of protection provided, but must be balanced against the aerodynamic and weight penalties incurred. For re-entry vehicles any unnecessary weight can severely reduce the vehicle's payload capacity and the missions it can carry out. For hypersonic cruise or glide vehicles designed for prolonged atmospheric flight a larger TPS can significantly reduce the vehicle's aerodynamic performance. Along with any superfluous weight, this can have a significant adverse effect on the range or speed of the vehicle.

One of the main difficulties engineers face when designing a TPS is obtaining data for the system at its operating conditions. Full flight experiments of hypersonic vehicles are expensive and this often means that only a limited number of flight-tests are conducted. In addition, it can be difficult to obtain accurate data on the aerothermal

environment and the TPS material response from flight tests. This leads to a reliance on ground-test facilities. However, such facilities are currently unable to match the combination of high thermal loads and high-speed flows that can be experienced during hypersonic flight [133]. Therefore, continuum Computational Fluid Dynamics (CFD) simulations and Direct Simulation Monte Carlo (DSMC) methods play a key role in understanding the TPS performance. Simulations can aid in the creation of more efficient prototype designs and reduce the number of flight tests required, resulting in significant performance benefits and cost savings. Consequently, a large amount of effort has been spent on the development of the models and methods required to simulate hypersonic flow-fields and the response of the TPS.

The development of upwind flux methods by Steger and Warming [259], and Roe [230] enabled the use of shock-capturing finite-volume solvers for more accurate modelling of high-speed flows that include shock waves. However, the ideal gas model that is applicable in the low supersonic regime does not account for the high-temperature and high-speed gas dynamics that arise in the hypersonic regime. The extreme post-shock temperatures result in the dissociation and ionisation of the chemical species within the gas, as well as the transfer of energy to internal modes, such as the rotational, vibrational and electronic modes, and transfer of energy to electrons. Both processes take a finite time to complete and these processes may occur throughout the shock layer as a result of the high flow speeds and, at high altitude, the low pressure. Therefore, one cannot assume that the fluid is in chemical or thermodynamic equilibrium and thermochemical nonequilibrium models must be used in order to accurately simulate hypersonic flows. Figure 1.1 shows a comparison between a nonequilibrium air simulation and an ideal gas simulation of a high-enthalpy flow experiment (see Section 5.4.1 for details). One can see that the shock standoff distance is significantly larger and the temperature in the shock layer is much higher in the ideal gas simulation. This illustrates the influence that the fluid model can have on aerothermal predictions in hypersonic flows.

Thermochemical nonequilibrium models (discussed in detail in Section 2.1) were first incorporated into finite-volume solvers in the late 1980's, with Ref. [41] being an early example. Similar thermochemical nonequilibrium models have been employed in a number of codes over the last 30 years, and have been used to obtain heat flux predictions for hypersonic flows that agree well with experimental data. Examples of hypersonic CFD solvers include the structured solver LAURA (Langley Aerothermodynamic Upwind Algorithm) [50] and unstructured solvers such as DPLR (Data Parallel Line Relaxation) [296], US3D (UnStructured 3D) [207] and LeMANS (The Michigan Aerothermodynamic Navier-Stokes solver) [242].

One challenge that has been encountered in hypersonic flow simulations is that the quality of the computational mesh can have a large impact on the accuracy of the surface results [39, 99, 179, 212]. Firstly, the mesh needs to be well aligned with the

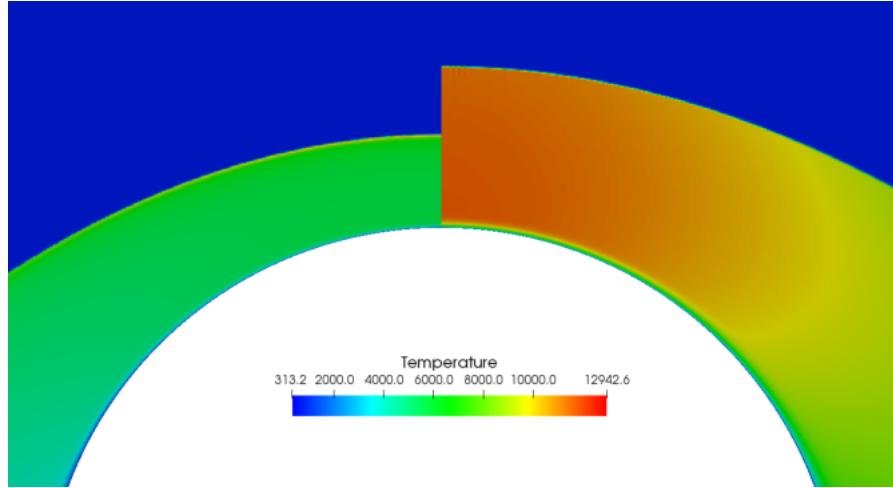


Figure 1.1: A comparison of the stagnation region in a simulation of a shocktube experiment when using a nonequilibrium model (left) and an ideal gas model (right). See Section 5.4.1 for the flow conditions.

shock structure in order to accurately capture the shock location and reduce errors in the stagnation region. It has been shown that poor shock alignment in the stagnation region can result in poor predictions of the pressure and entropy at the edge of the boundary layer, leading to unacceptable heat flux predictions [99]. Candler *et al.* demonstrated that triangular and tetrahedral grids with poor shock alignment created a “large, spurious velocity component” immediately behind the shock in the stagnation region, which greatly affected the flow field in the shock layer and the surface heat flux predictions [39]. In addition to shock alignment, the large gradients in the boundary layer require a highly resolved near-wall mesh, with a recommended near-wall cell Reynolds number less than one [23, 39, 191, 212]. Investigations have shown an error of up to 100% in the stagnation region heat flux results when using a cell Reynolds numbers as low as 10 [191], highlighting the importance of near-wall resolution. Finally, maintaining the orthogonality of the mesh to the surface is known to be important for predictions of skin friction, and is likely to impact heat flux results in a similar way [95].

The high-quality grid required for accurate surface heating results can be difficult and time-consuming to create for many hypersonic vehicles and flows. Firstly, the location of any shock waves, shock-shock interactions and shock-boundary-layer interactions are not known *a priori* so an adaptive process is often required to obtain accurate surface predictions. Shock structures will also change throughout a vehicle’s trajectory as the flow conditions change, requiring the process to be repeated for each condition of interest. Secondly, many hypersonic vehicles’ TPSs use ablative materials so the external geometry of the vehicle may change during flight due to pyrolysis, sublimation and spallation caused by heating and shear stress, as well as through erosion caused by dust, ice and water in the atmosphere (see Fig. 1.2). As changes to

the shape can have a large impact on the heat flux [9], and the ablative shape change is strongly coupled to the heat flux [179], modelling of a vehicle's shape change whilst maintaining a high-quality mesh is vital for accurate TPS analysis. This introduces significant complexity to the mesh generation process, as the updated mesh must account for the surface deformation and the change in the shock structure that this causes.

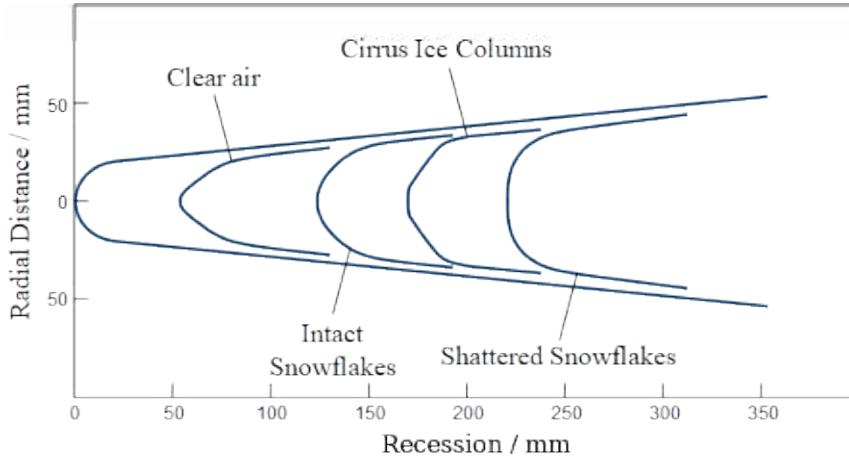


Figure 1.2: The influence of different weather particles on the nose-tip of a hypersonic vehicle (reproduced from Ref. [197] with the author's permission).

Flexible TPSs have been the focus of recent research due to their ability to reduce the ballistic coefficient of re-entry vehicles [68]. Such systems allow for potentially large changes in the vehicle geometry, and the deformation and heating of a flexible TPS will be highly coupled to the flow field. A high-quality mesh is needed throughout simulations of these systems in order to accurately assess the heat fluxes and forces acting on the flexible body. Finally, there are challenges for hypersonic vehicles that require control through flight, such as the ESA's Space Rider and the Sierra Nevada Corporation's Dream Chaser variants, that are not unique to hypersonic applications. For example, the aerodynamic shape of the vehicle can create complex shock structures, a new mesh needs to be created each time a control surface is moved, and assessment of the dynamic stability of the vehicle requires time accurate simulations of the vehicle in motion.

In simulations of many hypersonic vehicles, where the vehicle's geometry or the shock structure is changing throughout flight, manual mesh generation and shock alignment/refinement can consume significant resources. For each change in the vehicle's shape or configuration and for changes in the flow conditions, a new mesh must be created that adequately resolves the shock structures and boundary layer. This is often a time-consuming, iterative process, requiring large person and computer hours. This research aims to address this challenge by investigating the use of highly-automated meshing techniques for hypersonic aerothermodynamic

predictions. New automated meshing techniques could increase the use of time-accurate and through-trajectory simulations of ablative and flexible TPSs, and reduce the resources required to generate aerothermodynamic databases. In turn, this could lead to the design of more efficient systems. The remainder of this section reviews automated meshing approaches in the context of hypersonic flow simulations, outlines the approaches used and the scope of this research, and presents the organisation of the thesis.

1.2 Automated Meshing in Hypersonic Flow Simulations

Adaptive Mesh Refinement (AMR) is commonly used to refine shock structures in supersonic flow applications by automatically detecting areas of high truncation errors and refining the spatial discretisation in these areas. AMR methods can be classified as either p -refinement, h -refinement or r -refinement. In p -refinement, the order-of-accuracy of the underlying solver is increased, often by increasing the size of the numerical stencil. As shocks are singularities in the solution, h - and r -refinement, which increase the mesh resolution, are generally more suitable for supersonic applications. h -refinement increases resolution by adding extra cells to regions where there are large truncation errors. On the contrary, r -refinement keeps the total number of cells and the cell connectivity the same, but moves the locations of the grid points to increase resolution. Illustrations of both h - and r -refinement are shown in Fig. 1.3. Ideally, the AMR algorithms will be highly automated and produce a mesh that gives a similar numerical error to a uniformly fine mesh, but with a significantly lower computational expense. If both can be achieved, it can significantly reduce the manual and computational resources required in mesh generation.

Generally, AMR methods can be classified into Cartesian Adaptive Mesh Refinement (CAMR) methods where all of the elements in the mesh are Cartesian, or Unstructured Adaptive Mesh Refinement (UAMR) where the elements can have any shape. CAMR methods are more commonly used with h -refinement techniques, with an early example being that of Berger and Oliger [22]. This method was later modified to become the widely used block-Structured Adaptive Mesh Refinement (SAMR) method [21], which refines the grid by overlaying structured blocks. Cartesian methods have also been developed where the cell-based refinement is instead based on tree-based data structures [70, 224] or unstructured domains [4, 15]. The advantages of CAMR methods depend on the implementation but include high-quality grids, efficient memory access patterns and spatial integration, and implicit connectivity. The primary disadvantage of CAMR is the difficulty in accurately representing boundaries using Cartesian cells, requiring the use of immersed or embedded boundary methods, such cut-cell [127] or ghost-fluid methods [64]. As well as this, coarse-fine boundaries often result in “hanging nodes” where nodes on the refined mesh fall in between

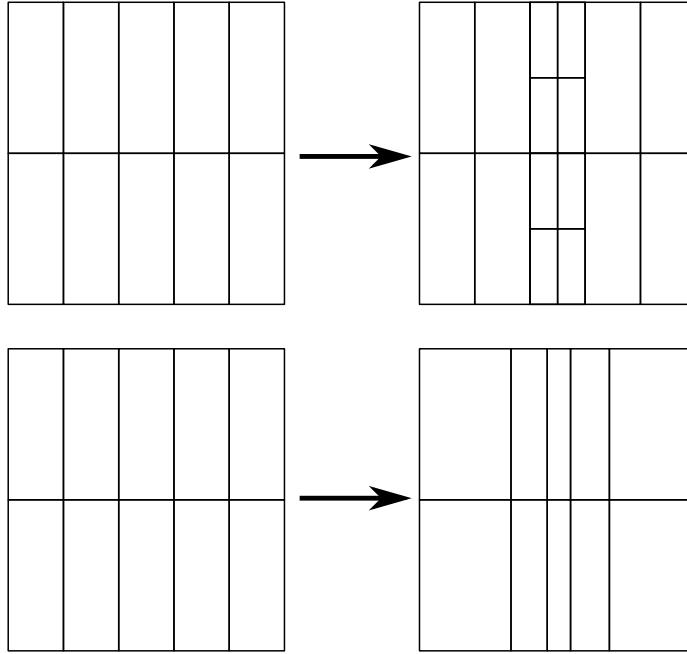


Figure 1.3: Illustrations of h -refinement (top) and r -refinement (bottom).

nodes on the adjacent, coarser mesh. This requires additional algorithms in order to ensure that the method remains conservative [64]. A comparison of a static, body-conforming mesh and a CAMR mesh is shown in Fig. 1.4, where the good shock refinement but poor boundary representation of the CAMR method can be seen.

UAMR can use grid elements of arbitrary shape enabling more accurate boundary

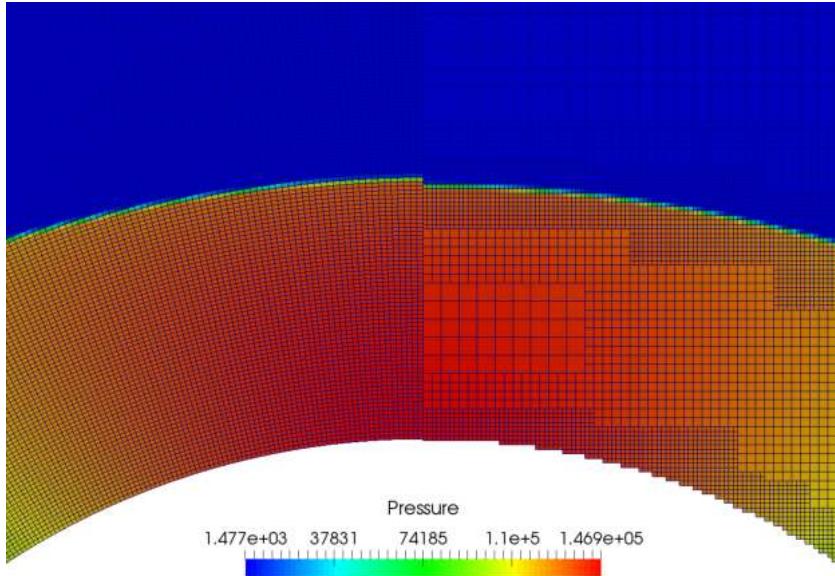


Figure 1.4: Comparison of a body-conforming mesh (right) and a CAMR mesh (left) in the stagnation region of a cylinder flow.

representations. UAMR is often referred to as anisotropic mesh adaptation and can utilise both h - and r -refinement techniques in order to improve the grid quality [87, 110]. Most early examples of these methods were based on the finite-element method [45], but they have since become more widely used with the finite-volume method as well [18, 28, 210]. Although UAMR methods are better at representing complex geometries than CAMR, they can be less computationally efficient and require more storage due to their unstructured nature. In addition, complex algorithms are required to ensure that adequate cell quality is maintained. For example, that the cell skewness and grid smoothness remain within acceptable bounds [71, 110]. Where h -adaptivity is used, the UAMR algorithms must typically ensure that the mesh is conforming, i.e. “the face of an element cannot be adjacent to a subdivided element face” [71], which usually leads to the refinement of a number of surrounding cells. In purely r -adaptive UAMR methods, the adaptation is naturally limited by the number of cells in the mesh at the start of the simulation, as this remains constant. To avoid coarsening the mesh in areas away from the shock that also require high resolution (e.g. the boundary layer), empirical weighting algorithms must be used [28, 210] that can reduce the level of automation. Finally, UAMR methods often require converged intermediate solutions and several applications of the mesh adaptation algorithm to reach a final converged solution [18, 28]. This can limit the methods efficiency in time-accurate simulations with unsteady shock structures or moving bodies.

AMR techniques have previously been applied to hypersonic flow simulations. CAMR has been used to refine both bow shocks [292] and shock-shock interactions [288, 289] in hypersonic simulations by adding cells to regions where discontinuities are detected. A simple r -adaptive approach, often referred to as “shock tailoring” has been widely used within hypersonic flow software. In this method the location of the shock is detected and then the nodes in each structured line of the mesh are scaled so that the freestream boundary is moved closer to the shock. The number of nodes in the mesh remains the same, but moving the boundary reduces the number of grid points outside of the shock layer, effectively increasing the resolution and aligning the grid to the shock. This method has been used in a number of hypersonic flow solvers including LAURA [99], DPLR [239], and US3D [40] but is largely limited to simple convex surfaces where the mesh extends from the surface to the freestream boundary along structured lines [40, 82]. r -adaptive methods around complex bodies have been demonstrated for hypersonic flows using both structured [210] and unstructured grids [18, 28]. The results showed that r -adaptive methods are able to effectively capture shock structures and can lead to “significant differences” in surface results [28].

One set of methods that can be used to simulate recessing surfaces or moving bodies are mesh deformation techniques, where the shape of the mesh changes to account for boundary motion. An early example of such a technique utilised a spring analogy to

control changes in the mesh topology [17]. In this method the elements connecting the nodes on the mesh are modelled as linear springs where the stiffness of each spring is inversely proportional to the distance between the mesh nodes. After a boundary has moved, the updated node locations can be found by iteratively solving a static equilibrium equation [17]. The original method could lead to invalid cells in the presence of large mesh deformation and extensions of the method, such as the use of additional torsional springs [83], have been proposed to improve robustness. An independently developed but analogous mesh morphing technique uses linear elastic theory [130] to control the mesh deformation. A more recent mesh-morphing technique is the Radial Basis Function (RBF) method introduced by De Boer [60] and developed further by Rendall and Allen [228]. This method determines the displacement of each node in the updated mesh by evaluating RBFs using the distance of each mesh node to each “control” location on the deforming surface. This method can be used with arbitrary mesh topologies and has been applied to a range of areas, including fluid-structure interaction problems [228] and surface deformation due to icing of wings [107].

Until recently these techniques have not been widely used within hypersonic flow solvers to account for surface recession. Most previous examples of mesh deformation have used a basic method that was first used in DPLR [239] and has since been implemented in LeMANS [178, 242] and US3D [40]. In this method, the cells near the wall move with the recession, and those at the outer boundary remain in place. The method requires that the mesh uses structured lines between the wall to the boundary, that the ratio of each cell height remains constant, and that the total distance from the wall to the external boundary remains constant. As such, this method does not allow for large deformations or concave regions. Other authors have used a looser coupling approach, where an external program was used to generate a mesh, taking a parameterised surface as an input [101, 151]. Recently, the more flexible mesh deformation methods discussed above have started to be adopted in hypersonic research, with linear elastic theory being used in the shape optimisation of hypersonic bodies [81] and the RBF method being used for surface shape change in an ablative material response solver [243] and a hypersonic flow solver [304]. A mesh deformation technique specifically designed for stability analysis of rotating bodies has been implemented in US3D and used to perform dynamic stability analysis of hypersonic vehicles [260]. In this method, the near-body region simply follows the rotation of the body, whilst the off-body region remains stationary. The deforming region then smoothly varies between the two [40, 260].

Whilst mesh morphing techniques are able to update the computational mesh as the surface deforms or the body moves, there are limitations. The mesh quality can reduce significantly if the mesh deformation becomes too large, if there are bodies in relative motion, or if the body undergoes a large rotation relative to the boundaries. In such

situations, automated re-gridding is required and the solution must be interpolated to the new mesh [131]. Whilst this has previously been demonstrated for specific scenarios [131], it requires additional automated meshing capabilities for arbitrary computational domains and incurs a considerable computational cost. In addition, the mesh morphing methods do not naturally incorporate mesh refinement around shocks, which is a key requirement for accurate heat flux predictions. There are previous examples in the hypersonic literature where the simple “shock tailoring” r -adaptive method has been combined with the basic mesh deformation method from DPLR. Examples of using this approach for coupled simulations with recessing surfaces can be found in the work of Nompelis *et al.* [206], Chen *et al.* [51] and Martin *et al.* [179]. However, this combined method has the limitations that are inherent to both methods, and is therefore limited to simple convex surfaces where structured lines are used between the surface and freestream boundaries [40, 82]. More advanced methods are required for geometrically complex bodies, bodies in motion or surfaces with concave regions caused by differential recession.

Another technique that can provide a high level of automation and flexibility is the overset gridding method. Overset (a.k.a. composite or Chimera) grid methods were introduced through the work of Starius [257], Kreiss [150], Henshaw and Chesshire [52, 119], and independently by Steger and Benek [20, 258]. In this method overlapping computational grids are joined into a single solution through the exchange of information as boundary conditions (see Fig. 1.5). Overset methods have been used within hypersonic flow simulations for manual grid refinement of shocks structures [28] and wake flows [200] by manually increasing the cells in one or more overset blocks. Overset methods are also commonly used for moving body simulations as the body-fitted mesh is able to move with the body against a fixed background mesh. This has previously been demonstrated for aerodynamic force assessments of hypersonic vehicles [196, 260]. However, there are disadvantages to using overset methods. Firstly, the algorithms required to assemble the different meshes into a single domain can increase the computational times and add complexity to the underlying software. Secondly, depending on the overset software, automation can actually be reduced, with manual shock refinement required and potentially large amounts of user input to specify block boundaries and connectivity [188]. This said, the level of automation is highly dependent on the specific software, and is not necessarily a consequence of using overset techniques. Finally, overset methods are not conservative if simple spatial interpolation methods are used to provide the required information at the domain boundaries’ ghost cells. Although overset methods offer great flexibility, there is relatively little research on the influence of overset boundaries on heat flux predictions for hypersonic vehicles and, as discussed above, previous work has shown that surface heating predictions are highly sensitive to the spatial discretisation.

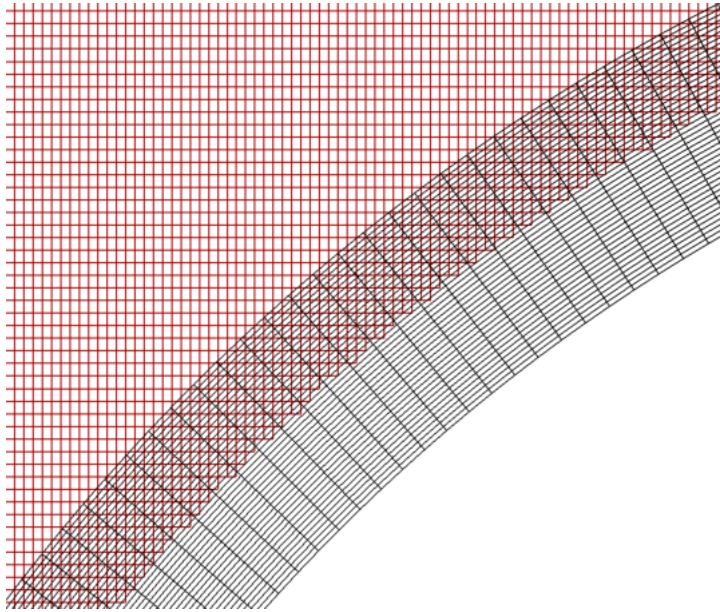


Figure 1.5: An enhanced view of an overset mesh with overlapping near-body and off-body domains.

The above review highlights a number of examples in the literature where researchers have used automated techniques to meet the demands of mesh generation for hypersonic flow simulations. However, no single method has been able to meet all of the challenges presented by recessing surfaces with concave regions, complex aerodynamic geometries and bodies in motion whilst maintaining the required shock and boundary layer resolution. One promising method that has not previously been applied to hypersonic aerothermal predictions is the combination of CAMR and overset methods. CAMR methods have an excellent shock-capturing capability, but the Cartesian cells make it difficult to accurately represent complex surfaces and create grids suitable for heat fluxes predictions. Overset grid methods offer a large amount of flexibility and automation, and outside of hypersonic research they have been used in simulations that include large surface deformations [84]. However, overset methods often require the background grid to be manually refined if the surface geometry or flow conditions (and therefore shock-structures) change [28, 200]. The limitations of the two methods - poor surface representation and inefficient shock-capturing - can be negated by combining them into a single overset-CAMR solver. In such a solver, body-fitted near-body grids can be used to accurately capture the boundary layer, whilst AMR in the off-body region can accurately capture shocks and flow features.

Early examples of overset-CAMR solvers include the work of Brislawn *et al.* [32], Meakin [185, 186], and Boden and Toro [25]. A series of investigations, detailed in Refs. [16, 120, 121, 122], coupled high-speed inviscid reacting flows to deforming solid geometries using the AMR and overset methods within the Overture framework [33]. The investigations demonstrated the capability of a combined meshing procedure to capture shocks and solid-body deformations in a time-accurate manner. However, the

use of inviscid flows gives limited insight into whether such a method could be used for heat flux and shear stress computations. In addition, simple, structured curvilinear near-body meshes were used as there was no need to account for boundary layers and the bodies investigated were convex in shape. More recently, for subsonic and transonic simulations, a more flexible, and highly automated, near-body “strand meshing” technique has been developed and combined with an off-body CAMR solver. This near-body meshing method was introduced by Meakin *et al.* [190] and enables high-resolution boundary layer meshes to be produced around complex bodies. The viscous strand/CAMR solver was shown to give good agreement with experimental results when simulating complex three-dimensional shapes [137, 154, 290, 291]. These investigations included simulations where fixed near-body meshes were in motion relative to a background grid, but there was no investigation of using strand mesh techniques to simulate deforming geometries. In addition, there has been no research into whether the automatically generated strand meshes are suitable for surface heat flux predictions.

1.3 Research Scope and Key Achievements

The strand/CAMR method offers a high-level of automation and flexibility, and has previously given accurate results for subsonic and transonic flows. However, it has not been applied to hypersonic flow problems and, as stated by Candler, “methods that show promise for transonic and low supersonic flows may not work at hypersonic conditions” [39]. Consequently, the first aim of this work is to determine whether a combined strand/CAMR solver is suitable for highly-automated hypersonic flow simulations. This will be assessed in terms of the accuracy of heat flux predictions and the level of automation in the presence of deforming surfaces, dynamic shock structures and surfaces in relative motion. A second aim of this research is to optimise the numerical integration techniques, overset methods and strand mesh generation algorithms for efficient and automated hypersonic aerothermal simulations. The final aim of this work is to utilise a strand/CAMR solver’s capabilities to obtain new insights into hypersonic flow phenomena. To achieve these aims, a strand/CAMR solver has been developed that utilises a set of governing equations and physical models suitable for hypersonic flow predictions.

In this work, Park’s two-temperature thermochemical nonequilibrium fluid model has been added to the AMROC framework. In addition, mapped mesh methods have been implemented within AMROC for the first time. These contributions enable simulations of thermochemical nonequilibrium flows on adaptive Cartesian meshes and on body-fitted and strand domains. The verification of these additions includes the first published use of the Method of Manufactured Solutions (MMS) with the two-temperature thermochemical model, as detailed in Section 3.2.5.

Implicit and Implicit/Explicit (IMEX) time-integration methods have been implemented within AMROC as part of this research. This enables larger time-steps to be taken on the strand domain, where the thin near-wall cells can significantly restrict the time-step. A range of linear solvers have been implemented that take advantage of the quasi-structured strand mesh. To the best of the author's knowledge, this work includes the first comparison of different linear solvers' performance on strand domains (see Section 3.3.3). The implicit and IMEX methods are shown to give a significant speed-up over the previous explicit methods. For example, a speed-up of over 21 is achieved for the test case in Section 6.1 when using the IMEX method. The implicit methods are a major contribution to the AMROC framework. Without this contribution many of the simulations in Chapter 5 and 6 would not have been computationally tractable.

A novel strand mesh generation algorithm has been developed in this work and is described in Section 4.2. The new method is able to create a strand mesh that is orthogonal at the wall and smooth away from the wall, with a low memory footprint. In Section 5.2 the novel technique is shown to produce results that are significantly less sensitive to user input than the previous method, clearly demonstrating the benefits of the new algorithm. The strand mesh generation algorithm has been coupled with a surface deformation algorithm. As far as the author is aware this is the first time that the strand meshing paradigm has been combined with surface deformation techniques.

An overset library has been implemented which joins the near-body strand domain with the off-body CAMR domain. A unique set of state-of-the-art overset grid assembly tools have been developed, which take advantage of the underlying strand and SAMR structures to minimise communication overheads, reduce donor search times and enable high levels of automation. A novel Locality Sensitive Hashing (LSH) technique has been developed which is shown to significantly reduce the number of near-body searches required even when the off-body domain is adapting. The performance of different unstructured donor search methods are investigated in Section 4.3.8. A novel inverse map set-up algorithm has been implemented that significantly reduces the inverse map set-up times. Using this algorithm, the inverse map methods are found to greatly outperform the more commonly used Alternating Digital Tree (ADT) method in both the set-up times and search times.

A significant contribution of this thesis is a detailed assessment of the influence of overset boundaries on hypersonic heat flux predictions. To the author's knowledge, this has not previously been studied in detail. The strand/CAMR surface heating results are compared with single domain and experimental results for blunt body test cases (see Section 5.4) and test cases with complex shock structures that cross the overset boundaries (see Sections 5.5 and 6.1). The overset results are found to be in

good agreement with the reference results. This gives confidence that the strand/CAMR method can be used for hypersonic heating assessments.

Another contribution of this research is the first application of the strand/CAMR paradigm to surface deformation simulations. The results in Section 5.6 demonstrate that the new solver is able to incorporate surface deformation with minimal user input. The solution does not need to be re-initialised after the surface deforms and the AMR is shown to automatically capture changes in the shock structures. These results indicate that the new solver is well suited to simulations of ablating hypersonic bodies. In addition, two free-body simulations are carried out in Section 5.7, further highlighting the automated meshing capabilities and flexibility of the new solver.

Finally, the new solver has been used to study dynamic, unsteady flow structures that have not previously been investigated. The strand/CAMR paradigm enables efficient resolution of shock structures in the off-body AMR domain, and accurate representation of the boundary layer in the strand domain. These capabilities are demonstrated in simulations of a Type IV shock interaction impinging on an ablated surface in Section 6.2. The results show that the shape of the ablated region can have a significant impact on the unsteady flow features. Two distinct modes of oscillating flow are identified, where the mode, frequency and amplitude of oscillation is dependent on the surface geometry. The effect of nose bluntness on the unsteady flow over a double-cone is investigated in Section 6.3. The simulations show that small changes in nose bluntness can have a large impact on the amplitude and frequency of the oscillating forces experienced by the double-cone. This coupling between nose bluntness and the flow features could be important for an ablating TPS, where the nose bluntness will change over time.

The solver is currently limited to two-dimensional and axisymmetric flows, with a single overset near-body grid which is based on a structured, curvilinear mesh. However, many of the algorithms have been implemented so that they can be extended to three dimensions, or used with unstructured near-body grids. The solver developed in this work can be considered a prototype solver that has been used to assess whether the strand/CAMR method is suitable for hypersonic flow problems, and used to demonstrate the capabilities enabled by the strand/CAMR paradigm.

1.4 Organisation of Thesis

In Chapter 2, a review of the components required in a hypersonic strand/CAMR solver is presented. A description of the thermochemical nonequilibrium encountered in hypersonic flight is given, along with a review of the most commonly used thermochemical models. The different elements of a strand/CAMR solver are then examined in detail.

Chapter 3 details the extensions that have been made to the AMROC framework in this work. The thermochemical nonequilibrium governing equations that have been incorporated within AMROC are discussed. The implementation of the new spatial and temporal integration routines is then described, and results from the verification and performance studies are presented. Chapter 4 details the automated meshing techniques implemented in this research. This includes a description of the novel strand mesh generation algorithm and the overset library developed in this work.

Verification and validation results from the combined strand/CAMR solver are shown in Chapter 5. The results include the validation of blunt body heat flux predictions, simulations of shock-shock and shock-boundary-layer interactions and demonstrations of automated mesh generation around deforming bodies and bodies in relative motion. The novel investigations of unsteady flows are presented in Chapter 6. Finally, Chapter 7 draws together the findings from the research and gives recommendations for future work.

Chapter 2

Literature Review

There are two areas that need to be understood to create an accurate and efficient hypersonic strand/CAMR solver. Firstly, an understanding of the fluid models that are suitable for hypersonic flow simulations is required. Secondly, the different components of a strand/CAMR solver must be understood, including the off-body and near-body solver methods, the different strand meshing techniques and the overset grid assembly algorithms. In this chapter an extensive review of hypersonic fluid models and strand/CAMR solver components is presented.

2.1 Physical Models

A vehicle travelling at hypersonic speeds in Earth's atmosphere creates a strong shock in the stagnation region. Due to the strength of the shock wave and the kinetic energy that is dissipated, the temperature increase across the shock can be thousands of Kelvin. The temperatures in the shock layer will almost certainly be high enough to excite a molecule's vibrational energy modes (800 K), and can exceed the temperature required to dissociate nearly all oxygen molecules and nearly all nitrogen molecules (4000 K and 9000 K, respectively, at 1 atm pressure [9]). The excitation of these internal energy modes and changes to the composition of the gas mean that the assumption of a constant ratio-of-specific-heats, γ , is no longer valid. Therefore, the air can no longer be modelled as a calorically perfect gas, although the ideal gas equation of state, $pv = RT$, can still hold [9]. In addition, the species collisions required to excite the internal energy modes and cause chemical reactions do not occur instantaneously. Consequently, due to the high post-shock velocities, the molecules in the flow may have travelled far from the shock before the changes occur. In such cases, the flow should be modelled as a thermochemical nonequilibrium flow.

To determine whether a flow is in thermodynamic or chemical nonequilibrium, one uses the ratio of the thermodynamic relaxation times and characteristic reaction times to the characteristic flow time. These ratios are known as Damköhler numbers, \mathcal{D} , and if $\mathcal{D} = \mathcal{O}(1)$ then the effects of nonequilibrium should be considered [272].

In a review paper [98] Gnoffo gives guidance on how both the chemical and thermodynamic Damköhler numbers can be approximated. The characteristic flow time is taken as the time for a molecule to transverse the shock layer and can be approximated as

$$\tau_f = D/V_\infty, \quad (2.1)$$

where D is the diameter of the nose of the vehicle and V_∞ is the freestream flow velocity. Using estimated values for the mole fractions behind a shock, one can use the species molar production rate equation to estimate the characteristic time for the production of species i , due to the forward reaction r , $\tau_{cf,r,i}$ [98]. Thus, the Damköhler number for each reaction can be established as

$$\mathcal{D}_{cf,r,i} = \frac{\tau_f}{\tau_{cf,r,i}}. \quad (2.2)$$

Similarly, Damköhler numbers can be obtained for the transfer of energy between the internal modes of a molecule. For example, the Damköhler number for the translational-vibrational energy transfer can be found by using the Millikan and White [192] relationship to estimate the characteristic translational-vibrational relaxation time for the interaction of two species s and r , $\tau_{v,s,r}$. The relationship takes the form $p\tau_{v,s,r} = p\tau_{v,s,r}(T)$, where $\tau_{v,s,r}(T)$ is determined by species specific gas constants and the pressure can be estimated using the freestream density and velocity. This gives an estimate for the Damköhler number for the translational-vibrational relaxation as

$$\mathcal{D}_{v,s,r} = \frac{\tau_f}{\tau_{v,s,r}} = DV_\infty\rho_\infty/\tau_{v,s,r}(T). \quad (2.3)$$

The Damköhler numbers for different flight regimes can be calculated and used to determine whether chemical or thermodynamic nonequilibrium should be modelled. Figure 2.1 shows a velocity-altitude map that is redrawn from a review paper by Tirsky [272]. One can see from the diagram that re-entry vehicles, such as the space shuttle, fall well within the range of thermochemical nonequilibrium. As the aim of this project is to develop a code that is capable of modelling flight regimes that include re-entry into Earth's atmosphere from low Earth orbit, thermochemical nonequilibrium effects must be accounted for.

It has been shown that it is possible to use the master equation and quantum state-to-state calculations to determine the macroscopic properties of a gas in thermochemical nonequilibrium, using very few assumptions [133]. In such methods,

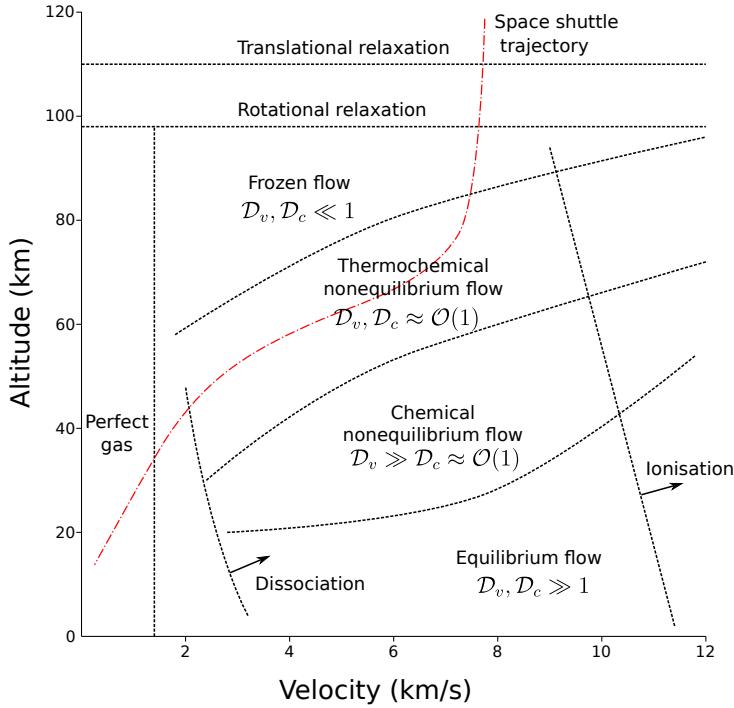


Figure 2.1: A velocity-altitude map which gives guidance on the flight regimes where nonequilibrium effects should be considered (redrawn from Ref. [272]).

each quantum vibrational state of a molecule is accounted for using a separate conservation equation. State-to-state models, such as the Forced Harmonic Oscillator (FHO) model [2, 171] or those based on Quasi-Classical Trajectory (QCT) modelling of molecular dynamics [79], are then used to determine the dissociation rates and population of each vibrational energy level. Using such methods means the assumption that the energy levels are populated in a Boltzmann distribution is not valid, so each state is associated to its own temperature.

Although these models use very few assumptions and have been shown to give accurate results [26, 113, 132, 201], the computational cost of such models is significant [133]. This is because each quantum state requires its own conservation equation and source terms to transfer energy between them. There are examples of such methods being used in full scale CFD simulations [26], which show more accurate prediction of shock stand-off distances due to the ability to account for non-Boltzmann vibrational energy distributions. However, the research is relatively recent, and, at the time of writing, state-to-state models have not reached a level of maturity for routine use in the engineering design process. This said, there are an increasing number of examples where state-to-state results are being used to modify simpler, less computationally expensive models to make them more accurate [49, 141, 152, 201, 208, 251, 252]. In addition, thermochemical libraries are being developed that include state-to-state

models [37], which will make it easier to utilise such methods within existing CFD solvers.

Although direct state-to-state simulations or models based on QCT results may be more widely adopted in the future they are still in their infancy compared to more commonly used models. The majority of practical CFD calculations that have been carried out over the past 30 years use simplified models of the thermochemical nonequilibrium. In this section, some of the most commonly used thermochemical nonequilibrium models are discussed.

2.1.1 Thermodynamic Model

In the development of a thermodynamic nonequilibrium model, the internal energy modes of each atom or molecule are first considered separately. The internal energy of a species can be broken down into four modes, the translational, rotational, vibrational and electronic modes [9]. The sum of the energy contained within each mode, along with the sensible energy, gives the specific internal energy for a given species. For the models discussed below, each species' individual energy mode is assumed to be in local equilibrium, meaning the energy in the translational mode has a Maxwell distribution and the energy in each internal mode has a Boltzmann distribution. This allows the energy in each mode of each species to be described by a single temperature [272]. Using this model the specific internal energy of a species, s , is given by the sum of the energy in each internal mode,

$$e_s = e_s^t(T_s^t) + e_s^r(T_s^r) + e_s^v(T_s^v) + e_s^{el}(T_s^{el}) + e_s^0, \quad (2.4)$$

where e_s^t is the translational energy, e_s^r is the rotational energy, e_s^v is the vibrational energy, e_s^{el} is the electronic energy and e_s^0 is the sensible (or reference) energy.¹

There are several ways that the total internal energy of the mixture can be modelled in a nonequilibrium scheme, and this depends on the fidelity of modelling needed and the Damköhler number of each internal mode. The time taken for different modes to relax to equilibrium varies, where the characteristic relaxation times have the following relationship [74, 133]:

$$\tau_0 \approx \tau^t < \tau^r \ll \tau^v \approx \tau^{el}, \quad (2.5)$$

where τ_0 is the mean collision time.

¹Splitting the energy into independent modes assumes the Born-Oppenheimer approximation [29] is correct. The validity of using this assumption in aerothermodynamics computations has been brought into question by Giordano [93]. State-to-state models have also shown non-Boltzmann distributions of internal energies behind strong shock waves [132, 201], questioning the assumption of local thermodynamic equilibrium in each mode. However, as Casseau *et al.* [44] recently argue, "this approach undoubtedly remains the reference as it has proved to give satisfactory results for numerous re-entry case scenarios".

The varying relaxation times of the energy modes often results in a number of different energy equations being used to describe the energy in the flow. The number of separate energy equations (and thus temperatures) will depend on the level of nonequilibrium that needs to be modelled. The more temperatures used, the greater the computational expense and complexity, as extra equations and source terms are needed. As such, one aims to use the minimum number of temperatures required to accurately model a given flow.

The extremely short relaxation time for the translational modes means that it is normally assumed the translational temperature of all species take on a Maxwell distribution at the same energy level. Therefore, the energy in the translational mode is represented by a single temperature, $T_s^t = T^t$ for all s . This assumption is common to all continuum flow solvers of which the author is aware. The short rotational relaxation time means that the rotational temperature is normally considered to be in equilibrium with the translational temperature ($T_s^r = T^t = T_{tr}$). However, this is not always the case (e.g. Refs. [143, 145, 146]) and Park argues that this assumption could lead to the shock stand-off distance being underestimated [218].

The vibrational relaxation times are greater than the rotational relaxation times, and the vibrational relaxation times of species differ. Simulations that account for these differences by using multiple vibrational energy equations have shown differences in species vibrational temperatures behind normal shocks [61, 246, 269] and in rapidly expanding flows [89, 146, 203]. Therefore, it may be necessary to use a model with multiple vibrational temperatures. This is especially true when modelling high-enthalpy wind tunnels where the rapidly expanding flow causes large differences in species vibrational temperatures [89, 146, 203]. The electronic energy levels of a molecule are generally considered to be in equilibrium with the vibrational energy. This assumption is based on experimental observations of the electronic and vibrational temperatures in mixtures of N_2 and sodium [215].

In weakly ionised flows, the electron temperature can be considered the same as the electronic (and thus, vibrational) temperature or can be considered separately. The assumption that the electron temperature is the same as the vibrational-electronic temperature is justified by the fast energy transfer between the vibrational mode and the electron translational mode, and because the electronic mode is excited by the electron translational mode [241]. However, simulations have shown that the level of vibrational-electron nonequilibrium can be large at high re-entry speeds [144, 145], so separate consideration of the electron temperature is important for accurate results. Accurate assessment of the electron density is important when trying to determine the level of radio blackout, as this is caused by high electron density [144, 145]. In addition, the electron density and temperature can have an influence on the rate of ablation of a TPS. Therefore, a separate electron temperature may be required to

accurately model radio blackout and ablative material response at high re-entry speeds.

The simplest thermodynamic nonequilibrium model considers the translational and rotational modes of all species to be in equilibrium ($T_s^r = T_s = T_{tr}$) and assumes that the vibrational, electronic and electron energies can be described by the same temperature ($T_s^v = T_s^{el} = T_e = T_{ve}$) [214]. This is known as the two-temperature model. This model assumes that the differences between the vibrational relaxation times of each species will not have a significant impact on the results. Gnoffo [97] justifies the two-temperature model by observing that even when the vibrational temperatures are different they are qualitatively similar and are all very different from the translational-rotational temperature profile. This model is one of the most commonly used within the hypersonic CFD community and has been shown to deliver good agreement with experimental results [44, 40, 57, 96, 118, 241, 273, 296].

Shang and Surzhikov [246, 247] compared the results from a model using multiple vibrational temperature equations and a separate electronic temperature to results obtained using two codes based on the two-temperature model (DPLR [296] and LAURA [96]). The results showed that electronic energy conservation and separate consideration of the vibrational energies makes very little difference to the predictions of either the convective-conductive heat transfer to the body or the chemical composition of the flow. The comparison was made for extremely high-speed re-entry flows, where the vibrational-electronic energy nonequilibrium is likely to be highest.

Taking into consideration the flow regime to be modelled (atmospheric flight for speeds up to low Earth orbit re-entry) and the computational cost and uncertainties associated with multi-temperature energy transfer [97, 246], it was decided to use Park's two-temperature model in this work. In the remainder of this section, a review of the models used within the two-temperature model is given.

2.1.1.1 Calculation of Internal Energies

All of the models described so far require the internal energy in each mode to be calculated. The energy in each internal mode is calculated by integrating the mode's specific heat at constant volume over temperature,

$$e_s^m = \int_{T_{ref}}^{T_m} C_{v,s}^m \, d\hat{T}, \quad (2.6)$$

where T_{ref} is a reference temperature.

The translational and rotational modes are fully excited at very low temperatures. Therefore, the specific heat capacities are constant at all temperatures of interest

($150 < T < 20,000$) [9]. Using statistical thermodynamics, the translational and rotational specific heat capacities at constant volume are shown to be

$$C_{v,s}^t = \frac{3}{2} \frac{R_u}{M_s} \quad (2.7)$$

and

$$C_{v,s}^r = \begin{cases} \frac{R_u}{M_s}, & \text{for diatomic molecules,} \\ 0, & \text{for atoms and electrons,} \end{cases} \quad (2.8)$$

where R_u is the universal gas constant and M_s is the molar mass of species s . The energies are therefore given by

$$e_s^t = \int_{T_{\text{ref}}}^{T_t} C_{v,s}^t d\hat{T} = C_{v,s}^t (T_t - T_{\text{ref}}) \quad (2.9)$$

and

$$e_s^r = \int_{T_{\text{ref}}}^{T_r} C_{v,s}^r d\hat{T} = C_{v,s}^r (T_r - T_{\text{ref}}). \quad (2.10)$$

In the two-temperature model, this gives the translational-rotational energy of a species as

$$e_s^{tr} = (C_{v,s}^t + C_{v,s}^r)(T_{tr} - T_{\text{ref}}). \quad (2.11)$$

There are two ways in which the vibrational-electronic energy of a molecule can be calculated at a given temperature. The first is to use thermodynamic curve fits [97] and the second is to employ a Rigid-Rotator Harmonic Oscillator (RRHO) model [9].

Vibrational-Electronic Energy Using Thermodynamic Curve Fits

Over many years NASA has produced tables of coefficients that enable one to obtain the thermodynamic properties of individual species as a function of temperature, at a standard pressure of 1 atm [109, 183, 184]. The most recent set of curve fits [183] give coefficients for over 2000 species, for a temperature range of 200 K to 20,000 K. In these databases, the reference temperature is taken to be 298.15 K [183].

The database gives curve fits for $C_p(T)/R_u$, $h(T)/R_u$ and $S(T)/R_u$ for each species. Thus, one needs to separate the vibrational-electronic energy from the other modes in a species. This is achieved by taking advantage of the constant specific heat capacities for the rotational and translational modes to find the specific heat for the combined vibrational-electronic modes as

$$C_{v,s}^{ve} = C_{v,s} - (C_{v,s}^t + C_{v,s}^r), \quad (2.12)$$

where $C_{v,s}$ is given by the relationship

$$C_{v,s} = C_{p,s} - \frac{R_u}{M_s}. \quad (2.13)$$

Using Eq. (2.6), the vibrational-electronic energy is given as

$$e_s^v(T_{ve}) + e_s^{el}(T_{ve}) = e_s^{ve}(T_{ve}) = \int_{T_{\text{ref}}}^{T_{ve}} C_{v,s}^{ve}(\hat{T}) d\hat{T}. \quad (2.14)$$

Substituting Eqs. (2.12) and (2.13) into (2.14) gives

$$e_s^{ve} = \int_{T_{\text{ref}}}^{T_{ve}} C_{p,s}(\hat{T}) - \left(\frac{R_u}{M_s} + C_{v,s}^t + C_{v,s}^r \right) d\hat{T}. \quad (2.15)$$

Using the definition of enthalpy,

$$h_s(T) = \int_{T_{\text{ref}}}^T C_{p,s}(\hat{T}) d\hat{T} + h_s^0, \quad (2.16)$$

where h_s^0 is the enthalpy of formation and is given in the database, the vibrational-electronic energy is given by

$$e_s^{ve}(T_{ve}) = h_s(T_{ve}) - h_s^0 - \int_{T_{\text{ref}}}^{T_{ve}} \left(\frac{R_u}{M_s} + C_{v,s}^t + C_{v,s}^r \right) d\hat{T}, \quad (2.17)$$

$$= h_s(T_{ve}) - \left(\frac{R_u}{M_s} + C_{v,s}^t + C_{v,s}^r \right) (T_{ve} - T_{\text{ref}}) - h_s^0. \quad (2.18)$$

Using Eq. (2.4) the specific internal energy of a species is then calculated as

$$e_s(T_{tr}, T_{ve}) = \int_{T_{\text{ref}}}^{T_{tr}} C_{v,s}^t + C_{v,s}^r d\hat{T} + e_s^{ve}(T_{ve,s}) + e_s^0, \quad (2.19)$$

$$= (C_{v,s}^t + C_{v,s}^r) (T_{tr} - T_{\text{ref}}) + e_s^{ve}(T_{ve,s}) + e_s^0, \quad (2.20)$$

where e_s^0 can be found from the enthalpy of formation,

$$e_s^0 = h_s^0 - \frac{R_u}{M_s} T_{\text{ref}}. \quad (2.21)$$

Vibrational-Electronic Energy using the RRHO Model

In this method, analytic functions are used to calculate the energy contained in the electronic and vibrational temperature modes. These functions are derived using a combination of classical thermodynamics and quantum mechanics (see Ref. [9] for details).

The specific vibrational energy is given by

$$e_s^v(T_{v,s}) = \begin{cases} \frac{R_u}{M_s} \frac{\theta_{v,s}}{\exp(\theta_{v,s}/T_{v,s}) - 1}, & \text{for diatomic molecules,} \\ 0, & \text{for atoms,} \end{cases} \quad (2.22)$$

where $\theta_{v,s}$ is the characteristic vibrational temperature of the species.

The specific electronic energy for all species is

$$e_s^{el}(T_{el,s}) = \frac{R_u}{M_s} \frac{\sum_{i=1}^{\infty} g_{i,s} \theta_{el,i,s} \exp(-\theta_{el,i,s}/T_{el,s})}{\sum_{i=1}^{\infty} g_{i,s} \exp(-\theta_{el,i,s}/T_{el,s})}, \quad (2.23)$$

where $g_{i,s}$ is the degeneracy at energy level i for species s and $\theta_{el,i,s}$ is the characteristic electronic temperature at energy level i for species s . Although the sum is from $i = 1$ to $i = \infty$, only the lower electronic energy levels are usually considered in the hypersonic regime. This is because the contribution of the higher energy levels is negligible at the temperatures of interest due to the large characteristic electronic temperatures for the higher levels [241].

Using Eq. (2.4) and the two-temperature model, the internal energy of a heavy species is given by,

$$e_s(T_{tr}, T_{ve}) = (C_{v,s}^t + C_{v,s}^r) (T_{tr} - T_{ref}) + e_s^v(T_{ve}) - e_s^v(T_{ref}) + e_s^{el}(T_{ve}) - e_s^{el}(T_{ref}) + e_s^0, \quad (2.24)$$

and for an electron is

$$e_e(T_{ve}) = C_{v,s}^t (T_{ve} - T_{ref}) + e_s^0, \quad (2.25)$$

where e_s^0 is the same as in Eq. (2.21). Often when using the RRHO model the reference temperature is taken to be zero [41, 103, 241]. This gives the species energies as

$$e_s(T_{tr}, T_{ve}) = (C_{v,s}^t + C_{v,s}^r) T_{tr} + e_s^v(T_{ve}) + e_s^{el}(T_{ve}) + h_s^0, \quad (2.26)$$

and electron internal energy as

$$e_e(T_{ve}) = C_{v,s}^t T_{ve} + h_e^0. \quad (2.27)$$

Comparison of Vibrational-Electronic Energies

The thermodynamic curve fits are constructed using an extended version of the RRHO model that includes corrections for various effects, such as anharmonic oscillations, rotational stretching and vibration-rotation interactions [183]. Hence, the data in the curve fits is thought to be more accurate than using the RRHO model alone [98, 245]. Figures 2.2 and 2.3 show the vibrational-electronic energy for atomic and molecular oxygen and nitrogen calculated using the two different methods.

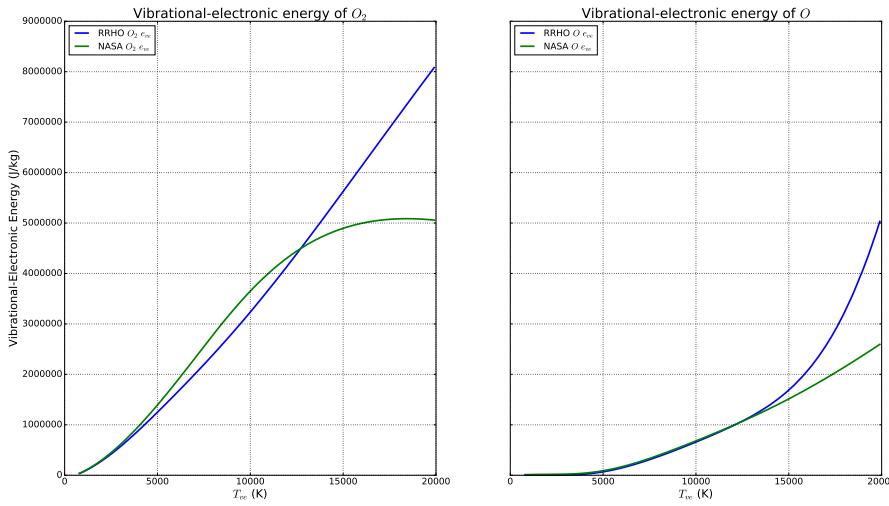


Figure 2.2: The vibrational-electronic energy of O_2 and O between 800 K and 20,000 K.

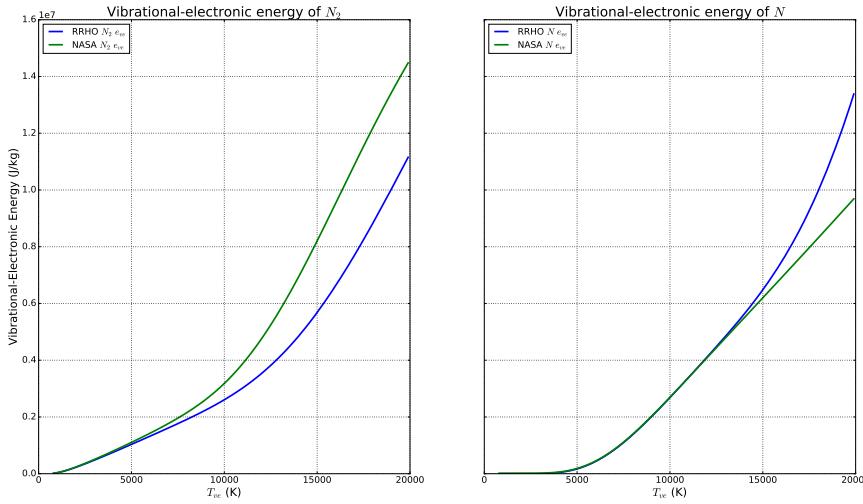


Figure 2.3: The vibrational-electronic energy of N_2 and N between 800 K and 20,000 K.

One can see that at low temperatures there is very little difference between the two methods. However, as the molecules reach temperatures that are close to those

required for all of the molecules to become dissociated one can see that differences begin to appear. As the temperature increases further the differences become larger. Similarly, for the monatomic case, where there is no vibrational energy present, the results begin to diverge as the temperature approaches that required to ionise most of the atoms, and continue to diverge above this temperature.

As the majority of the differences are seen above the temperature where the species will either dissociate or ionise, any difference in the computational results is likely to be small, as the molecules will quickly dissociate at this temperature. Therefore, both models are likely to yield similar computational results. The advantage of the curve fitting method is that it is more accurate and possibly more computationally efficient as it only requires simple table look-up functions. A disadvantage of using the curve fitting method is that the vibrational and electronic energies cannot be considered separately.

2.1.2 Energy Transfer Between Modes

In models based on the partition of the internal energies of a molecule, the energy exchange between these modes must be accounted for.

2.1.2.1 Translational-Vibrational Energy Exchange

The translational-vibrational exchange is generally modelled using the Landau-Teller [155] equation. This method calculates the energy exchange based upon a characteristic translational-vibrational relaxation time, $\tau_{v,s}^{T-V}$, and the difference in vibrational energy of a species at T_{tr} and T_{ve} ,

$$Q_s^{T-V} = \rho_s \frac{de_s^v}{dt} = \rho_s \frac{e_s^v(T_{tr}) - e_s^v(T_{ve})}{\tau_{v,s}^{T-V}}. \quad (2.28)$$

It should be noted that when the curve fitting method is used to calculate the thermodynamic properties, the vibrational and electronic modes cannot be separated. As such the equation becomes,

$$Q_s^{T-V} = \rho_s \frac{de_s^v}{dt} = \rho_s \frac{e_s^{ve}(T_{tr}) - e_s^{ve}(T_{ve})}{\tau_{v,s}^{T-V}}. \quad (2.29)$$

When using the RRHO model, one can use the original equation (Eq. (2.28)).

The translational-vibrational relaxation time for a molecule is calculated as a molar average of the relaxation time due interactions with all of the species present in the

flow (including when $s = r$),

$$\tau_{v,s}^{T-V} = \frac{\sum_{r=1}^N X_r}{\sum_{r=1}^N \frac{X_r}{\tau_{s-r}^{T-V}}}, \quad (2.30)$$

where X_r is the species concentration of species r .

The characteristic relaxation time can be calculated in a number of ways [103]. The first method is based on the theory of Landau and Teller [155], where the relaxation time is given by

$$\tau_{s-r}^{T-V} = \frac{A}{p} \exp \left[\frac{B}{T^{1/3}} + C \right], \quad (2.31)$$

where the constants A , B , and C are found using experimental data.

The second method is based on an empirical relationship developed by Millikan and White [192], which was established by analysing experimental data of vibrational relaxation rates,

$$p \tau_{s,r}^{T-V} = \exp \left[A_{s,r} \left(T_{tr}^{-1/3} - B_{s,r} \right) - 18.42 \right] \text{ (in atm-sec)}, \quad (2.32)$$

where $A_{s,r}$ and $B_{s,r}$ are constants that depend on the two interacting species, s and r . These values were originally calculated using the species' molar mass and characteristic vibrational temperatures as

$$A_{s,r} = 1.16 \times 10^{-3} \mu_{s,r}^{1/2} \theta_{v,s}^{4/3}, \quad (2.33)$$

$$B_{s,r} = 0.015 \mu_{s,r}^{1/4}, \quad (2.34)$$

where

$$\mu_{s,r} = \frac{M_s M_r}{M_s + M_r}. \quad (2.35)$$

However, it was found that although these values work well to describe the energy exchange between two molecular species, they are less accurate when describing a collision between a molecule and an open shell atom [201, 270]. Park [216] proposed new constants to account for this effect, which have been shown to give good agreement with experimental results [117].

The Millikan and White approach has been shown to under-predict relaxation times at higher temperatures [74, 97]. In order to correct for this problem, Park proposed a correction factor [213] based on the species' average molecular speed,

$$\bar{c}_s = \sqrt{\frac{8R_u T_{tr}}{M_s \pi}}, \quad (2.36)$$

limited collision cross-section,

$$\sigma_s = C \left(\frac{50000}{T_{tr}} \right)^2 \quad (2.37)$$

and number density, n_s . The constant C in Eq. (2.37) was given as 10^{-20} in the original work, but has since been given as 3×10^{-21} for O_2 and N_2 by other researchers [43].

Several variations of the correction factor can be found in the literature. Some methods add the correction factor for each species once [57, 193, 241, 273, 284],

$$\tau_{v,s}^{T-V} = \frac{\sum_{r=1}^N X_r}{\sum_{r=1}^N \frac{X_r}{\tau_{s-r}^{T-V}}} + \tau_s^p, \quad (2.38)$$

where the Park correction factor is given as

$$\tau_s^p = \frac{1}{\bar{c}_s \sigma_s n_s}. \quad (2.39)$$

In other methods [44, 61, 74, 191], the correction is applied for each pair of colliding species,

$$\tau_{v,s}^{T-V} = \frac{\sum_{r=1}^N X_r}{\sum_{r=1}^N \frac{X_r}{\tau_{s-r}^{T-V} + \tau_{s-r}^p}} \quad (2.40)$$

In this case, the correction factor is given by either [44, 191]

$$\tau_{s-r}^p = \frac{1}{\bar{c}_s \sigma_s n_{s,r}}, \quad (2.41)$$

where $n_{s,r}$ is the number density of the two species,

$$n_{s,r} = \begin{cases} n_s & \text{if } s = r, \\ n_s + n_r & \text{otherwise,} \end{cases} \quad (2.42)$$

or, a correction based on the collision of hard spheres [74] in the form

$$\tau_{s-r}^p = \sqrt{\frac{\pi k_B \mu_{s,r} T_{tr}}{8 N_A}} \frac{1}{\sigma_s p}, \quad (2.43)$$

where k_B is the Boltzmann constant, N_A is the Avagadro constant.

The Millikan and White model, with the addition of a Park relaxation time, appears to be the most commonly used approach in CFD software modelling hypersonic flow [41, 44, 57, 96, 117, 118, 193, 216, 241, 247, 273, 296]. It should be noted that the

Millikan and White data is correlated for vibrational relaxation that occurs after a shock wave, and experimental data has shown that vibrational relaxation happens much more rapidly in expanding flows. As a result, the relaxation time is often divided by a constant in expanding flows to account for this effect [219].

Another model used for calculating the relaxation time is the Schwartz, Slawsky and Herzfeld (SSH) model [244]. However, it is more complex and does not appear to give significantly improved relaxation times when compared to the modified Millikan and White method [74, 103]. In addition, it is only theoretically valid for interactions between two molecules.

Finally, recent research using quantum state-to-state methods [10, 152] has lead to new temperature and pressure dependent relationships for the vibrational relaxation of O_2 and N_2 . These have been applied in simulations of shock waves using a modified Landau-Teller type equation [153, 201, 208]. These new models show some improvements over traditional models, although the improvements are limited as they still utilise a single vibrational temperature and a Landau-Teller type equation [201].

2.1.2.2 Heavy Particle-Electron Energy Exchange

The traditional model for the energy exchange between heavy particles and free electrons was originally developed by Lee [157], and has been used in a number of CFD codes [41, 96, 241, 247, 269, 296]. The energy exchange is modelled using the formula

$$Q^{T-e} = 3R_u \rho_e (T - T_{ve}) \sqrt{\frac{8R_u T_{ve}}{\pi M_e}} \sum_{s \neq e} \frac{\rho_s N_A}{M_s^2} \sigma_{er}, \quad (2.44)$$

where N_A is the Avagadro number and σ_{er} is the effective collision cross section between an electron and heavy particle. This is taken to be

$$\sigma_{er} = 10^{-20} \text{ m}^2 \quad (2.45)$$

for neutral-electron collisions and

$$\sigma_{er} = \frac{8\pi}{27} \frac{e^4}{k^2 T_{ve}^2} \ln \left[1 + \frac{9k^3 T_{ve}^3}{4\pi n_e e^6} \right] \quad (2.46)$$

for electron-ion collisions [241].

The review paper [247] gives two more recent formulas for the transfer between free electrons and heavy particles due to collisions. These are based on electron energy transition models and are given as

$$Q^{T_i-e} = 1.21 \times 10^{20} X_e X_i \frac{T_{tr} - T_{ve}}{T_{ve}^{3/2}} \ln \Lambda \quad (2.47)$$

for ions and

$$Q^{T_n-e} = 3.378 \times 10^{10} X_e X_n \sqrt{T_{ve}} (T_{tr} - T_{ve}) \left[1 - \left(1 + \frac{T_{ve}}{T_{tr}} \right)^{-1} \right] \quad (2.48)$$

for neutral species.

2.1.3 Chemistry Model

The Damköhler number is of order one for many reactions in air at hypersonic flow speeds. Hence, a finite rate reaction model is needed, with separate conservation equations for each species and source terms to account for net species production rates. A derivation of the species net molar production rate equation for a single reaction can be found in Ref. [9]. The production rate of species s due to reaction r is given as

$$\dot{n}_{sr} = (\beta_{sr} - \alpha_{sr}) \left[k_{f,r} \prod_{i=1}^{N_s} \left(\frac{\rho_i}{M_i} \right)^{\alpha_{ir}} - k_{b,r} \prod_{i=1}^{N_s} \left(\frac{\rho_i}{M_i} \right)^{\beta_{ir}} \right] \quad (2.49)$$

where β and α are the stoichiometric coefficients², $k_{f,r}$ is the forward reaction rate, $k_{b,r}$ is the backward reaction rate and N_s is the number of species. It is important to note that the species concentrations and reaction rate constants should be in the same physical units. Reaction rate data used to determine the forward and backward reaction rates is often in units of cm and grams, whereas density is often in kg/m³, so conversion factors may need to be used.

The total production of a species in units of mass is then given by summing the production rate from each reaction and multiplying the result by the molar mass of the species,

$$\dot{w}_s = M_s \sum_{r=1}^{N_r} \dot{n}_{sr}, \quad (2.50)$$

where N_r is the number of reactions involving species s .

The forward reaction rate is found using the Arrhenius equation, which is a function of the forward rate controlling temperature $T_{c,f}$, and is given by

$$k_{f,r}(T_{c,f}) = A_r T_c^{\eta_{f,r}} \exp [-\theta_r / T_{c,f}], \quad (2.51)$$

where A_r is the reaction rate constant, θ_r is the activation temperature and $\eta_{f,r}$ is a constant. The backwards reaction rate is normally calculated by its relationship to the

²The stoichiometric coefficient is the number appearing before the chemical compound in the reaction equation.

equilibrium constant, $K_{eq,r}$ and forward reaction rate,

$$k_{b,r} = \frac{k_{f,r}}{K_{eq,r}}. \quad (2.52)$$

When calculating the backwards reaction rate, the forward reaction rate and equilibrium constant are both evaluated using the backward rate controlling temperature, $T_{c,b}$. This depends on the reaction and is not necessarily the same as the forward rate controlling temperature [57, 241].

The equilibrium constant, $K_{eq,r}$, can either be found analytically using Gibb's free energy minimisation, or using temperature dependent curve fits developed from experimental data. Gibb's free energy minimisation is the most accurate method, but curve fits can reduce the computational cost. Tchuen [269] compares several of the curve fits that have been developed for air, namely those of Park [216], Gardiner [91], Moss [250], Dunn and Kang [75] and one developed by Tchuen based on the Dunn and Kang model and the Gupta equilibrium curve fits [109]. The work reveals that the calculation of the shock stand-off distance and cold wall heat flux for a blunt body is highly sensitive to the equilibrium model used, with differences of up to 60% being observed for the heat flux. After comparisons with experimental data, Tchuen recommends using the Park reaction rate coefficients [216] for predicting the hypersonic flow around blunt bodies [269].

2.1.4 Thermodynamic-Chemical Coupling

2.1.4.1 Reaction Rates

The forward reaction rate for dissociation reactions should be based on both the vibrational energy and the translational-rotational energy, due to the physical processes that cause dissociation reactions to occur. The four main models that are used to account for the coupling between the thermodynamic nonequilibrium and dissociation reactions are that of Marrone and Treanor [177], Knab *et al.* [148], Macheret and Rich [172], and Park [214].

The most common model found in the literature is Park's two-temperature model [214]. This model uses a combination of the two-temperatures to establish the rate controlling temperature for dissociation reactions,

$$T_{c,f} = T_{tr}^q T_{ve}^{(1-q)}, \quad (2.53)$$

where q is typically between 0.7 and 0.5. The Park model is likely to be the most common as it is simple to implement, computationally efficient and the required

constants are available in the literature [215, 216, 217]. However, it has limited theoretical underpinning.

Marrone and Treanor [177] established a model in 1963 in which the forward reaction rate of a dissociation reaction is calculated using the translational-rotational temperature and is then multiplied by a correction factor, that is a function of both the translational-rotational temperature and the vibrational temperature,

$$k_{f,r} = Z(T_{tr}, T_v) k_{f,r}(T_{tr}). \quad (2.54)$$

The correction factor is determined using the vibrational partition function, $Q_{v,i}(T)$, at various temperatures. This accounts for the effect of preferential dissociation of molecules in a higher vibrational state:

$$Z(T_{tr}, T_v) = \frac{Q_{v,s}(T_{tr})Q_{v,s}(T_F)}{Q_{v,s}(T_v)Q_{v,s}(-U)}. \quad (2.55)$$

The simplified vibrational partition function for a truncated harmonic oscillator is used and is given by

$$Q_{v,s}(T) = \frac{1 - \exp\left(\frac{-T_D}{T}\right)}{1 - \exp\left(\frac{-\theta_{v,s}}{T}\right)}, \quad (2.56)$$

where T_D is the characteristic dissociation temperature, defined as $T_D = D/k_B$, where D is the dissociation energy and k_B is the Boltzmann constant. T_F is a pseudo-temperature which is defined as

$$\frac{1}{T_F} = \left(\frac{1}{T_v} - \frac{1}{T_{tr}} - \frac{1}{U} \right), \quad (2.57)$$

where U is a modelling parameter called the characteristic probability temperature which controls the amount of preferential dissociation from higher vibrational energy levels (an infinite value of U results in equal dissociation probability from all levels [61, 103]). The value of U is usually [61]

$$\frac{D}{6k_B} \leq U \leq \frac{D}{3k_B}. \quad (2.58)$$

Knab *et al.* [148] extended the model of Marrone and Treanor to include exchange reactions, associative ionisation and electron impact ionisation reactions by modifying the form of the correction factor. More recently, Chaudhry and Candler have fitted the parameters within the Marrone and Treanor model to match results given by detailed quantum state-to-state models [49]. In this model, the characteristic probability temperature is given by

$$\frac{1}{U} = \frac{1}{T_{tr}} + \frac{1}{U^*}. \quad (2.59)$$

Results from QCT analysis of O_2 and N_2 dissociation due to interactions with atomic and molecular nitrogen and oxygen were used to fit the free parameters in the model and are tabulated in Ref. [49].

The model of Macheret and Rich [172] also uses a correction factor for dissociation reactions. However, the correction factor is determined based on the FHO model and, unlike the Treanor and Marrone model, has no user input parameters.

A comparison of the models of Park, Marrone and Treanor, and Macheret with state-to-state simulations showed that the Marrone and Treanor derived models were the most accurate when predicting dissociation rates of oxygen [113]. Comparisons have been made on the effect of the models on the heat flux and shock standoff distances. Some comparisons have shown limited differences [61, 209, 247]. However, the modified Marrone and Treanor model developed by Chaudhry and Candler showed differences in both the heat flux and the species mass fractions when compared with Park's. At some densities there was a normalised heat flux difference of approximately 25% for a re-entry body travelling at 5 km/s in air when a non-catalytic wall boundary condition was used [49]. The authors attributed this to the Park model predicting higher levels of dissociation. Recently, different reaction rates and models, including the modified Marrone and Treanor model of Chaudhry and Candler, were used to predict the electron density in the RAM-C II trajectory [240]. Little difference between the models was observed, but it was noted that the use of the two-temperature model may be the limiting factor when predicting highly ionised flows.

2.1.4.2 Energy Conservation

When a molecule dissociates, this is accompanied by a loss of vibrational energy that needs to be accounted for in the vibrational energy equation. The most common model used (for example Refs. [41, 57, 97, 96, 146, 241, 247, 296]) is an empirical model, where the net production rate of a molecule is multiplied by the average vibrational energy of the molecule and a constant,

$$Q_s^{C-V} = c_1 \dot{w}_s e_s^v. \quad (2.60)$$

In a preferential model it is assumed the molecules in the higher energy states are more likely to dissociate, and those that recombine will initially be in a higher vibrational state. In this model, the value of c_1 is greater than one. In a non-preferential model c_1 is equal to one.

Other models [97, 241] use the dissociation energy of the molecule, D_s , giving

$$Q_s^{C-V} = c_2 \dot{w}_s D_s, \quad (2.61)$$

where $c_2 \approx 0.3$.

A similar function to the above is also used to account for the electronic energy change due to reactions,

$$Q_s^{C-el} = c_1 \dot{w}_s e_s^{el}. \quad (2.62)$$

It should be noted that both of the above models are empirical and merely attempt to approximate the loss of energy in a manner that is computationally efficient.

The model of Marrone and Treanor [177] calculates the energy change due to dissociation and recombination as [61]

$$Q_s^{C-V} = e_{p,s}^v \dot{\omega}_s^f + e_{d,s}^v \dot{\omega}_s^b, \quad (2.63)$$

where $\dot{\omega}_s^f$ and $\dot{\omega}_s^b$ are the production and destruction rates of molecule s . The production and destruction energies are given by

$$e_{p,s}^v = \frac{R_u}{M_s} \frac{\theta_{v,s}}{\exp(-\theta_{v,s}/U) - 1} - \frac{R_u}{M_s} \frac{T_{D,s}}{\exp(-T_{D,s}/U) - 1} \quad (2.64)$$

and

$$e_{d,s}^v = \frac{R_u}{M_s} \frac{\theta_{v,s}}{\exp(\theta_{v,s}/T_F) - 1} - \frac{R_u}{M_s} \frac{T_{D,s}}{\exp(T_{D,s}/T_F) - 1}, \quad (2.65)$$

where T_F and U are the same as in Eqs. (2.57) and (2.58). The use of the characteristic probability temperature U allows for preferential dissociation to be incorporated into the energy conservation. Again, Knab extended the model to account for exchange and associative ionisation reactions [61, 148].

In addition to vibrational energy loss, ionisation reactions also change the level of energy in the vibrational-electronic mode. Firstly, the energy carried by free electrons in ionisation reactions involving neutral species are normally accounted for using

$$Q^{C-e} = \dot{w}_e e_e. \quad (2.66)$$

Secondly, the energy lost from the electronic state due to an ionisation impact involving a free electron must be considered [57, 97, 96, 144, 241, 296]. For most models of air this only involves the reactions



and



The energy loss is calculated using the net production rate of free electrons due to these reactions $\dot{w}_{e,s}$ and the first ionisation energy per unit mass of the species \hat{I}_s , i.e.

$$Q^I = \sum_s \dot{w}_{e,s} \hat{I}_s \text{ for } s = N^+, O^+. \quad (2.69)$$

2.1.5 Transport Properties

The viscosity, thermal conductivity and diffusion coefficients of a gas are required to accurately determine the conditions that a TPS must withstand. The convective heat fluxes, diffusive heat fluxes and shear stresses are calculated using the thermal conductivity, diffusion coefficient and viscosity values directly, so their accurate assessment is key to obtaining the correct results. For an ideal gas the transport properties may follow simple relationships, such as Sutherland's law for viscosity [265] and the constant Prandtl number relationship for conductivity. For equilibrium mixtures, the transport properties have been tabulated as a function of temperature and pressure (for example Ref. [108]), and curve fits of this data can be used to obtain the transport properties. However, for thermochemical nonequilibrium flows the properties must be computed as a function of the species mass fractions and the thermodynamic conditions. A brief review of the most commonly used models is given here. The full equations are omitted for brevity, but they can be found within the given references.

The accurate evaluation of the transport properties of a thermochemical nonequilibrium mixture requires collision cross-sections to be calculated for all pairs of colliding species [98]. The calculation of these collision integrals is computationally expensive, so they are often taken from the published translational temperature dependent curve fits found in the literature (for example Refs. [42, 109, 175, 295]). Once the collision integrals are known the transport properties can be calculated. Transport properties for mixtures are obtained using the Chapman-Enskog procedure [48] to obtain a solution of the Boltzmann equation [109, 176, 211, 245]. However, it involves inverting a linear system consisting of N_s equations, which is computationally expensive. Various approximations have been made to compute the mixture properties from the individual species transport properties.

A mixing rule introduced by Wilke [287] uses simplifying assumptions to obtain a solution to the first-order Chapman-Enskog relation. This method gives the mixture translational thermal conductivity and viscosity as a weighted sum of the individual species properties. The properties of a single species are found using simple curve fits, such as those given by Blottner *et al.* [24], or calculated as functions of the translational temperature, collision cross-sections and molar mass [98, 109] using equations from kinetic theory. However, the assumptions used by Wilke are not entirely valid for weakly ionised mixtures [211]. The Armaly-Sutton mixing rule [11] aimed to address

this issue by reducing the number of assumptions. The Armaly-Sutton mixing rule can be written in a form similar to the Wilke mixing rule, however, the scale factor is modified due to the differing assumptions used in the two methods [211]. Another commonly used mixture model is the Gupta-Yos model [109]. This uses similar assumptions to the Wilke mixing rule but retains the mixture collision integrals in the formulation, leading to more accurate results [109]. More recently, computationally efficient methods for solving the first-order Chapman-Enskog equation have been employed that lead to the accurate calculation of the translational thermal conductivity and viscosity of weakly ionised mixtures [94, 176].

To assess the thermal conductivity of internal modes, the Eucken approximation [80] is the most commonly used [109, 176, 211], often with the modification suggested by Hirschfelder [123]. In this method, the mixture conductivity of each mode is calculated using the species concentrations, the mode's specific heat capacity for each species and the binary diffusion coefficients for all pairs of species.

In addition to the calculation of the mixture viscosity and conductivities, the diffusive flux of each species needs to be evaluated. The most physically accurate method of finding the diffusive fluxes is to solve the Stefan-Maxwell equation [7]. However, this is computationally expensive as it requires the evaluation of multi-component diffusion coefficients and is solved iteratively [7, 98, 237]. Simplified models have been created where an effective binary diffusion coefficient for each species is calculated using the reciprocal molar averages of the binary diffusion coefficients. Fick's law can then be applied to calculate the mass flux as a function of the species mass fraction gradient. As Fick's law does not conserve mass, a modified version of Fick's law was proposed that ensures mass conservation [266]. Finally, the diffusion coefficient for electrons must be linked to the diffusion of ions in order to ensure charge neutrality. Therefore, an ambipolar diffusion model is often used where the diffusion coefficient for electrons is calculated using the diffusion velocity of the ions [97].

Comparisons of the various models can be found in the literature. Palmer and Wright [211] compared the Wilke, Armaly-Sutton and Gupta-Yos mixing rules against the exact solutions of the Chapman-Enskog equations. The authors found that the Armaly-Sutton mixing rule was the most accurate over the temperature range of interest, with the Gupta-Yos method being accurate until significant ionisation occurred. The Wilke mixing rule was found to be the least accurate of the three and only applicable at relatively low temperatures. When the Wilke and Gupta-Yos methods were applied to simulations of a blunt body travelling at hypersonic speeds by Alkandry *et al.*, the models gave differences of up to 60% from each other [7]. The difference was most marked in higher temperature post-shock conditions.

Analysis of the effect of the different diffusion models on heat transfer have also been conducted [7, 69, 266]. They show that heat flux results using the modified version of

Fick's law, which ensures mass conservation, agree well with exact results (i.e. those using the Stefan-Maxwell model) for a range of conditions. As such, it is a computationally efficient method of calculating the diffusive fluxes in a multi-component mixture [7, 98].

2.2 Strand/Cartesian Adaptive Mesh Refinement Solver Review

The strand/Cartesian Adaptive Mesh Refinement (CAMR) paradigm enables high levels of automation when generating computational grids around complex bodies. A strand/CAMR tool-set consists of a number of components: a CAMR solver, a strand mesh generator, a strand mesh solver, and overset grid assembly software. Each of these components is reviewed in this section.

2.2.1 Cartesian Adaptive Mesh Refinement Solver

When using an AMR solver as a background solver in overset simulations, most authors have used a h -refinement based Cartesian Adaptive Mesh Refinement solver [16, 120, 186, 190]. CAMR solvers have a number of properties that make them suitable for highly automated overset simulations:

- On a Cartesian mesh, various grid quality metrics such as skewness and smoothness are optimal. As a result, the use of Cartesian meshes for finite-volume methods has been shown to increase the accuracy of the difference schemes [5].
- The spatial integration routines on a Cartesian mesh are often more efficient than on unstructured meshes as they do not require additional numerics for unaligned cell faces.
- Cartesian meshes naturally fit into a structured computational framework. This enables efficient data access, as neighbouring elements can be grouped in memory, and means that the connectivity does not need to be stored and queried.
- The combination of structured cell storage and the alignment of the cells in space simplifies the implementation of high-order-accurate spatial reconstruction schemes for finite-volume methods. For example, one-dimensional Monotonic Upstream-centered Scheme for Conservation Laws (MUSCL) [279] or Weighted Essentially Non-Oscillatory (WENO) [164] methods can be used on a dimension-by-dimension basis [34, 305].
- Cartesian grids can be generated trivially, only requiring the user to input the dimensions of the Cartesian domain and the number of cells (or cell length) in each dimension.

- When combined with AMR, Cartesian grids can simplify the evaluation of some types of refinement criteria, such as error estimates through Richardson extrapolation, as the grid refinement is uniform in each spatial dimension [159].
- In overset simulations the solid bodies must be removed from the background mesh. Cartesian grids enable efficient hole-cutting methods such as Cartesian hole-maps [188] to be used directly on the cells, rather than auxiliary Cartesian grids.
- The point search and interpolation algorithms required for overset simulations are efficient and simple to implement on structured Cartesian grids.

CAMR methods were first used for hyperbolic flow problems by Berger and Oliger [22] and have since been used in simulations of a wide range of phenomena including incompressible flow simulations [181], compressible ideal gas [21] and reacting gas [62, 120] simulations, magnetohydrodynamics simulations [104, 168] and flow simulations using lattice Boltzmann methods [67, 274]. Regardless of the problem being studied, a CAMR solver requires a method for determining which areas should be refined and a data structure to store the refinement and connectivity information.

Refinement criteria are used to determine the parts of the domain that are likely to contain high truncation errors and should therefore be refined. The choice of refinement criteria and the tolerances must be carefully considered to ensure that an optimal amount of the domain is refined. A wide range of different methods have been implemented within finite-volume AMR solvers including Richardson-type error estimates [22], the Operator Recovery Error Source Detector (ORESD) method of Lapenta [156], first and second order flow field gradient methods [62, 88, 120], multi-resolution methods [116], and adjoint based methods [202]. Most error estimation methods rely on user input criteria, which can require tuning for the specific simulation. To reduce the amount of tuning required, non-dimensional values can be used [134], and the criteria can be de-coupled from the grid resolution (e.g. in scaled gradient methods [62]). Comparisons of different refinement criteria have been carried out, with Li [159] finding that a Richardson-type method and the ORESD method were better able to capture errors and were less sensitive to user input than a second-order gradient method. Kamkar *et al.* [134] demonstrated that non-dimensionalised parameters were able to better refine wake vortex structures with less tuning than dimensional parameters. More recently, Deiterding *et al.* [65] found that multi-resolution methods performed better than scaled first-order gradient methods, giving similar global errors for a lower cell count. The investigation of robust refinement criteria that are able to effectively reduce errors, whilst not being overly sensitive to user input, is an area of on-going research.

As new cells are added in refinement regions the CAMR solver must keep track of the dynamic mesh topology. Different data structures have been used, with most falling

into the categories of unstructured, tree-based or block-structured. Unstructured methods generally refine individual cells and store the cell locations and connectivity explicitly. Examples of unstructured CAMR solvers include Cart3D [4] and FARCOM [15]. However, the unstructured nature of the mesh negates some of the advantages of using a Cartesian mesh, such as implicit connectivity and efficient data structures. Tree- and block-based methods retain these advantages to a greater degree and appear to be more widely used in the literature.

Tree-based Adaptive Mesh Refinement (TAMR) methods, introduced by Powell and co-workers [224, 70], refine cells (or fixed $N \times N$ blocks of cells) and store the refinement and connectivity using binary-tree, quad-tree or oct-tree structures, in one, two and three dimensions, respectively. Each time a cell or block is refined a new level or node in the tree structure is created (see Fig. 2.4). The tree structure contains all of the connectivity information that is required by the solver [70].

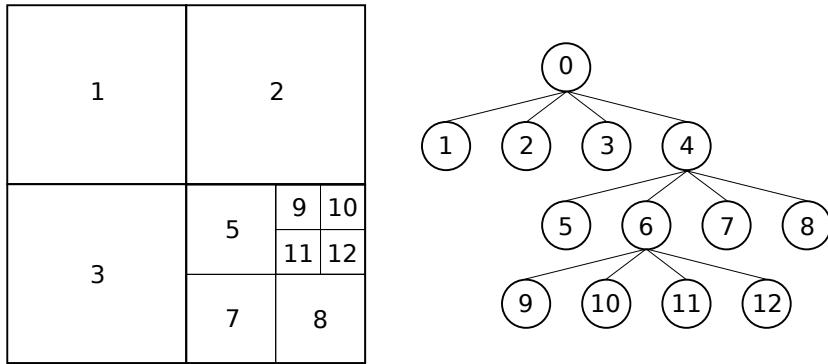


Figure 2.4: A TAMR quad-tree structure for a two-dimensional mesh.

In the SAMR technique, introduced by Berger and Colella [21], cells that are marked for refinement are formed into rectangular patches which are then overlaid with a structured patch of finer cells [62]. This process occurs recursively on each layer to produce a hierarchy of refined grids, as shown in Fig. 2.5. The use of structured patches means that all of the advantages of a single-level structured Cartesian mesh are retained within a given patch. SAMR methods can use tree-based structures to store the refinement patch hierarchy and global indices for the cells so that connectivity information does not need to be explicitly stored [64]. It should be noted that TAMR and SAMR methods are not mutually exclusive and can be combined within a single solver (e.g. Ref. [278]).

Both TAMR methods and SAMR methods have inherent advantages and disadvantages. TAMR methods will generally give a lower cell count as only marked cells are refined [5], but the more localised refinement can lead to higher synchronisation overheads than for SAMR grids [64]. The patch based data storage used in a SAMR solver can be more efficient as neighbouring cells on each patch are grouped in memory and no tree-traversal is required to determine connectivity [5].

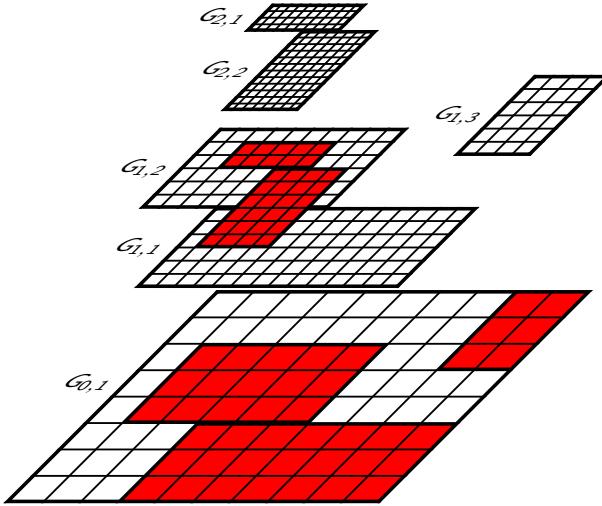


Figure 2.5: A SAMR refinement hierarchy with structured rectangular refinement blocks (© Ralf Deiterding).

However, SAMR solvers will generally carry out integration across the entire grid for all levels, even though the coarser level results are overwritten by the finer level results. This can negate some of the benefits of the SAMR data structure, but generally the integration on the coarse levels is a fraction of the cost of the integration on the fine level, especially when temporal refinement is used [64].

The algorithms required for efficient TAMR and SAMR on distributed memory machines add complexity to the underlying software, with SAMR methods generally needing more complex algorithms [5, 64, 302]. Unless careful thought is given to the design of the software, CAMR solvers can be more difficult to maintain and extend than non-adaptive solvers. Fortunately, a number of TAMR and SAMR frameworks have been produced over the years that aim to separate the complex mesh refinement algorithms from the physics being studied. This allows practitioners in specialist areas to use CAMR methods without requiring a detailed understanding of the underlying CAMR techniques. TAMR frameworks that can be used to build adaptive solvers include PARAMESH [174], p4est [36] and, more recently, Athena++ [263]. An extremely wide range of frameworks have been developed using SAMR methods, and a thorough review of existing frameworks is given within Ref. [73]. Some notable examples of SAMR frameworks include AMROC [64], SAMRAI [126], Chombo [3] and AMReX (previously BoxLib) [301].

CAMR has been shown to be highly efficient, reducing the computational times by a factor of 5 to 20 when compared to uniform high-resolution grids, whilst giving similar errors [139]. CAMR is known to be particularly effective for simulations that

contain discontinuities in some areas and smoothly varying regions elsewhere [128, 159]. This makes it well suited to the simulating the hypersonic flow fields that are being investigated in this work.

Although there are many advantages to using CAMR solvers there are also costs. Firstly, there is a computational cost incurred during the refinement process and in the synchronisation of the refinement regions. In simulations where large portions of the domain would need to be refined, and the refinement criteria are difficult to define (e.g. turbulence dominated wake flows), then high-order, non-adaptive methods can be more efficient [128]. However, a CAMR solver is likely to be more efficient than a Cartesian solver with a uniformly fine mesh when the refinement regions cover less than one third of the computational domain [128]. Secondly, all CAMR methods result in hanging nodes along refinement boundaries that require special treatment to ensure that the fluxes in this region are conservative [21, 35, 64]. It has also been shown that these coarse-fine boundaries can reduce the order-of-accuracy to first-order [159], so adequately refining important features is vital to obtain an accurate solution. The most relevant drawback of CAMR methods for hypersonic heat flux predictions is that the Cartesian nature of the grid makes the accurate representation of complex bodies difficult. In turn, this can lead to inaccurate boundary layer simulations and subsequently poor heat flux and shear stress predictions.

Complex geometries can be included on a Cartesian domain by creating an “embedded” boundary within the domain. Boundary conditions are then applied within the Cartesian cells at the edge of the embedded boundary [64]. Different approaches have been used to set the boundary conditions in the Cartesian cells. One such method approximates the geometry with the Cartesian cells, where each cell is marked as either inside or outside of the boundary and those immediately outside the boundary are used as ghost cells [64]. This creates a staircase-like surface when the geometry is not aligned with the Cartesian domain, which can artificially thicken the boundary layer. A demonstration of this can be seen in Fig. 2.6, which shows the velocity field in the double-wedge experiment simulated in Section 6.1. In this figure the results are shown for a body conforming mesh and for a Cartesian mesh that uses the embedded boundary method. One can see that the boundary-layer and recirculation region on the Cartesian domain are significantly larger due to the artificial roughness created by the stair-case geometry. In turn, this results in clear differences in the off-body structure where the separation shock is much further forward in the Cartesian simulation. Both the thicker boundary layer and the subsequent changes to the shock structure would severely reduce the accuracy of the heating predictions.

To overcome issues with inaccurate geometry representations, “cut-cell” methods were introduced that use the actual geometry of the boundary and partition cells along the line/plane of the boundary [127]. The fluxes in the boundary cells are then

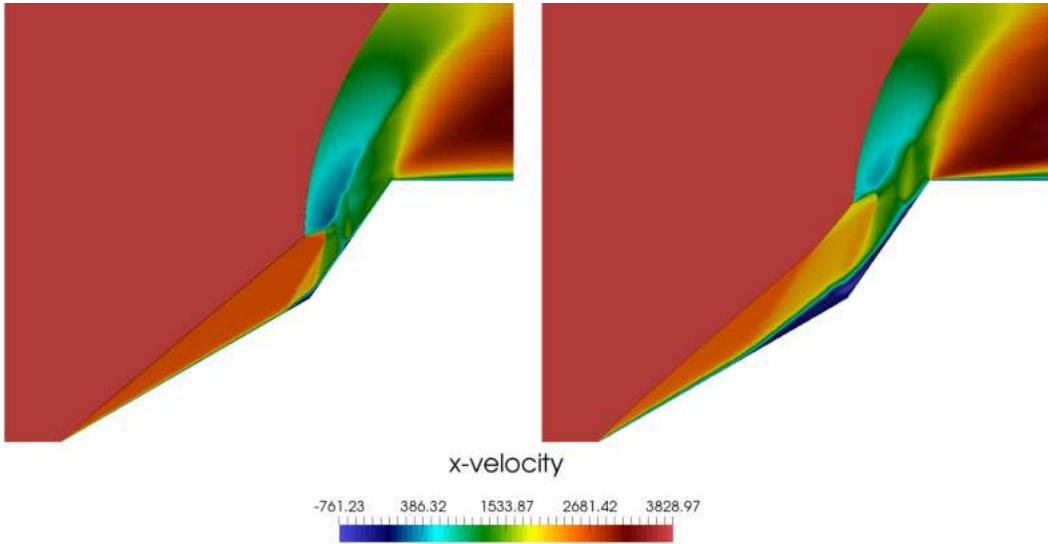


Figure 2.6: A comparison of the velocity field over a double wedge when using a body conforming grid (left) and a Cartesian grid with the embedded boundary method (right).

calculated using the cut-cell geometry. Whilst this can give better boundary representation, it can also result in arbitrarily small cells, which can severely restrict the time step. The time-step can be increased using cell merging techniques, but cell merging does not overcome the fundamental issue that is inherent to Cartesian domains: adequate boundary-layer resolution for arbitrary geometries.

As discussed in the introduction, the accurate prediction of heat fluxes requires extremely high resolution near the wall, with the Reynolds number based on the near-wall cell height spacing being less than or equal to one. This often results in a near-wall cell height of $1\text{ }\mu\text{m}$ or less. For a Cartesian grid to achieve this resolution around an arbitrary surface the number of cells required becomes very large. For example, in Fig. 2.6 the largest dimension of the base mesh is 0.11 m. Using 100×100 cells on the base mesh, 10 levels of refinement, each with a refinement factor of two, are required to reach $1\text{ }\mu\text{m}$ resolution at the wall. This set-up was tested using AMROC and the total cell count was found to be 1.13×10^6 for the above geometry. This is very large for a two-dimensional domain and the mesh was only refined at the wall and not refined for any shock structures. This demonstrates how obtaining sufficient boundary layer resolution in vehicle-scale, three-dimensional simulations using only Cartesian cells could make the computations intractable. Mapped and unstructured grids are able to achieve sufficient resolution with a manageable total cell count by using high aspect ratio cells at the wall and then rapidly increasing the cell size away from the wall. This is something that a purely Cartesian domain is not able to do.

The inability to efficiently and accurately represent complex geometries and resolve high Reynolds number boundary-layers is the motivation for introducing a near-body strand mesh and overset techniques.

2.2.2 Strand Mesh Generation

In a strand/CAMR solver, the mesh in the near-body region is generated using a “strand meshing” technique. These methods aim to create a high-quality boundary layer mesh with a high level of automation. In the original strand meshing process, first described by Meakin *et al.* [190], a mesh is generated in four stages.

1. The locations of the strands are set as the vertices of an input, tessellated surface.
2. The growth vector for each strand is calculated.
3. Strands are grown from the surface by placing nodes along each growth vector using a stretching function, as shown in Fig. 2.7.
4. Invalid cells that result from strands crossing each other are removed from the mesh by “clipping” the strands at the required node index, illustrated in Fig. 2.8.

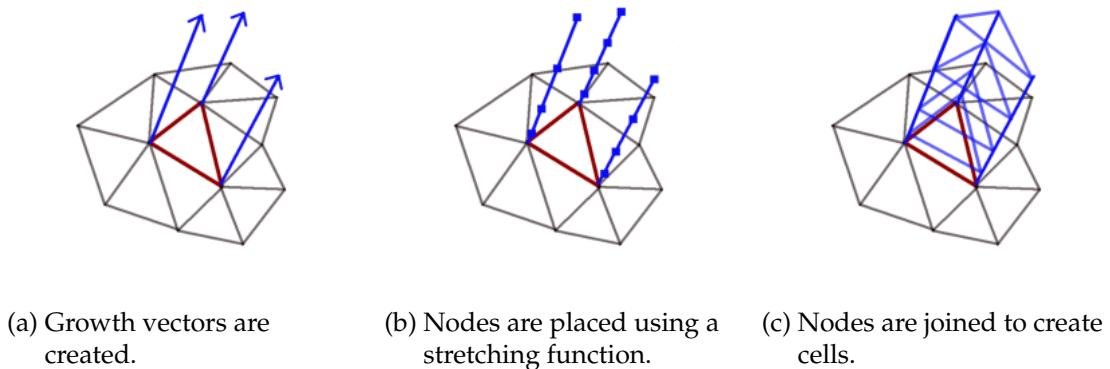


Figure 2.7: An example of the original strand mesh generation procedure.

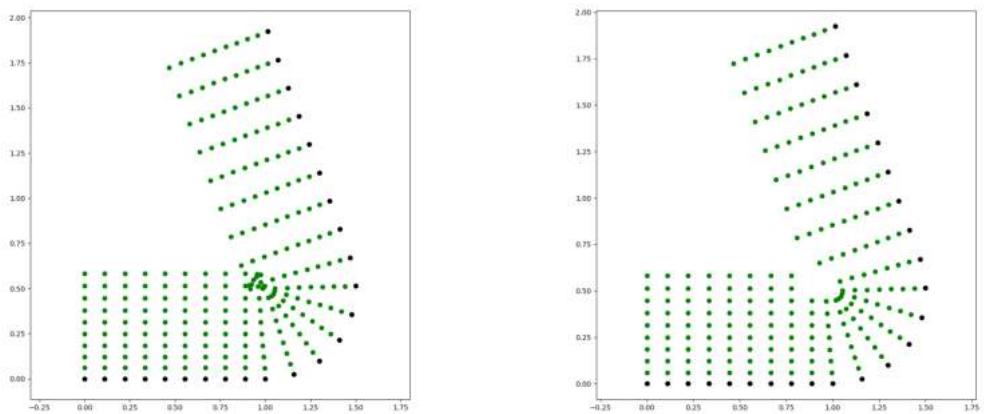


Figure 2.8: The nodes of a two-dimensional strand mesh before clipping (left) and after clipping (right).

For the above process, the method used to determine the growth vectors has the greatest influence on the overall mesh quality (for a constant surface mesh). As such, an algorithm was developed to determine the growth vectors, which included a user-input parameter to control the smoothness of the final mesh [137]. In this algorithm, the growth vector is first set as the normal vector at each surface node. Each growth vector is then iteratively updated by taking the average of the growth vectors of the surrounding vertices. Each iteration reduces the difference between neighbouring growth vector directions and improves the resolution at under-resolved convex features, as shown in Fig. 2.9. The process is terminated when the changes in the growth vectors fall below the user-input smoothing residual.

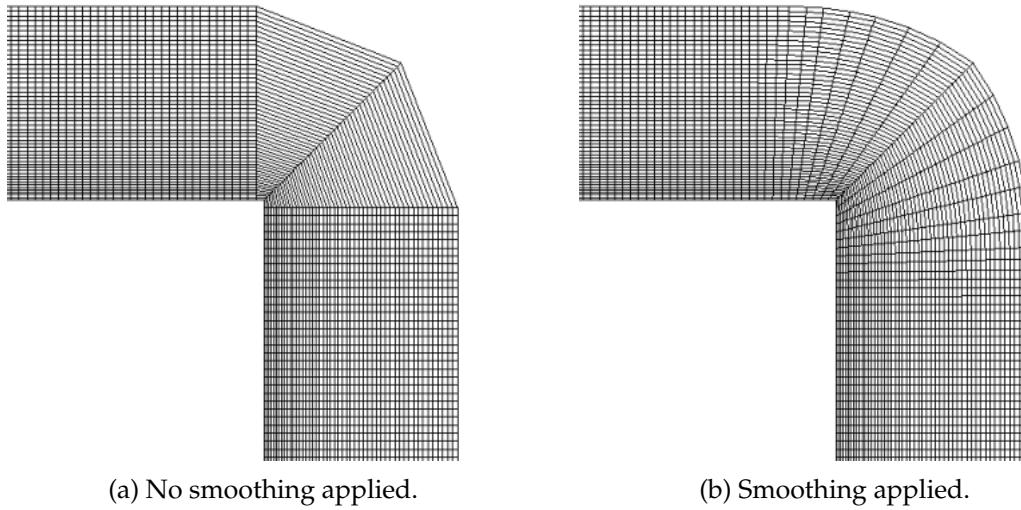


Figure 2.9: An illustration of the effect of strand mesh smoothing for a surface with convex corners.

Meshes generated using the above approach have two major benefits. First, the meshes are created with a high level of automation, where the user is only required to input the strand length, L_s , number of nodes, N_T , the initial cell spacing at the surface, Δs_0 and a desired smoothing residual r_0 , along with a discretised surface. Secondly, meshes defined in this way have an extremely low memory footprint. The strand length and stretching parameters are constant across the whole mesh and each vertex on the surface mesh has an associated location, growth vector and clipping index. Therefore, the entire mesh geometry can be derived by storing $6V + 2$ floating point numbers in three dimensions, or $4V + 2$ floating point numbers in two dimensions, and V integers for the clipping index, where V is the number of surface vertices. As such, these values can be stored locally on every processor, which can significantly reduce the amount of communication required during the overset grid assembly.

This simple approach was shown to give accurate results around complex bodies [137, 290]. However, some limitations of this strand mesh generation procedure were

noted. Firstly, at highly convex parts of the body (e.g. sharp corners) it was difficult to obtain adequate resolution without considerable smoothing. Whilst additional smoothing increased the resolution away from the surface, it increased the non-orthogonality of the cells joining the surface. This could have adverse impacts on the solution, especially for high Reynolds number boundary layer flows. Katz *et al.* [137] showed that there was an optimal level of smoothing for a given configuration. However, this was obtained through trial and error, which is not conducive to automated mesh generation. For hypersonic flows the non-orthogonality at the surface could have an influence on the heat flux prediction.

In an attempt to overcome the resolution/surface orthogonality trade-off and reduce the required user input, a “multi-strand” method was developed by Haimes [111]. In this method multiple strands are grown from nodes on edges or convex corners. This reduces the non-orthogonality of strands close to the corners as less smoothing is required to obtain the same resolution (shown in Fig. 2.10). However, investigations have shown that the simpler single strand method with growth vector smoothing gives better results than the multi-strand method in a variety of situations, including laminar flow around a square cylinder and turbulent flow over a NACA0012 wing [154, 294]. One possible reason for the poorer results of the multi-strand method is the large changes in cell sizes in the multi-strand region, close to the surface [254]. More recent efforts have therefore focused on multiple-level strand meshing [234, 254].

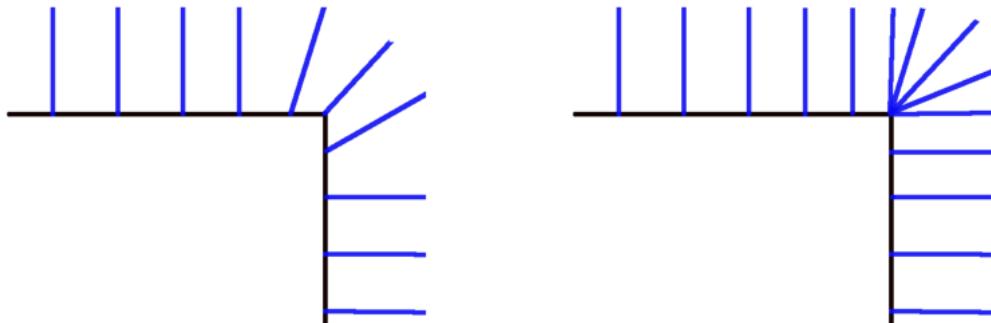
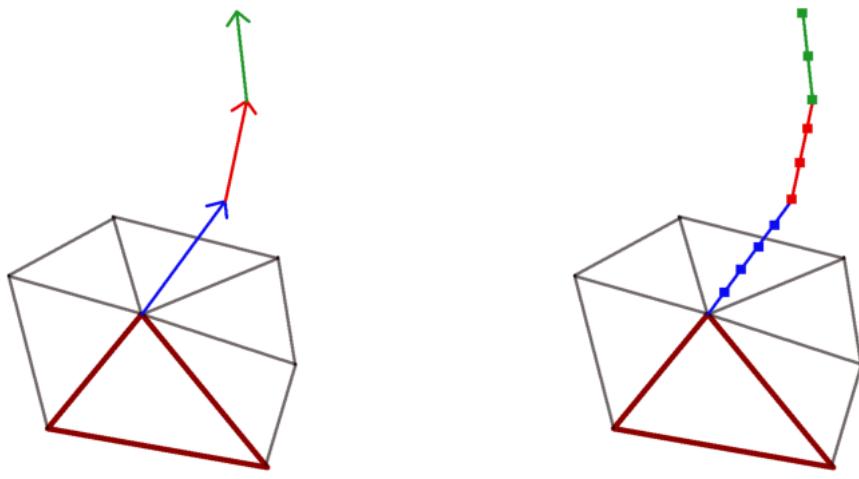


Figure 2.10: An illustration of the difference in growth vectors for a smoothed strand mesh (left) and a multi-strand mesh (right) at a convex part of the surface.

The multi-level method creates strands from several shorter segments, each with different growth vectors. First, an initial, small strand level is created where the growth vectors are normal to the surface [254]. The ends of these strands create a closed surface and the strand ends act as the new nodes from which the next level of strands are grown. Each level is able to use different growth vectors to the previous level (see Fig. 2.11) and a different number of nodes. The multi-level strand mesh can be combined with the multi-strand method where multiple strands are created at the same node.



(a) Different growth vectors for each level of the multi-level strand mesh. (b) Nodes distributed along the multi-level strand growth vectors.

Figure 2.11: An illustration of a multi-level strand mesh.

In the multi-level method, a new growth vector smoothing process was developed to be applied to each layer. For each level of strands, the initial growth vector at each node is determined by finding the closest point between the node and an iso-surface that is equidistant above the previous surface. This method minimises the occurrence of strand crossing, and therefore clipping, in concave regions of the mesh [254] (see Fig. 2.12a). The strand normal vectors are then smoothed using an elastic spring analogy. In this method, an imaginary spring connects each node on the iso-surface to its neighbouring nodes. The stiffness of each spring is inversely proportional to the edge length on the surface the strand was grown from (see Fig. 2.12b). This results in a force on each node that is dependent on the distance to the surrounding nodes and the stiffness value. The resulting equations of motion are integrated forward in time, with certain constraints on the node motion. These are a maximum distance constraint, an angle constraint based on “visibility” of the surface nodes, and a length constraint that ensures the strand remains on or above the iso-surface [233]. The multi-level strand mesh generation technique was used to generate meshes around complex bodies and the meshes were compared with those produced by automated advancing front and tetrahedral merging methods [233, 254]. They were shown to be of similar or better quality, based on a number of grid metrics such as smoothness, orthogonality and resolution. The results from the strand/CAMR solver were compared with experimental results and were shown to be in good agreement.

One disadvantage of the multi-level approach is that there is a greater level of user input required, thus the level of automation is reduced. Namely, the length of the strands on each level and the number of levels must be specified by the user and the

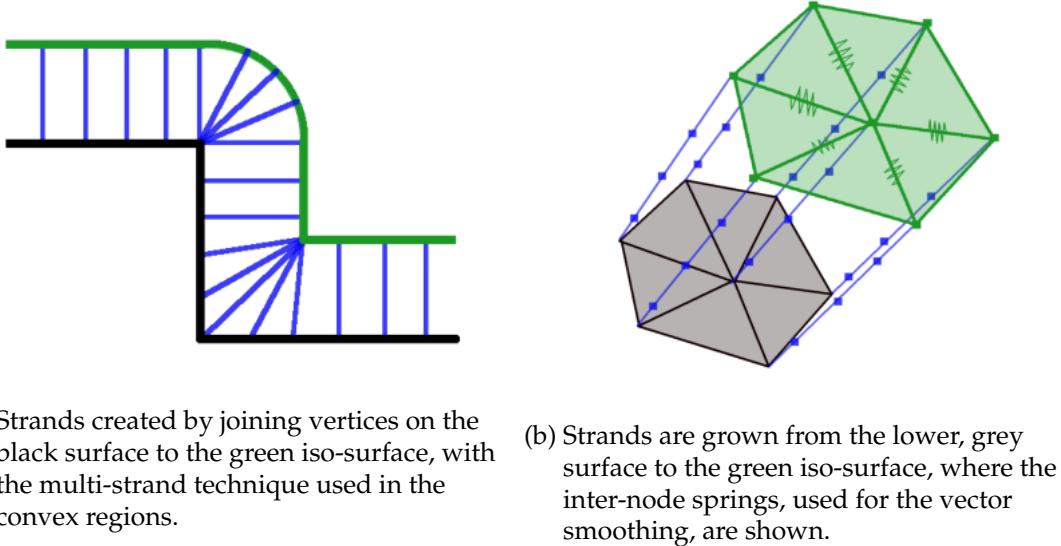


Figure 2.12: An illustration of the multi-level iso-surface and spring analogy smoothing technique.

smoothing operation lacks a simple exit criteria, so the user must set a specific number of iterations or a minimum walk threshold [233, 254]. Another disadvantage is that the multi-level method has a higher memory footprint when compared to the original strand meshing technique, as multiple growth vectors are associated with each surface vertex and, if the iso-surface method is used [234], each strand on each level is allowed to have a different length.

2.2.3 Strand Mesh Solution Techniques

In the first implementations of a strand/CAMR solver, a generic unstructured solver was used in the near-body strand mesh region [137, 190]. However, this failed to take advantage of the unique characteristics of the strand mesh. Each strand can be stored in memory as a one-dimensional structured mesh, whilst each layer parallel to the surface can be stored as an unstructured, two-dimensional mesh. This split data structure can not only reduce the computation times compared to a fully unstructured solver, but also simplifies the use of certain solution techniques. Firstly, efficient higher-order flux schemes can be implemented [138]. Secondly, it is simpler to take advantage of efficient, directionally split, time integration schemes [136].

The high-order scheme described by Katz and Work [138] utilises the strand structure by combining a finite difference method in the strand direction with a flux correction method in the unstructured layers. In the strand direction Summation-By-Parts (SBP) finite difference operators [85] are used to ensure high-order accuracy and discrete conservation in the finite difference flux calculations. In the unstructured layers, a flux correction scheme is used that maintains third-order accuracy for the inviscid fluxes

[135] and fourth-order accuracy for the viscous fluxes [223]. The two are combined by incorporating the high-order flux derivatives in the strand direction as a source term when integrating the unstructured layer in time. The results from this scheme showed that fourth-order accuracy could be obtained with only a 50% increase in the computational time required for a second-order-accurate scheme [138].

The strand structure enables implicit time integration methods to be used that take advantage of the structured data layout and strong temperature, density and velocity gradients in the wall-normal direction [136, 296] (see Fig. 2.13). The structured strand data makes it significantly easier to use line-relaxation methods to solve the implicit linear system. The line-relaxation method has been shown to be efficient for hypersonic flow simulations as it couples the solution of the linear system to the flow features in the boundary layer [296]. In a strand grid the block tri-diagonal Jacobian required for this method can be created easily as the cell indices in each strand are already aligned with the wall-normal direction. This is in contrast to a fully unstructured grid where algorithms are required to find lines in the mesh that emanate from the wall and then reorder the cell indices along these lines [241].

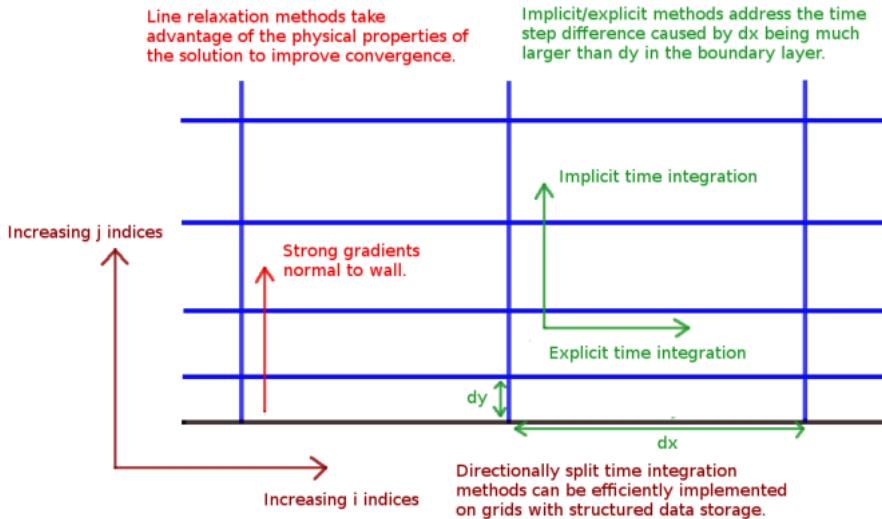


Figure 2.13: An illustration showing the structured, high aspect ratio strand mesh and how this influences the time integration methods.

The structured data layout along each strand can also be used in the implementation of directionally partitioned, implicit/explicit methods [275]. Strand meshes often use high-aspect-ratio cells close to the wall. The stable time-step for a explicit method is linked to the geometry of the cell (see Section 3.3.1.1 for details). As a result, the explicit time step can be orders of magnitude larger in the wall-aligned direction than in the wall-normal direction. In an implicit/explicit method, the update in the strand direction would be implicit, and the update in the unstructured layers would be explicit. This enables large time steps whilst increasing the efficiency of the method

compared to a fully implicit method. In a dimensionally partitioned method the Jacobian only needs to be created for fluxes in the wall-normal direction and the resulting block tri-diagonal Jacobian can be efficiently inverted using block Lower-Upper decomposition [136, 241, 296, 281]. The implicit/explicit method was shown to greatly reduce the time to steady-state when compared with a fully explicit method [275]. However, it has been shown to result in slower convergence when compared to fully implicit methods [136] due to the limited time step in the explicit direction [136]. The advantage gained from an implicit/explicit method is likely to be highly dependent on the flow and the ratio of the minimum face length in each dimension. For time-accurate simulations, the strand structure would make it simpler to use high-order-accurate implicit/explicit methods, such as those discussed in Ref. [12]. These have been used in other areas of computational fluid dynamics where the high aspect ratio cells result in small explicit time steps [55, 166, 277, 285].

2.2.4 Overset Grid Assembly

The near-body strand mesh domains and the off-body CAMR domain must be assembled into a single computational domain using overset (a.k.a Chimera) methods. As discussed in Section 1.2, overset grid methods in CFD were introduced through the work of Starius [257], Kreiss [150], Henshaw and Chesshire [52, 119], and independently by Steger and Benek [20, 258]. In overset simulations, near-body meshes are overlaid onto a background mesh and holes are cut in the background mesh around the surface of the body. The separate, overlapping, computational domains are joined through the exchange of information as boundary conditions. Performing the hole cuts and exchanging the required information between the domains is generally known as overset domain assembly. A large variety of overset domain assembly algorithms have been described in the literature and implemented into libraries such as Overture [33], PEGASUS [231], CHIMPS [8], SUGGAR++ [205], PUNDIT [232] and TIOGA [30], to name a few. In this section, various methods that have been used to carry out overset domain assembly are discussed.

The boundaries of each domain are determined using a process known as hole-cutting. This is where cells are effectively blanked out of one domain due to the presence of another domain. This could either be where cells are cut from the background mesh due to the presence of an overset near-body grid, or where cells are cut from overlapping near-body grids. This results in the cells of each mesh being classified as either solve cells, where the solution is updated as part of the integration, or hole cells, where they are not updated, but are either blanked out or used to set the boundary conditions. Figure 2.14 shows a simple example where a single overset domain is placed onto a Cartesian background domain, and a hole is cut in the background

domain. To calculate the geometry of the hole and classify the cells two different types of hole cutting can be used, explicit hole cutting and implicit hole cutting.

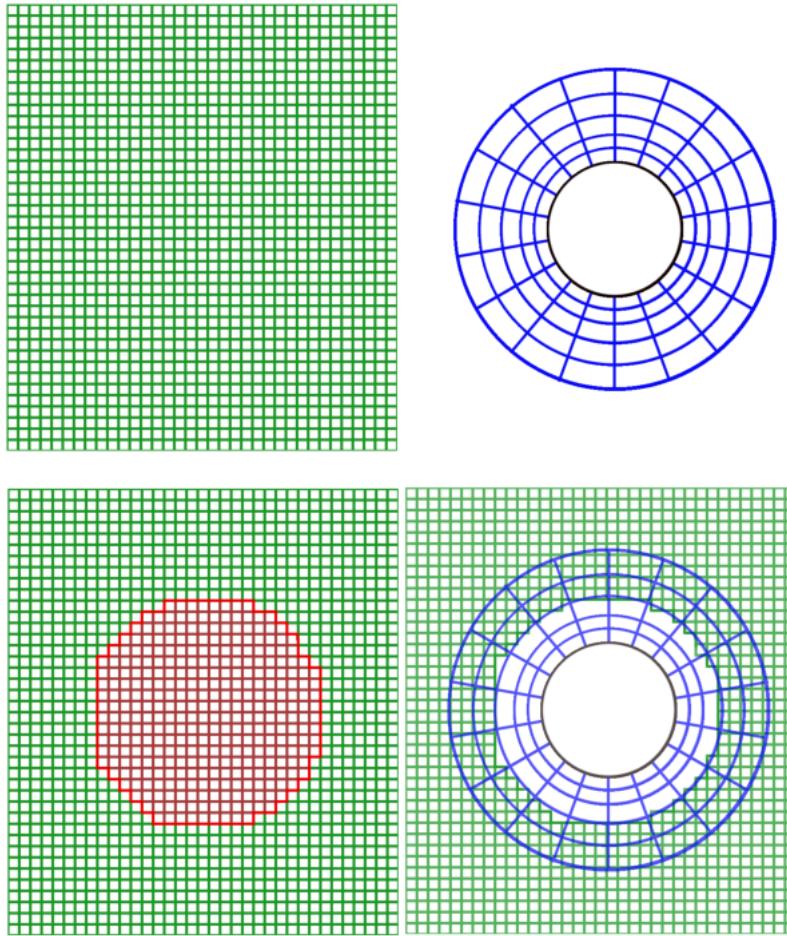


Figure 2.14: An example of a body-fitted domain overset onto a background Cartesian domain. The dark red cells in the Cartesian domain show the hole-cut.

Explicit hole cutting uses the geometry of the bodies and grids directly in order to establish the hole cuts. For an overset mesh on a Cartesian background mesh, a “level curve” at the exterior of the overset mesh can be used as the surface of the hole that needs to be cut out of the background domain [19]. If a level curve is not available, surface normal vector tests [19] or vector ray casting [189, 222] can be used to find the points within the overset solid bodies. However, these methods can require significant user input [264], especially if there are multiple, possibly overlapping overset domains. Another explicit technique offering greater autonomy is the hole map technique [188, 264]. In this method, a Cartesian grid overlays the bodies, and each Cartesian voxel can then be rapidly classified as either inside, outside or on the boundary of a solid body by using flood-fill algorithms. The majority of solver cells

can then be rapidly classified as either in or out of the hole depending on the classification of the Cartesian boxes they are in. Only cells that are within an ambiguous box need to undergo further testing.

Implicit hole cutting methods were developed to reduce the user input required for overlapping near-body domains [158]. This method first determines if two cells from different domains overlap. If they overlap, cell attributes, such as cell volume, aspect-ratio, wall-distance (or combinations of these values [167]) are used to determine which cell should be marked as a solve cell and which as a hole cell. This defines the hole and solve points for each domain with minimal additional input from the user. It provides an automated method of removing overlapping cells from body-fitted domains whilst optimising the quality of the body-fitted meshes. The disadvantage of the implicit method is that it can be a computationally expensive procedure [47] due to the need for exhaustive cell-to-cell comparisons over the entire domain. To reduce the computational expense, bounding boxes and oriented bounding boxes can be created surrounding each mesh and used to quickly down-select the cells that may overlap [264]. For example, any cells that fall outside of the bounding box of the other mesh are not considered.

The holes within a background domain result in internal boundaries that require boundary conditions. Similarly, a near-body domain requires boundary conditions at its outer edge. These are known as inter-grid boundary conditions and must be set using information from another grid. This is commonly achieved by interpolating the solution from one grid to fill boundary cells on the other grid [19], but can also be achieved by interpolating fluxes to the overset boundary [53, 58]. Boundary cells that receive information are known as receptors, receivers or fringe points. The cells from a different mesh that provide the required information are known as donor cells. In order to exchange information, the domains must have an overlap. The amount of overlapping cells required will depend on the number of ghost cells on each mesh and the number of cells that are needed around each receptor to enable the interpolation. For a well-defined problem, the donor cells must also be solve cells, and this must be accounted for when carrying out the hole cutting procedure. Figure 2.15 shows an example of ghost cells on the Cartesian and overset domains.

Once the receptors have been determined on each mesh, the flux or solution points in these cells become “query” points that require data from another mesh. The domain assembly software must determine which cells from other meshes are required to interpolate data to the query point i.e. which cells become donor cells. Determining the donor cells on a Cartesian grid is trivial, as the global indices of the cells surrounding a given query point can be located using the Cartesian box dimensions and step size. However, finding donor cells on a body-fitted or unstructured mesh is more difficult, and the computational cost can be high. Minimising this cost is one of the main objectives when creating an efficient overset grid assembly tool-set.

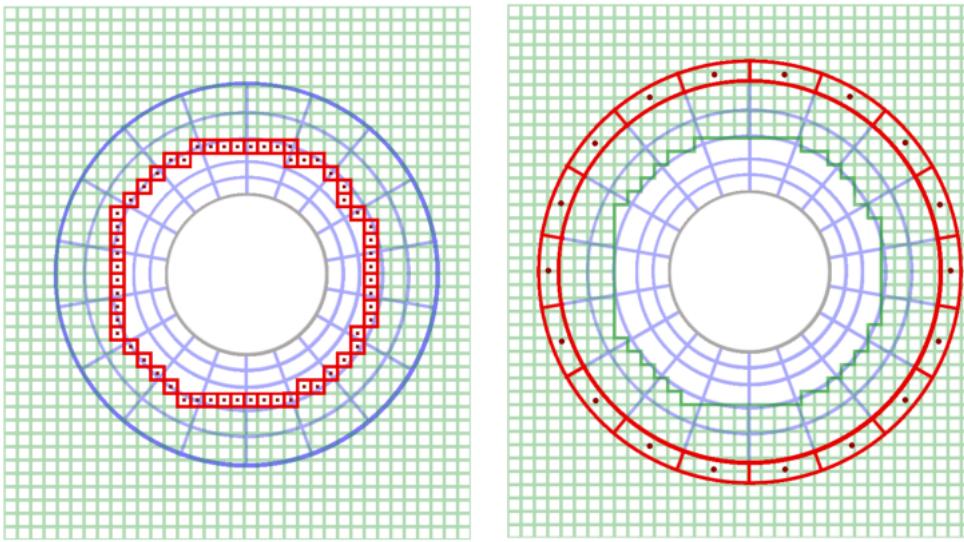


Figure 2.15: An example of ghost cell locations for the overset boundaries.

The dark red points are the receptor centres on the Cartesian domain (left) and the near-body domain (right).

A simple and robust donor search technique is an exhaustive search, where an inclusion test is carried out on each cell to determine whether it contains the query point. However, this algorithm is inefficient as there is the potential for a large number of cells to be tested unnecessarily. Therefore, spatial decomposition algorithms are used to reduce the number of cells that are tested. These methods only test cells that are already determined as being close to the query point.

One spatial decomposition method from the literature uses a Cartesian approximate inverse map [140, 187, 232] to group unstructured/curvilinear cells into spatial “buckets”. The Cartesian auxiliary mesh overlays the unstructured/curvilinear mesh and if a cell overlaps with a bucket the cell is associated with the bucket (see Fig. 2.16). Given a query point location, the corresponding index of the Cartesian bucket can be identified trivially. The cells stored in this bucket can then be retrieved and an inclusion test performed on each cell. The inverse map can be constructed using each computational cell’s bounding box or exact cell geometry. When using the exact cell geometry this is referred to as Structured Auxiliary Meshes with Element Storage [140] or Exact Inverse Maps [232]. Using exact geometries makes the creation of the map more complex and costly, but leads to lower memory usage and faster donor search times than when using bounding boxes [140]. A benefit of either inverse map method is that query points that lie outside of the domain can be quickly identified, as they will either fall outside of the inverse map or the inverse map cell will be empty.

A second spatial decomposition method that is commonly found in the overset literature is the Alternating Digital Tree (ADT) method [27]. This method has

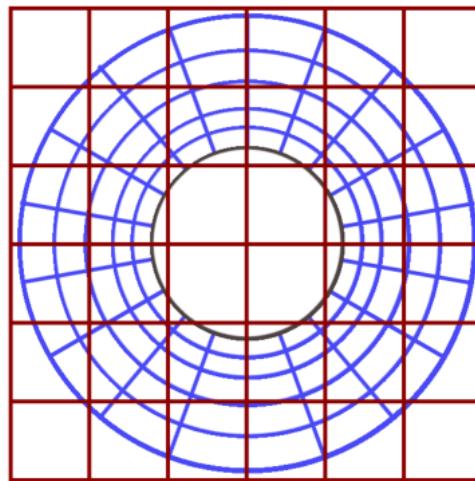


Figure 2.16: An auxiliary, Cartesian mesh (dark red) is used to group cells from a mapped/unstructured mesh (blue) into spatial buckets.

similarities to the inverse map methods in that it uses cell bounding box information to reduce the number of cells that are checked in the donor search. However, instead of placing the bounding boxes of the potential donor cells into an inverse map they are instead stored in a tree structure. The tree is created using the minimum and maximum points in each dimension of the bounding box for each cell. Once the tree structure has been created, bounding boxes that contain the query point can be found using tree-based range (a.k.a. window) searches, where the range is defined by the query point location and the minimum and maximum points of all of the cells. The query point may reside within several cell bounding boxes so an inclusion test is then carried out for each cell. Comparisons between tree-based and inverse map methods have shown that map methods generally result in much faster search times, but can have higher set-up times [112, 140].

Another donor search method that has been used extensively in overset libraries is the “stencil walk” procedure. In this method, the connectivity information of the mesh is used to walk towards the query point. First, a starting cell is selected and a test is performed to determine whether the interpolation point lies within the cell. If the test fails, the the cell’s geometry is used to determine the direction the search should progress, and the adjacent cell in this direction is then tested. The search progresses from cell to cell until the cell containing the query point is found. On a structured curvilinear grid, the inclusion test and search direction use a mapped computational space for each cell [188], as shown in Fig. 2.17a. By assuming the mapping remains roughly constant in a given area, the cell indices can be increased by more than one as the geometry of the mesh can be inferred from the structured nature of the grid. This enables multiple cells to be traversed without being tested which can speed up the algorithm [264] and is known as “stencil jumping”. On an unstructured grid, the

stencil walk is performed by walking through cell faces from one cell to the next until the cell containing the query point is found, as shown in Fig. 2.17b. There are various ways to determine the direction of the unstructured walk, such as testing the angle between each face normal and the vector from the face centre to the query point [205], or testing which face a line between the cell centre and the query point intersects [253]. For both curvilinear and unstructured grids, walking methods can suffer from robustness issues if the walk goes across a solid boundary or an exterior boundary. Nakahashi *et al.* [198] and Murray [197] alleviated this problem by adding subsidiary grids inside solid boundaries that enable the walking algorithm to walk through the body.

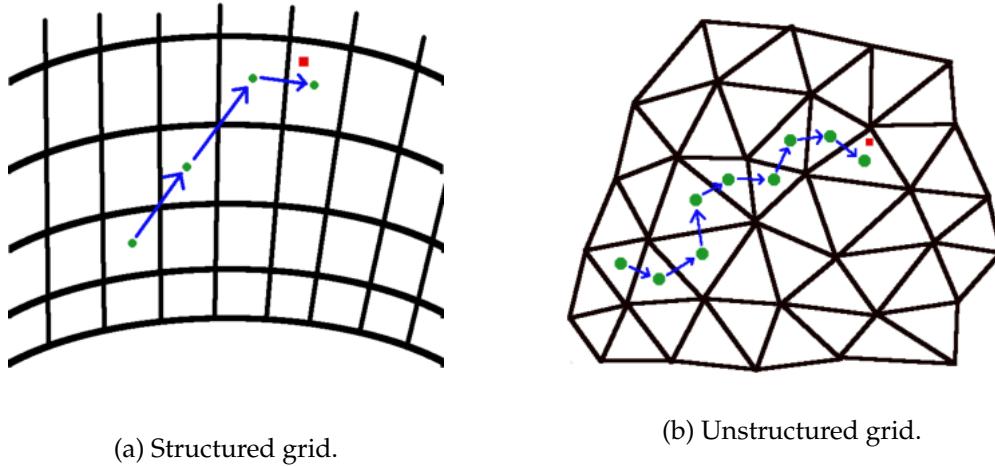


Figure 2.17: Stencil walk illustration on structured and unstructured grids. On a structured grid, both i and j indices can be traversed in one step.

For a stencil walking algorithm, the starting point has a major impact on the computational cost of the search algorithm. Spatial partitioning strategies can be used to improve the performance of the stencil walk by improving the starting location of the walking algorithm. For example, the stencil walk can be combined with an inverse map or ADT [232], where the starting cell is selected as a cell in the same bucket or as a cell with an overlapping bounding box. Alternatively, kd-trees containing cell centre or node locations can be used to perform nearest neighbour searches in order to select the initial cell in the stencil walk [124]. The performance of a walking algorithm can also be improved if the query points are ordered in an efficient manner. By selecting a query point close to the previous one and starting the walking algorithm in the cell where the previous query point was found, the number of steps taken can be greatly reduced [46]. If the query points are the cell centres on a Cartesian background grid they can be ordered in an efficient way using the global Cartesian cell indices. When the mesh is in motion, one can use the “temporal coherence” of the connectivity each time it needs to be regenerated [124]. This is simply the idea that geometry of two meshes will only differ by a small amount at each time step - in explicit methods, the mesh can only move by less than a single cell width. As such, the previous donor

element is likely to be the same as or very close to the required donor element and should be checked first.

Most modern CFD solvers run on multiple processors and the domain assembly process must account for this. The simplest method is to gather all of the domains' information onto a single processor, perform the domain connectivity and interpolation, and then send out the required information to each processor. However, this is likely to be inefficient as all other processors will be idle during the overset grid assembly. The efficiency can be improved by running the domain connectivity algorithm on all of the processors [205, 253], where, ideally, the work load of the domain assembly should be evenly split over the available processors. To reduce the communication required between processors, spatial decomposition algorithms can again be used in a pre-processing step. For example, bounding boxes can be shared between the processors to establish which processors may have overlapping grids and therefore may need to exchange receptor and donor information. However, multi-processor domain assembly can still lead to inefficient load balancing, especially when there are only a small number of processors with overlapping grids. The load balancing can be improved by either placing overlapping grids on the same processor [205] or increasing the number of processors that contain receptor or donor points [232].

2.3 Chapter Summary

Extensive reviews of the physical models most commonly used in hypersonic flow simulations and the components of a strand/CAMR solver have been conducted. The knowledge gained from these reviews was used in the development of the hypersonic flow model and strand/CAMR solver in this work. In the next chapter, the governing equations and strand mesh solution methods are presented, whilst the automated meshing techniques are detailed in Chapter 4.

Chapter 3

Hypersonic Flow Model and Numerical Solution

In this chapter the governing equations for hypersonic flows and the numerical methods used to solve these equations are presented. A set of governing equations for simulating hypersonic flows is derived based on the models discussed in Section 2.1. The governing equations are solved within the the SAMR framework AMROC (Adaptive Mesh Refinement in Object-oriented C++) [64]. In previous work, a shock capturing finite-volume method was integrated with the framework to allow highly adaptive simulations of flow fields that include discontinuities [62, 63]. As part of this research project, a hypersonic fluid model has been incorporated within an AMROC patch integrator, enabling adaptive simulations of thermochemical nonequilibrium flows. In addition, the spatial and time integration routines have been extended so that solutions to viscous, wall-bounded hypersonic flow problems can be efficiently obtained. The Cartesian solver has been modified to allow mapped, body-fitted meshes to be used, and implicit time integration methods have been implemented within AMROC for the first time. The implicit methods utilise efficient solution techniques that take advantage of the quasi-structured layout of a strand mesh and the expected physical results.

3.1 Governing Equations

Following the review of different thermochemical models for hypersonic flow simulations, it was decided to use Park's two-temperature model [215] in this work (see Section 2.1). The governing equations for the two-dimensional, two-temperature Navier-Stokes equations are given by

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} - \frac{\partial \mathbf{f}^v}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} - \frac{\partial \mathbf{g}^v}{\partial y} = \mathbf{w}, \quad (3.1)$$

The vector of conserved variables \mathbf{q} , the inviscid flux vectors \mathbf{f} and \mathbf{g} , and the source vector \mathbf{w} take the form

$$\mathbf{q} = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{N_s} \\ \rho u \\ \rho v \\ \rho e^{ve} \\ \rho E \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} \rho_1 u \\ \vdots \\ \rho_{N_s} u \\ \rho u^2 + p \\ \rho u v \\ \rho v \\ \rho e^{ve} u \\ u(\rho E + p) \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} \rho_1 v \\ \vdots \\ \rho_{N_s} v \\ \rho u v \\ \rho v^2 + p \\ \rho e^{ve} v \\ v(\rho E + p) \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_{N_s} \\ 0 \\ 0 \\ Q_{ve} \\ 0 \end{bmatrix}. \quad (3.2)$$

The viscous flux vectors, \mathbf{f}^v and \mathbf{g}^v are given by

$$\mathbf{f}^v = \begin{bmatrix} -J_{x,1} \\ \vdots \\ -J_{x,N_s} \\ \tau_{x,x} \\ \tau_{y,x} \\ \kappa_{ve} \frac{\partial T_{ve}}{\partial x} - \sum_{s=1}^{N_s} J_{x,s} e_{ve} \\ \kappa_{tr} \frac{\partial T_{tr}}{\partial x} + \kappa_{ve} \frac{\partial T_{ve}}{\partial x} + u \tau_{x,x} + v \tau_{y,x} - \sum_{s=1}^{N_s} J_{x,s} h_s \end{bmatrix}, \quad (3.3)$$

and

$$\mathbf{g}^v = \begin{bmatrix} -J_{y,1} \\ \vdots \\ -J_{y,N_s} \\ \tau_{x,y} \\ \tau_{y,y} \\ \kappa_{ve} \frac{\partial T_{ve}}{\partial y} - \sum_{s=1}^{N_s} J_{y,s} e_{ve} \\ \kappa_{tr} \frac{\partial T_{tr}}{\partial y} + \kappa_{ve} \frac{\partial T_{ve}}{\partial y} + u \tau_{x,y} + v \tau_{y,y} - \sum_{s=1}^{N_s} J_{y,s} h_s \end{bmatrix}, \quad (3.4)$$

where κ_{tr} and κ_{ve} are the translational-rotational and vibrational-electronic thermal conductivities, respectively.

The species diffusion is based on a modified version of Fick's diffusion law [266], which ensures the sum of the diffusion fluxes is zero. In the x -direction this is given for all species other than electrons as,

$$J_{x,s \neq e} = -\rho D_s \frac{\partial Y_s}{\partial x} - Y_s \sum_{r \neq e} (-\rho D_r \frac{\partial Y_r}{\partial x}), \quad (3.5)$$

where Y_s is the mass fraction of species s and D_s is the average diffusion coefficient of species s . A similar expression is obtained in the y -direction, where the x derivatives are replaced by the y derivatives.

For flows containing electrons the electron diffusive flux is calculated using the assumption of ambipolar diffusion. This guarantees charge neutrality and gives the diffusive flux for electrons as

$$J_{x,e} = M_e \sum_{i=1}^{N_i} \frac{J_{x,i} C_i}{M_i} \quad (3.6)$$

where N_i is the number of ions and C_i is the charge of the ion.

The viscous stress tensor, $\tau_{i,j}$, is given by

$$\tau_{i,j} = \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) - \delta_{i,j} \frac{2}{3} \mu \nabla \cdot \mathbf{u}, \quad (3.7)$$

where $\delta_{i,j}$ is the Kronecker delta.

The two-dimensional axisymmetric governing equations are given by

$$\frac{\partial(y\mathbf{q})}{\partial t} + \frac{\partial(y\mathbf{f})}{\partial x} - \frac{\partial(y\mathbf{f}^v)}{\partial x} + \frac{\partial(y\mathbf{g})}{\partial y} - \frac{\partial(y\mathbf{g}^v)}{\partial y} = y\mathbf{w} + y\mathbf{w}^{\text{axi}}, \quad (3.8)$$

where \mathbf{q} , \mathbf{f} , \mathbf{f}^v , \mathbf{g} , \mathbf{g}^v and \mathbf{w} are the same as above,

$$\mathbf{w}^{\text{axi}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ (p - \tau_{\theta\theta})/y \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.9)$$

and

$$\tau_{\theta\theta} = 2\mu - \frac{2}{3}\mu \nabla \cdot \mathbf{u}. \quad (3.10)$$

In the axisymmetric equations the divergence of the velocity vector (required in the source term, and within the viscous flux vectors) is given by

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{1}{y} \frac{\partial(yv)}{\partial y}. \quad (3.11)$$

The closing equations are calculated using the open-source thermochemical library, Mutation++ [245]. Mutation++ is a library that contains state-of-the-art thermochemical models in a computationally efficient, object-oriented C++ package. The library has been designed for use in high-enthalpy flow simulations and is able to

calculate the thermodynamic properties and source terms required for Park's two-temperature model. The main advantage in using this library is that it saves significant work re-implementing and debugging thermochemical models that are readily available in an open-source library. This will allow for greater focus on the automated mesh generation techniques. The disadvantage of using Mutation++ is that some of the models are not state-of-the-art, although most are. Areas where the models can be improved are discussed below. This said, the open-source nature of the code and its object-oriented design enables different models to be added quickly, if required. The relevant thermochemical constants used in Mutation++ library are given in Appendix A.1.

The mixture density, ρ , is calculated as the sum of the partial densities ρ_s ,

$$\rho = \sum_{s=1}^{N_s} \rho_s, \quad (3.12)$$

and the pressure, p , is determined using the ideal gas equation and Dalton's Law of partial pressures,

$$p = \sum_{s=1}^{N_s} p_s = \sum_{s \neq e} \rho_s \frac{R_u}{M_s} T_{tr} + \rho_e \frac{R_u}{M_e} T_{ve}, \quad (3.13)$$

where R_u is the Universal Gas Constant and M_s is the molecular weight of species s .

The specific internal energy, e_s , is calculated by the Mutation++ library using the RRHO model, given by Eqs. (2.11), (2.22), (2.23) and (2.24), with $T_{ref} = 298.15\text{ K}$. The total specific energy can then be calculated using

$$E = \sum_{s=1}^{N_s} \frac{\rho_s}{\rho} e_s + \frac{1}{2} (u^2 + v^2). \quad (3.14)$$

For reasons discussed in Section 2.1.1 the calculation of internal energies using curve fits is more accurate and likely to be more computationally efficient than the RRHO method used in the two-temperature model in Mutation++. Internal energy calculations using the NASA curve fits are implemented within Mutation++ for single temperature mixture. Thus, it would be relatively simple to create a two-temperature model using the curve fits if required. However, this would prevent the vibrational and electronic temperatures from being considered separately.

The thermodynamic state can be set by passing the species densities and either the mixture internal energies or temperatures to Mutation++. When the internal energies are used to set the state, the temperatures must be determined. They are calculated using a using a Newton-Raphson method, where

$$\frac{\partial e^{tr}}{\partial T^{tr}} = C_v^{tr} = \sum_{s=1}^{N_s} \frac{\rho_s}{\rho} C_{v,s}^{tr} \quad (3.15)$$

and

$$\frac{\partial e^{ve}}{\partial T^{ve}} = C_v^{ve} = \sum_{s=1}^{N_s} \frac{\rho_s}{\rho} C_{v,s}^{ve}. \quad (3.16)$$

The species net production rates \dot{w}_s , are calculated using the reaction rate constants of Park [216], and Park's two-temperature model is used to account for the role of vibrational energy in dissociation reactions (Eq. (2.53)). As discussed in Section 2.1.4, Park's thermo-chemical coupling model has limited theoretical basis. A more theoretically sound model, such as Marrone and Treanor's [177] could be more accurate. This would also enable the incorporation of QCT results into the dissociation model, as demonstrated by Chaudhry and Candler [49], who are planning more work in this area. This said, many authors have found the differences in heat flux to be small [61, 209, 247], and the Park model seems to be the most widely used. As such, it is sufficient for this work.

The backward reaction rates are determined using an equilibrium constant that is calculated as a function of Gibbs free energy [245]. In Section 2.1.3 the work of Tchuen [269] was discussed, which compared various equilibrium constant curve fits and found Park's curve fits [216] to be the most accurate. However, he did not compare the results to the calculation of the equilibrium constant using Gibbs free energy minimisation. In addition to this, the Park curve fits are derived from experimental data, whereas the Gibbs-minimisation method has a stronger theoretical underpinning. The disadvantage of using this method is that it is likely to be more computationally expensive than the using curve fits.

Within Mutation++, the vibrational-electronic source term is calculated as

$$Q_{ve} = \sum_{s \neq e}^{N_s} \left(Q_s^{T-V} + Q_s^{C-V} + Q_s^{C-el} \right) + Q^{T-e} + Q^{C-e} + Q^I. \quad (3.17)$$

The translational-vibrational energy exchange, Q_s^{T-V} , is modelled using the difference in vibrational energy only,

$$Q_s^{T-V} = \rho_s \frac{de_s^v}{dt} = \rho_s \frac{e_s^v(T_{tr}) - e_s^v}{\tau_{v,s}^{T-V}}, \quad (3.18)$$

and the vibrational relaxation time is given by the Millikan and White formula, with modified constants [216] and the Park correction time,

$$p \tau_{s,r}^{T-V} = \exp \left[A_{s,r} \left(T_{tr}^{-1/3} - B_{s,r} \right) - 18.42 \right] \text{ (in atm-sec)}, \quad (3.19)$$

$$\tau_{v,s}^{T-V} = \frac{\sum_{r=1}^N X_r}{\sum_{r=1}^N \frac{X_r}{\tau_{s-r}^{T-V} + \tau_{s-r}^p}}. \quad (3.20)$$

The Park correction time is applied for each colliding pair, and takes the form given by Duffa [74],

$$\tau_{s-r}^p = \sqrt{\frac{\pi k_B \mu_{s,r} T_{tr}}{8N_A}} \frac{1}{\sigma_s p}. \quad (3.21)$$

The temperature in the calculation of the collision cross section, σ_s , is limited to 20,000 K,

$$\sigma_s = 3 \times 10^{-21} \left(\frac{50,000}{\min(T_{tr}, 20,000)} \right)^2. \quad (3.22)$$

As previously discussed in Section 2.1.1, the above translational-vibrational exchange model is largely empirical but does not give significantly different results to more theoretically sound models, such as the SSH models, or Landau-Teller formulations based on quantum state-to-state calculations [74, 103, 201].

The source terms accounting for the change in the vibrational and electronic energy due to chemical reactions, Q_s^{C-V} , Q_s^{C-el} , are given by Eqs. (2.60) and (2.62), respectively. In both cases a non-preferential model is used so $c_1 = 1$. This is another area where the model within Mutation++ could be improved, and the Marrone and Treanor model [177], or modified Marrone and Treanor model [49, 148] could be employed, as it has been shown to more closely match state-to-state results [113]. The source terms accounting for energy transfer to and from electrons, Q^{T-e} , Q^{C-e} , Q^I , are given by Eqs. (2.44), (2.66) and (2.69), respectively.

In Mutation++ the mixture viscosity and translational thermal conductivity can be calculated using Wilke's mixing rule [287], the Gupta-Yos mixing rule [109] or by solving the linear system derived from the Boltzmann equations using the Chapman-Enskog procedure [94, 176]. The inclusion of the Chapman-Enskog method is one of the strengths of Mutation++, as it is the most accurate [211] and it has been implemented in a computationally efficient manner [94, 245]. Therefore, this is the default method used in this work. The internal energy conductivities are calculated using the Eucken approximation and the collision cross sections are taken from the literature [245]. The average diffusion coefficient for each species is calculated as the reciprocal molar average of the binary diffusion coefficient relative to all other species,

$$D_s = \frac{(1 - x_s)}{\sum_{r \neq s} x_r / \mathcal{D}_{sr}}, \quad (3.23)$$

where x_s is the mole fraction and \mathcal{D}_{sr} is the binary diffusion coefficient, calculated using the collision cross-sections of the pair of species.

In addition to the transport property calculations in Mutation++, the simple Blottner/Eucken/Wilke model has also been implemented. In this model the viscosity of each species is calculated using

$$\mu_s = 0.1 \exp [(A_s \ln(T_{tr}) + B_s) \ln(T_{tr}) + C_s], \quad (3.24)$$

where the constants A_s , B_s and C_s are species specific. The conductivities are then found using the Eucken approximation,

$$\kappa_s^{tr} = \mu_s \left(\frac{5}{2} C_{v,s}^t + C_{v,s}^r \right), \quad (3.25)$$

$$\kappa_s^{ve} = \mu_s C_{v,s}^{ve}, \quad (3.26)$$

and the mixture properties are obtained from the individual species properties using the Wilke mixing rule [287],

$$\mu = \sum_{s=1}^{N_s} \mu_s \frac{x_s}{\phi_s}, \quad \kappa^{tr} = \sum_{s=1}^{N_s} \kappa_s^{tr} \frac{x_s}{\phi_s}, \quad \kappa^{ve} = \sum_{s=1}^{N_s} \kappa_s^{ve} \frac{x_s}{\phi_s}, \quad (3.27)$$

where

$$\phi_s = \sum_{r=1}^{N_s} x_r \left[\left(1 + \sqrt{\frac{\mu_s}{\mu_r}} \left[\frac{M_r}{M_s} \right]^{\frac{1}{4}} \right)^2 \frac{1}{\sqrt{8 + 8 \frac{M_s}{M_r}}} \right]. \quad (3.28)$$

To complete the simpler transport model, an average diffusion coefficient is calculated for all species. The average diffusion coefficient is obtained via a user defined Lewis number, Le , and is given by

$$D = \frac{Le \kappa^{tr}}{\rho C_p^{tr}}. \quad (3.29)$$

Finally, the frozen speed-of-sound (i.e. the speed-of-sound at a fixed chemical composition) is calculated as

$$a = \sqrt{\left(1 + \frac{R_u}{\rho C_v^{tr}} \sum_{s \neq e} \frac{\rho_s}{M_s} \right) \frac{p}{\rho}}. \quad (3.30)$$

For flows without electrons $a = \sqrt{\hat{\gamma} p / \rho}$, where $\hat{\gamma}$ is the frozen ratio-of-specific-heat-capacities,

$$\hat{\gamma} = \frac{\sum_{s=1}^{N_s} Y_s C_{p,s}^{tr}}{\sum_{s=1}^{N_s} Y_s C_{v,s}^{tr}} \quad (3.31)$$

$$= \frac{\sum_{s=1}^{N_s} Y_s \frac{R_u}{M_s}}{\sum_{s=1}^{N_s} Y_s C_{v,s}^{tr}} + 1. \quad (3.32)$$

3.2 Spatial Integration

AMROC discretises the governing equations using the finite-volume method. In this method, the domain is broken down into a set of computational cells and volume integrals are applied to each cell. Taking the governing equations from Section 3.1 and integrating both sides over a volume gives

$$\int_V \frac{\partial \mathbf{q}}{\partial t} dV + \int_V \left[\frac{\partial \mathbf{f}}{\partial x} - \frac{\partial \mathbf{f}^v}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} - \frac{\partial \mathbf{g}^v}{\partial y} \right] dV = \int_V \mathbf{w} dV. \quad (3.33)$$

Using Green's theorem to convert the volume integrals of the partial derivatives into surface integrals, then rearranging leads to

$$\int_V \frac{\partial \mathbf{q}}{\partial t} dV = - \oint_{S(V)} ([\mathbf{f} - \mathbf{f}^v] dy + [\mathbf{g} - \mathbf{g}^v] dx) + \int_V \mathbf{w} dV. \quad (3.34)$$

Applying this equation to a single computational cell, c , leads to the following,

$$\frac{\partial \mathbf{Q}_c}{\partial t} = -\frac{1}{V_c} \left(\sum_{f=1}^{N_f^c} \left[(\mathbf{F} - \mathbf{F}^v)_f \hat{n}_f^x + (\mathbf{G} - \mathbf{G}^v)_f \hat{n}_f^y \right] A_f \right)_c + \mathbf{W}_c, \quad (3.35)$$

where V_c is the cell volume (area in two-dimensional simulations), N_f^c is the number of cell faces, \hat{n}_f is the outward facing unit normal of face f , and A_f is the face area (or length for two-dimensional faces). The spatial integration of the governing equations involves the evaluation of the Right Hand Side (RHS) of Eq. (3.35).

The source term \mathbf{W}_c is evaluated at each of the cell centres, whilst the flux terms are evaluated at the centre of each face, using information from either side of the face and accounting for the face geometry. Previously within AMROC, the structured, Cartesian nature of the grid resulted in the flux evaluation for the two-dimensional solver being simplified to

$$\mathbf{R}_{i,j}^f = \frac{1}{\Delta x \Delta y} \left(\Delta y \left[(\mathbf{F} - \mathbf{F}^v)_{i+1,j} - (\mathbf{F} - \mathbf{F}^v)_{i,j} \right] + \Delta x \left[(\mathbf{G} - \mathbf{G}^v)_{i+1,j} - (\mathbf{G} - \mathbf{G}^v)_{i,j} \right] \right), \quad (3.36)$$

where $\mathbf{R}_{i,j}^f$ denotes the flux residual for the cell with indices i, j , and $i \in [1, N_x]$, $j \in [1, N_y]$ and N_x and N_y are the number of domain cells in the x and y direction, respectively. The fluxes are evaluated at the cell interfaces, giving

$$\mathbf{F}_{i,j}^{(v)} = \mathbf{F}_{i-1/2,j}^{(v)}(\mathbf{Q}^l, \mathbf{Q}^r), \quad (3.37)$$

and

$$\mathbf{G}_{i,j}^{(v)} = \mathbf{G}_{i,j-1/2}^{(v)}(\mathbf{Q}^l, \mathbf{Q}^r), \quad (3.38)$$

where \mathbf{Q}^l and \mathbf{Q}^r are the conserved variables on the left and right side of the face, respectively.

In this work, the spatial integration within AMROC has been extended to allow for the use of mapped domains in the near-body region. Previously, non-Cartesian geometries have been represented in AMROC using an embedded boundary method where cells inside a surface are removed from the domain. This can lead to a “stair casing” effect, created by Cartesian cells representing non-Cartesian boundaries and inaccurate predictions of the heat loads (see Fig. 2.6). As discussed in the Section 1.1, for accurate heating assessments for hypersonic vehicles the grid must be aligned with the surface of the body and the cell must be small in the direction normal to the wall ($Re_{\Delta s} \leq 3$ [23, 39, 191, 212]). A method for two-dimensional geometry transformations has been incorporated into AMROC as part of this research. This enables body-fitted meshes with adequate cell resolutions to be used. In this method, a Cartesian computational domain, in (ξ, η) space, is mapped to a physical domain, in (x, y) space.

To ensure consistency across the domains of the overset solver, both the background SAMR solver and the strand solver use the mapped mesh implementation. The mapping for the Cartesian domain is simply $(x, y) = (\xi, \eta)$. This ensures that both solvers use the same flux functions, reducing the amount of source code to be maintained and simplifying the overset solver compilation.

Accounting for the geometry transformation, the flux evaluation has been modified to become

$$\begin{aligned} \mathbf{R}_{i,j}^f = & \frac{\Delta\eta\Delta\xi}{V_{i,j}} \left(\frac{1}{\Delta\xi} \left[\left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^v \right)_{i+1,j} - \left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^v \right)_{i,j} \right] + \right. \\ & \left. \frac{1}{\Delta\eta} \left[\left(\hat{\mathbf{G}} - \hat{\mathbf{G}}^v \right)_{i,j+1} - \left(\hat{\mathbf{G}} - \hat{\mathbf{G}}^v \right)_{i,j} \right] \right), \end{aligned} \quad (3.39)$$

where $\hat{\mathbf{F}}$, $\hat{\mathbf{F}}^v$, $\hat{\mathbf{G}}$ and $\hat{\mathbf{G}}^v$ are the face-aligned, physical fluxes per computational unit length, given by

$$\hat{\mathbf{F}}_{i,j}^{(v)} = \bar{\mathbf{F}}_{i-1/2,j}^{(v)} \frac{A_{i-1/2,j}}{\Delta\eta}, \quad (3.40)$$

and

$$\hat{\mathbf{G}}_{i,j}^{(v)} = \bar{\mathbf{G}}_{i,j-1/2}^{(v)} \frac{A_{i,j-1/2}}{\Delta\xi}. \quad (3.41)$$

The methods used to determine these fluxes is described below.

3.2.1 Inviscid Fluxes

The inviscid flux at each face is calculated by transforming the conserved variables either side of the cell face to be normal to the face. The normal inviscid flux is then calculated with the transformed values, before the flux is rotated back to align with the face in physical space. The inviscid fluxes are therefore given by

$$\bar{\mathbf{F}}_{i-1/2,j} = T_{i-1/2,j}^{-1} \mathbf{F}_n(T_{i-1/2,j} \mathbf{Q}_{i-1/2,j}^l, T_{i-1/2,j} \mathbf{Q}_{i-1/2,j}^r), \quad (3.42)$$

and

$$\bar{\mathbf{G}}_{i,j-1/2} = T_{i,j-1/2}^{-1} \mathbf{F}_n(T_{i,j-1/2} \mathbf{Q}_{i,j-1/2}^l, T_{i,j-1/2} \mathbf{Q}_{i,j-1/2}^r). \quad (3.43)$$

The matrices $T_{i-1/2,j}$ and $T_{i,j-1/2}$ transform the momentum components to be normal to the cell face and are given by

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{n}^x & \hat{n}^y & 0 & 0 \\ 0 & 0 & 0 & -\hat{n}^y & \hat{n}^x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.44)$$

where the face unit normals are either those of the left-hand face or bottom cell face for $\mathbf{F}_{i-1/2,j}$ and $\mathbf{G}_{i,j-1/2}$, respectively.

A flux function is used to calculate the normal, inviscid flux through the face, which is a function of the variables either side of the face. The normal flux is given by

$$\mathbf{F}_n(\mathbf{Q}^l, \mathbf{Q}^r) = \begin{bmatrix} \rho_1 u \\ \vdots \\ \rho_{N_s} u \\ \rho u^2 + p \\ \rho u v \\ \rho e^{ve} u \\ u(\rho E + p) \end{bmatrix}, \quad (3.45)$$

and is determined using an inviscid flux scheme or approximate Riemann solver. This idea was first proposed by Godunov in the late 1950's [100]. Consequently, there are a wide number of flux schemes that have been developed over many years and used in supersonic flow simulations. In this work, several inviscid flux schemes have been implemented: the van Leer flux vector splitting method [280], the Advection

Upstream Splitting Method (AUSM) [163] and some of its derivatives [160, 162], and the Harten-Lax-van Leer-Contact (HLLC) scheme [276].

3.2.1.1 van Leer Flux Scheme

The van Leer method uses flux splitting to find the flux across the face as

$$\mathbf{F}_n = \mathbf{F}_L^+ + \mathbf{F}_R^- . \quad (3.46)$$

The flux is calculated according to the Mach number of the flow either side of the cell face [106],

$$\mathbf{F}_L^+ = \begin{cases} \mathbf{F}_L & \text{if } M_L > 1 , \\ \mathbf{F}_{vL,L}^+ & \text{if } -1 \leq M_L \leq 1 , \\ \mathbf{0} & \text{if } M_L < -1 , \end{cases} \quad (3.47)$$

and,

$$\mathbf{F}_R^- = \begin{cases} \mathbf{F}_R & \text{if } M_R < -1 , \\ \mathbf{F}_{vL,R}^- & \text{if } -1 \leq M_R \leq 1 , \\ \mathbf{0} & \text{if } M_R > 1 , \end{cases} \quad (3.48)$$

where $\mathbf{F}_{L/R}$ is the flux given in Eq. (3.2) evaluated using the variables in the left or right cell.

The van Leer fluxes $\mathbf{F}_{L/R}^\pm$ for the two-temperature model are given as

$$\mathbf{F}_{L/R}^\pm = \pm \frac{\rho a}{4} (M \pm 1)^2 \begin{bmatrix} \rho_1 / \rho \\ \vdots \\ \rho_{N_s} / \rho \\ u - (u \mp 2a) / \hat{\gamma} \\ \rho e_{ve} / \rho \\ (\rho E + p) - \frac{(u \mp a)^2}{\hat{\gamma} + 1} \end{bmatrix}_{L/R} . \quad (3.49)$$

The van Leer method is known to introduce dissipation and is not suitable for calculating boundary layer flows [170]. This said, the positivity preserving property of the van Leer method makes it an attractive method to use for inviscid hypersonic flows with strong expansion regions, e.g. in the base region of a re-entry vehicle. For example, NASA's inviscid Cart3D solver uses the van Leer method, with the authors stating that such positivity preserving methods are likely to be the most robust [4].

3.2.1.2 Advection Upstream Splitting Method (AUSM)

The AUSM scheme was developed by Liou [163] in an attempt to combine the computational efficiency of the flux vector splitting methods with the accuracy of the flux difference splitting methods, such as the Roe scheme [230]. The limited numerical dissipation in the AUSM scheme makes it suitable for simulating the high velocity gradients encountered in a hypersonic boundary layer.

In the AUSM scheme the flux is split into two distinct parts: the convective flux, F^c , which travels at the flow speed, u , and the pressure flux, F^p , which propagates at the speed-of-sound, a . For the one-dimensional ideal gas Euler equations the flux is split as [163]

$$\mathbf{F}_n = u \begin{bmatrix} \rho \\ \rho u \\ \rho E + p \end{bmatrix} + \begin{bmatrix} 0 \\ p \\ 0 \end{bmatrix} = \mathbf{F}^c + \mathbf{F}^p. \quad (3.50)$$

At a cell interface the velocity is defined using information from both cells. In a similar way to the van Leer scheme, the contributions from the left and right cells are determined based on the flow velocities and the speed-of-sound,

$$\mathbf{F}^c = u_{1/2} \begin{bmatrix} \rho \\ \rho u \\ \rho E + p \end{bmatrix}_{L/R} = M_{1/2} a_{L/R} \begin{bmatrix} \rho \\ \rho u \\ \rho E + p \end{bmatrix}_{L/R}, \quad (3.51)$$

where the values from the left hand cell are used if $M_{1/2} \geq 0$ and the values from the right hand cells are used if $M_{1/2} < 0$. Thus, the convective flux vector can be given by,

$$\mathbf{F}^c = \frac{1}{2} (M_{1/2} + |M_{1/2}|) a_L \begin{bmatrix} \rho \\ \rho u \\ \rho E + p \end{bmatrix}_L + \frac{1}{2} (M_{1/2} - |M_{1/2}|) a_R \begin{bmatrix} \rho \\ \rho u \\ \rho E + p \end{bmatrix}_R. \quad (3.52)$$

The interface Mach number is calculated by using split Mach numbers to combine contributions from the left and right cells,

$$M_{1/2} = \mathcal{M}^+(M_L) + \mathcal{M}^-(M_R), \quad (3.53)$$

where the Mach numbers are split using the same method as in the van Leer flux scheme above. This uses a combination of waves speeds ($M \pm 1$),

$$\mathcal{M}^\pm = \begin{cases} \pm \frac{1}{4}(M \pm 1)^2 & \text{if } |M| \leq 1, \\ \frac{1}{2}(M \pm |M|) & \text{otherwise.} \end{cases} \quad (3.54)$$

The pressure flux is found in a similar way by using an interface pressure, which is calculated by splitting the pressure and taking contributions from both cells,

$$\mathbf{F}^p = p_{1/2} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad (3.55)$$

where,

$$p_{1/2} = \mathcal{P}^+(p_L, M_L) + \mathcal{P}^-(p_R, M_R). \quad (3.56)$$

Liou states that there are many ways to split the pressures using polynomial expansions of the characteristic wave speeds ($M \pm 1$), and gives two suggestions in Ref. [163]. The pressure splitting is implemented in AMROC as follows:

$$\mathcal{P}^\pm = \begin{cases} \frac{p}{4}(M \pm 1)^2(2 \mp M) & \text{if } |M| \leq 1, \\ \frac{p}{2} \frac{(M \pm |M|)}{M} & \text{otherwise.} \end{cases} \quad (3.57)$$

The AUSM method is trivially extended to the two-temperature model to give the fluxes as

$$\begin{aligned} \mathbf{F}_n^{AUSM} = & \frac{1}{2} (M_{1/2} + |M_{1/2}|) a_L \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{N_s} \\ \rho u \\ \rho e_{ve} \\ \rho E + p \end{bmatrix}_L \\ & + \frac{1}{2} (M_{1/2} - |M_{1/2}|) a_R \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{N_s} \\ \rho u \\ \rho e_{ve} \\ \rho E + p \end{bmatrix}_R + p_{1/2} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \quad (3.58)$$

A range of methods based on the AUSM method have been developed. These were primarily developed to overcome problems with carbuncles, pressure overshoots in the post-shock region and pressure oscillations in the boundary layer [161]. The first method proposed was the AUSMDV method [283], which introduces additional dissipation in regions of high pressure gradients, reducing pressure overshoots in the post-shock region. This was followed by the AUSM⁺ scheme [160], which introduced

a common interface speed-of-sound to calculate the left and right Mach numbers,

$$M_{L/R} = \frac{u_{L/R}}{a_{LR}}, \quad (3.59)$$

where a_{LR} the common interface speed-of-sound $a_{LR} = f(\mathbf{Q}_L, \mathbf{Q}_R)$. The flux equation then becomes

$$\mathbf{F}^c = \frac{1}{2} (M_{1/2} + |M_{1/2}|) a_{LR} \begin{bmatrix} \rho \\ \rho u \\ \rho E + p \end{bmatrix}_L + \frac{1}{2} (M_{1/2} - |M_{1/2}|) a_{LR} \begin{bmatrix} \rho \\ \rho u \\ \rho E + p \end{bmatrix}_R. \quad (3.60)$$

In addition to the common speed-of-sound, alternative functions to calculate the split Mach numbers, $\mathcal{M}^+(M_L)$ and $\mathcal{M}^-(M_R)$, and pressures, $\mathcal{P}^+(p_L, M_L)$ and $\mathcal{P}^-(p_R, M_R)$, were also developed as part of the AUSM⁺ scheme. These were devised to satisfy a larger number of desired properties than the original splitting was able to, and are given by

$$\mathcal{M}^\pm = \begin{cases} \pm \frac{1}{2}(M \pm 1)^2 \pm \beta(M^2 - 1)^2 & \text{if } |M| \leq 1, \\ \frac{1}{2}(M \pm |M|) & \text{otherwise,} \end{cases} \quad (3.61)$$

and

$$\mathcal{P}^\pm = \begin{cases} \frac{p}{4}(M \pm 1)^2(2 \mp M) \pm p\alpha M(M^2 - 1)^2 & \text{if } |M| \leq 1, \\ \frac{p}{2} \frac{(M \pm |M|)}{M} & \text{otherwise.} \end{cases} \quad (3.62)$$

The recommended values of β and α are $1/8$ and $3/16$, respectively [160].

The AUSM⁺ scheme was shown to give better results than the original scheme [160], but failed to entirely alleviate the problem of overshoots in pressure behind the shock and oscillations in the pressure in boundary layers. This led to further modifications to the method resulting in the AUSMPW⁺ [142] and the AUSM⁺-up [162]. Both methods are similar to the AUSM⁺ scheme but add dissipative terms to the fluxes and modify the definition of the common speed-of-sound. Both are shown to reduce the issues of pressure overshoots and pressure oscillations.

In this research, the original AUSM method, the AUSM⁺ and the AUSM⁺-up have been implemented within AMROC for the thermochemical nonequilibrium solver.

3.2.1.3 Harten-Lax-van Leer-Contact (HLLC) Scheme

The HLLC scheme is an extension by Toro of the earlier Harten-Lax-van Leer (HLL) scheme [276]. The original HLL scheme only allowed for two characteristic waves and three states, however, the Euler equations have three characteristic waves. The HLLC scheme restores the third wave, known as the contact wave, which leads to a

reduction in the numerical dissipation compared to the HLL scheme. The fluxes are derived by applying the integral form of $\partial \mathbf{Q} / \partial t + \partial \mathbf{F} / \partial x$ to a single control volume (the derivation of the scheme can be found in Ref. [276]).

In the HLLC scheme, the flux is determined by the wave speed at the cell interface,

$$\mathbf{F}_n^{HLLC} = \begin{cases} \mathbf{F}_L & \text{if } S_L \leq 0, \\ \mathbf{F}_L^* & \text{if } S_L < 0 \leq S_*, \\ \mathbf{F}_R^* & \text{if } S_* < 0 \leq S_R, \\ \mathbf{F}_R & \text{if } S_R \geq 0, \end{cases} \quad (3.63)$$

where $S_{L,R,*}$ are the interface wave speeds. The left and right fluxes are simply the inviscid flux vector, \mathbf{F} , using the left or right conserved variables, respectively. $\mathbf{F}_{L/R}^*$ are determined using the left and right fluxes and the intermediate states $\mathbf{Q}_{L/R}^*$

$$\mathbf{F}_L = \mathbf{F}(\mathbf{Q}_L), \quad (3.64)$$

$$\mathbf{F}_L^* = \mathbf{F}_L + S_L (\mathbf{Q}_L^* - \mathbf{Q}_L), \quad (3.65)$$

$$\mathbf{F}_R^* = \mathbf{F}_R + S_R (\mathbf{Q}_R^* - \mathbf{Q}_R), \quad (3.66)$$

$$\mathbf{F}_R = \mathbf{F}(\mathbf{Q}_R), \quad (3.67)$$

where the interface states are given by

$$\mathbf{Q}_{L/R}^* = \rho_{L/R} \left(\frac{S_{L/R} - u_{L/R}}{S_{L/R} - S_*} \right) \begin{bmatrix} Y_{1,L/R} \\ \vdots \\ Y_{N_s,L/R} \\ S_* \\ v_{L/R} \\ w_{L/R} \\ E_{L/R} + (S_* - u_{L/R}) \left[S_* + \frac{p_{L/R}}{\rho_{L/R}(S_{L/R} - u_{L/R})} \right] \end{bmatrix}. \quad (3.68)$$

There are various estimates for the wave speeds that can be used. In this work the Roe averaged variables are used to determine the left and right wave speeds

$$S_L = \tilde{u} - \tilde{a}, \quad S_R = \tilde{u} + \tilde{a}, \quad (3.69)$$

where

$$\tilde{u} = \frac{\sqrt{\rho_L} u_L + \sqrt{\rho_R} u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \quad \tilde{a} = \frac{\sqrt{\rho_L} a_L + \sqrt{\rho_R} a_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \quad (3.70)$$

and the contact wave speed is given by

$$S_* = \frac{p_R - p_L + \rho_L u_L (S_L - u_L) - \rho_R u_R (S_R - u_R)}{\rho_L (S_L - u_L) - \rho_R (S_R - u_R)}. \quad (3.71)$$

The small amount of numerical dissipation in the HLLC scheme makes it suitable for simulating high-enthalpy boundary layer flows. However, the HLLC method has been known to produce carbuncles when simulating high Mach number flows over blunt bodies.¹

3.2.1.4 Arbitrary Lagrangian-Euler Fluxes

Overset meshing techniques are well suited to free-body simulations, as the near-body domain can move through the background domain. With a moving near-body domain the flux calculations must account for the motion of the mesh through the fluid. As such, the flux scheme must be suitable for calculations in the Eulerian frame of reference, where the mesh is static and the fluid is moving, and in the Lagrangian frame of reference, where the mesh is moving but the free-stream velocity is zero. Both fluid and mesh motion can be present at the same time, so both frames of reference must be accounted for by using an Arbitrary Lagrangian-Euler (ALE) flux function for the inviscid fluxes.

In this work, the AUSM and HLLC methods have been extended to account for arbitrary mesh motion. The implementation of the AUSM-ALE and HLLC-ALE methods are based on Ref. [256] and Ref. [169], respectively, where both have been extended to account for the thermochemical nonequilibrium flow model. In both schemes the face velocities, u_L and u_R are modified to be the fluid velocity relative to the face velocity. This is given by

$$\hat{u}_{L/R} = u_{L/R} - s, \quad (3.72)$$

where s is the face velocity, evaluated at the same location as the flux (i.e. the face centre). An additional term must also be added to the energy equation to account for the work done on the control volume. In both cases, this is simply the interface pressure flux multiplied by the face velocity s .

3.2.1.5 High-Order Reconstruction

Extrapolation of the conserved and primitive variables to the cell faces is required in order to obtain a high order-of-accuracy in space. In AMROC, this is carried out using a second-order-accurate MUSCL (Monotonic Upstream-centered Scheme for Conservation Laws) scheme. In the MUSCL method, a scalar value \hat{s} is extrapolated to a cell face using,

$$\hat{s}_{i+1/2} = \hat{s}_i + \frac{1}{2}\Psi\left(r_{i-1/2}^+\right)\Delta_{i-1/2}, \quad (3.73)$$

¹Carbuncles are numerically induced changes to the bow shock that typically occur close to the stagnation region, leading to poor surface predictions and solver instabilities.

where \hat{s}_i is the value of the scalar at the cell centre, Ψ is the gradient limiter and $\Delta_{i-1/2} = \hat{s}_i - \hat{s}_{i-1}$ is the gradient. The gradient limiter ensures that the extrapolation is Total Variation Diminishing (TVD), where no new minima or maxima are created in the reconstructed values. There are a number of gradient limiters available within AMROC including the min-mod and van Albada, and the input to each is

$$r_{i-1/2}^+ = \frac{\Delta_{i+1/2}}{\Delta_{i-1/2}}, \quad (3.74)$$

where $\Delta_{i+1/2} = \hat{s}_{i+1} - \hat{s}_i$. The min-mod limiter is the most diffusive and is given by

$$\Psi(r_{i-1/2}^+) = \max(0, \min[1, r_{i-1/2}^+]), \quad (3.75)$$

whilst the van Albada limiter is given by

$$\Psi(r_{i-1/2}^+) = \frac{(r_{i-1/2}^+)^2 + r_{i-1/2}^+}{(r_{i-1/2}^+)^2 + 1}. \quad (3.76)$$

As the limiters in AMROC are constructed using the assumption of a uniform Cartesian mesh, the reconstruction and limiting must be performed in computational space. This requires the extrapolated variable s to be transformed to computational space using

$$\hat{s} \equiv |\mathcal{J}|s, \quad (3.77)$$

where

$$|\mathcal{J}| = \frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \frac{\partial x}{\partial \eta}. \quad (3.78)$$

The variables used in the reconstruction do not have to be the conserved variables, but can be any set of variables that can be used to define the conserved and primitive variables at the face. Often, using a combination of conserved and primitive variables results in a more robust solver, as certain physical constraints, such as positive pressure, can be enforced [62]. In this work, the species densities, x and y momentum, vibrational-electronic temperature and pressure are used in the reconstruction. Before determining the remaining variables at the face, the reconstructed variables must be transformed back to physical space, using

$$s = \frac{\hat{s}}{|\mathcal{J}|}. \quad (3.79)$$

3.2.2 Viscous Fluxes

The viscous flux at each face in the mapped grid is calculated by evaluating the viscous fluxes in the x and y direction and multiplying each vector by the unit normal

for the face. This gives the face fluxes as

$$\bar{\mathbf{F}}_{i,j}^v = (\mathbf{F}^v \hat{n}^x + \mathbf{G}^v \hat{n}^y)_{i-1/2,j} \quad (3.80)$$

and

$$\bar{\mathbf{G}}_{i,j}^v = (\mathbf{F}^v \hat{n}^x + \mathbf{G}^v \hat{n}^y)_{i,j-1/2}, \quad (3.81)$$

where \mathbf{F}^v and \mathbf{G}^v are given in Eqs. (3.3) and (3.4), respectively.

In order to calculate the viscous fluxes on the mapped grid the derivatives of the flow variables with respect to the x and y directions must be calculated at each face. To calculate the derivatives in physical space, using the variables stored on the computational mesh, one must use

$$\frac{\partial \phi}{\partial x} = \left(\frac{\partial \phi}{\partial \xi} \right) \left(\frac{\partial \xi}{\partial x} \right) + \left(\frac{\partial \phi}{\partial \eta} \right) \left(\frac{\partial \eta}{\partial x} \right) \quad (3.82)$$

and

$$\frac{\partial \phi}{\partial y} = \left(\frac{\partial \phi}{\partial \xi} \right) \left(\frac{\partial \xi}{\partial y} \right) + \left(\frac{\partial \phi}{\partial \eta} \right) \left(\frac{\partial \eta}{\partial y} \right). \quad (3.83)$$

The derivatives of the computational coordinates with respect to the physical coordinates are obtained from the mapping, either from the analytic solution (if available) or using second-order-accurate numerical derivatives. The derivatives of the flow variables on the computational domain are found using second-order-accurate central-difference formulas, resulting in an overall second-order-accurate method.

Different stencils for calculating the derivatives must be used at the $i - 1/2, j$ face and the $i, j - 1/2$ face. For a generic variable ϕ , the gradient of ϕ with respect to the computational coordinates, is evaluated at the face $i - 1/2, j$ using

$$\left(\frac{\partial \phi}{\partial \xi} \right)_{i-1/2,j} = \frac{\phi_{i,j} - \phi_{i-1,j}}{\Delta \xi} \quad (3.84)$$

and

$$\left(\frac{\partial \phi}{\partial \eta} \right)_{i-1/2,j} = \frac{1}{2} \left[\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2\Delta \eta} + \frac{\phi_{i-1,j+1} - \phi_{i-1,j-1}}{2\Delta \eta} \right]. \quad (3.85)$$

The stencil for the $i - 1/2, j$ face in both directions is shown in Fig. 3.1.

Similarly, the derivatives of ϕ with respect to computational coordinates at the face $i, j - 1/2$ are found using,

$$\left(\frac{\partial \phi}{\partial \eta} \right)_{i,j-1/2} = \frac{\phi_{i,j} - \phi_{i,j-1}}{\Delta \eta} \quad (3.86)$$

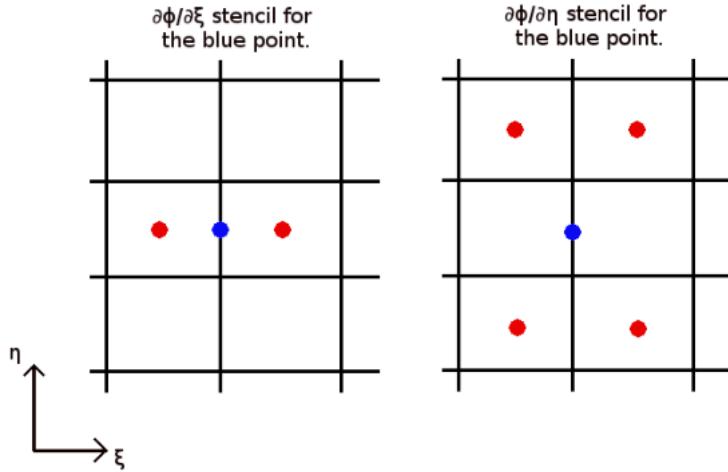


Figure 3.1: The stencil for calculating the ξ derivative (left) and η derivative (right) at the blue point (the $i - 1/2, j$ face).

and

$$\left(\frac{\partial \phi}{\partial \xi} \right)_{i,j-1/2} = \frac{1}{2} \left[\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2\Delta\xi} + \frac{\phi_{i+1,j-1} - \phi_{i-1,j-1}}{2\Delta\xi} \right]. \quad (3.87)$$

The face values for the transport properties are simply taken to be the average of the two values either side of the face. Once the transport properties and derivatives in both the x and y directions have been obtained at each face the viscous fluxes can be evaluated.

3.2.3 Boundary Conditions

AMROC uses ghost cells to define the boundary conditions at the edge of the computational domain. The values in the ghost cells are calculated so that the correct Dirichlet or von Neumann boundary condition is obtained at the boundary face. As such, the ghost cell values depend on the flux calculations used, and must account for the higher-order reconstruction of the variables to the boundary faces. All of the variables used in the flux calculations must be specified in each ghost cell so that the correct boundary condition is obtained.

Within the newly developed solver, the inviscid fluxes are computed using the conserved variables for the mass, momentum and energy equations, along with the pressure, ratio-of-specific-heat-capacities and speed-of-sound that are stored in each cell's primitive variable vector. The viscous fluxes are computed using the mass fractions and velocities that are calculated using conserved variables, along with the two temperatures which are stored in the primitive variable vector. It should be noted that the fictitious values in the ghost cells may not be, and are not required to be,

thermodynamically consistent. The only constraint is that the values produce the correct, physically realisable boundary condition.

3.2.3.1 Supersonic inflow

For a supersonic inflow, the values on the boundary are those specified by the user. Therefore, ghost cell values are simply set as the inflow variables. The conserved variables in the ghost cell, \mathbf{Q}_{gc} , are therefore given by

$$\mathbf{Q}_{gc} = \mathbf{Q}_{in} \quad (3.88)$$

and the primitive variables in the ghost cell, \mathbf{P}_{gc} , are given by

$$\mathbf{P}_{gc} = \mathbf{P}_{in}, \quad (3.89)$$

where \mathbf{Q}_{in} and \mathbf{P}_{in} are the conserved and primitive inflow variables, respectively.

3.2.3.2 Supersonic Outflow

For a supersonic outflow there are no gradients in the flow variables at the boundary. Therefore, the values in the ghost cells are simply set as the values from the domain cell at the boundary. Thus, the conserved variables are given by

$$\mathbf{Q}_{gc} = \mathbf{Q}_{nc} \quad (3.90)$$

and the primitive variables are given by

$$\mathbf{P}_{gc} = \mathbf{P}_{nc}, \quad (3.91)$$

where the subscript nc denotes the domain cell closest to the boundary, i.e. the neighbouring cell.

3.2.3.3 Symmetry/Slip Wall

For a symmetry/slip wall boundary condition, the momentum of the flow normal to the wall must be zero at the wall cell face and the momentum in the tangential direction to the wall must be unchanged. This is achieved by setting the wall-normal momentum in the ghost cells so that (1) the interpolated normal velocity at the wall cell face is zero, (2) the interpolated tangential velocity is equal to the value in the domain cell and (3) the pressure either side of the face is equal.

The boundary condition must account for both the second-order-accurate reconstruction as well as the mapping of the grid. The higher-order reconstruction is accounted for by setting the value in multiple layers of ghost cells, and using the respective layer of cells from the domain to determine the ghost cell values. In order to account for the mapping, the flow in the domain cell must be transformed to align with the boundary cell face. This is achieved applying the transformation matrix of the wall cell face to the vector of conserved variables in the domain cell,

$$\hat{\mathbf{Q}}_{dc} = T_w \mathbf{Q}_{dc}, \quad (3.92)$$

where T_w is the transformation matrix for the cell face at the boundary. This gives the wall-normal momentum in the domain cell as $\hat{\mathbf{Q}}_{dc}^{\rho u}$, and the wall-aligned momentum in the domain cell as $\hat{\mathbf{Q}}_{dc}^{\rho v}$.

The ghost cell values are then set by linearly reconstructing the momentum at the wall in order to obtain zero momentum in the wall-normal direction. In each of the ghost cells, the transformed x -momentum variable is set as

$$\hat{\mathbf{Q}}_{gc}^{\rho u} = \frac{-\frac{d_{gw}}{d_{gd}} \hat{\mathbf{Q}}_{dc}^{\rho u}}{1 - \frac{d_{gw}}{d_{gd}}}, \quad (3.93)$$

where d_{gw} and d_{gd} are the distances from the ghost cell centre to the wall face centre and the ghost cell centre to the domain cell centre, respectively. The reconstructed tangential momentum is unchanged, so the ghost cell values are simply copied from the transformed domain variables,

$$\hat{\mathbf{Q}}_{gc}^{\rho v} = \hat{\mathbf{Q}}_{dc}^{\rho v} \quad (3.94)$$

The vector of conserved variables is then transformed back using the inverse matrix,

$$\mathbf{Q}_{gc} = T_w^{-1} \hat{\mathbf{Q}}_{gc}. \quad (3.95)$$

For a symmetry boundary/slip wall, the vector of primitive variables are simply copied from the relevant domain cell,

$$\mathbf{P}_{gc} = \mathbf{P}_{dc}. \quad (3.96)$$

3.2.3.4 Adiabatic No-slip Wall

For a no-slip wall, the magnitude of the velocity at the boundary is zero. For an adiabatic no slip wall, there is no heat flux or mass diffusion into the wall either. The adiabatic wall boundary condition uses the same wall-normal velocity as the slip wall

boundary condition given in Eq. (3.93). However, the reconstructed tangential velocity at the wall must also be zero. This is achieved by setting the transformed γ -momentum in the ghost cell using

$$\hat{\mathbf{Q}}_{\text{gc}}^{\rho v} = \frac{-\frac{d_{gw}}{d_{gd}} \hat{\mathbf{Q}}_{\text{dc}}^{\rho v}}{1 - \frac{d_{gw}}{d_{gd}}} , \quad (3.97)$$

for each layer of ghost cells.

The primitive variables are again simply copied from the relevant domain cells,

$$\mathbf{P}_{\text{gc}} = \mathbf{P}_{\text{dc}} . \quad (3.98)$$

3.2.3.5 Isothermal No-slip Wall

The isothermal boundary condition is similar to the adiabatic boundary condition, except the user now specifies the temperature at the wall. The temperature at the wall is reconstructed in a similar way to the velocities, using

$$\mathbf{P}_{\text{gc}}^{T_{tr}} = \frac{\mathbf{P}_w^{T_{tr}} - \frac{d_{gw}}{d_{gd}} \mathbf{P}_{\text{dc}}^{T_{tr}}}{1 - \frac{d_{gw}}{d_{gd}}} \quad (3.99)$$

and

$$\mathbf{P}_{\text{gc}}^{T_{ve}} = \frac{\mathbf{P}_w^{T_{ve}} - \frac{d_{gw}}{d_{gd}} \mathbf{P}_{\text{dc}}^{T_{ve}}}{1 - \frac{d_{gw}}{d_{gd}}} . \quad (3.100)$$

For robustness, the minimum temperature in the ghost cell is limited to 50 K.

3.2.3.6 Specified Mass Fraction, Isothermal No-slip Wall

The surface material used for a vehicle can have an influence on the chemical reactions at the surface. Some materials promote recombination of atomic species at the surface, which can lead to large diffusive heat fluxes that must be accounted for. To simulate this, the user is able to specify the chemical composition at the boundary in the form of mass fractions. The momentum and temperature variables in the ghost cell are set in the same way as the isothermal boundary. The mass fractions must be set without influencing the inviscid fluxes and derived variables. Therefore, the density in the ghost cell is set to be the same as in the domain cell.

$$\rho_{\text{gc}} = \sum_{s=1}^{N_s} \rho_{s,\text{dc}} . \quad (3.101)$$

The mass fractions in the ghost cells are set so that the interpolated mass fractions at the wall are those specified by the user,

$$Y_{s,gc} = \frac{Y_{s,w} - \frac{d_{gw}}{d_{gd}} Y_{s,dc}}{1 - \frac{d_{gw}}{d_{gd}}} . \quad (3.102)$$

The species mass fractions in the ghost cells are limited to zero and re-normalised if necessary. The species densities are set using the density and the normalised mass fractions,

$$\hat{\mathbf{Q}}_{gc}^{\rho_s} = Y_{s,gc} \rho_{gc} . \quad (3.103)$$

3.2.3.7 Blowing Wall

The final boundary condition implemented within the solver is a blowing boundary condition where the user must specify the mass flow rate, mass fractions and temperature of the fluid blowing through the wall. This boundary condition is used to simulate ablating materials on the surface of re-entry vehicles. Both the density at the wall and the wall velocity are calculated from a specified the mass flow rate. The method used to determine the density and velocity is the same as that described in Refs. [180, 271], where the surface momentum balance (neglecting the diffusive fluxes) is solved. The surface momentum balance across the boundary face is given by

$$p_{nc} + \rho_{nc} v_{nc}^2 = p_w + \rho_w v_w^2 , \quad (3.104)$$

Substituting in the ideal gas relationship for pressure gives a quadratic equation for the wall density,

$$\rho_w^2 R_w T_{tr}^w - \rho_w [p_{nc} + \rho_{nc} v_{nc}^2] + \dot{m}^2 = 0 , \quad (3.105)$$

where T_{tr}^w and \dot{m} are the user specified temperature and mass blowing rate, respectively. R_w is the mixture gas constant, determined using the user specified mass fractions,

$$R_w = \sum_{s=1}^{N_s} Y_{s,w} R_U / M_s . \quad (3.106)$$

Once the density at the wall has been determined, the pressure at the wall is given by

$$p_w = \rho_w R_w T_{tr}^w , \quad (3.107)$$

and the velocity at the wall by

$$v_w = \dot{m} / \rho_w . \quad (3.108)$$

The pressure and momentum variables in the ghost cells are then set so that the linearly interpolated values at the wall are those calculated above,

$$\mathbf{P}_{\text{gc}}^p = \frac{\mathbf{P}_w^p - \frac{d_{gw}}{d_{gd}} \mathbf{P}_{\text{dc}}^p}{1 - \frac{d_{gw}}{d_{gd}}} \quad (3.109)$$

and

$$\hat{\mathbf{Q}}_{\text{gc}}^{\rho v} = \frac{\hat{\mathbf{Q}}_w^{\rho v} - \frac{d_{gw}}{d_{gd}} \hat{\mathbf{Q}}_{\text{dc}}^{\rho v}}{1 - \frac{d_{gw}}{d_{gd}}}. \quad (3.110)$$

3.2.4 Axisymmetric Formulation

In order to simulate axisymmetric flows using a two-dimensional domain, the axisymmetric finite-volume formulation of Yu [299] is used. The two-dimensional axisymmetric governing equations are given by

$$\frac{\partial(y\mathbf{q})}{\partial t} + \frac{\partial(y\mathbf{f})}{\partial x} - \frac{\partial(y\mathbf{f}^v)}{\partial x} + \frac{\partial(y\mathbf{g})}{\partial y} - \frac{\partial(y\mathbf{g}^v)}{\partial y} = y\mathbf{w} + y\mathbf{w}^{\text{axi}}, \quad (3.111)$$

where

$$\mathbf{w}^{\text{axi}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ (p - \tau_{\theta\theta})/y \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (3.112)$$

and

$$\tau_{\theta\theta} = 2\mu - \frac{2}{3}\mu\nabla \cdot \mathbf{u}. \quad (3.113)$$

In the source term, and within the viscous flux vectors, the divergence of the velocity vector is given by

$$\nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{1}{y} \frac{\partial(yv)}{\partial y}. \quad (3.114)$$

Again, Green's theorem is used to transform Eq. (3.111) into a volume integral. The volume of a single cell, V^{axi} , is calculated by integrating around an axis of revolution to give

$$V_{i,j}^{\text{axi}} = 2\pi\hat{y} V_{i,j}, \quad (3.115)$$

where \hat{y} is the cell centroid y coordinate. Similarly, the area of a face becomes the area of a surface of revolution and is given by

$$A_f^{\text{axi}} = 2\pi\bar{y} A_f, \quad (3.116)$$

where \bar{y} is the y coordinate of the centre of the face. Using these definitions and assuming that the cell y -coordinates are constant, the finite-volume equation for each cell is given by

$$\frac{\partial \mathbf{Q}_{i,j}}{\partial t} = -\frac{1}{\hat{y}_{i,j} V_{i,j}} \sum_{f=1}^{N_f} \left[(\mathbf{F} - \mathbf{F}^v)_f \hat{n}_f^x + (\mathbf{G} - \mathbf{G}^v)_f \hat{n}_f^y \right] \bar{y}_f A_f + \mathbf{W}_{i,j} + \mathbf{W}_{i,j}^{\text{axi}}. \quad (3.117)$$

To change from a two-dimensional simulation to an axisymmetric simulation, Eqs. (3.39), (3.40) and (3.41) are modified to give the required flux residual. To do so, one simply multiplies the face area, A_f , by the corresponding face centre y -coordinate, \bar{y}_f , and the cell area, $V_{i,j}$, by the cell centroid y -coordinate, $\hat{y}_{i,j}$. In addition to this, the axisymmetric source term must be added and the divergence terms in the viscous flux vectors must be changed to the radial coordinate version given in Eq. (3.114).

The velocity divergence used in the axisymmetric source term requires the velocity gradients to be evaluated. The source term is calculated at the cell centres so the gradients used must be found at the cell centres on the mapped mesh. On a mapped mesh, the derivative of a variable ϕ is given by

$$\begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = \mathcal{J}^{-1} \begin{bmatrix} \frac{\partial \phi}{\partial \xi} \\ \frac{\partial \phi}{\partial \eta} \end{bmatrix}, \quad (3.118)$$

where the mapping Jacobian is given by

$$\mathcal{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \quad (3.119)$$

resulting in

$$\begin{bmatrix} \frac{\partial \phi}{\partial x} \\ \frac{\partial \phi}{\partial y} \end{bmatrix} = \frac{1}{\frac{\partial x}{\partial \xi} \frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi} \frac{\partial x}{\partial \eta}} \begin{bmatrix} \frac{\partial y}{\partial \eta} \frac{\partial \phi}{\partial \xi} - \frac{\partial y}{\partial \xi} \frac{\partial \phi}{\partial \eta} \\ -\frac{\partial x}{\partial \eta} \frac{\partial \phi}{\partial \xi} + \frac{\partial x}{\partial \xi} \frac{\partial \phi}{\partial \eta} \end{bmatrix}. \quad (3.120)$$

The derivatives of the spatial coordinates and flow variables w.r.t. computational space can be calculated to second-order-accuracy using a four point stencil (see Fig. 3.2).

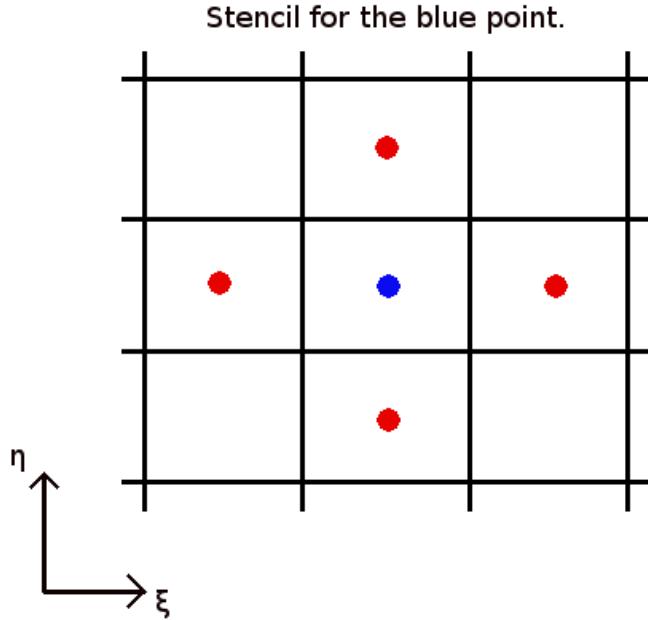


Figure 3.2: The stencil for calculating the derivatives at the centre of a cell on a mapped mesh.

3.2.5 Method of Manufactured Solutions

The Method of Manufactured Solutions (MMS) [229] was used to verify the implementation of the spatial integration on mapped meshes. The MMS process is a way of rigorously determining the order-of-accuracy of a numerical method. If the observed order-of-accuracy is approximately equal to the theoretical order-of-accuracy, then this provides evidence that the method has been implemented correctly. The process of using MMS to verify the spatial integration can be summarised by the following steps:

1. Determine spatially varying, analytic functions of the conserved variables, to give a set of “manufactured solutions” for the governing equations.
2. Substitute the manufactured solutions into the governing equations and apply the differential operators to obtain an analytic expressions for the variation of the solutions in space. The functions can be substituted either into the strong form of the governing equation, Eq. (3.1), or the weak form, Eq. (3.34)
3. Incorporate the analytic expressions into the solver as either a source term (when using the strong form of the governing equations) or as additional fluxes at each face (when using the weak form of the governing equations).
4. Run the modified solver on three uniformly refined meshes, until each simulation has converged.

5. Determine the error in the solution on each mesh by comparing the final conserved variable fields with the analytic functions determined in Step 1.
6. Calculate the observed order-of-accuracy using the error from each mesh and the mesh refinement factor.

The user has some freedom over the form of the function used for the manufactured solutions. The main limitations are that the functions should give physical solutions, e.g. the pressure must be positive, and that the functions must give a non-trivial solution when the differential operators are applied i.e. one must be able to differentiate the functions the number of times required by the governing equations. These constraints mean that exponential and trigonometric functions are often used for the manufactured solutions.

3.2.5.1 Scalar Diffusion

The first MMS test was conducted using a simple passive scalar test case to verify the implementation of the diffusive fluxes on the mapped mesh. The governing equation for the passive scalar was given by

$$\nabla \cdot (\kappa \nabla c) = 0, \quad (3.121)$$

which, in two-dimensions, gives

$$\frac{\partial}{\partial x} \left(\kappa \frac{\partial c}{\partial x} \right) + \frac{\partial}{\partial y} \left(\kappa \frac{\partial c}{\partial y} \right) = 0. \quad (3.122)$$

This is the steady-state diffusion equation and utilises the same methods as the species diffusion and thermal diffusion terms in the governing equations. This test case was used as a simpler test case than the full governing equations, that is still able to verify the diffusive flux calculations within the solver.

Solutions were created for the scalar concentration, c and the scalar diffusion coefficient, κ , as

$$c = c_0 + c_x \sin(a_x^c x \pi) + c_y \cos(a_y^c y \pi), \quad (3.123)$$

$$\kappa = \kappa_0 + \kappa_x \sin(a_x^\kappa x \pi) + \kappa_y \cos(a_y^\kappa y \pi), \quad (3.124)$$

where the coefficients are given in Table 3.1. Applying Eq. (3.121) to the analytic functions gives

$$\begin{aligned} \nabla \cdot (\kappa \nabla c) = & a_y^c a_y^\kappa c_y \kappa_y \pi^2 \sin(a_y^c \pi y) \sin(a_y^\kappa \pi y) - \\ & (a_y^c)^2 c_y \pi^2 \cos(a_y^c \pi y) \left(\kappa_0 + \kappa_x \sin(a_x^\kappa \pi x) + \kappa_y \cos(a_y^\kappa \pi y) \right) - \\ & (a_x^c)^2 c_x \pi^2 \sin(a_x^c \pi x) \left(\kappa_0 + \kappa_x \sin(a_x^\kappa \pi x) + \kappa_y \cos(a_y^\kappa \pi y) \right) \\ & + a_x^c a_x^\kappa c_x \kappa_x \pi^2 \cos(a_x^c \pi x) \cos(a_x^\kappa \pi x). \end{aligned} \quad (3.125)$$

This function was implemented as a source term for the passive scalar equation within AMROC. It can be seen that even a simple governing equation can give a long and complex source term. To avoid errors in the derivation of the source term, the computational algebra software Maxima was used.

Variable (ϕ)	ϕ_0	ϕ_x	a_x^ϕ	ϕ_y	a_y^ϕ
c	100	20	1.75	17.5	2.5
κ	0.05	0.0025	1.25	0.005	0.75

Table 3.1: The constants used in the diffusion MMS test case.

Two mappings were used for the mesh, a Cartesian mapping and a linear mapping given by

$$x(\xi, \eta) = \xi + \eta \tan(\phi) - b, \quad (3.126)$$

$$y(\xi, \eta) = 2(\eta + \xi \tan(\phi)), \quad (3.127)$$

where, $\xi \in [0, 1]$, $\eta \in [0, 1]$, $b = 0.5$ and $\phi = 15^\circ$. The simulations were run on three uniformly refined grids of 25×25 , 49×49 and 97×97 and each simulation was run until the scalar residual converged. Figure 3.3 shows the scalar variable field for the three meshes on the mapped domain.

The results from the tests cases are shown in Tables 3.2 and 3.3, and the observed error convergence is shown relative to theoretical first- and second-order-accuracy errors in Fig. 3.4. The results show that the errors on both the Cartesian and mapped domain are converging with approximately second-order-accuracy. These test results provide strong evidence that the gradient construction and diffusive flux computations on the mapped mesh have been implemented correctly.

Mesh Dimension	c Error	Observed Order
25x25	4.615×10^{-2}	N/A
49x49	1.144×10^{-2}	2.01
97x97	2.864×10^{-3}	2.00

Table 3.2: Results from the diffusion MMS case on the Cartesian mesh.

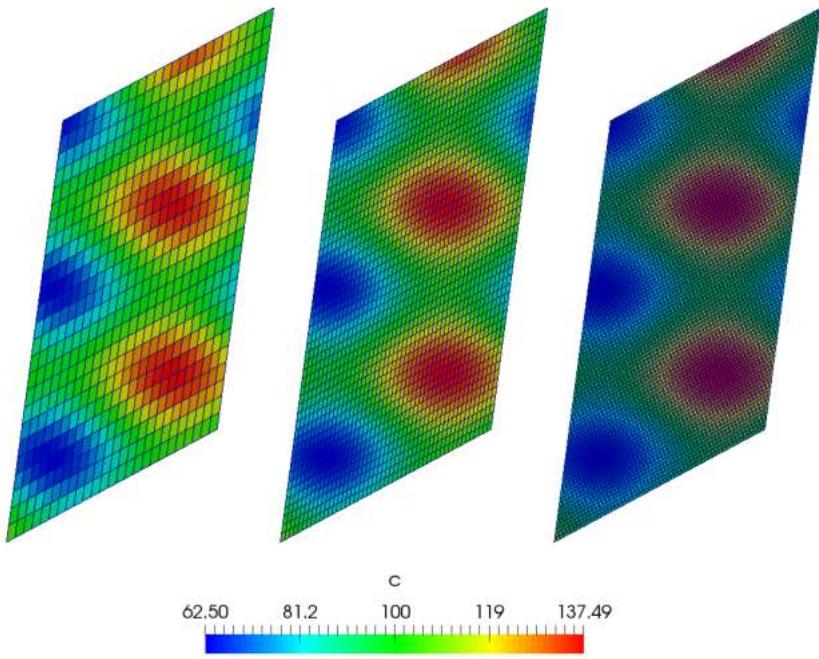


Figure 3.3: The concentration, c , through the domains in the diffusion MMS case on the mapped mesh.

Mesh Dimension	c Error	Observed Order
25x25	9.103×10^{-2}	N/A
49x49	2.432×10^{-2}	1.91
97x97	6.261×10^{-3}	1.96

Table 3.3: Results from the diffusion MMS case on the mapped mesh.

3.2.5.2 Euler and Navier-Stokes Equations

Manufactured solutions were created for the variables ρ_{O_2} , ρ_N , u , v , $e_{O_2}^{ve}$, e_N^{ve} , T_{tr} and μ . The solutions took the form:

$$\phi = \phi_0 + \phi_x f_x(a_\phi^x \pi x) + \phi_y f_y(a_\phi^y \pi y), \quad (3.128)$$

where the functions f_x and f_y were either sin or cos. To test all aspects of the convective and viscous fluxes, three different sets of constants were used to test the supersonic Euler, subsonic Euler and subsonic Navier-Stokes equations. The inviscid tests were conducted with both positive and negative velocities in order to test the different branches of the flux schemes. In the Navier-Stokes case, the transport coefficients were chosen to give viscous fluxes of similar magnitude to the inviscid fluxes. The verification of the viscous fluxes is not dependent on the inviscid flux scheme, so the Navier-Stokes test case was only run with the AUSM flux scheme. The constants and functions used in the solutions are detailed in Tables A.5 to A.7 in

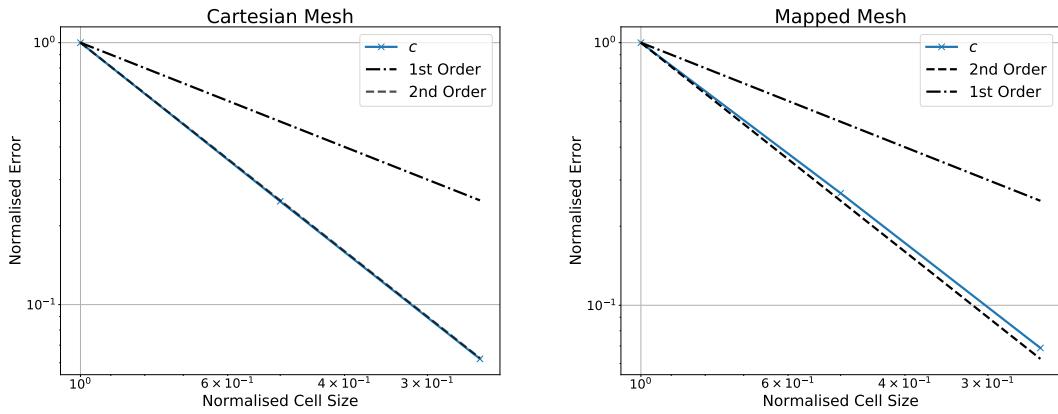


Figure 3.4: The normalised errors for the Cartesian (left) and mapped (right) MMS diffusion test case, compared with the theoretical first- and second-order-accuracy errors.

Appendix A.2. To the authors knowledge this is the first time that MMS has been used for a thermochemical nonequilibrium model. These results were published in Ref. [13], and, more recently, similar methods were the focus of a paper published in the Journal of Computational Physics by other authors [86].

The solutions were substituted into the governing equations in order to create the required analytic source terms. The differentiation was carried out using the computer algebra system Maxima. An analytic mapping was used that was designed to test the mapped reconstruction and flux calculations on a nonlinearly mapped grid,

$$x = \xi + 0.125 \sin(0.75 \pi \eta) \quad (3.129)$$

$$y = \eta + 0.075 \sin(1.25 \pi \xi) . \quad (3.130)$$

The mesh created with this mapping, along with an example density field given by the manufactured solution, is shown in Fig. 3.5.

For each set of solution constants, the simulations were run on three uniformly refined grids. The solutions were run until the density residual converged, which was at a L_2 norm of less than 1×10^{-11} for some of tests on the coarsest grids but significantly lower for the majority of cases. The convergence of the density residual for the inviscid subsonic and supersonic HLLC cases is shown in Fig. 3.6.

Some example error convergence plots for the different cases are shown in Fig. 3.7 and the full results for the different tests are given in Appendix A.2. One can see that the observed order of accuracy is approximately equal to two for all flow regimes, and all flux schemes. The MMS results provide strong evidence that the spatial integration of the fluxes has been implemented correctly. In addition, because the solutions were specified with primitive variables, the tests confirm that the equation-of-state within Mutation++ is implemented as expected.

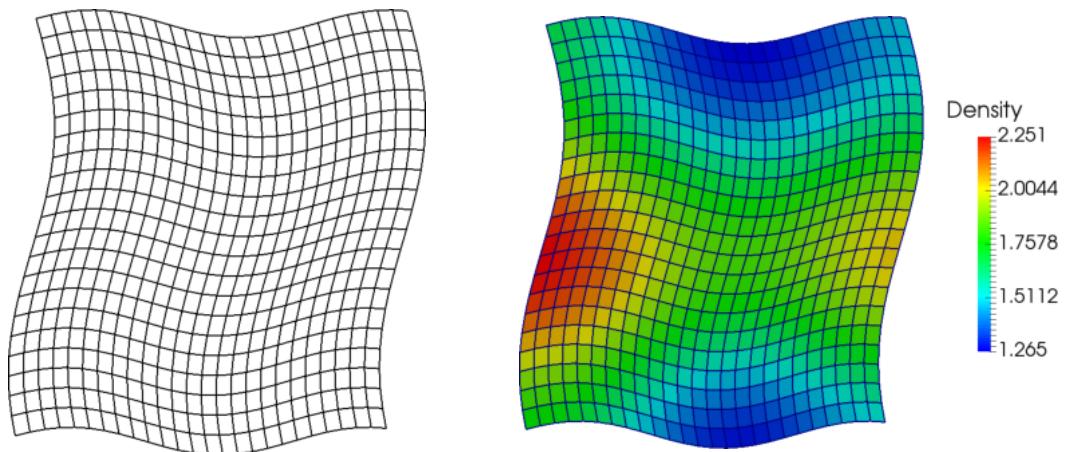


Figure 3.5: The sinusoidal mapping and an example density field for the Euler and Navier-Stokes MMS test cases.

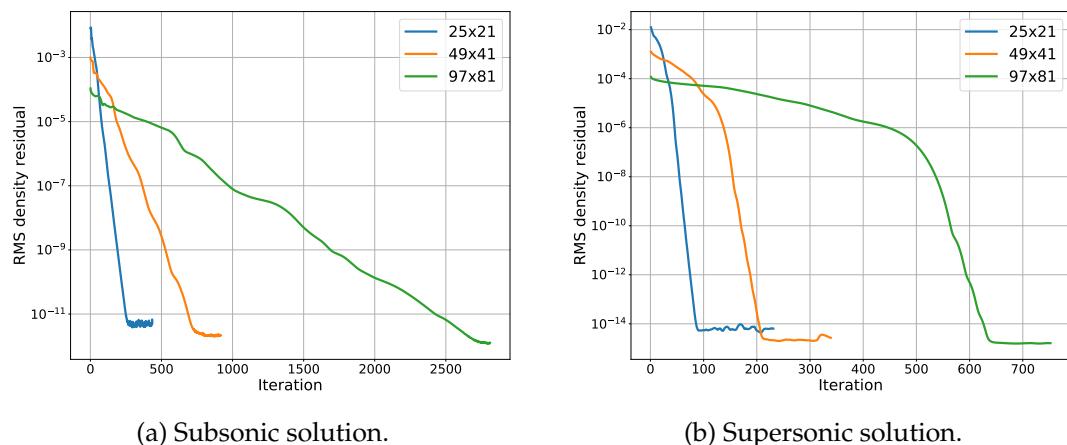


Figure 3.6: The convergence of the density residuals in the subsonic and supersonic MMS test cases for the HLLC flux scheme.

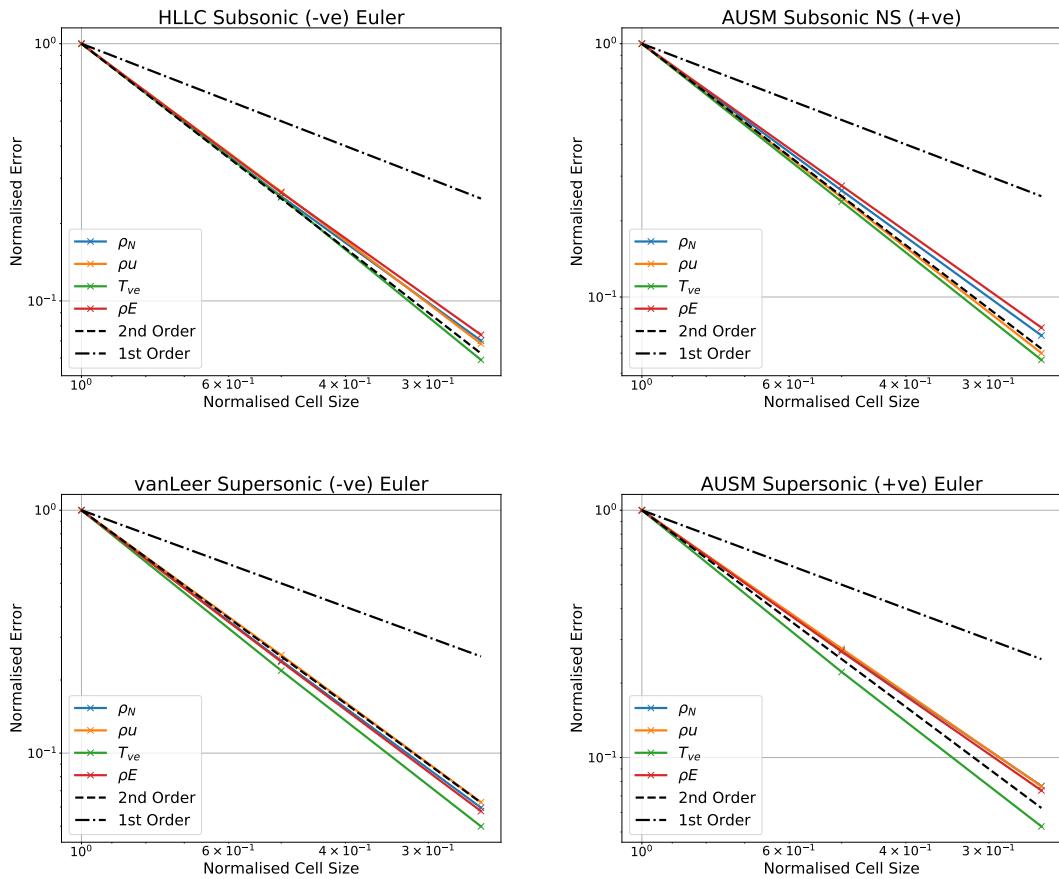


Figure 3.7: The normalised error convergence for different MMS cases, compared with the theoretical convergence for first- and second-order-accuracy.

3.3 Time Integration

In AMROC, time integration is either used to advance the solution to steady-state or to determine how the solution varies over time. Starting from Eq. (3.34), both sides of the equation are integrated in time to find the conserved variable update,

$$\int_{t_s} \int_V \frac{\partial \mathbf{q}}{\partial t} dV dt = - \int_{t_s} \left[\oint_{S(V)} ([\mathbf{f} - \mathbf{f}^v] dy + [\mathbf{g} - \mathbf{g}^v] dx) + \int_V \mathbf{w} dV \right] dt. \quad (3.131)$$

There are a number of ways that the time integration can be carried out depending on the temporal accuracy required and the numerical stiffness of the equations. An explicit, second-order-accurate time integration method was previously implemented in AMROC and is still used in the off-body CAMR region. However, in this work, implicit time integration methods have been implemented for use on the uniform near-body region. Before discussing the new methods that have been introduced in this work, the existing time integration methods are described.

3.3.1 AMROC Time Integration

Using the finite-volume method, described in Section 3.2, Eq. (3.131) can be rewritten as

$$\int_{t_s} \frac{\partial \mathbf{Q}_{i,j}}{\partial t} dt = - \int_{t_s} \mathbf{R}_{i,j} dt, \quad (3.132)$$

where the residual for each cell, $\mathbf{R}_{i,j}$, in the Cartesian framework of AMROC, is given by

$$\begin{aligned} \mathbf{R}_{i,j}(\mathcal{Q}) = & \frac{1}{\Delta x \Delta y} \left(\Delta y \left[(\mathbf{F} - \mathbf{F}^v)_{i+1,j} - (\mathbf{F} - \mathbf{F}^v)_{i,j} \right] + \right. \\ & \left. \Delta x \left[(\mathbf{G} - \mathbf{G}^v)_{i+1,j} - (\mathbf{G} - \mathbf{G}^v)_{i,j} \right] \right) - \mathbf{W}_{i,j}. \end{aligned} \quad (3.133)$$

The variable \mathcal{Q} denotes a vector containing the conserved variables in all cells in the domain. This is because the flux and source terms for cell i, j can be functions of a number of different cells' conserved variables.

The residual contributions for each cell can be partitioned, with one partition for each spatial dimension, and one for the source term,

$$\mathbf{R}_{i,j}(\mathcal{Q}) = \mathbf{R}_{i,j}^x(\mathcal{Q}) + \mathbf{R}_{i,j}^y(\mathcal{Q}) + \mathbf{R}_{i,j}^S(\mathcal{Q}), \quad (3.134)$$

where,

$$\mathbf{R}_{i,j}^x(\mathcal{Q}) = \frac{1}{\Delta x} \left[(\mathbf{F} - \mathbf{F}^v)_{i+1,j} - (\mathbf{F} - \mathbf{F}^v)_{i,j} \right], \quad (3.135)$$

$$\mathbf{R}_{i,j}^y(\mathcal{Q}) = \frac{1}{\Delta y} \left[(\mathbf{G} - \mathbf{G}^v)_{i+1,j} - (\mathbf{G} - \mathbf{G}^v)_{i,j} \right], \quad (3.136)$$

$$\mathbf{R}_{i,j}^S(\mathcal{Q}) = \mathbf{W}_{i,j}. \quad (3.137)$$

For the mapped spatial integration, there are analogous flux residuals in the computational ξ and η directions,

$$\mathbf{R}_{i,j}^\xi(\mathcal{Q}) = \frac{\Delta\eta}{V_{i,j}} \left[(\hat{\mathbf{F}} - \hat{\mathbf{F}}^v)_{i+1,j} - (\hat{\mathbf{F}} - \hat{\mathbf{F}}^v)_{i,j} \right], \quad (3.138)$$

$$\mathbf{R}_{i,j}^\eta(\mathcal{Q}) = \frac{\Delta\xi}{V_{i,j}} \left[(\hat{\mathbf{G}} - \hat{\mathbf{G}}^v)_{i+1,j} - (\hat{\mathbf{G}} - \hat{\mathbf{G}}^v)_{i,j} \right]. \quad (3.139)$$

The three partitions above can be integrated in time together or consecutively, where the updated conserved variables calculated in the integration of each partition are then used in the next. This is known as operator splitting and was previously implemented within AMROC. For example, using first-order-accurate splitting to integrate the source term separately gives

$$\mathbf{Q}_{i,j}^* = \mathbf{Q}_{i,j}^n - \int_{t_n}^{t_n + \Delta t} \mathbf{R}_{i,j}^\xi(\mathcal{Q}^n) + \mathbf{R}_{i,j}^\eta(\mathcal{Q}^n) dt, \quad (3.140)$$

$$\mathbf{Q}_{i,j}^{n+1} = \mathbf{Q}_{i,j}^* - \int_{t_n}^{t_n + \Delta t} \mathbf{R}_{i,j}^S(\mathcal{Q}^*) dt. \quad (3.141)$$

It should be noted that different time integration techniques can be used for the different partitions. For example, multi-step and implicit Ordinary Differential Equation (ODE) integrators have been implemented within AMROC for the time integration of the stiff source terms that can often arise in reacting flows.

Second-order-accurate splitting in time, known as Strang splitting had also been implemented in AMROC. In this method the source term is applied before and after

the flux update, using half time-steps,

$$\mathbf{Q}_{i,j}^* = \mathbf{Q}_{i,j}^n - \int_{t_n}^{t_n + \Delta t / 2} \mathbf{R}_{i,j}^S(\mathcal{Q}^n) dt, \quad (3.142)$$

$$\mathbf{Q}_{i,j}^{**} = \mathbf{Q}_{i,j}^* - \int_{t_n}^{t_n + \Delta t} \mathbf{R}_{i,j}^\xi(\mathcal{Q}^*) + \mathbf{R}_{i,j}^\eta(\mathcal{Q}^*) dt, \quad (3.143)$$

$$\mathbf{Q}_{i,j}^{n+1} = \mathbf{Q}_{i,j}^{**} - \int_{t_n + \Delta t / 2}^{t_n + \Delta t} \mathbf{R}_{i,j}^S(\mathcal{Q}^{**}) dt. \quad (3.144)$$

Dimensional operator splitting has also been implemented within AMROC, where the residual in the x and y (or ξ and η) direction are integrated separately. Combined with a first-order-accurate source term splitting this gives the update as

$$\mathbf{Q}_{i,j}^* = \mathbf{Q}_{i,j}^n - \int_{t_n}^{t_n + \Delta t} \mathbf{R}_{i,j}^\xi(\mathcal{Q}^n) dt, \quad (3.145)$$

$$\mathbf{Q}_{i,j}^{**} = \mathbf{Q}_{i,j}^* - \int_{t_n}^{t_n + \Delta t} \mathbf{R}_{i,j}^\eta(\mathcal{Q}^*) dt, \quad (3.146)$$

$$\mathbf{Q}_{i,j}^{n+1} = \mathbf{Q}_{i,j}^{**} - \int_{t_n}^{t_n + \Delta t} \mathbf{R}_{i,j}^S(\mathcal{Q}^{**}) dt. \quad (3.147)$$

One of the benefits of splitting the flux in this way is that the stable time step can be increased for explicit, viscous computations.

Within AMROC, the time integration of the flux residuals was carried out using explicit methods. When using explicit methods, a maximum time step is imposed on the integration, above which the simulation becomes unstable. The maximum stable time step that can be taken is determined using the Courant-Friedrichs-Lowy (CFL) condition.

3.3.1.1 Courant-Friedrichs-Lowy Condition

For an explicit simulation on an unstructured finite-volume mesh the CFL condition can be written as [194]

$$\sum_f \left[\frac{\lambda_f^v A_f}{d_f} + \lambda_f^c A_f \right] - \frac{V_c}{\Delta t} \leq 0, \quad (3.148)$$

where λ_f^v and λ_f^c are the viscous and convective spectral radii, respectively, and d_f is the distance between the cell centres either side of the face. In this work, λ_f^c is given by

$$\lambda_f^c = |\mathbf{u} \cdot \hat{\mathbf{n}}_f| + a_f, \quad (3.149)$$

and λ_f^v is given by

$$\lambda_f^v = \max\left(\frac{4}{3} \left|\frac{\mu}{\rho}\right|_f, \left|\frac{\mu}{\rho c_v}\right|_f, D_{s,f}^{\max}\right). \quad (3.150)$$

Rearranging Eq. (3.148) gives

$$\frac{\Delta t}{V_c} \sum_f \left[\frac{\lambda_f^v}{d_f} + \lambda_f^c \right] A_f \leq 1. \quad (3.151)$$

When dimensional splitting is used within AMROC, the CFL condition must be evaluated in each dimension separately, giving,

$$\max \left(\left[\frac{\lambda_{i-1/2,j}^v}{d_{i-1/2,j}} + \lambda_{i-1/2,j}^c \right] A_{i-1/2,j} + \left[\frac{\lambda_{i+1/2,j}^v}{d_{i+1/2,j}} + \lambda_{i+1/2,j}^c \right] A_{i+1/2,j}, \left[\frac{\lambda_{i,j-1/2}^v}{d_{i,j-1/2}} + \lambda_{i,j-1/2}^c \right] A_{i,j-1/2} + \left[\frac{\lambda_{i,j+1/2}^v}{d_{i,j+1/2}} + \lambda_{i,j+1/2}^c \right] A_{i,j+1/2} \right) \frac{\Delta t}{V_{i,j}} \leq 1. \quad (3.152)$$

for an explicit method.

The relationship between the maximum time step and the cell volume, V_c , and cell-centre-to-cell-centre distance, d_f , means that the size of the cell has a large impact on the maximum stable time step. For the near-wall cells, d_f is equal to the wall cell spacing, Δs_0 and, as discussed previously, Δs_0 is often very small in hypersonic simulations where it is recommended that $\text{Re}_{\Delta s} \leq 3$ [23, 39, 191, 212]. The near-wall cells can also be highly stretched with d_f and A_f being orders of magnitude different in the wall-normal and wall-aligned directions. This can lead to large differences in the stable time step in the two directions.

3.3.2 Implicit Time Integration

The high-aspect-ratio cells close to the wall place a severe restriction on the maximum stable time step when using explicit time integration methods. Implicit methods can be used to increase the maximum stable time step and therefore reduce the number of iterations and the computational time required to obtain solutions. Implicit methods utilise the current state and future state(s) to relax the stability criteria that explicit methods are subject to. The stable CFL number for implicit methods can often be far greater than one.

Any or all of the partitions in the operator splitting methods can be integrated using an implicit method. In this work, only the fluxes are integrated implicitly, with the source term integrated using one of the ODE methods already implemented within AMROC. This is because it is the fluxes that impose the restriction on the time step when using explicit methods. As discussed above, the flux integration can be split into two partitions based on the directions of the mesh, \mathbf{R}^ξ and \mathbf{R}^η , and different time integration methods can be used for each direction. Therefore, implicit methods can be used in one direction and explicit methods in the other, leading to IMEX methods. By only integrating implicitly in the wall-normal direction large time steps can still be used, with increased efficiency relative to fully implicit methods. IMEX methods have been implemented and investigated as part of this work.

Considering the entire computational domain as a single system, the ODE to be integrated implicitly at each time step is given by

$$\frac{\partial \mathcal{Q}}{\partial t} = \mathbf{f}(\mathcal{Q}), \quad (3.153)$$

where $\mathbf{f}(\mathcal{Q})$ is a vector containing the flux residuals from all of the cells in the domain,

$$\mathbf{f}(\mathcal{Q}) = \begin{bmatrix} -\left(\mathbf{R}_{1,1}^F(\mathcal{Q})\right) \\ \vdots \\ -\left(\mathbf{R}_{N_\xi, N_\eta}^F(\mathcal{Q})\right) \end{bmatrix}. \quad (3.154)$$

For an unpartitioned flux evaluation the cell flux residual, $\mathbf{R}_{i,j}^F$, is given by

$$\mathbf{R}_{i,j}^F = \mathbf{R}_{i,j}^\xi(\mathcal{Q}) + \mathbf{R}_{i,j}^\eta(\mathcal{Q}), \quad (3.155)$$

and for a partitioned flux evaluation $\mathbf{R}_{i,j}^F = \mathbf{R}_{i,j}^\xi$ or $\mathbf{R}_{i,j}^F = \mathbf{R}_{i,j}^\eta$ depending on the partition being integrated.

In this work, the first-order-accurate backwards Euler method has been implemented where the conserved variable update is given by

$$\frac{\mathcal{Q}^{n+1} - \mathcal{Q}^n}{\Delta t} = \mathbf{f}(\mathcal{Q}^{n+1}), \quad (3.156)$$

and $\mathbf{f}(\mathcal{Q}^{n+1})$ is approximated using a first-order-accurate Taylor expansion,

$$\mathbf{f}(\mathcal{Q}^{n+1}) = \mathbf{f}(\mathcal{Q}^n) + \frac{\partial \mathbf{f}(\mathcal{Q}^n)}{\partial \mathcal{Q}} \Delta \mathcal{Q}. \quad (3.157)$$

Substituting Eq. (3.157) into Eq. (3.156) and re-arranging gives

$$\left[\frac{I}{\Delta t} - \frac{\partial \mathbf{f}(\mathcal{Q}^n)}{\partial \mathcal{Q}} \right] \Delta \mathcal{Q} = \mathbf{f}(\mathcal{Q}^n), \quad (3.158)$$

which can be solved to find the conserved variable update.

Finding the solution of Eq. (3.158) requires the construction and solution of a linear system of equations. The methods implemented and used in this research are discussed below.

3.3.2.1 Linear System Construction

The update equation, Eq. (3.158), leads to a set of simultaneous equations which must be solved to find the update in each cell. The update equation can therefore be written as a linear system of equations,

$$A\mathbf{x} = \mathbf{b}. \quad (3.159)$$

When creating the linear system the two-dimensional cell indices are mapped to a one-dimensional index, $(i, j) \rightarrow k$, where $k \in [1, N_c]$, and N_c is the number of cells in the grid. This means that each cell is represented by a row of the linear system.

The solution vector, \mathbf{x} , contains the update to the vector of conserved variables for each cell,

$$\mathbf{x} = \begin{bmatrix} \Delta \mathbf{Q}_1^n \\ \Delta \mathbf{Q}_2^n \\ \vdots \\ \Delta \mathbf{Q}_{N_c-1}^n \\ \Delta \mathbf{Q}_{N_c}^n \end{bmatrix}, \quad (3.160)$$

and the RHS vector contains the flux residuals for each cell,

$$\mathbf{b} = \begin{bmatrix} -\mathbf{R}_1^F(\mathcal{Q}) \\ \vdots \\ -\mathbf{R}_{N_c}^F(\mathcal{Q}) \end{bmatrix}. \quad (3.161)$$

The A matrix contains the terms in the square brackets on the Left Hand Side (LHS) of Eq. (3.158). This includes the flux Jacobian, which is the derivative of the flux residuals w.r.t. the \mathcal{Q} vector. This is a large square matrix consisting of $N_c \times N_c$ smaller matrices, each of dimension $N_e \times N_e$, where N_e is the number of conserved variables,

$$\frac{\partial \mathbf{f}}{\partial \mathcal{Q}} = - \begin{bmatrix} \frac{\partial \mathbf{R}_1^F}{\partial \mathcal{Q}_1} & \cdots & \frac{\partial \mathbf{R}_1^F}{\partial \mathcal{Q}_{N_c}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{R}_{N_c}^F}{\partial \mathcal{Q}_1} & \cdots & \frac{\partial \mathbf{R}_{N_c}^F}{\partial \mathcal{Q}_{N_c}} \end{bmatrix}. \quad (3.162)$$

The conserved values \mathcal{Q} used to calculate the flux residual for each cell, \mathbf{R}_c^F , depends on the flux reconstruction method used. For higher-order methods, the flux residual

can use information from a large number of surrounding cells. Therefore, the flux Jacobian may include derivatives w.r.t. a large number of different cells' conserved variables. However, including all of these cells can make the flux Jacobian difficult to construct. It has also been shown that steady-state solvers converge fastest when a first-order approximation of the inviscid fluxes is used to form the Jacobian, even when second-order-accurate reconstruction is used (for example, Ref. [136]). In addition, it can be assumed that the viscous fluxes are primarily a function of the variables in the cells either side of a face. Therefore, the residuals used to create the Jacobian matrices depend only on the cell and its closest neighbours,

$$\mathbf{R}_{i,j}^F = \left[\mathbf{R}_{i,j}^\xi(\mathbf{Q}_{i-1,j}, \mathbf{Q}_{i,j}, \mathbf{Q}_{i+1,j}) + \mathbf{R}_{i,j}^\eta(\mathbf{Q}_{i,j-1}, \mathbf{Q}_{i,j}, \mathbf{Q}_{i,j+1}) \right] \quad (3.163)$$

Using the above simplification each row of the Jacobian contains five matrices with non-zero entries when using unpartitioned time integration, or three when using separate partitions for each mesh direction. This, combined with the structured nature of AMROC, means that the Jacobian has a block penta- or tri-diagonal structure with the entries on the main tri-diagonal being dependent on the mapping $(i, j) \rightarrow k$. For example, using unpartitioned integration and choosing the mapping $(i, j) \rightarrow k$ so that the i values are continuous, $k = i + N_\xi(j - 1)$, gives each row of the Jacobian as a block penta-diagonal system,

$$\begin{aligned} \frac{\partial \mathbf{R}_{i,j}^F}{\partial \mathbf{Q}} = & \left[\mathbf{0}, \dots, \mathbf{0}, \frac{\partial \mathbf{R}_{i,j}^\xi}{\partial \mathbf{Q}_{i-1,j}}, \frac{\partial \mathbf{R}_{i,j}^\xi}{\partial \mathbf{Q}_{i,j}}, \frac{\partial \mathbf{R}_{i,j}^\xi}{\partial \mathbf{Q}_{i+1,j}}, \mathbf{0}, \dots, \mathbf{0} \right] + \\ & \left[\mathbf{0}, \dots, \mathbf{0}, \frac{\partial \mathbf{R}_{i,j}^\eta}{\partial \mathbf{Q}_{i,j-1}}, \mathbf{0}, \dots, \mathbf{0}, \frac{\partial \mathbf{R}_{i,j}^\eta}{\partial \mathbf{Q}_{i,j}}, \mathbf{0}, \dots, \mathbf{0}, \frac{\partial \mathbf{R}_{i,j}^\eta}{\partial \mathbf{Q}_{i,j+1}}, \mathbf{0}, \dots, \mathbf{0} \right]. \end{aligned} \quad (3.164)$$

Modifying the mapping such that the j values are contiguous in memory, $k = j + N_\eta(i - 1)$, would mean that the $j \pm 1$ Jacobians would form the central tri-diagonal instead. The only other terms that contribute to the LHS matrix A are scalars on the diagonal. As such, the LHS matrix always has a tri- or penta-diagonal form and this is graphically depicted for the mapping $k = j + N_\eta(i - 1)$ in Fig. 3.8.

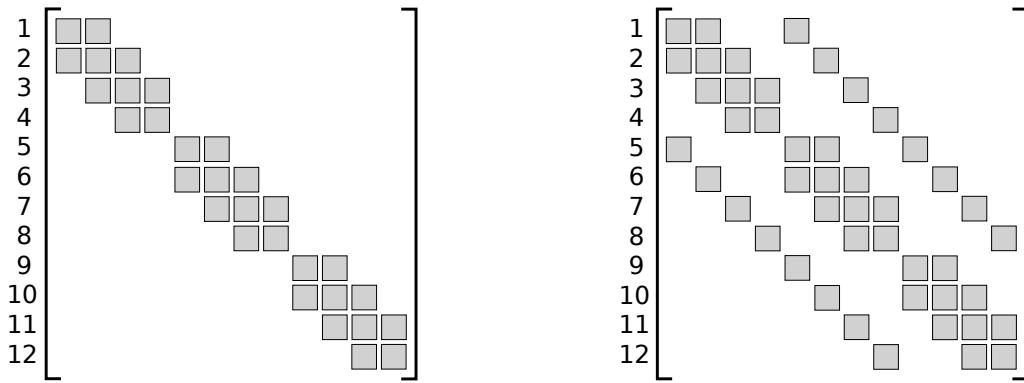
3.3.2.2 Jacobian Calculation

The Jacobians can be calculated using either analytic derivatives of the flux functions w.r.t. the conserved variables or by numerical/automatic differentiation, and each method has its advantages and disadvantages. Using analytic derivatives is normally significantly more computationally efficient, but they can be difficult to obtain (or require simplification), are prone to errors during their implementation and must be derived and coded for each flux scheme, transport model, governing equation etc. Numerical or automatic differentiation methods are normally faster to implement,

(1,4) → 4	(2,4) → 8	(3,4) → 12
(1,3) → 3	(2,3) → 7	(3,3) → 11
(1,2) → 2	(2,2) → 6	(3,2) → 10
(1,1) → 1	(2,1) → 5	(3,1) → 9

η
↑
 ξ

- (a) A structured grid with a mapping $(i, j) \rightarrow k$ that places consecutive j values along the central tri-diagonal.



- (b) The block tri-diagonal matrix of a partitioned system (left) and the block penta-diagonal matrix of an unpartitioned system (right).

Figure 3.8: A depiction of the index mapping and LHS matrix.

easier to debug and can be more accurate than simplified analytic derivatives.

However, the accuracy of numerical, finite-difference derivatives depends on the step size and for both methods the flux functions must be evaluated multiple times to obtain the derivatives w.r.t. each conserved variable.

In this work, numerical differentiation is used to obtain the flux Jacobians. Specifically, the complex-step method is used, which is a numerical method that avoids the pitfalls of the finite-difference method. It is similar to dual number automatic differentiation, but can be used with AMROC's FORTRAN77 code base as it uses complex numbers rather than non-standard dual number types. Although it is less computationally efficient than using analytic derivatives, using this method has enabled the rapid implementation and testing of implicit methods within the new solver for both inviscid and viscous flows.

The complex-step method is based on the Taylor expansion of a complex number, $z = x + ih$,

$$f(z) = f(x) + ihf'(x) - h^2f''(x)/2! - ih^3f'''(x) + \dots . \quad (3.165)$$

When h is chosen to be a number smaller than the square root of the floating point precision, the parts of the right hand side that contain exponents of h greater than two

are negligible. Taking the imaginary part of both sides, neglecting higher-order h terms and dividing by h gives

$$f'(x) = \frac{\text{Im}[f(z)]}{h}. \quad (3.166)$$

This means that the derivative of a real valued function can be found by using a complex value function instead, modifying the complex part by a step size h and then dividing the imaginary part of the output by the step size. The method avoids the two main causes of error that arise in the finite-difference method: the round-off error that arises in the subtraction when using small step-sizes, and the error in the secant method when using larger step sizes.

The error of the complex-step and finite-difference derivatives relative to the analytic derivatives for the frozen speed-of-sound are shown in Fig. 3.9. The speed-of-sound was chosen as it incorporates a number of different flow properties. The results demonstrate the step-size independence and improved accuracy of the complex-step method for calculating derivatives.

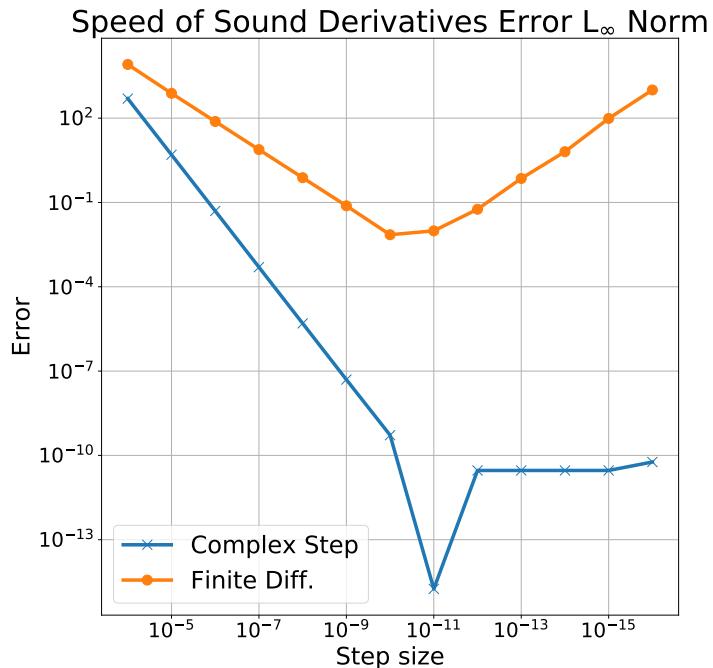


Figure 3.9: The complex-step and finite-difference L_2 norm error for various step sizes.

Although the complex-step method has advantages over the finite-difference method, there are also some disadvantages. Firstly, there is an additional computational cost when using the complex-step method as each function must be evaluated using complex variables. Secondly, the complex-step method does not work when calculating variables using external libraries that only accept real numbers as inputs.

The solver in this work makes extensive use of Mutation++, which is implemented using real numbers. Therefore, many of the functions which compute the variables required for the flux calculations (speed-of-sound, pressure, viscosity etc.) were re-written in FORTRAN77 with complex variables as the inputs and outputs. The final disadvantage of using the complex-step method over the finite-difference method is that a function must have an analytic solution in order to obtain the derivative. To circumvent this problem, the analytic derivative can be approximated and then used directly as the complex part of the function output.

When using numerical differentiation the derivatives for each variable is calculated by modifying that variable, calling the function and extracting the derivative. Therefore, the function must be called once per derivative. A naive implementation would construct the LHS Jacobian by modifying each conserved variable in turn and calling the flux functions, which would result in $N_e \times N_c$ flux computations. However, not all of the cells interact, so the number of flux evaluations can be significantly reduced by identifying cells that can be modified at the same time. These cells can be identified using graph colouring [56, 92]. The principal of graph colouring is that any number of variables can be modified at the same time as long as they do not appear within the same row of the Jacobian [56, 92]. As previously discussed, the LHS matrix for the structured AMROC grid is either tri- or penta-diagonal. Therefore, the Jacobian can be constructed in either $N_e \times 3$ or $N_e \times 5$ calls to the flux functions, respectively. The structured nature of the mesh makes the graph simple to determine, as cells with consecutive i and j values are in the same row of the Jacobian. Thus, adjacent cells are given different “colours” and no two variables from the adjacent cells are modified at the same time.

3.3.2.3 Linear System Solution

The large, sparse nature of the linear system makes the use of direct solvers impractical. Therefore, a number of iterative linear solvers have been created as part of this research to efficiently solve the block tri- or penta-diagonal linear system at each update.

The LHS matrix A can be written as

$$A = A_L + A_D + A_U, \quad (3.167)$$

where, for an unpartitioned, penta-diagonal system

$$A_L = A_L^O + A_L^C, \quad A_U = A_U^C + A_U^O, \quad (3.168)$$

and for a partitioned system,

$$A_L = A_L^C, \quad A_U = A_U^C. \quad (3.169)$$

In the above, A_D represents the block diagonal, $A_{L/U}^C$ represent the lower and upper blocks of the central block tri-diagonal, and $A_{L/U}^O$ are the block diagonals offset from the central diagonal (see Fig. 3.8). The structured nature of the grid means that each diagonal can be saved in a dense format, using the minimum possible amount of storage and enabling efficient memory access. In this work, the matrix is stored in this format and sparse matrix/vector operations have been implemented that are optimised for this structure. In addition, all of the dense matrix and vector operations are carried out using efficient BLAS and LAPACK routines.

The fixed-point iterative solvers that have been implemented in this work are the Jacobi solver, Symmetric Gauss-Seidel (SGS) solver, Lower-Upper Symmetric-Gauss-Seidel (LU-SGS) solver and line-relaxation solver. In addition a preconditioned, restarted Generalized Minimal Residual (GMRES) Krylov subspace solver has been implemented that can use any of the fixed-point solvers as a preconditioner.

The simplest solver is the Jacobi solver. In this method, each iterative update is given by

$$\mathbf{x}^m = A_D^{-1} \left[\mathbf{b} - (A_L + A_U) \mathbf{x}^{m-1} \right], \quad (3.170)$$

where m is the iteration index.

The SGS method uses a similar approach to the Jacobi method but updates the solution values used on the RHS during each iteration, rather than at the end of each iteration. This can significantly increase the speed of convergence relative to the Jacobi method. The convergence rate will depend on the order in which the updates are carried out, so consecutive forward and backward sweeps can be used to reduce the influence of the cell ordering. The forward sweeps update the solution in each cell starting from $k = 1$ then increasing k using

$$\mathbf{x}^* = A_D^{-1} \left(\mathbf{b} - A_L \mathbf{x}^* - A_U \mathbf{x}^{m-1} \right). \quad (3.171)$$

The backward sweep updates the solution values starting from $k = N_c$ and decreasing k using

$$\mathbf{x}^m = A_D^{-1} \left(\mathbf{b} - A_L \mathbf{x}^* - A_U \mathbf{x}^m \right). \quad (3.172)$$

The LU-SGS method is similar to the SGS and is derived by re-writing the matrix A as

$$A = A_L + A_D + A_U \quad (3.173)$$

$$= (A_L + A_D) A_D^{-1} (A_U + A_D) - A_L A_D^{-1} A_U. \quad (3.174)$$

Neglecting the terms $A_L A_D^{-1} A_U$, this gives

$$(A_L + A_D)\mathbf{x}^* = \mathbf{b}, \quad (3.175)$$

$$A_D^{-1}(A_U + A_D)\mathbf{x}^m = \mathbf{x}^*. \quad (3.176)$$

Thus, the forward and backward sweeps of the LU-SGS method can then be written as

$$\mathbf{x}^* = A_D^{-1}(\mathbf{b} - A_L \mathbf{x}^*), \quad (3.177)$$

and,

$$\mathbf{x}^m = A_D^{-1}(\mathbf{b} - A_D \mathbf{x}^* + A_U \mathbf{x}^m). \quad (3.178)$$

It can be seen that the LU-SGS method reduces the number of matrix-vector multiplications required compared with the SGS method. In the original LU-SGS method the diagonal matrix of each cell was approximated using only diagonal elements. This greatly simplifies the inversion of the diagonal matrix and significantly reduces the operation count relative to the SGS method [298]. In this work, the full diagonal matrices are used, but future work could investigate the use of an approximate diagonal matrix. The LU-SGS solver can be used as a solver on its own [298], but is more commonly used as a preconditioner in a Krylov subspace method.

The line-relaxation (a.k.a. line-implicit) solver is also analogous to the Jacobi method. However, rather than inverting only the block diagonal of the A matrix the entire central block tri-diagonal is inverted. Therefore, the iterative line-relaxation method is given by

$$\mathbf{x}^m = A_T^{-1} \left[\mathbf{b} - (A_L^O + A_U^O)\mathbf{x}^{m-1} \right], \quad (3.179)$$

where $A_T = A_L^D + A_D + A_U^D$ is the central block tri-diagonal of A , and $A_L^O + A_U^O$ is the remaining off-diagonal matrices of A . This is shown in Fig 3.10. The central block tri-diagonal is directly inverted using block Lower-Upper (LU) factorisation [281] (see Appendix A.3 for details).

The line-relaxation method takes advantage of the coupling between the mesh and the flow field to efficiently obtain solutions in viscous, wall bounded flow simulations [296]. The flow gradients are usually much stronger and the cells are thinner in the wall-normal direction. Therefore, using LU factorisation to solve the system exactly in this direction greatly improves the convergence rate [296]. In a structured strand mesh this is achieved by reordering the two-dimensional mesh indices so that the cells in the wall-normal direction contribute to the central block tri-diagonal system.

As previously discussed, when using a partitioned method where only the wall-normal fluxes are integrated implicitly, a block tri-diagonal LHS matrix is created. Using the block tri-diagonal LU factorisation method, the solution for this

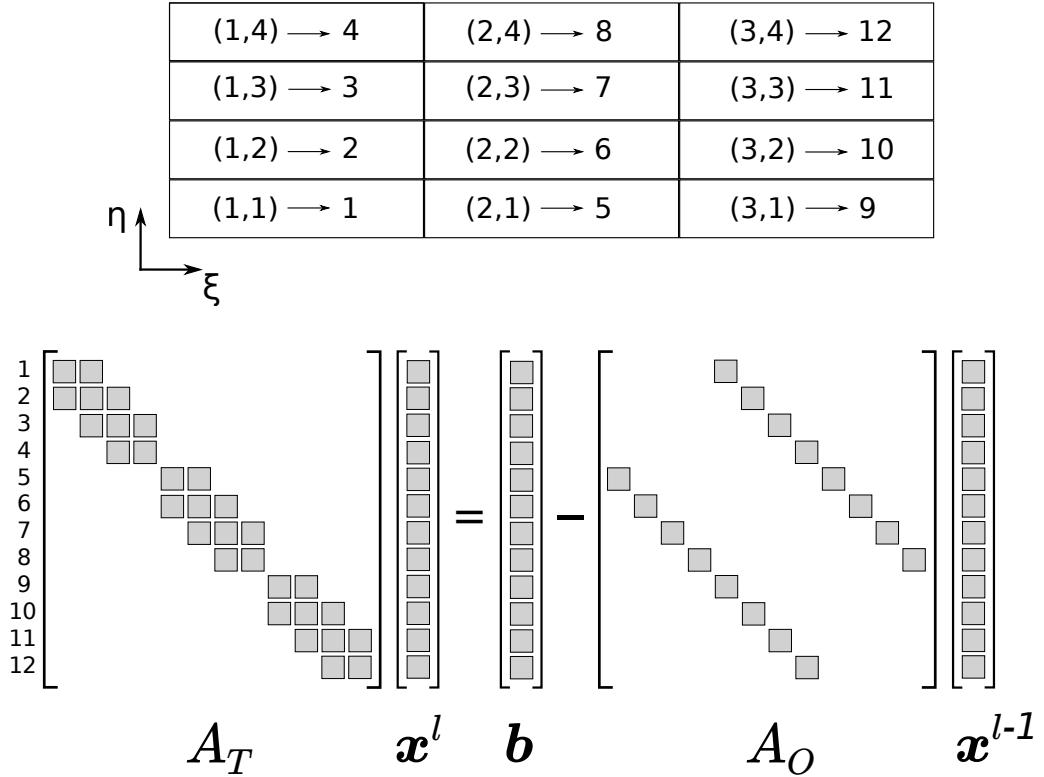


Figure 3.10: A depiction of the index mapping and linear system when using a line-implicit method, where the lines are constructed in the η direction.

system can be obtained directly. This can make dimensionally partitioned time integration methods more computationally efficient.

The above fixed-point iteration methods all use a relatively small amount of memory as only the off-diagonal matrices and inverted (tri-)diagonal matrices need to be saved. The inversion only needs to be performed once and can be re-used for each iteration, making the iterations relatively cheap. However, they can suffer from slow convergence. Therefore, Krylov subspace methods are often used instead, as they have better convergence properties than fixed-point iteration methods [72].

The GMRES method is a Krylov subspace method that is widely used within CFD solvers. For the derivation of the GMRES method and a discussion on the advantages relative to the fixed-point iteration methods described above, see Ref. [72]. For ill-conditioned systems, such as those typically encountered in flow simulations, the GMRES method can require a significant number of iterations before converging to a solution. To improve the convergence rate one instead solves a preconditioned system. One can either solve a left-preconditioned system

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}, \quad (3.180)$$

where M is an approximation to A , or a right-preconditioned system,

$$AM^{-1}\mathbf{u} = \mathbf{b}, \quad (3.181)$$

where $\mathbf{x} = M^{-1}\mathbf{u}$. In practice the solution of the preconditioned system does not require the inversion of a matrix M , but usually involves one or two iterations of an iterative solver using the LHS matrix, A . As such, any of the above iterative solvers can be used as a preconditioner for the GMRES method.

The main advantage of the GMRES method is its convergence properties. In practical computations, Krylov subspace methods, such as the GMRES method, have been shown to converge in significantly fewer iterations than fixed-point solvers [6, 72]. A disadvantage of the GMRES solver is that it requires significantly greater storage than the fixed-point methods. The original GMRES method required m_{\max} vectors of equal dimension to the solution vector, where m_{\max} is the maximum number of iterations allowed. To reduce the memory requirements, a restarted GMRES solver can be used, where the solver can “restart” after a user-specified number of iterations. The ability to restart means that the number of additional vectors is reduced to m_{restart} , which can be much smaller than m_{\max} . However, this requires the Krylov subspace to be rebuilt, which can cause the solver to stagnate and lead to slower convergence or a failure to converge. Another disadvantage is that the cost of the method increases significantly with each additional iteration. This makes the use of a preconditioner to aid convergence extremely important.

When the fixed-point iteration methods are used as solvers or preconditioners one must account for the distributed nature of the grid. In AMROC, the grid can be distributed across multiple processors, or multiple structured blocks can exist on a single processor that contains a large number of cells. Solving a global system over a decomposed domain requires iterative updates to the solution [72]. In this work a minimum overlap additive Schwarz method is used where each grid block is solved independently and ghost cell communication is then used to obtain the solution at the block boundary. This is then used to update the RHS of the equation before the next iteration. For example, when using the line-relaxation method, each iterative update is modified to become

$$\mathbf{x}^m = A_T^{-1} \left[\mathbf{b} - (A_L^O + A_U^O) \mathbf{x}^{m-1} - A_G \mathbf{x}_G^{m-1} \right], \quad (3.182)$$

where A_G contains the ghost cell Jacobians and \mathbf{x}_G^{m-1} the ghost cell solutions. Similarly, for a dimensionally partitioned system the solution update would be given by

$$\mathbf{x}^m = A_T^{-1} \left(\mathbf{b} - A_G \mathbf{x}_G^{m-1} \right). \quad (3.183)$$

Parallel communication is performed after every iteration of the Jacobi, SGS, LU-SGS or line-relaxation methods, including when they are used as preconditioners for the GMRES solver. The above update procedure is depicted for a partitioned method in Fig. 3.11.

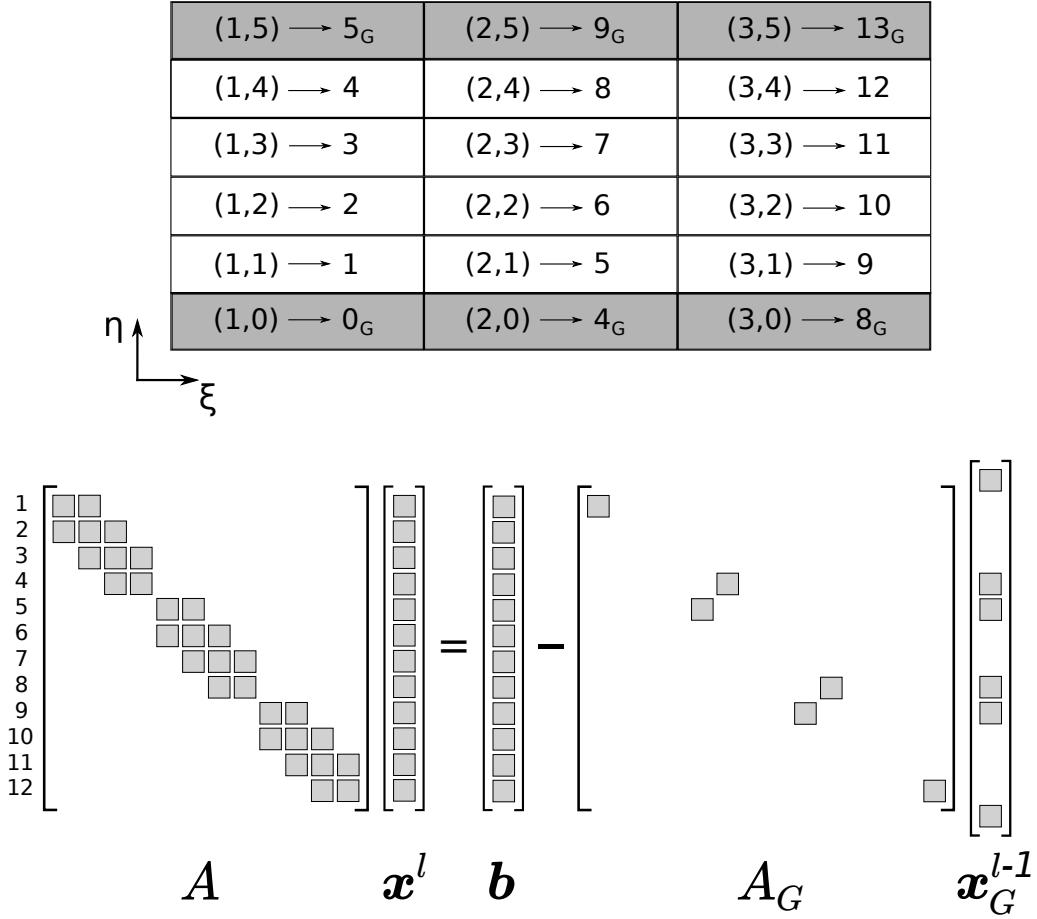


Figure 3.11: A depiction of the additive Schwarz method for η implicit time integration.

The physical boundary conditions for the linear system solution are set using an analogous process to the processor boundaries, where the solution in the boundary cells are updated after each iteration. The solution in the boundary cells is related to the solution in the interior cells in the same way as the physical boundary conditions (see Section 3.2.3). However, as the linear solver only considers the conserved variables the temperature boundary conditions cannot be set directly and are therefore updated explicitly.

A termination criterion is required for all of the above solvers and a measure of the relative error is used in this work. For the fixed-point solvers this is based on the

difference in the solution vector updates

$$\epsilon^l = \frac{\| \mathbf{x}^m - \mathbf{x}^{m-1} \|}{\| \mathbf{x}^m \|}. \quad (3.184)$$

In contrast, the GMRES solver minimises the residual and therefore the stopping criterion uses the relative residual,

$$\epsilon^l = \frac{\| A\mathbf{x}^m - \mathbf{b} \|}{\| \mathbf{b} \|} \quad (3.185)$$

for the right-preconditioned system or

$$\epsilon^l = \frac{\| M^{-1}A\mathbf{x}^m - M^{-1}\mathbf{b} \|}{\| M^{-1}\mathbf{b} \|} \quad (3.186)$$

when using the left-preconditioned system. When the linear solver error estimate ϵ^l falls below a user specified tolerance the iterations are terminated and the solution is returned.

A set of unit tests have been implemented to verify the solvers and sparse matrix and vector operations. The solvers are verified using randomly generated A matrices and \mathbf{x} vectors which are then multiplied together to find the \mathbf{b} vectors. The solvers are then used to obtain a solution vector, $\hat{\mathbf{x}}$, from A and \mathbf{b} . The tests showed that all of the solvers were able to obtain solutions with an L2 norm error close to machine precision for block diagonal, tri-diagonal and penta-diagonal systems.

3.3.3 Implicit Test Cases

The motivation behind introducing an implicit time integration scheme is to provide a significant speed up over explicit methods due to the longer time-steps that can be used. When using the new implicit methods, the user must input the linear solver termination criteria, determine which linear solver to use and decide whether to use a partitioned or unpartitioned method. These choices all influence the computational efficiency of the implicit method. Three steady-state test cases were used to understand the influence of these choices on the computational time. These test cases were the simulation of viscous supersonic flow over a cylinder, and the simulations of viscous subsonic and supersonic flow over a flat plate.

For the subsonic flat plate case the freestream Mach number was 0.2 and the freestream Reynolds number $3 \times 10^5 \text{ m}^{-1}$. For both supersonic simulations, the Mach number was 3 and Reynolds number $1.5 \times 10^6 \text{ m}^{-1}$. In all cases a pure nitrogen flow was used with a freestream temperature of 273.15 K. The subsonic flat plate case was used to study a flow without shocks and enabled comparisons with the analytic Blasius solution. The supersonic cases were chosen as examples of high Reynolds

number, compressible flows common to hypersonic flight, with the cylinder simulation giving a shock dominated flow and the flat plate simulation a viscous dominated flow. The supersonic cases used a relatively low enthalpy so that the flux partitions were the primary reason for numerical stiffness, rather than the thermochemical nonequilibrium source terms.

For the flat plate simulation a rectangular domain, with dimensions of 0.15 m by 0.45 m and 100×125 cells was used. Along the lower boundary a slip region from 0.15 m was included prior to the start of the wall boundary condition, giving the flat plate a length of 0.3 m. The spacing in the x direction was varied so that the start of the flat plate was refined. For the subsonic case, the wall spacing was $20 \mu\text{m}$ and for the supersonic cases the wall spacing was $10 \mu\text{m}$. The mesh was stretched away from the wall using a hyperbolic tangent function. An image of the flat plate mesh for the subsonic case is shown in Fig. 3.12. The CFL number was increased from an initial value of 1 to a final value of 500 in the subsonic simulation and 1000 in the supersonic simulation.

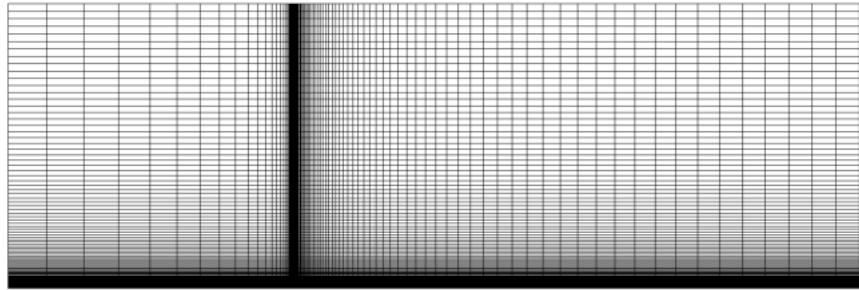


Figure 3.12: The mesh used for the flat plate simulation, stretched in both the wall-normal and wall-aligned directions.

The cylinder simulation used a cylinder with a radius of 45×10^{-3} m. A symmetry boundary condition was used along the stagnation streamline. The mesh was fitted to the cylinder surface and extended to 0.15 m in both the vertical and horizontal direction. The mesh used 75 cells in the wall-tangential direction and 250 cells in the wall-normal direction. The near-wall spacing was set as $10 \mu\text{m}$ and the mesh was stretched away from the wall using a hyperbolic tangent function. The maximum stable CFL number was empirically determined to be approximately 100, which is significantly lower than in the flat plate cases. This is likely due to the strongly nonlinear normal shock in the stagnation region, which results in the linearised time integration method giving inaccurate predictions, leading to instabilities at long time-steps.

3.3.3.1 Linear Solver Termination Criteria

In steady-state simulations one is seeking the root of the nonlinear equation $\mathbf{f}(\mathcal{Q}) = 0$. When integrating in time to steady-state the aim is to obtain a solution at each time step that is sufficiently accurate that it allows the nonlinear problem to converge. Using a higher termination criteria means that there will be a larger error in the solution but the number of linear solver iterations will be lower. The aim of these simulations is to examine the most efficient choice for the linear solver termination criteria for steady-state simulations, i.e. the maximum criterion that allows the solution to converge to steady-state.

The linear solver termination criteria was varied from 1×10^{-2} to 1×10^{-6} in factors of 10. In all cases the criteria had limited impact on the rate of convergence and the final result. The density residual convergence for each criterion is graphically depicted in Fig. 3.13. The jumps in the residual are where the CFL condition was increased at a given physical time. It can be seen that the convergence in all cases is similar, but oscillations are present in the density residual when using the highest termination criteria. Similar results were obtained in the supersonic flat plate and cylinder cases.

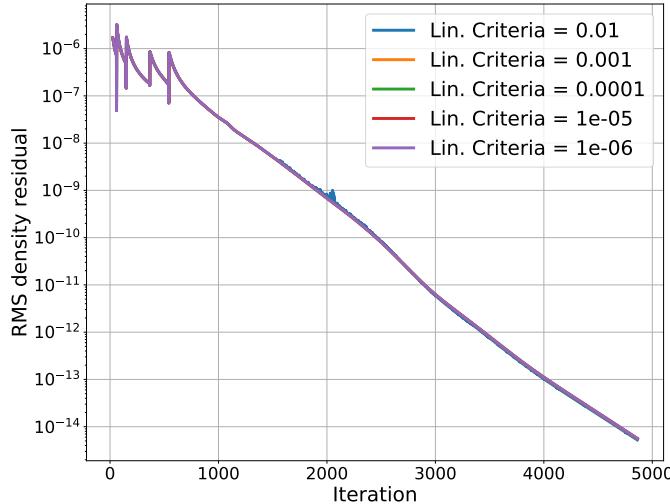


Figure 3.13: The density residual convergence using different linear solver termination criteria (the curves overlap).

The boundary layer velocity profile is compared against the analytic Blasius solution in Fig. 3.14. Excellent agreement is obtained with the Blasius solution providing good verification evidence for the implementation of the viscous fluxes on the mapped mesh. The analytic solution was also used to assess the error when using the different linear solver termination criteria. The error was the same for all criteria. The results from this test suggest that a linear solver termination criteria of 1×10^{-3} would be sufficient in most steady-state simulations.

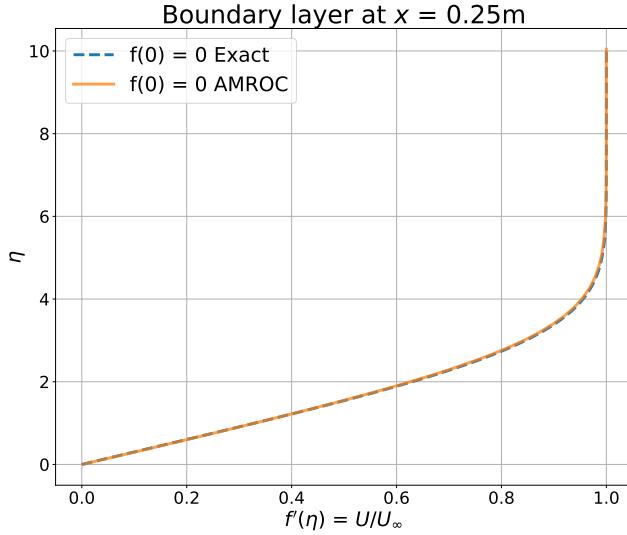


Figure 3.14: A comparison with the analytic solution for the subsonic flat plate test case.

3.3.3.2 Linear Solver Efficiency

For each test case, the different linear solvers were used to advance the solution to steady-state. The time spent in each linear solver and the average number of linear iterations is shown in Table 3.4 for the subsonic flat plate case. The results show that the LU-SGS preconditioned GMRES solver provided the best performance, whereas the simulation failed to converge when using the line-relaxation preconditioned GMRES solver. This highlights how the choice of preconditioner can have a large impact on the GMRES solver's efficacy. The convergence failure using the line-relaxation preconditioner is likely due to a restarted GMRES solver being used, and the solver requiring significantly more iterations than m_{restart} . The line-relaxation solver required more than 38 iterations on average, and at higher CFL numbers required up to 62 iterations, which is significantly higher than m_{restart} which was set to be 20.

	GMRES(LU-SGS)	SGS	Line Rel.
Time (s)	394.1	486.5	629.7
Normalised Time	1.0	1.235	1.598
Avg. Iterations	11.38	19.46	38.59
Max. Iterations	17	28	62

Table 3.4: Linear solver data for the subsonic flat plate test case.

The set of results for the supersonic flat plate test case are shown in Table 3.5. For this test case, the LU-SGS preconditioned GMRES solver failed to converge and the line-relaxation solver gives the best performance. This is in contrast to the subsonic

test case where the line-relaxation preconditioned GMRES solver failed to converge and the line-relaxation solver performed worst out of those that did converge. These results highlight the interaction between the physical flow and the linear solver efficiency. For the subsonic test case, where each cell face contained left and right going waves, the direction independent solvers performed best.

	GMRES(Line Rel.)	SGS	Line Rel.
Time (s)	309.9	851.5	242.9
Normalised Time	1.276	3.506	1.0
Avg. Iterations	8.1	43.2	10.31
Max. Iterations	28	92	31

Table 3.5: Linear solver data for the supersonic flat plate test case.

For the cylinder test case, all solvers were able to converge to the same solution and the performance of the solvers is much more even than in the previous two cases. This could be due to the lower maximum CFL number used in this case. The line-relaxation solver gives the best performance followed closely by the LU-SGS preconditioned GMRES solver and the SGS solver. The GMRES solvers required the least iterations on average, but each iteration was more costly than the fixed-point methods. For this computation, the LU-SGS preconditioner is more effective as it reduces the iterations by approximately one third relative to the line-relaxation preconditioner.

	GMRES(Line Rel.)	GMRES(LU-SGS)	SGS	Line Rel.
Time (s)	417.8	344.7	359.8	340.8
Normalised Time	1.23	1.01	1.05	1.0
Avg. Iterations	9.9	6.7	11.5	11.3
Max. Iterations	15	10	18	17

Table 3.6: Linear solver data for the cylinder test case.

Considering the above, it is difficult to give general recommendations for the solver that should be used. The results show that it is dependent on the flow features. Both the SGS and line-relaxation methods give competitive performance relative to the GMRES solver in certain conditions. In addition, the GMRES solver uses more memory and is more sensitive to user set-up, requiring the correct preconditioner and m_{restart} value in order to obtain convergence. The line-relaxation solver gave the best performance in the two supersonic cases so may be the best choice for hypersonic flow simulations. The widely used hypersonic solvers DPLR [296] and US3D [207] use the line-relaxation method by default, which supports this conclusion.

3.3.3.3 Partitioned vs Unpartitioned Integration

In all of the tests the dimensionally partitioned methods required significantly smaller time-steps in order to be stable. The stable time-step often required the CFL number for the explicit partition to be less than 0.5 as would be expected for the forward Euler method. Whilst the partitioned method does give a reduced time per iteration, the significantly larger number of iterations required in these test cases meant that the unpartitioned methods were less efficient. These results are in agreement with previous studies investigating different solution techniques for strand mesh grids [136]. Possible exceptions to this situation occur when the time-step is limited for some other reason, resulting in a low explicit CFL in the wall-normal direction but a CFL greater than one in the wall-aligned direction. For example, this may occur on stretched grids when the maximum stable implicit time step is the limiting factor, when physical processes limit the time-step, or when the explicit off-body grid limits the near-body time step.

3.3.3.4 Implicit Method Speed-Up

The primary reason for introducing implicit time integration is to reduce the computational time required to carry out simulations. The test cases have shown that a very high CFL number can be used with the new implicit method, with a CFL condition of 1000 used for the supersonic flat plate simulation. Whilst this will result in significantly fewer time-steps to reach steady-state than when using an explicit method, the implicit methods incur additional costs. Firstly, the Jacobian needs to be created at each time-step, then the linear system must be solved. The above simulations were also run using an explicit method in order to examine the speed-up enabled by the new implicit method. The implicit simulations used a linear solver termination criteria of 1×10^{-4} , the most efficient linear solver for the test case and the empirically determined maximum stable CFL number.

A comparison of density residual convergence w.r.t. the iteration number and time is shown in Fig. 3.15 for the cylinder test case when using explicit and implicit methods. Both simulations converge to approximately the same residual, however the implicit simulation converges in significantly fewer iterations, with a shorter run-time. The cylinder test case had the lowest implicit CFL number and should therefore result in the lowest speed-up, but still gives a speed-up of approximately 7.3 to reach the same steady-state residual.

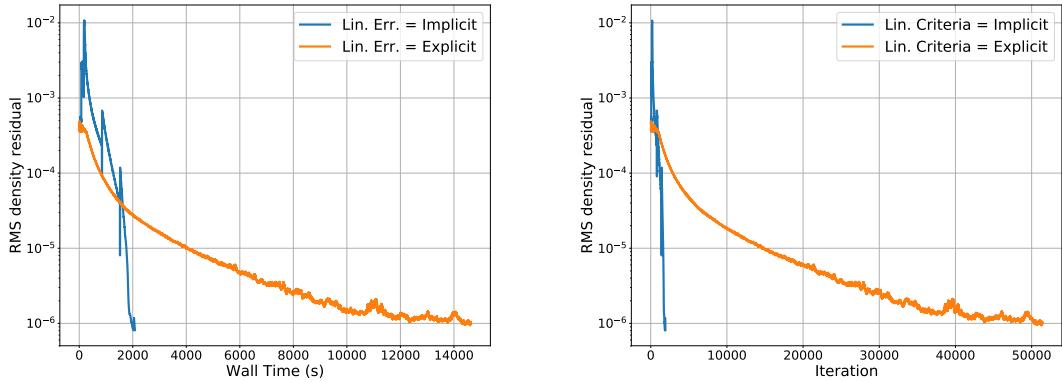
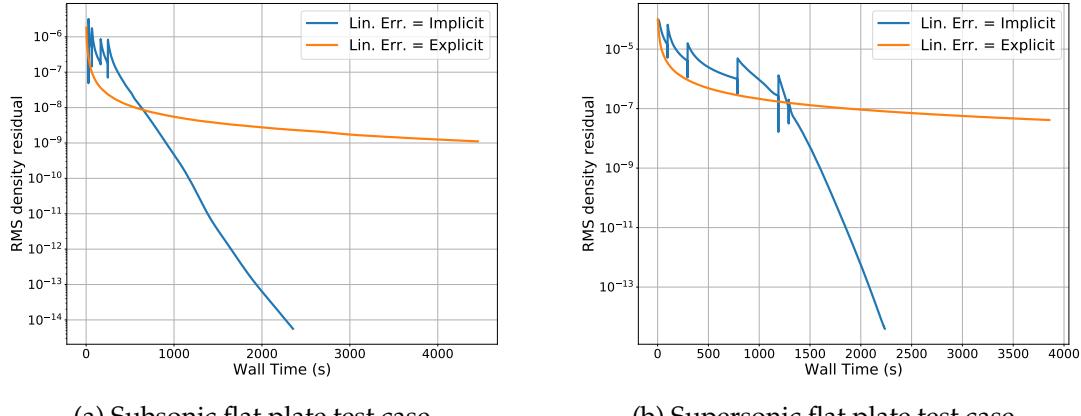


Figure 3.15: The density residual convergence plotted against the wall time (left) and iteration count (right) when using implicit and explicit time integration.

Comparisons of the density residual convergence for the flat plate test cases are shown in Fig. 3.16. The explicit flat plate simulations were not run to steady-state as they would have required a prohibitive number of time-steps. The results show that the residual in the explicit case is orders of magnitude higher than in the implicit case, even though it was run for longer, and the rate of convergence is very slow. As such, the speed up obtained from the implicit scheme is likely to be significantly larger than the factor of 7.3 obtained for the cylinder case.



(a) Subsonic flat plate test case.

(b) Supersonic flat plate test case.

Figure 3.16: The density residual convergence plotted against the wall time when using implicit and explicit time integration.

In Table 3.7 a breakdown of the times required for the different components of the implicit method is shown. It can be seen that the Jacobian calculation dominates the computational time for all three test cases. Whilst the complex-step method makes the Jacobian computation robust and simple to extend to new flux schemes it requires the complex flux functions to be called $N_e \times 6$ times for an unpartitioned method. This makes the complex step method computationally expensive and highly sensitive to

the number of equations and dimensions, as well as to the cost of the flux evaluations. The implementation of analytic Jacobians would likely give a significant speed up over the complex-step method and further increase the computational efficiency relative to the explicit method.

	Flat plate (M=3)	Flat plate (M=0.2)	Cylinder
RHS evaluation (s)	99.3	98.1	103.6
Jacobian creation (s)	1575.5	1547.9	1663.4
Linear solve (s)	242.9	394.1	340.8

Table 3.7: The total time spent in each part of the implicit update for the three steady-state test cases.

3.4 Chapter Summary

Several additions and modifications to the AMROC framework were required to enable hypersonic flow simulations using a strand/CAMR technique. A hypersonic fluid model has been integrated with AMROC, enabling the simulation of flow in thermochemical nonequilibrium. The spatial integration routines within the AMROC solver have been greatly extended to enable viscous computations on mapped grids. The hypersonic fluid model and mapped mesh methods were verified using the MMS technique. This is the first time this method has been applied to thermochemical nonequilibrium models. Implicit methods have been implemented to allow for more efficient time-integration techniques in high Reynolds number simulations. This included the development of several linear solvers that have been tightly integrated with the AMROC data structures and account for domain decomposition on distributed memory machines. The implicit methods are a substantial addition to the AMROC framework and were shown to give large reductions in computational time over the previously implemented explicit methods.

As a result of the above work, AMROC can now efficiently obtain solutions on strand grids. In the next chapter the strand mesh generation algorithms and overset grid assembly methods required to create a highly automated strand/CAMR solver are described in detail.

Chapter 4

Automated Meshing Techniques

A strand/CAMR solver enables highly-automated mesh generation by combining an off-body CAMR solver with a near-body strand solver using overset grid assembly algorithms. In this chapter the specific implementation of the underlying automated meshing and grid assembly techniques used in this research is discussed. First, a brief review of the pre-existing functionality within the AMROC SAMR framework is given, with a focus on the areas that are most relevant to this work. Then, the strand mesh generator and overset grid assembly library that have been developed by the author are presented. A novel strand mesh generation algorithm is described. This aims to create near-body meshes that are more suitable for hypersonic boundary layer flow simulations than previous strand meshing techniques. The implementation of the parallel, distributed-memory overset grid assembly toolkit is discussed. Verification and performance studies conducted as part of this research are also presented.

4.1 AMROC Framework

The off-body solver is based on the adaptive computational framework AMROC (Adaptive Mesh Refinement in Object-oriented C++), which implements the SAMR algorithm of Berger and Colella [21]. As discussed in Section 2.2.1, the SAMR patches retain all of the inherent advantages of Cartesian grids on each patch, including optimal mesh quality, structured data layouts and simplified extension to higher-order methods. AMROC can accommodate a large number of refinement levels, each with different refinement factors. Dynamic load balancing on distributed-memory machines is achieved by using efficient space filling curve methods [64]. The AMROC framework contains refinement criteria based on the “scaled gradient” approach and Richardson-type error estimates, enabling robust, automated grid refinement. More recently, a multiresolution criteria has been implemented within the AMROC framework [65], giving improved performance over the scaled gradient method, but

this has not been used in this work. AMROC is designed as an extensible framework and a variety of different governing equations have previously been implemented within the patch integrators to create a number of different SAMR solvers. The features within AMROC that enable highly automated off-body mesh generation and coupling to a near-body solver are reviewed in this section. A more complete introduction to the AMROC framework can be found in Ref. [64].

AMROC has a proven ability to perform highly-automated mesh refinement. This has been shown for a range of flow models including ideal gas [62], reacting gas [62, 63], magnetohydrodynamic [168] and cell-based lattice-Boltzmann models [67]. The distributed memory SAMR methods within AMROC have been shown to be effective with both second-order-accurate MUSCL schemes and higher-order WENO methods [305].

AMROC contains embedded boundary methods that enable sections of the AMR grid to be removed from the computation and boundary conditions imposed. These algorithms have been utilised in the hole-cutting stage of the overset grid assembly. The holes are cut in the grid using level-set algorithms and the cells on the boundary can be flagged for refinement [64]. The ability to refine the boundaries enables complex hole-cuts to be accurately represented and the refinement level to be selected based on the overlapping near-body cell sizes. The level-set method cuts a hole in the domain by calculating the signed distance from a level surface, where cells inside the surface have a negative distance. One method for specifying the level surface that is particularly useful for automated mesh generation is the the Closest Point Transform (CPT) method of Mauch [182]. In this method the level surface can be described using an arbitrary, triangulated surface. The Eikonal equation is then solved to determine the distance of each cell from the surface. A combination of the method of characteristics and polygon/polyhedral scan conversion is used to solve the equation. This gives a highly efficient method, which has a computational complexity that is linear with respect to the number of grid points on the background grid and the number of points on the level surface [66, 182].

The CPT method has been tested and verified for moving surfaces in Fluid-Structure Interaction (FSI) computations performed using the AMROC framework and a separate solid mechanics solver [54, 66]. These FSI computations are similar to overset computations in that two separate solvers are coupled by boundary conditions. In this previous work, a method for processor-to-processor communication between the solvers was developed that utilised overlapping bounding boxes to determine the communication pattern [66]. This idea, and the underlying algorithms, have been utilised and further developed in this work.

Finally, SAMR grid interpolation routines previously implemented in AMROC have been used within the overset grid assembly algorithms. In these methods, linear

interpolation is used when all of the surrounding values are known, and constant interpolation is used if only a limited number of surrounding values are known. This results in either second- or first-order-accurate interpolation, respectively. As the near-body overset boundaries will fall within the AMR domain, enough points will be available for second-order-accurate interpolation to be used.

As part of this research, a new refinement criterion has been added to the AMROC framework. The refinement criterion flags cells containing interpolation points for refinement. This reduces the truncation error close to interpolation points, which should lead to the exchange of more accurate solution variables during overset grid assembly.

4.2 Strand Mesh Implementation

Strand mesh generation algorithms have been incorporated into the mapped solver to enable near-body meshes to be created and updated with minimal user input. A novel strand mesh generation algorithm has been developed that is well suited to simulations of high-speed viscous flows. In addition, a surface motion algorithm has been integrated with the strand mesh generator to enable automated mesh updates.

4.2.1 Strand Generation

The strand meshing method, mesh quality problems and proposed solutions were first reviewed in Section 2.2.2, but are summarised here for completeness. Strand mesh generation techniques grow strands, which are then connected to create cells. In the original method, the strand growth vectors are adjusted from an initial growth vector that is taken as the normal of the surrounding surface faces. The growth vectors were modified to increase the mesh “smoothness”, which can be defined as the maximum volume ratio between each cell and its neighbours. The smoothing process previously implemented was able to reduce the large jumps in cell size and low resolution at convex corners that arise when using the surface normal vectors alone. However, the original smoothing method creates strands that are not orthogonal to the surface. This results in a trade-off between mesh smoothness away from the surface and orthogonality at the surface. Mesh orthogonality is particularly important when computing heat fluxes and shear stressed in hypersonic flows as the large near-wall gradients must be accurately captured [95]. Two methods have been proposed to overcome the resolution/orthogonality trade-off: the “multi-strand” method [111] and the “multi-level” method [234, 254]. However, the multi-strand method has been shown to give less accurate results than the original method [154, 294] and the multi-level method reduces the level of automation and increases the memory required to store the mesh.

To address some of the short-comings of the previous strand meshing techniques, a new strand mesh generation technique has been developed. The new method aims to create a mesh with orthogonal strands at the surface and high smoothness away from the surface, whilst maintaining a high level of automation and a small memory footprint. The new technique is similar to the multi-level strand method, but only uses two growth vectors in order to minimise the memory foot-print. An inner growth vector, \hat{v}_I , is used close to the wall and is typically orthogonal to the surface (although smoothing can be applied by the user if desired). An outer growth vector, \hat{v}_O , is used further from the boundary and the orientation is set to create a smooth mesh. Using just two vectors maintains a compact mesh description, but it can lead to poor mesh quality when the angle between the two vectors is large. To overcome this issue a

circular arc is used to smoothly transition between the inner and outer growth vectors, as shown in Fig. 4.1. The circular arc can be fully defined by the two growth vectors, along with the number of arc nodes, N_c , and the number of inner strand nodes, N_I . Both additional values can be specified by the user, but values of $N_c = 4$ and N_I equal to the number of ghost cells are suitable default values. The availability of default values that work for a wide range of simulations reduces the amount of user input required, increasing automation compared to previous methods.

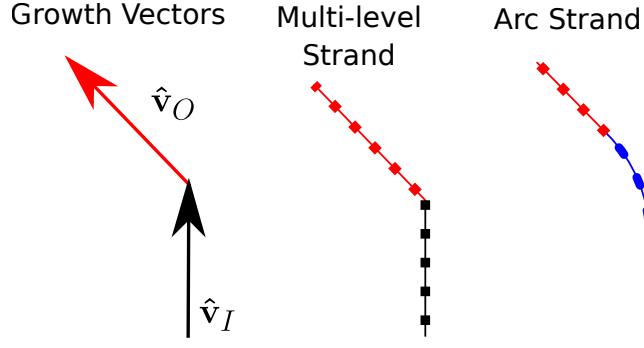


Figure 4.1: Visualisations of the growth vectors and different strand types.

The values of N_c and N_I are used globally on the strand mesh, so an “arcing strand” mesh only requires one additional growth vector per surface node and two additional integers. As such, the entire mesh geometry can be stored using only $10V + 4$ numbers in three dimensions, or $7V + 4$ numbers in two dimensions, where V is the number of surface vertices (cf. $7V + 2$ numbers in three dimensions, or $5V + 2$ numbers in two dimensions for the original strand mesh algorithm).

The first step in the arcing strand mesh generation is to initialise the inner and outer growth vectors at each vertex as the vectors normal to the surface. The normal is calculated as the average of the face normal vectors surrounding the vertex. The outer growth vectors (and, optionally, the inner growth vectors) then undergo smoothing, where an iterative process is used to reduce the angle between neighbouring strand growth vectors [137, 190]. The smoothing is carried out using the same method detailed by Katz *et al.* [137]. This method solves a local minimisation problem at each vertex, which aims to minimise the angle between the strand growth vector at the vertex and the surrounding strand growth vectors. To achieve this, the function to be minimised is

$$f(\mathbf{n}_i) = \sum_{j=1}^{V_n} (1 - \mathbf{n}_i \cdot \mathbf{n}_j), \quad (4.1)$$

where \mathbf{n}_i is the unit strand growth vector of a vertex, V_n is the number of vertices surrounding vertex i , and \mathbf{n}_j is the unit strand growth vector at neighbouring vertex j . The minimisation problem is solved under the constraint that the resulting strand growth vector is a unit vector. As such, the problem is solved iteratively, where, in

each iteration, k , the strand growth vector at each vertex is modified using

$$\mathbf{n}_i^k = \frac{\sum_j \mathbf{n}_j^{k-1}}{|\sum_j \mathbf{n}_j|}. \quad (4.2)$$

The smoothing residual for each vertex's growth vector is calculated as a function of the angle between the updated vector and the previous vector,

$$r_i^k = (1 - \mathbf{n}_i^k \cdot \mathbf{n}_i^{k-1}). \quad (4.3)$$

The iterative procedure is stopped once the root-mean-square of the residuals from all of the nodes (i.e. the overall smoothing residual) has fallen below a user tolerance. This stopping condition can be used to tune the level of smoothing applied to the mesh. Figures 4.2a to 4.2d show the effect of smoothing on a square surface. One can see that if no smoothing is applied, there are large changes in cell volumes at the corner of the box. As the smoothing residual stopping condition is lowered, the mesh becomes smoother in the corner regions of the box. In this work, the default outer smoothing residual is 1×10^{-7} and the inner vector is not smoothed.

Once the inner and outer growth vectors, $\hat{\mathbf{v}}_I$ and $\hat{\mathbf{v}}_O$, have been determined, the nodes on the strands can be found using a stretching function, and the inner, arc and total node information. The strands are constructed in such a way that the combined length of the inner strand, arc and outer strand is equal to the strand length input by the user. The distance along the strand of node N_i is given by,

$$L(N_i) = L_s s(N_i), \quad (4.4)$$

where L_s is the strand length and the stretching function, $s(N_i)$, is the hyperbolic tangent function of Vinokur [282],

$$s(N_i) = 1 + \frac{\tanh\left[\delta\left(\frac{N_i}{(N_T-1)} - 1\right)\right]}{\tanh(\delta)}. \quad (4.5)$$

The growth rate, δ , is determined by solving $\Delta s_0 - L_s s(1) = 0$ numerically using a Newton-Raphson method. The strand length and initial spacing are held constant throughout the computation, so the value of δ can be calculated on initialisation and stored, and is the same for all strands.

The location of a node on the inner strand is simply given by

$$\mathbf{p}(N_i) = \mathbf{p}_0 + L(N_i) \hat{\mathbf{v}}_I, \quad (4.6)$$

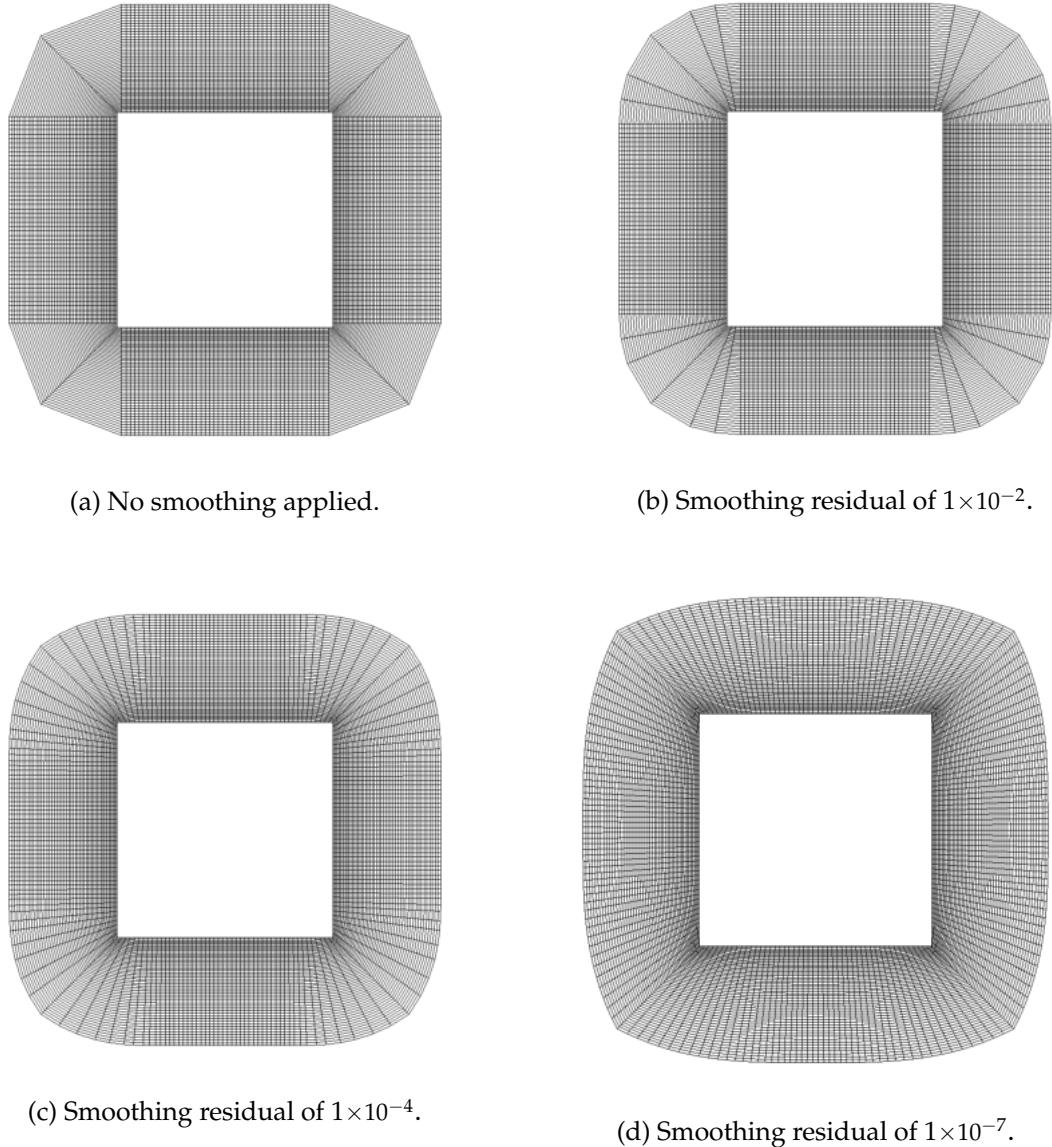


Figure 4.2: A strand mesh generated around a square with different amounts of strand growth vector smoothing.

where \mathbf{p}_0 is the location of the surface vertex. Whether or not an arc is needed is determined using the angle between the inner and outer growth vectors,

$$\theta_c = \arccos(\hat{\mathbf{v}}_I \cdot \hat{\mathbf{v}}_O). \quad (4.7)$$

If θ_c is less than the square-root of machine precision the outer growth vector is assumed to be the same as the inner growth vector and no arc is created.

When an arc is required, the locations of the nodes on the arc are determined by rotating an arc radius vector about an axis at the centre of the arc. In order to do so, the arc rotational axis and arc radius unit vector must be found. These vectors can be

determined under the assumption that the inner and outer growth vectors are tangential to the start and end of the arc:

- The axis of rotation for the arc is the unit vector that is orthogonal to both growth vectors (see Fig. 4.3a),

$$\hat{\mathbf{v}}_A = \frac{(\hat{\mathbf{v}}_I \times \hat{\mathbf{v}}_O)}{|\hat{\mathbf{v}}_I \times \hat{\mathbf{v}}_O|}. \quad (4.8)$$

- The arc radius unit vector is the vector that points from the centre of the arc to the final inner node. This is a vector that is orthogonal to the inner growth vector (as the inner growth vector is tangential to the arc), and in the same plane as both growth vectors (see Fig. 4.3b). Therefore, this vector is given by

$$\hat{\mathbf{v}}_c = \frac{(\hat{\mathbf{v}}_I \times \hat{\mathbf{v}}_A)}{|\hat{\mathbf{v}}_I \times \hat{\mathbf{v}}_A|}, \quad (4.9)$$

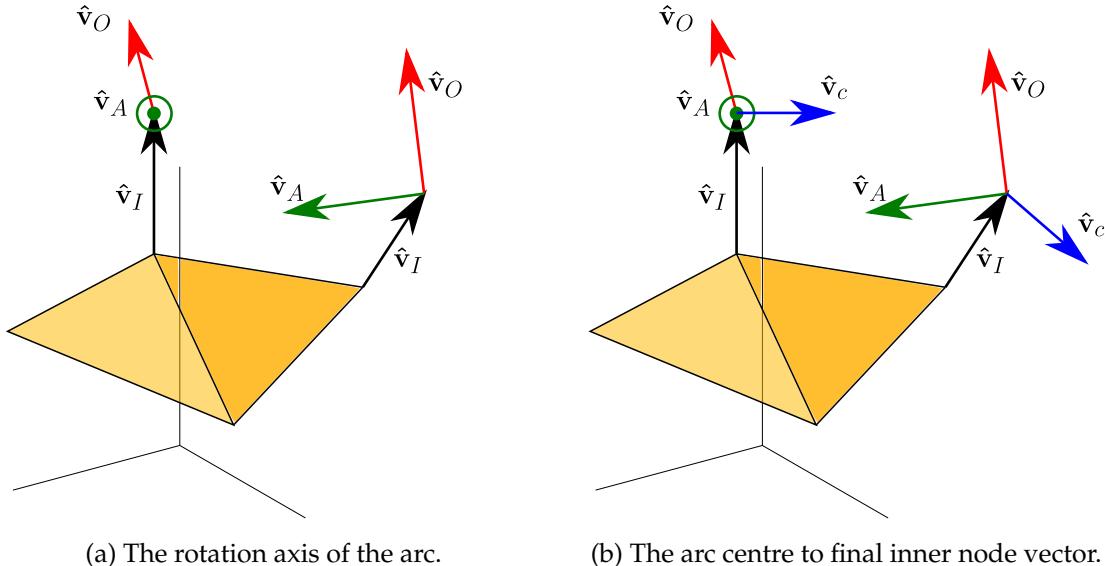


Figure 4.3: Visualisations of the vectors used to create the arc.

The location of the centre of the arc is obtained using the arc length, arc radius and the arc radius unit vector (see Fig. 4.4a):

- The length of the arc, L_c , is determined as the difference in length between the node at the end of the arc and the node at the end of the inner strand,

$$L_c = L(N_c + N_I) - L(N_I). \quad (4.10)$$

- As the arc is tangential to both growth vectors, θ_c is equal to the sweep angle of the arc. The radius of the arc is therefore given by

$$R_c = L_c / \theta_c. \quad (4.11)$$

- The center of the arc is then obtained using

$$\mathbf{p}_c = \mathbf{p}(N_I) - \hat{\mathbf{v}}_c R_c . \quad (4.12)$$

Any number of the above vectors and variables can be stored for each strand in order to reduce the computational expense of finding nodes on the arc. How many parameters are stored (if any) will depend on the number of times the locations of the strand nodes must be found, and on the memory requirements.

Once the required arc parameters have been determined the location of each node on the arc can be found, as shown in Fig. 4.4b:

- The arc angle of node N_i , where $N_I < N_i \leq (N_I + N_c)$, is given by

$$\theta_{c,i} = \frac{L(N_i) - L(N_I)}{R_c} \quad (4.13)$$

- The vector $\hat{\mathbf{v}}_c$ is rotated by $\theta_{c,i}$ about the axis, $\hat{\mathbf{v}}_A$, to give a unit vector pointing from the center of the arc to the node, $\hat{\mathbf{v}}_{c,i}$.
- The node location is then given by

$$\mathbf{p}(N_i) = \mathbf{p}_c + \hat{\mathbf{v}}_{c,i} R_c . \quad (4.14)$$

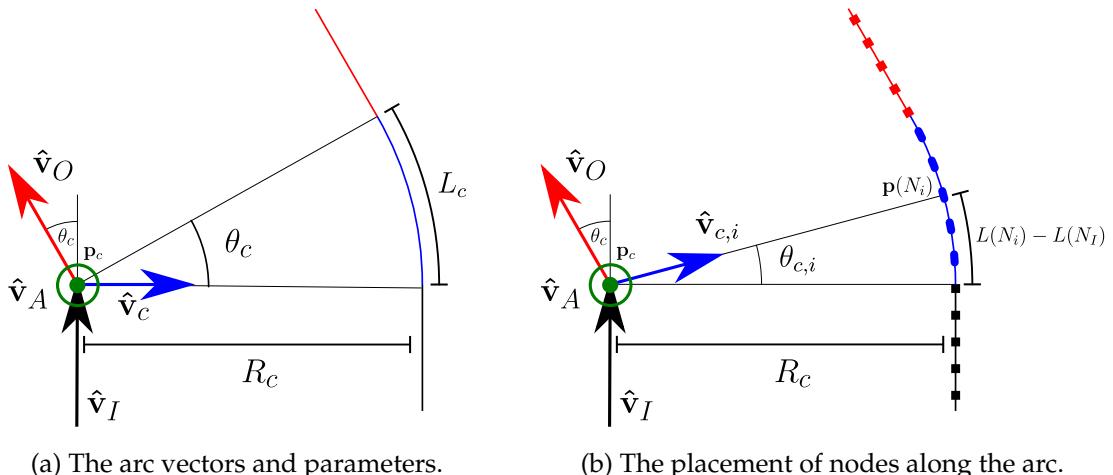


Figure 4.4: Visualisations of the arc vectors and geometric parameters and the placement of nodes along the arc.

The above algorithm has been implemented within a two-dimensional solver, but requires three-dimensional vectors. (In the current two-dimensional implementation three-dimensional growth vectors are used with the z component set to zero.) As such, the algorithm would not require any modifications if used for three-dimensional strand mesh generation.

The locations of nodes on the outer section of the strand are given using the outer growth vector and the distance along the strand from the end of the arc to the node,

$$\mathbf{p}(N_i) = \mathbf{p}(N_I + N_c) + (L(N_i) - L(N_c + N_I))\hat{\mathbf{v}}_O \quad (4.15)$$

It should be noted that the number of arc nodes can be set to zero and this will result in an immediate change from the inner to outer growth vectors, similar to multi-level strand meshes. Also, The method described above is not limited to two levels but could be extended to multiple levels at the cost of additional storage.

A comparison between an arcing strand mesh and a single strand mesh is shown in Fig. 4.5. One can see that a similar level of smoothness is obtained away from the surface, but the arced strands maintain orthogonality at the wall before smoothly transitioning to the outer strand growth vectors.

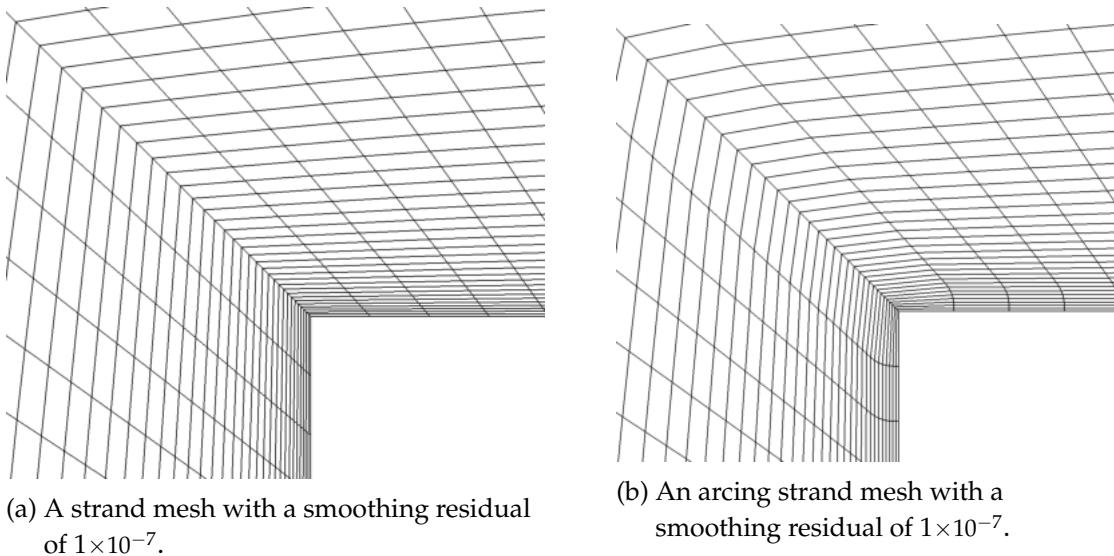


Figure 4.5: A comparison of a strand mesh and arcing strand mesh in the convex region of a surface.

4.2.2 Strand Clipping

Once the nodal positions are established the strands must be checked for any intersections and clipped if necessary. Axis-Aligned Bounding Boxes (AABBs) are created around the three segments of each strand, which can be rapidly tested for intersection. If any are found to overlap then further intersection tests must be carried out. In two-dimensions, the intersection tests can be carried out by equating the parametric forms of the line and arc segments. For example, the distances λ_i and λ_j

where two strand cross are given by

$$\lambda_i = \frac{n_j^x(p_i^y - p_j^y) - n_j^y(p_i^x - p_j^x)}{n_i^x n_j^y - n_j^x n_i^y}, \quad (4.16)$$

$$\lambda_j = \frac{n_i^x(p_i^y - p_j^y) - n_i^y(p_i^x - p_j^x)}{n_i^x n_j^y - n_j^x n_i^y}, \quad (4.17)$$

where \mathbf{p}_i and \mathbf{p}_j are the start positions of strands, and \mathbf{n}_i and \mathbf{n}_j are the strand growth vectors. If the crossing distance is between zero and the strand length, the strand is clipped to an index that reduces the strand length to below the crossing distance. Once the clipping index of each strand has been calculated, the mesh is checked for any strands that protrude above their neighbours and must therefore be reduced. Finally, the clipping index is further reduced by the number of ghost cells to ensure that the ghost cells used in the calculation all have positive volumes.

Figure 4.6 shows the results of the clipping algorithm applied to a strand mesh generated on a double wedge/cone configuration, with different amounts of smoothing. When limited smoothing is applied, one can see that there are a number of strands that have been clipped in the concave region of the mesh. If the outer vectors of the strand mesh are smoothed to a residual of 1×10^{-7} no clipping is required. The reduction in clipping is another benefit obtained when using smoothing, in addition to the smoother grid.

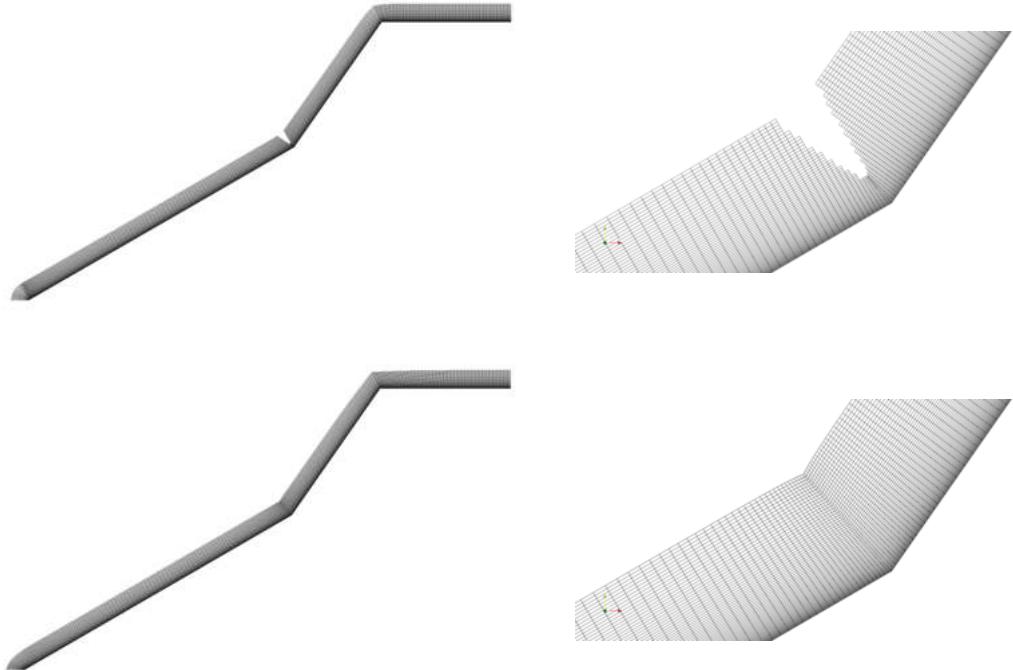
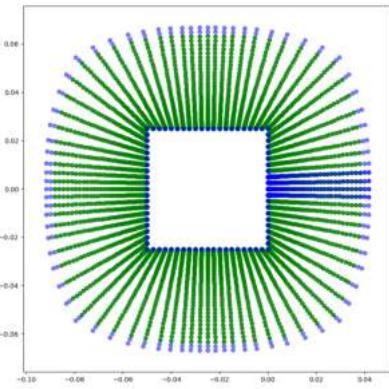


Figure 4.6: A double wedge with strand clipping applied and different amounts of smoothing.

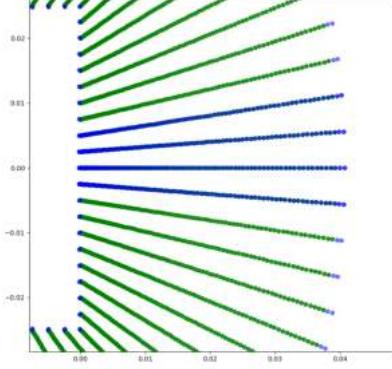
4.2.3 Ghost Cell Creation

Once the strand growth vectors and clipping indices have been determined the ghost cells must be created at each end of the surface, for example at the exit and symmetry boundaries in Fig. 4.6. Ghost cells must also be created below the surface ($N_i < 0$) and at the end of the strands ($N_i > N_T$).

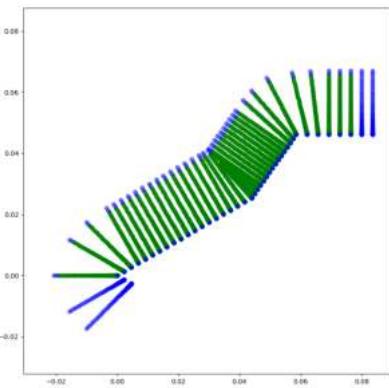
For a periodic surface, the ghost cell vertices and growth vectors are the same as the domain cell that is used to create the periodic boundary condition. The ghost cell surface locations and growth vectors are therefore simply be copied from the corresponding domain cells (see Figs. 4.7a and 4.7b). For a surface where there is a boundary at both ends, the ghost cells are created by reflecting the domain cells across



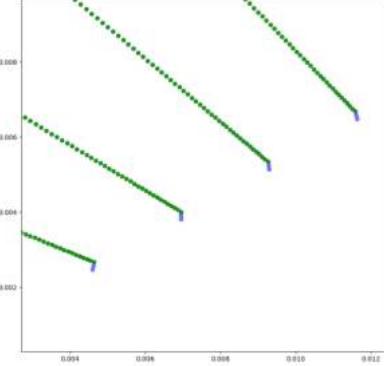
(a) Periodic ghost cell example.



(b) The strand growth normal vectors and surface positions are copied.



(c) Non-periodic ghost cell example.



(d) The surface ghost cells are mirrored about the boundary.

Figure 4.7: Ghost cell creation for a periodic and non-periodic surface. The nodes of the ghost cells are shown in blue, whilst the domain cells are shown in green.

the boundary. This is achieved by rotating the domain growth vectors and surface locations 180 degrees about the boundary growth vector, as shown in Fig. 4.7c. Therefore, when there are symmetry, inflow or outflow boundaries along the sides of the strand mesh the user must input a vector giving the direction of the boundary growth vector. This vector is held constant in the smoothing process.

At the surface of the body, ghost cells are created that are symmetric about the surface in order to simplify the implementation of boundary conditions. To ensure equal spacing from the wall the cells are grown on the interior of the surface using the same growth rate. If the arc-strand growth vectors are normal to the surface, the normal vector is simply multiplied by -1. Otherwise, the growth vector is mirrored about the surface by rotating it 180 degrees around a vector that is tangential to the surface node. The result of this ghost cell creation algorithm can be seen in Fig. 4.7d.

4.2.4 Strand Mesh Motion

The automated meshing enabled by a strand/CAMR solver means that the surface can change shape and the mesh can be regenerated with minimal user input. The high level of automation can be utilised when simulating ablating TPSs, which recess as material is lost through sublimation, pyrolysis and erosion. The method developed in this work is based on the assumption that the strand/CAMR solver will be “loosely” coupled to an ablating material solver. This coupling method uses steady-state fluid results to provide the boundary conditions for unsteady ablative material response simulations. The unsteady material response simulation is terminated after a specified time, change in surface temperature, or change in volume due to ablation. The boundary data from the material response solver is then sent to the fluid solver and used in the next steady-state flow simulation. This process is repeated until the end of the simulation time.

The loose coupling technique is common in hypersonic flow modelling due to the large differences in timescales between the flow and the material response [38, 151]. For example, the characteristic flow time may be $\mathcal{O}(1 \times 10^{-5} \text{ s})$, whilst the material response may be $\mathcal{O}(1\text{s})$. When using loose coupling, the mesh can be regenerated and the flow-field reset to an initial condition. However, to avoid having to re-initialise the flow-field after each surface update, a mesh motion algorithm has been developed to allow surfaces to be modified during a simulation.

For two surfaces with the same number of vertices and the same vertex connectivity, vertex motion vectors, $\mathbf{v}_{m,s}$, can be determined using the start and end location of each vertex (see Fig. 4.8). Each vertex can then be moved along its respective motion vector using

$$\mathbf{p}_0^{n+1} = \mathbf{p}_0^n + (\phi^n + \Delta\phi)\mathbf{v}_{m,s}. \quad (4.18)$$

At the start of the motion $\phi^0 = 0$ and the motion is complete when $\phi = 1$. The distance each node travels in a single time step is determined by $\Delta\phi$. The maximum value of $\Delta\phi$ must be found using a stability condition.

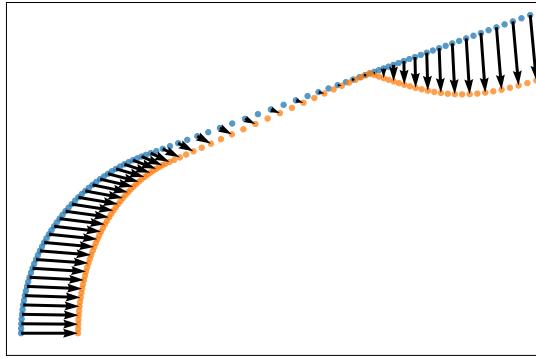


Figure 4.8: Example surface motion vectors for a recessing surface.

When using loose coupling to the material response solver the surface motion does not need to be considered in the fluid equations. As such, the stability of the mesh motion is solely dependent on the mesh geometry. In an overset simulation, the maximum motion of a surface vertex is limited by the change in the shape of the level surface used to create a hole in the background mesh. The level surface is able to deform by a maximum of $N_g \min(\Delta\mathbf{x}_g)$, where N_g is the number of ghost cells on the background mesh and $\Delta\mathbf{x}_g$ is a vector containing the dimensions of the Cartesian ghost cells on the background mesh. Limiting the change in the level surface in this way ensures that all of the cells on the background mesh will contain valid data after the level surface is updated and the hole cutting on the background mesh is performed.

When moving the surface, an initial value of $\Delta\phi$ is selected based on the minimum number of surface motion steps that is input by the user. The positions of the surface vertices are then updated using Eq. (4.18) and the effective hole motion is found from the level surface generated using the strand mesh geometry. If the motion is larger than $\alpha[N_g \min(\Delta\mathbf{x}_g)]$ then the value of $\Delta\phi$ is reduced and the process is repeated. The variable $\alpha \in (0, 1]$ is a user input value that reduces the maximum change in background hole size. Once a value of $\Delta\phi$ has been found that results in the change in the level surface being less than $\alpha[N_g \min(\Delta\mathbf{x}_g)]$, the strand mesh surface can be updated using Eq. (4.18).

4.3 Overset Domain Assembly

Overset techniques are used to connect overlapping computational domains in order to create a single solution. This is achieved by passing information between the domains as boundary conditions. Establishing where the boundaries are on each domain and exchanging the required boundary information between them is generally known as overset domain assembly.

In a strand/CAMR solver, the CAMR domain is used as the background domain and the strand mesh domains are overlaid. Holes must be cut in the CAMR mesh so that, at a minimum, there are no Cartesian cells within the solid bodies used for the strand mesh. The boundary conditions set at the edge of the hole must then be determined using data from the strand domain. The overset boundary cells at the edge of the hole are termed the “receptor” cells. The cells from the strand domain that are used to supply the boundary data for these cells are termed the “donor” cells. Analogously, a strand grid’s receptor cells are located at the exterior of the strand mesh, or at holes that have been cut to accommodate other solid bodies. These must be filled by donor information from either the Cartesian domain, or from other near-body domains. Once all of the data has been exchanged, the boundary conditions can be set to create a continuous solution across all domains.

The overset domain assembly process can be broken down into the following stages:

1. Perform hole cutting.
2. Identify receptor and potential donor cells.
3. Establish communication pattern.
4. Send/receive receptor cell locations.
5. Interpolate data to receptor locations.
6. Send/receive interpolated data.

A flow diagram of the overset grid assembly operations carried out on each processor is shown in Fig. 4.9.

In this work, a set of overset domain assembly algorithms has been developed to enable simulations combining strand and SAMR meshes, each running on multiple processors with distributed memory. When designing the overset domain assembly library the aim was to enable a high level of automation and minimise the computational cost. For high levels of automation, the hole cutting and communication set-up should require minimal user input. For good performance across multiple processors, inter-processor communication should be minimised,

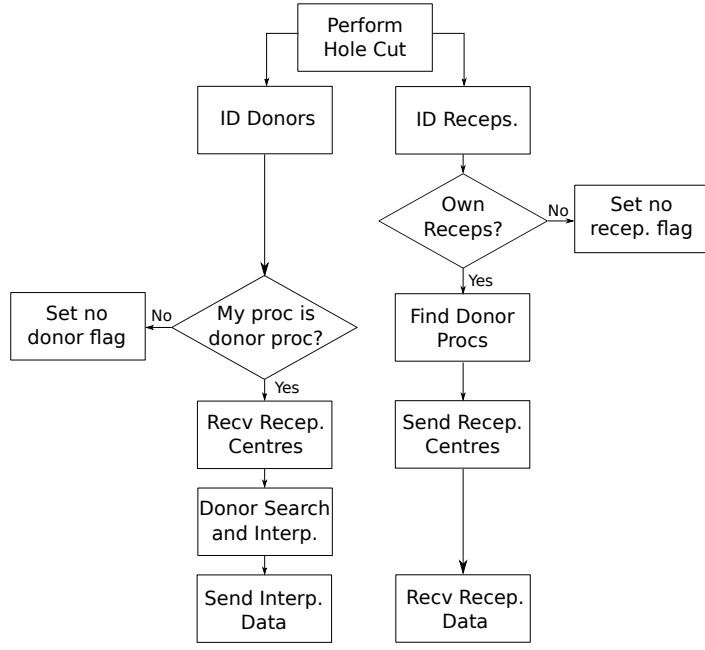


Figure 4.9: A flow chart illustrating the overset grid assembly operations carried out on every processor.

processors should communicate directly, unnecessary donor searches should be avoided (i.e. when the receptor point is outside the processors domain) and the donor search algorithm should be efficient [124, 232]. The algorithms implemented in this work use state-of-the-art methods that account for all of the above factors.

To aid the description, a strand cylinder on a background SAMR mesh is used, where both domains are split across two processors. The two meshes and the parallel domain decomposition of the domains are shown in Fig. 4.10. Both domains use separate MPI groups with separate root processors.

Currently, only a single strand domain can be used. However, the majority of the algorithms could be re-used if the solver is extended to enable multiple near-body domains.

4.3.1 Hole Cutting

Hole-cutting is where cells are blanked out of one domain due to the presence of another domain or solid body, as discussed in Section 2.2.4. The reader is referred back to Fig. 2.14 for an illustration of a hole cut on a Cartesian background domain. In this work, the hole cutting on the background SAMR domain is carried out using the CPT level-set algorithms that were already available within AMROC [182] (see Section 4.1). To maximise the level of automation, the level surface is created using the external boundary of the strand mesh and a cell offset specified by the user. The overlap

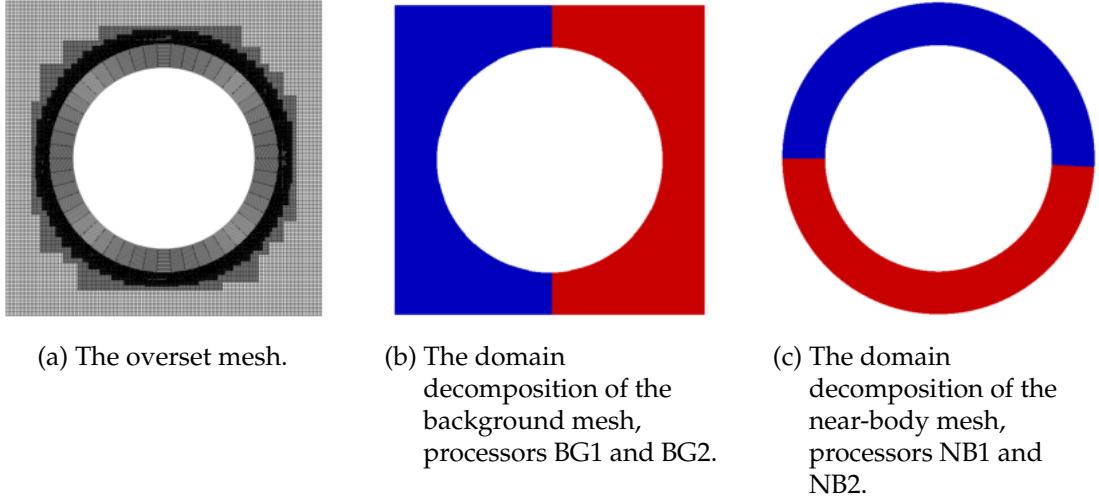


Figure 4.10: The overset mesh and domain decomposition used in the overset grid assembly description.

should be as small as possible, but there must be enough overlapping cells to carry out the interpolation. Smaller overlaps are beneficial as they reduce the total number of solve cells in the computation. In addition, large overlaps may lead to the solutions diverging on each domain if the cell sizes are different. This can create pressure oscillations at the overset boundary.

The hole-cutting level surface algorithm requires a closed surface, so additional user input is needed when simulating strand domains that are not fully enclosed within the background domain. For example, when there are planes of symmetry, the user is able to input a set of points that close the surface. For most geometries this consists of a one or two user input points. The surface describing the hole is calculated on the root processor of the strand solver then sent to the root processor of the SAMR solver, where it is broadcast to the other SAMR processors. The hole cutting algorithm is then called on the background domain. It should be noted that this communication step could be removed as the memory efficient description of the strand mesh would enable each background processor to compute and store the strand mesh geometry. This could be implemented in future work to further increase the efficiency of the background hole cutting routines.

Hole cutting is not currently required on the strand domain, as only a single near-body domain is used. The implementation of multiple near-body domains is discussed in more detail in Section 7.2.

4.3.2 Receptor and Potential Donor Identification

The receptors cells are identified as the overset ghost cells on each domain. The background receptors cells are the ghost cells that were created by the level-set hole

cutting algorithm, and the strand receptor cells are those at the end of each (clipped) strand. In the finite-volume method used in this work, the data is stored at the cell centres, so the cell centre locations of the ghost cells become the receptor points. (An illustration of the ghost cells and receptor centre locations for both domains can be found in Fig. 2.15.) The potential donor cells are any cells that have not been cut from the mesh, either due to hole-cutting or clipping. Once the receptor locations and potential donor cells have been identified, the geometric information required by the communication algorithms can be obtained.

4.3.3 Communication Pattern Set-up

In order to make the inter-processor overset communication as efficient as possible a point-to-point communication pattern is established. This allows information to be exchanged directly between processors with overlapping domains, rather than through root processors. As discussed in Section 4.1, the algorithm developed to establish the communication pattern is an extension of the method detailed in Ref. [66]. In this method, bounding box overlap tests are used to determine which processors should communicate. For an overset method, the boxes used in the overlap tests bound the locations of receptor cells and potential donor cells on each processor. The original method in Ref. [66] utilised Axis-Aligned Bounding Boxes (AABBs). In this work the method has been extended to enable the use of Oriented Bounding Boxes (OBBs) alongside AABBs. An OBB is likely to give a smaller box with a better fit to the cell locations, reducing the probability of false positives in the overlap calculations.

An AABB is simply defined by its lower and upper corners i.e. $(x_{\min}, y_{\min}, z_{\min})$ and $(x_{\max}, y_{\max}, z_{\max})$, and is constructed by finding the maximum and minimum coordinates in each dimension. An OBB is defined by a centre, \mathbf{c} , an orthonormal set of axes describing the orientation, $A = [\mathbf{a}_1, \dots, \mathbf{a}_D]$, and extents along each axis, \mathbf{e} . A schematic of a two-dimensional OBB is shown in Fig. 4.11.

An OBB is created by first finding the orientation axes, then finding the extents, and finally the centre. In this work a covariance-based approach is used to find the orientation axes. In this method, the axes are equivalent to the unit eigenvectors of the covariance matrix of the set of points to be bounded, as described in Ref. [105]. Once the axes are established, the extents can then be found by transforming the points to the new axis system and finding the minimum and maximum in each dimension using

$$e_i = \frac{1}{2} \left(\max_k (\mathbf{a}_i \cdot \mathbf{p}_k) - \min_k (\mathbf{a}_i \cdot \mathbf{p}_k) \right), \quad (4.19)$$

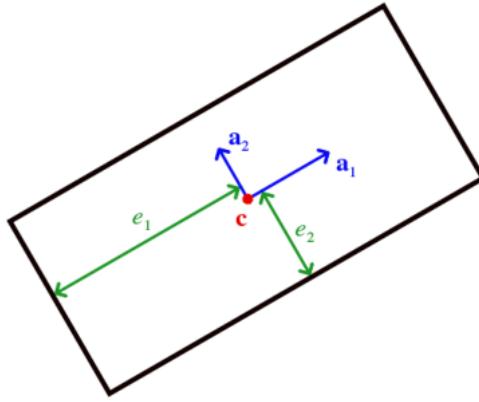


Figure 4.11: The OBB parameters for a two-dimensional OBB.

where $[\mathbf{p}_1, \dots, \mathbf{p}_n]$ are the set of point to be bounded and $k = 1, \dots, n$. The centre of the OBB is found as the mid-point of the range along each axis,

$$\mathbf{c} = \sum_{i=1}^{N_D} \frac{1}{2} \left(\max_k (\mathbf{a}_i \cdot \mathbf{p}_k) + \min_k (\mathbf{a}_i \cdot \mathbf{p}_k) \right) \mathbf{a}_i. \quad (4.20)$$

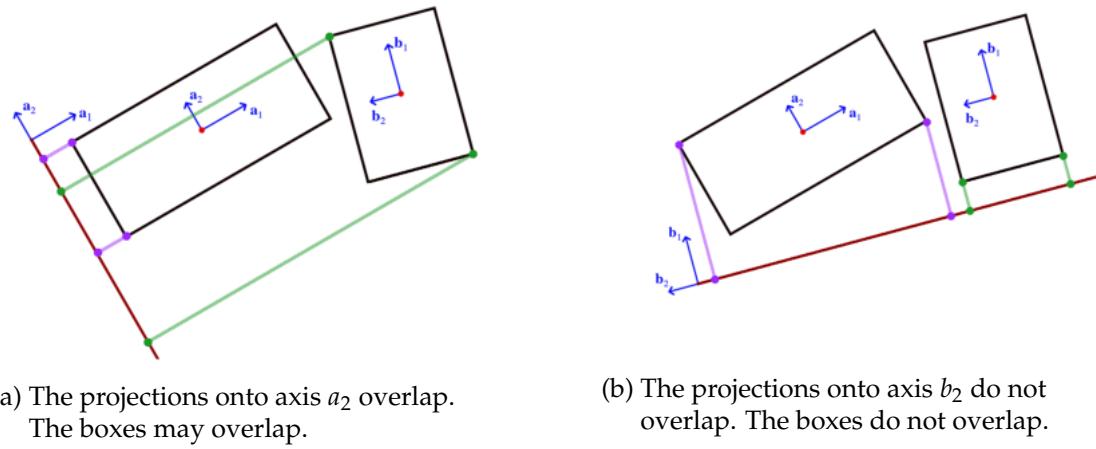
The covariance based approach described above is used within other domain connectivity packages (for example Ref. [232]) as it is computationally efficient and has been shown to give an OBB that is within 10% of the minimum OBB volume for the vast majority of cases using randomly distributed points [105].

OBBs are used to bound the receptor locations on both the SAMR and strand domain, as an OBB is likely to give a better fit to the sets of receptor locations, which may not be aligned with the Cartesian axes. For the donor bounding boxes, AABBs are used for the SAMR grid and OBBs are used for the near-body grids. AABBs are used for the SAMR donor cells as the information is readily available in the AMROC data structures. This makes the cost of constructing the donor cell bounding boxes negligible. In addition, the SAMR domains are themselves axis-aligned, so there is little benefit to be gained from orienting the bounding box. OBBs are used to bound the near-body donor cells to ensure a tight fit to a wide range of geometries. Only the cells adjacent to the boundaries are used to create the OBBs as this can improve the OBB fitting when cells are clustered in one area of the mesh, for example, at the wall in stretched strand grids [105]. Using only boundary-neighbour cells can significantly reduce the computational cost of constructing the donor bounding box, as it only uses a fraction of the total number of cells on the near-body domain.

Once the receptor and donor bounding boxes have been constructed, each processor sends its global rank, donor bounding box and receptor bounding box to its root processor. Each root processor packages this information into vectors and exchanges

this data with the other root processor. Overlap tests are then carried out between the receptor bounding boxes on one domain with the donor bounding boxes from the other domain. These tests are carried out on each root processor in order to determine the communication pattern.

The overlap tests for the OBBs are based on the separating axis theorem, where the corners of each box are projected onto each axis in turn and one-dimensional overlap checks are then carried out [105]. The boxes overlap if and only if all of the projections overlap, so if an overlap check returns false then the function can exit early. A visualisation of the overlap algorithm is shown in Fig. 4.12. When one of the boxes is an AABB knowledge of the axis orientation can be used to reduce the number of calculations. Details of the projection algorithm can be found in Ref. [105].



(a) The projections onto axis a_2 overlap.
The boxes may overlap.

(b) The projections onto axis b_2 do not overlap.
The boxes do not overlap.

Figure 4.12: The overlap algorithm projects the corners of each box onto each axis to determine whether the boxes overlap.

The result of the communication set-up procedure, is that each processor receives four vectors. These vectors contain the information required to determine the point-to-point communication:

1. A vector containing the receptor boxes from other processors that overlap with the donor box of this processor i.e. the send-to boxes vector.
2. A vector containing the global rank of the processor that owns each send-to box i.e. the send-to ranks vector.
3. A vector containing the donor boxes from other processors that overlap with the receptor box of this processor i.e. receive-from boxes vector.
4. A vector containing the global rank of the processor that owns each receive-from box i.e. the receive-from ranks vector.

The bounding boxes and communication vectors must be updated each time the mesh changes. For a static near-body mesh with a fixed domain decomposition, the

bounding boxes may only need to be calculated once on initialisation. For the background domain, the dynamic load balancing can cause changes in the domain decomposition and the AMR can result in receptor cell changes, even when the near-body mesh is static. Therefore, the background boxes and communication vectors must be updated more regularly, and the communication pattern re-established.

4.3.4 Receptor Exchange

In the receptor exchange each processor must send the receptor locations to other processors so that data can be interpolated to the receptor location. Spatial decomposition methods are used to minimise the number of receptors that are sent to donor processors. Minimising the number of receptors not only reduces the amount of data that must be communicated, but also reduces the number of donor searches that are carried out for points that do not reside within a donor processor's domain.

Before the receptors are exchanged, each processor groups its receptors into "spatial buckets" (also described as vision space bins or voxels [140, 232, 253]). These buckets can then be tested for intersections with the receive-from boxes, and only the receptors within overlapping buckets are sent to the donor processor.

Two different strategies are used to create the spatial buckets. On the near-body domain, the buckets are created using an auxiliary, oriented Cartesian grid that is based on the receptor OBB created during the communication set-up. Each bucket in the auxiliary grid is itself an OBB with the same axis as the larger receptor OBB. The receptors are sorted into their respective buckets by transforming each receptor to the OBB coordinate system and finding the Cartesian bucket index using

$$\tilde{\mathbf{x}}_R = A (\mathbf{x}_R - \mathbf{c}) + \mathbf{e}, \quad (4.21)$$

and

$$\mathbf{i} = \left\lfloor \frac{\tilde{\mathbf{x}}_R}{\Delta \tilde{\mathbf{x}}} \right\rfloor. \quad (4.22)$$

In the above, \mathbf{x}_R is the location of the receptor centre, \mathbf{i} is a vector containing the bucket index in each dimension, $\Delta \tilde{\mathbf{x}}$ is a vector containing the bucket length in each dimension, and $\lfloor \cdot \rfloor$ is the floor operator.

On the background domain a receptor bucket is created by placing an OBB around the receptors within a single SAMR block. The SAMR blocks already group clusters of cells together and the resulting OBBs are tighter fitting than those in the auxiliary grid. Figure 4.13 shows the two types buckets created in the cylinder example.

The receptor processor carries out OBB overlap tests between the receptor buckets and the receive-from box of each donor processor. Only receptor locations stored in

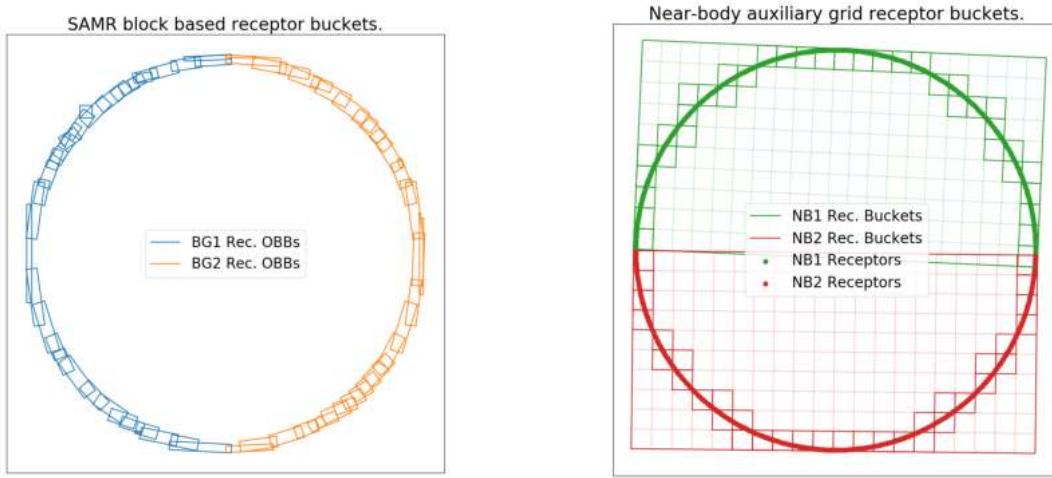


Figure 4.13: A visualisation of the receptor spatial buckets on the background processors (left) and near-body processors (right) in the cylinder example.

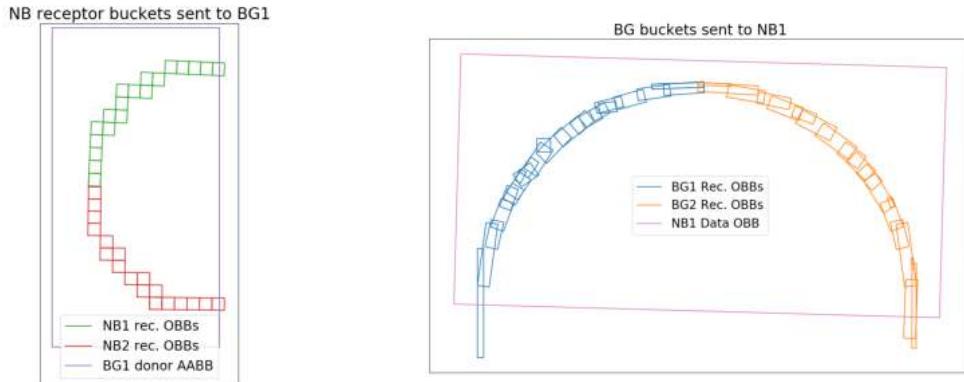


Figure 4.14: The receptor buckets that are exchanged following overlap tests between the near-body processors and background processors.

overlapping buckets are sent to the receive-from processor. Figure 4.14 shows the buckets that are sent once the overlap tests have been carried out for the cylinder example (cf. Fig. 4.13).

On the SAMR domain the number of receptor buckets is determined by the number of SAMR blocks containing receptors. For the near-body auxiliary grid the number buckets within the OBB must be chosen, and it has an impact on the memory requirements and performance [140, 253]. With more buckets, the space is discretised more finely, so fewer receptors are likely to be sent unnecessarily. However, more overlap comparisons must be carried out and more storage is required. In this work $n_R/2$ buckets are used , where n_R is the number of receptors within the OBB. This value was previously shown to give a good compromise between performance and memory, when using a similar spatial decomposition strategy [253].

The receptor buckets are created at the same time as the receptor bounding boxes are formed so that the receptors are only processed once. Therefore, the buckets must also be recreated when the receptor bounding boxes are updated, for example when the domain decomposition or mesh changes. As stated above, for a static near-body mesh the process may only need to be carried out once. For the background mesh the process needs to be carried out each time the AMR is updated.

4.3.5 Donor Search and Interpolation

The receptor cell centres sent from a receptor processor become interpolation points on the receiving, donor processor. Each interpolation point must be searched for and, if it is within the donor processor's data domain, the variables at the interpolation point must be determined.

Pre-existing AMROC interpolation routines are used on the background SAMR domain, as discussed in Section 4.1. On a given SAMR block, the cell indices containing the point can be found easily using the AMROC data structure,

$$\mathbf{i} = \left\lfloor \frac{\mathbf{x}_p - \mathbf{x}_0}{\Delta \mathbf{x}} \right\rfloor, \quad (4.23)$$

where \mathbf{i} is a vector containing the cell index in each dimension, \mathbf{x}_p is the location of the interpolation point, \mathbf{x}_0 is the lower corner of the block and $\Delta \mathbf{x}$ is the vector of Cartesian cell face lengths and $\lfloor \cdot \rfloor$ is the floor operator.

On the mapped mesh, the cells surrounding a given interpolation point cannot be found directly as the identity of the cells is dependent on the mapping. As discussed in Section 2.2.4, various methods are described in the literature for identifying the donor cell that contains an interpolation point. In this work, three classes of method have been implemented and tested: a structured stencil walking method, range queries in higher dimensions, and inverse map methods.

4.3.5.1 Stencil Walking Method

A stencil walking algorithm for structured grids has been implemented using the method described in Ref. [188]. The method tests to see if the interpolation point lies within a cell, and if not, whether the point lies within a cell with a higher or lower i or j index.

To determine if a point, \mathbf{x}_p , lies within a cell, the coordinates of the point are found in computational space, $\mathbf{s}_p = (\xi, \eta)$. The mapping must be such that the computational coordinates within the cell are given by $0 < \xi < 1$ and $0 < \eta < 1$, where the lower left hand corner of the cell is the origin of the coordinate system, as shown in Fig. 4.15.

Thus, if either of the coordinates of \mathbf{s}_P is greater than one or less than zero, the point must lie outside of the cell.

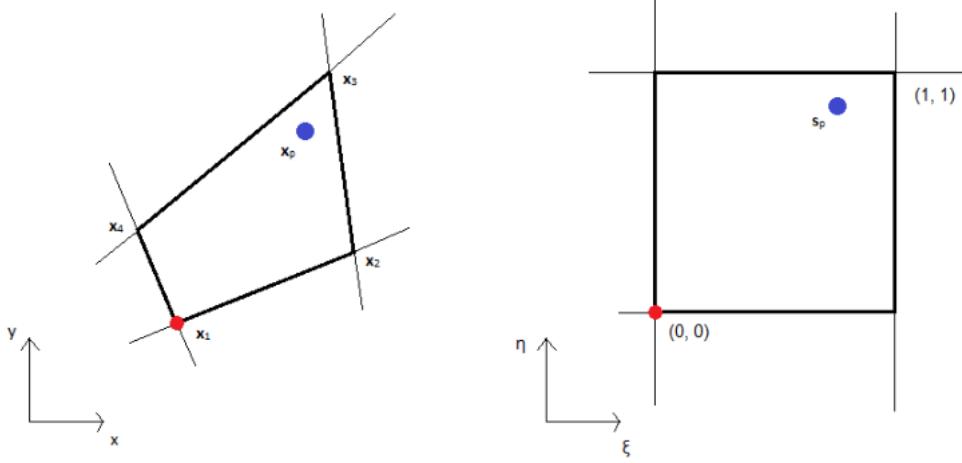


Figure 4.15: The cell in physical (x, y) space is mapped to (ξ, η) space.

In this work the mapping between the computational coordinates and the physical coordinates is based on bi-linear interpolation. As such, the values of \mathbf{s}_P are equivalent to the interpolation weights required to interpolate a value from the cell nodes to the interpolation point, \mathbf{x}_P . Thus, a point \mathbf{x} is given as a function of computational coordinates \mathbf{s} by

$$\mathbf{x}(\mathbf{s}) = \mathbf{x}_1 + (-\mathbf{x}_1 + \mathbf{x}_2)\xi + (-\mathbf{x}_1 + \mathbf{x}_4)\eta + (\mathbf{x}_1 - \mathbf{x}_2 + \mathbf{x}_3 - \mathbf{x}_4)\xi\eta, \quad (4.24)$$

where \mathbf{x}_{1-4} are the coordinates in physical space of the cell nodes
 $n_1 = (i, j)$, $n_2 = (i + 1, j)$, $n_3 = (i + 1, j + 1)$, $n_4 = (i, j + 1)$ (see Fig. 4.15).

Given a set of nodes at locations \mathbf{x}_{1-4} , and an interpolation point, \mathbf{x}_P , the values of the computational coordinates, \mathbf{s}_P , are found by solving the equation

$$\mathbf{x}(\mathbf{s}_P) - \mathbf{x}_P = 0. \quad (4.25)$$

The equation can be solved directly, however, it is difficult to determine *a priori* which solution of the quadratic equation should be chosen. Therefore, a Newton-Raphson method is used to solve the equation, where the value of \mathbf{s}_P is found iteratively using

$$\Delta \mathbf{s} = (\mathbf{x}_P - \mathbf{x}(\mathbf{s})) \left[\frac{\partial \mathbf{x}(\mathbf{s})}{\partial \mathbf{s}} \right]^{-1}, \quad (4.26)$$

and the Jacobian is determined analytically from Eq. (4.24).

Once the iterative scheme has converged to \mathbf{s}_P , the computational coordinates can be used to determine whether the interpolation point lies within the cell, and if not, the direction of the stencil walk. If both computational coordinates are between zero and

one then \mathbf{x}_p lies within the cell. If the value of ξ or η is less than one then the corresponding index value should be decreased. Alternatively, if the value of ξ or η is greater than one, then the corresponding index should be increased.

Different strategies have been used to improve the efficiency of the stencil walk algorithm on a mapped mesh. As discussed in Section 2.2.4, the choice of the starting location for the search has a large impact on the robustness and cost of the search, and various methods have been proposed to obtain a good starting guess. In this work, the starting node is chosen to be the boundary node closest to the interpolation point. The closest node is evaluated using the squared distance to remove the computational cost of calculating square-roots. The method requires no storage, is relatively cheap to calculate, ensures the number of steps is less than half the minimum number of cells in any dimension, and minimises the risk of a boundary in the mesh impeding the search, resulting in false negatives.

If the interpolation point does not lie within the domain then the search should fail. In this work, the search is deemed to have failed once the search algorithm has tried to step outside of the domain in one direction five times. This criteria allows for the search to walk along the edge of the domain for a number of steps, which may be required for certain mesh topologies. If the search does fail, then it is restarted from a different point as the results of the search are dependent on the starting point and mesh topology. The need for multiple searches results in a high computational cost for points that lie outside the domain. The receptor exchange algorithms described above help to minimise the number of failed searches.

4.3.5.2 Range Query Methods

Point-box intersection tests can be re-formulated as range searches in higher dimensional space and used to conduct donor searches. The basic idea of the method is that any range in \mathbb{R}^N can be mapped to a single point in \mathbb{R}^{2N} . For example, a three-dimensional bounding box given by the ranges $[x_{\min}, x_{\max}]$, $[y_{\min}, y_{\max}]$ and $[z_{\min}, z_{\max}]$ can be mapped to a single point: $(x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max})$. This is shown graphically for a set of one-dimensional ranges in Fig. 4.16. The ranges in the left hand graph map to points of the same colour in the right hand graph. In donor searches, AABBs bounding each potential donor cell are the ranges used to create the higher dimensional points.

Once all of the cells' AABBs have been converted to the higher dimensional points, intersection tests can be carried out using range queries. To find all of the bounding boxes that a point, \mathbf{x}_p , intersects, one performs a range query in the higher dimensional space to find points that lie in the range $[\mathbf{r}_{\min}, \mathbf{r}_{\max}]$, where

$$\mathbf{r}_{\min} = (\mathbf{x}_{\min}^\delta, \mathbf{x}_p), \quad (4.27)$$

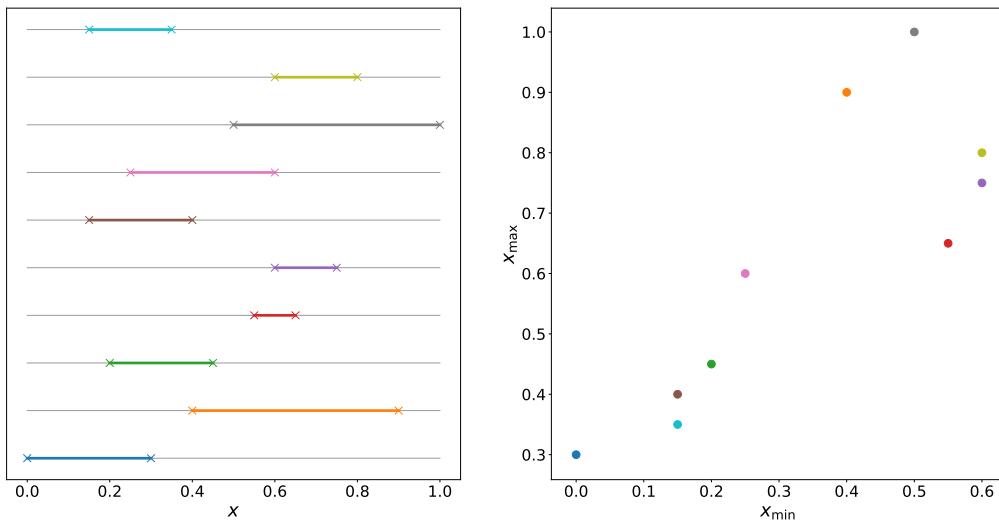


Figure 4.16: Examples of one-dimensional ranges transformed to two-dimensional points.

and

$$\mathbf{r}_{\max} = (x_p, x_{\max}^g), \quad (4.28)$$

where x_{\min}^g and x_{\max}^g are the global minimum and maximum points of all of the computational cell bounding boxes. This is depicted in one dimension in Fig. 4.17, where the point marked by the \times in the left hand image is mapped to a range “window” on the right hand image. It can be seen that the ranges that the point intersects with are all within the window. Any method can be used to carry out the range queries.

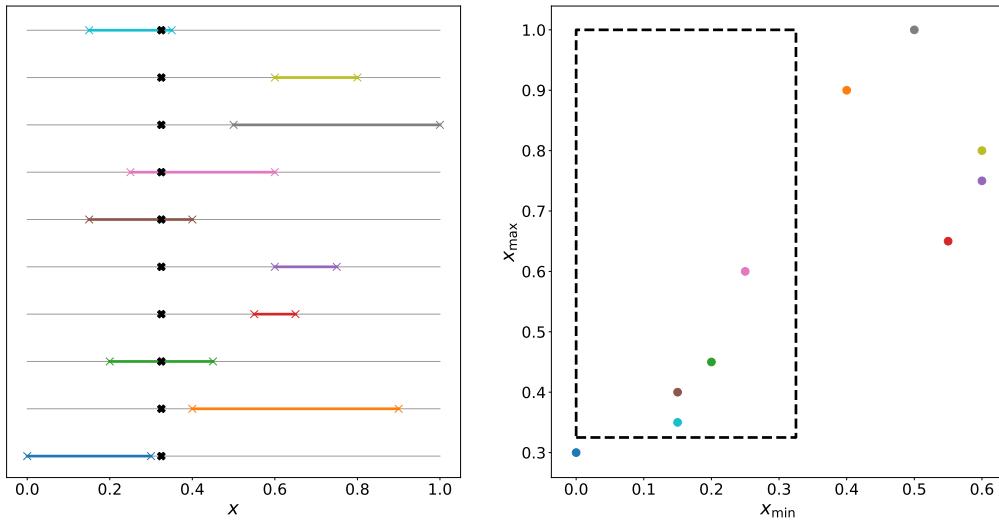


Figure 4.17: An example of an intersection search for a point and a set of one-dimensional ranges.

Bonet [27], was one of the first to use this method in computational meshing applications. In Bonet's work the \mathbb{R}^{2N} points were stored in in *kd*-trees, and this data structure was used to carry out the range searches. This method is usually referred to as an Alternating Digital Tree (ADT). This is the most commonly used range search method in donor search algorithms [232, 264], as it is not disadvantaged by the sparsity of the higher dimensional space that is a consequence of mapping of boxes to higher dimensions (see Fig. 4.16). The *kd*-tree is created by recursively partitioning the space, and storing points either side of the partition in separate branches. The partition is carried out on different dimensions at each level of the tree. The value used to partition the space is chosen as the median value of all the points, as this creates a balanced tree, with an approximately equal number of points either side of the partition. The downside of using the median value is that the points must be sorted during the creation of the tree. Once a partition contains less than a user specified number of points, the points are stored in a "leaf". The partitioning used to create a *kd*-tree for the one-dimensional example is depicted in Fig. 4.18.

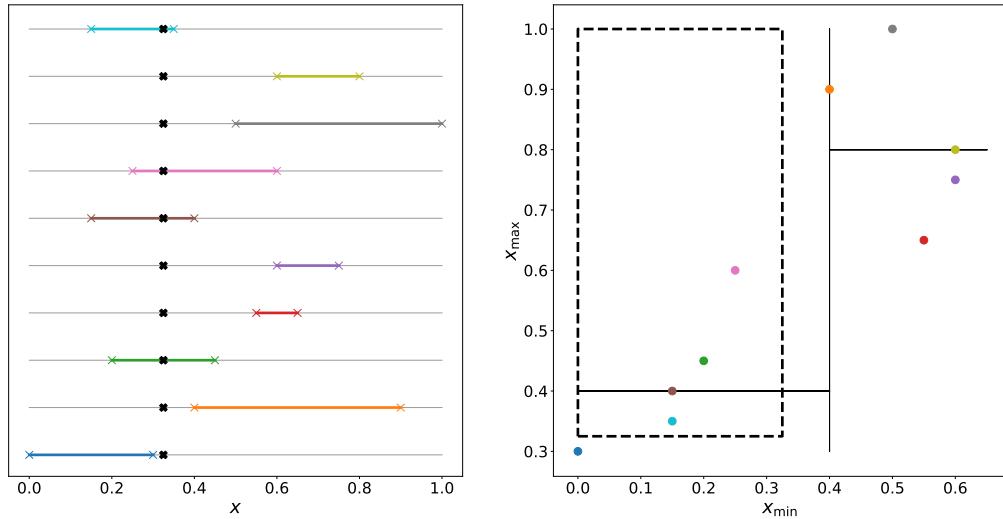


Figure 4.18: An illustration of the *kd*-tree domain decomposition, overlaid with a range search window.

For a *kd*-tree, the computational complexity for the creation time is $\mathcal{O}(N \log N)$ and for storage is $\mathcal{O}(N)$, where N is the number of points and the points are pre-sorted in each dimension. The *kd*-tree data structure allows for efficient range searches as large number of points can be quickly excluded. For example, in Fig. 4.18 only the branches on the left hand side of the first partition need to be traversed, excluding half of the points. The complexity in time of range queries using a *kd*-tree is $\mathcal{O}(\log N + \tilde{I})$, where \tilde{I} is the number of points returned.

Another method that is commonly used for range searches is the cell or bucket array based method. Similar to the receptor bucketing algorithm described above, this method uses a k dimensional grid and stores the points in the buckets of the grid.

Range searches are then conducted by mapping the bounds of the range to minimum and maximum bucket indices. The points in all of the buckets that are entirely inside the minimum and maximum indices are then returned. For buckets on the boundaries of the range, each point within the bucket must be tested for inclusion. A depiction of the bucketing method range search for the one-dimensional example is shown in Fig. 4.19.

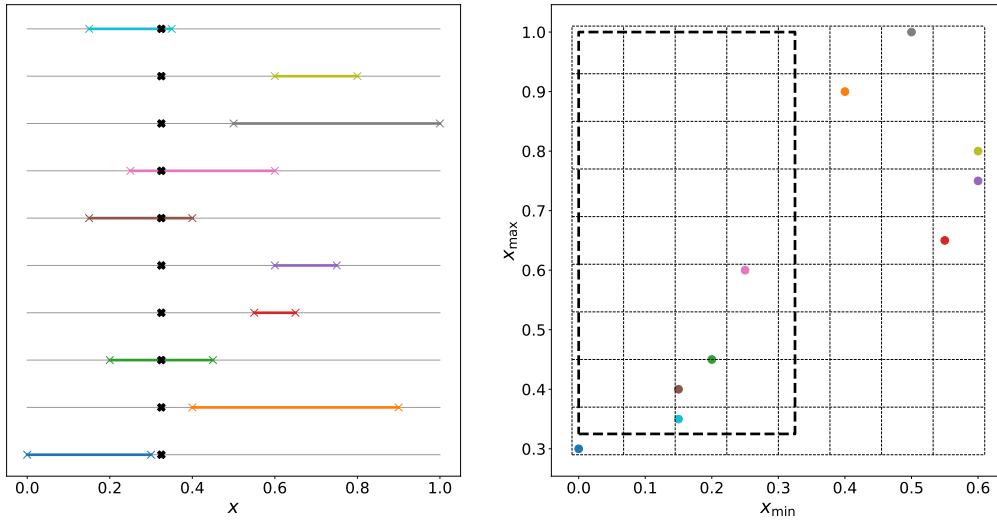


Figure 4.19: An example of a bucket array overlaid with a range search window.

The computational complexity for the creation time and storage of the bucket array is given by $\mathcal{O}(N + M)$ where M is the number of buckets in the array. The complexity in time of the range search is given by $\mathcal{O}(J + \tilde{I})$, where J is the number of cells that overlap the range window. As can be seen in Fig. 4.19 the large range window and sparse distribution of the higher-dimensional points can result in a significant number of empty buckets being returned, which could impact the performance. From the above it is clear that the size of the buckets is an important factor in determining the performance and storage. Although the kd-tree method (a.k.a. the ADT method) is widely used for donor searches, no examples of bucket array range searches could be found in the literature.

Within the AMROC eco-system a range query library is available. This library contains an implementation of the kd-tree method and the bucket array method. Both have been integrated with the donor search routines. The donor search routine calculates the AABB of each cell, maps them to higher-dimensional space and constructs the kd-tree and bucket array. The interpolation points are used to set the ranges used in the range queries, which return the computational cells with overlapping AABBs. An inclusion test is then carried out on each of the computational cells using the same bi-linear interpolation methods as used in the stencil walk. Once

the cell containing the interpolation point is found, the search exits and the next interpolation point is searched for.

4.3.5.3 Inverse Map Methods

Inverse map methods [187, 232], also referred to as Structured Auxiliary Meshes [140], can be used to carry out robust donor searches. In these methods a bucket array is first created that encompasses the computational domain. A cell is allocated to a bucket if it overlaps with the bucket. The donor search is then carried out by determining the bucket indices of the interpolation point and conducting inclusion tests on all of the computational cells within the bucket. Interpolation points that are mapped to an empty bucket cannot be within any cells, and can therefore be rapidly excluded.

The performance and storage of an inverse map method is influenced by the bucket sizes and the cost of the bucket-cell overlap tests. Smaller buckets discretise the domain more finely and will result in fewer inclusion tests being carried out as fewer computational cells will overlap with each bucket. However, the storage is increased when more buckets are used. Cell-bucket overlap tests can be carried out using the exact geometry of the computational cell or a volume which bounds the cell, such as an OBB, AABB or sphere. Using the exact cell geometry will result in a more precise inverse map, which could reduce the number of inclusion tests that are conducted. However, the computational cost of creating the map is increased due to the more complex overlap tests that must be carried out.

Some differences in the methods are illustrated in Fig. 4.20. It can be seen that the OBB grid gives smaller buckets, even though both bucket arrays have the same number of buckets, and the same minimum and maximum points in both dimensions. The figure also shows that the exact cell geometry will overlap with fewer OBB buckets than the cell AABB.

Comparisons of the performance of inverse maps using different bucket sizes and both exact cell geometries and cell bounding boxes was carried out in Ref. [140]. As expected, the results showed that the search speed increased as the bucket size was reduced, but the memory usage increased as well. The authors empirically found that the highest speed-to-memory ratio was achieved when buckets 5-10 times larger than the average cell were used. The results also showed that the search times when using exact cell overlaps were faster than for the bounding box overlaps in some cases, but slower in others. In addition, the set-up times for the exact method were significantly longer than when using bounding boxes.

In this work, two types of inverse map have been implemented and tested: an oriented bucket array with axis-aligned cell bounding boxes, referred to as an Oriented Inverse Map (OIM), and an axis-aligned bucket array with axis-aligned cell

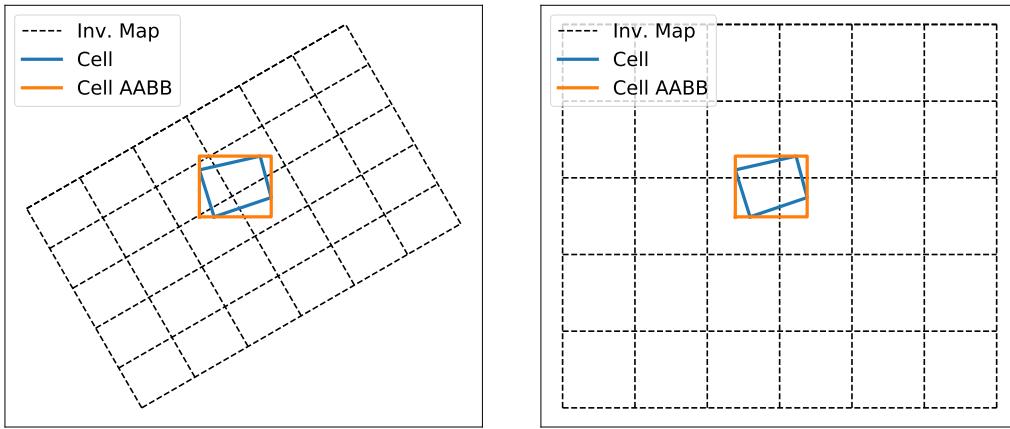


Figure 4.20: Examples of OBB (left) and AABB (right) inverse maps. The blue outlines represent an exact cell geometry whilst the orange outlines represent the AABB of the cell.

bounding boxes referred to as an Axis-Aligned Inverse Map (AAIM). Axis-aligned cell bounding boxes are used as the inverse maps may need to be recreated frequently as the near-body mesh changes due to surface recession. The additional cost of creating the exact inverse map is likely to outweigh the benefits of conducting fewer inclusion tests. In addition, storing the cell bounding boxes enables efficient AABB inclusion tests to be carried out for each interpolation point, before the more expensive bi-linear interpolation inclusion test is carried out.

4.3.5.4 Mapped Mesh Interpolation

Once the cell containing the interpolation point has been identified, a dual-grid method [204] is used to interpolate the data. In this method the data is interpolated using bi-linear interpolation from surrounding cell centres to give a monotonic, second-order-accurate interpolation scheme. The interpolation weights, based on the surrounding cell centre locations, are found using Eqs. (4.24), (4.25), (4.26). If a donor processor does not contain the query point the receptor data is set to be a null value that signifies that the interpolation was not carried out.

It should be noted that using linear interpolation to ghost cells does not ensure conservation across the overset boundary [255, 268]. This has been shown to lead to slower convergence and oscillations in the pressure close to the overset boundary [268]. Methods that improve the level of conservation by considering the fluxes at the overset boundary have previously been developed [53, 58, 268]. However, these methods are considerably more complex and the more commonly used interpolation methods have been shown to give excellent results in a wide range of practical

simulations [137, 154, 196, 204, 260, 290, 291]. The implementation of conservative methods is beyond the scope of this work, but could be considered in future research.

4.3.5.5 Locality Sensitive Hashing

Robust donor searches on arbitrarily mapped domains can be the most computationally costly part of the overset domain assembly. Therefore, an effective way to reduce the computational cost is to reduce the number of searches that are required. In this work, the number of near-body donor searches has been significantly reduced by using knowledge of the underlying structure of the SAMR background grid. A SAMR grid that is distributed across a number of processors can contain a large proportion of receptors that are located at the same point in space. This is because communication across block and processor boundaries requires overlapping cells. Figure 4.21 shows the receptor locations for the cylinder case when a large number of ghost cells are used. In this case, approximately one quarter of the receptor cells around the cylindrical hole are co-located.

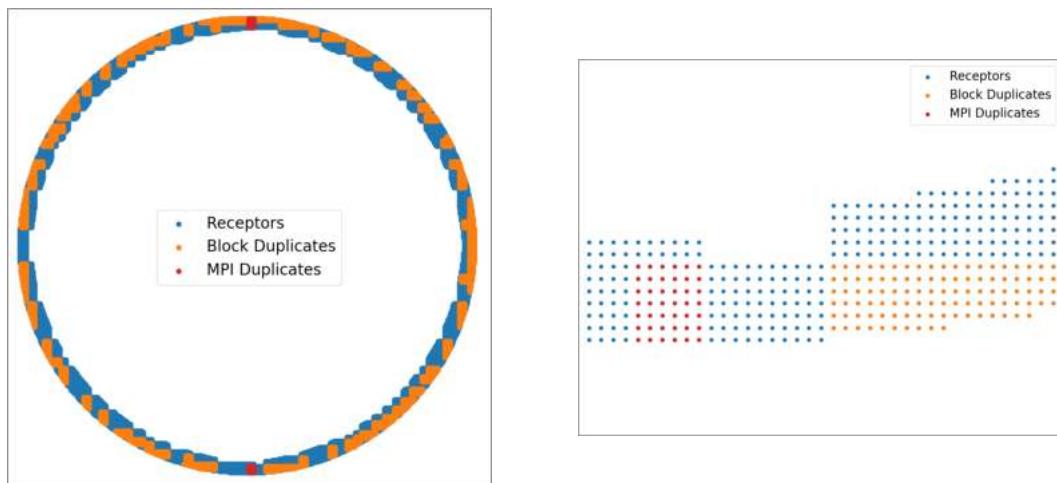


Figure 4.21: The duplicate receptors created in the cylinder example when using a large number of ghost cells.

Many points will also be co-located before and after the SAMR mesh is adapted or redistributed across processors. When such an update occurs, the number and locations of the receptors may change or the processor requesting the data at a given location may change. However, if the mapped mesh is static and the hole remains in the same location, many of the SAMR receptors will be in the same locations, so a new donor search will not be required. Therefore, the number of donor searches can be reduced by screening for co-located interpolation points.

In this work, repeated searches are avoided through the use of Locality Sensitive Hashing (LSH). This involves the use of a hash function that maps each receptor's

coordinates to a near-unique integer and gives the same integer for two co-located coordinates. The hash function will map two points to the same integer even when there are minor differences in coordinates due to floating point precision. In the function, each spatial coordinate is multiplied by a tolerance, cast to an unsigned integer and then the integers are combined using bit-wise operations. The code used in the hash function is given in Appendix A.4.

The hash function, and a function to check for hash collisions, are used to create a hash table which stores the donor search results for each location. For each interpolation point the hash table is checked and if the donor search for a given location has already been carried out then the interpolation parameters are simply copied from the table. If the location is not in the table then the donor search is carried out and the result is stored in the hash table. The average cost of accessing existing elements in a hash table is $\mathcal{O}(1)$ and using LSH leads to a significant reduction in the number of searches on the mapped grid. For the initial search, any co-located interpolation points on SAMR block or processor boundaries are only searched for once. After SAMR mesh updates, which are typically every time step on the coarse SAMR mesh, searches are only carried out for new locations.

Table 4.1 shows the number of searches undertaken by a near-body processor in the cylinder example. The number of the initial searches is significantly reduced and very few subsequent searches are needed, even though the SAMR mesh is changing (as shown by the changes in the number of interpolation points sent from the SAMR mesh). This behaviour was seen for a range of simulations. The hash collision results show that the hash function is effective at creating distinct integers for each location. In this scenario a hash collision is when two points give the same hash but are located in different positions.

Time step	No. Points	No. Searches	Hash Collisions
1	11,471	8,290	0
2	11,317	283	0
3	11,193	0	0

Table 4.1: The reduction in searches when using LSH. Data is from one near-body processor, using the receptors shown in Fig. 4.21.

To the best of the author’s knowledge, this is the first time that LSH has been used to remove duplicate donor searches in an overset grid assembly library. This new method significantly reduces the number of searches required when using a static near-body domain with an adaptive background domain. Consequently, it could lead to performance improvements in other strand/CAMR solvers.

4.3.6 Boundary Data Exchange

Once the interpolation has been carried out, the data at each interpolation point is sent to the processor that requested the data. The data is received by the processor, unpacked using the overlapping bucket information and stored so that it can be used to set the boundary conditions.

4.3.7 Verification

Order-of-accuracy tests have been carried out to verify the donor search and interpolation algorithms on the mapped domain. In these tests, a passive scalar is interpolated to the receptor locations of the AMR mesh. The magnitude of the scalar is set to be a nonlinear function of the spatial coordinates. The order-of-accuracy of the interpolation routines can then be calculated by refining the near-body mesh and evaluating the interpolation error at each receptor location. The L_∞ norm of the absolute difference between the analytic and interpolated values is used as this highlights any failed searches. The linear interpolation used in this work is expected to converge with second-order-accuracy when interpolating a smooth, nonlinear function. The nonlinear function used to carry out the tests is given by

$$c = c_0 - c_x \sin(a_x x) + c_y \cos(a_y y). \quad (4.29)$$

Two different geometries were used for the order-of-accuracy tests in order to assess the robustness of the donor search algorithms. The cylindrical geometry, shown in Figs. 4.22, was chosen as it is an analogue of an overset simulation of a wing or vehicle, where the mesh is periodic and contains an internal boundary. The sinusoidal geometry, shown in Fig. 4.23, was chosen to test the donor search on a nonlinear mapping, and to contrast with the cylinder mesh, as it is not periodic and does not contain an internal boundary.

The results from the tests are shown in Table 4.2 and 4.3. A refinement factor of two was used and the tables show the L_∞ norm on each mesh, along with the observed order-of-accuracy. One can see that the mapped interpolation routines are converging at approximately second-order-accuracy as expected. On each mesh, the tests were carried out with different numbers of processors across the near-body and background domains and with different parallel distribution patterns within the mapped domain. The same tests were carried out for each of the near-body donor search methods. All tests gave the exact same results. The AMROC SAMR interpolation routines were not tested as they were implemented and verified in previous work [182].

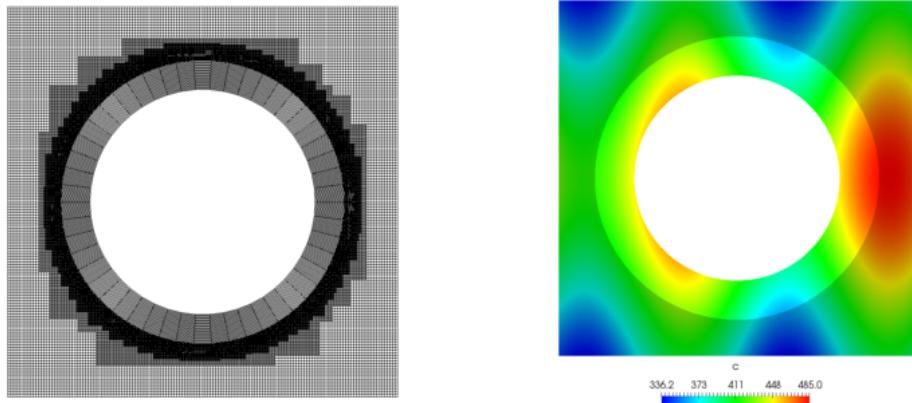


Figure 4.22: The cylindrical overset mesh (left) and scalar field (right) used in the interpolation order-of-accuracy tests.

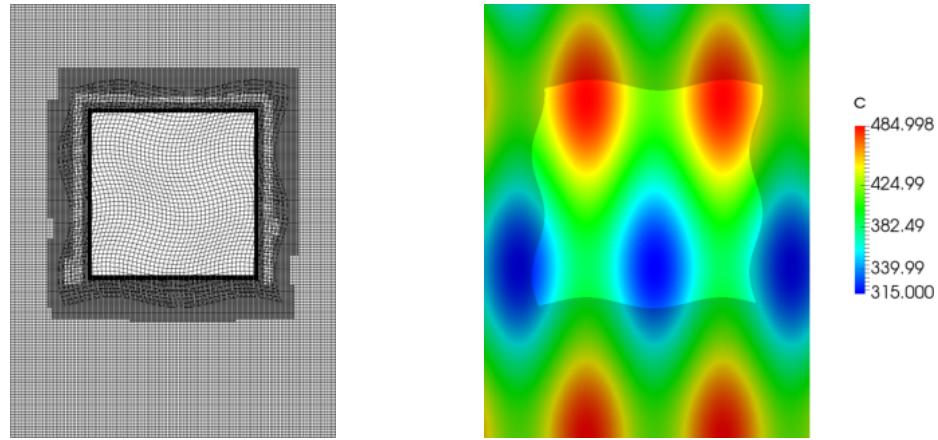


Figure 4.23: The sinusoidal overset mesh (left) and scalar field (right) used in the interpolation order-of-accuracy tests.

Mesh Dim.	L_∞ Norms	Observed Order	Interpolation Points
50×50	1.060782	N/A	22,710
99×99	0.2765216	1.94	21,749
197×197	0.0697965	1.99	20,473
393×393	0.01788589	1.96	20,711

Table 4.2: The results from the order-of-accuracy test on the cylindrical mesh.

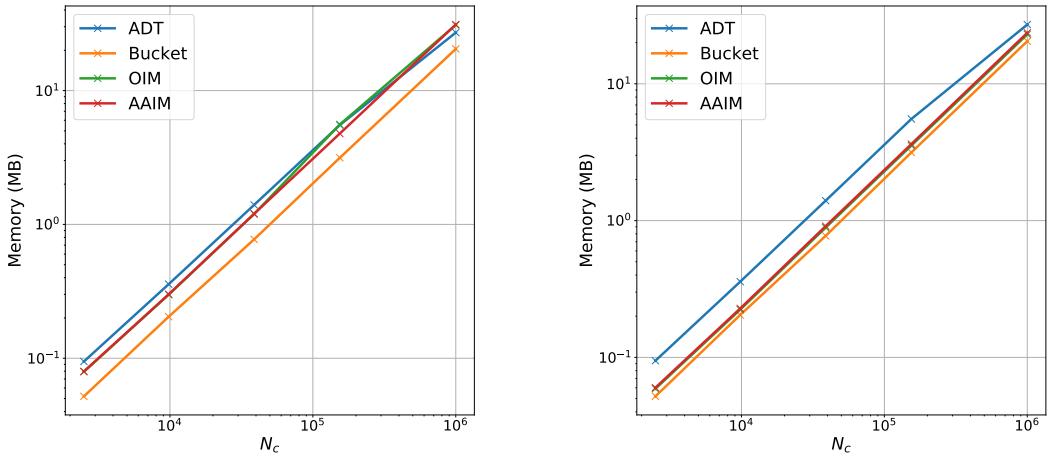
Mesh Dim.	L_∞ Norms	Observed Order	Interpolation Points
50×50	0.2280968	N/A	11,782
99×99	0.0589774	1.95	11,686
197×197	0.01488462	1.99	11,637
393×393	0.004015702	1.89	11,615

Table 4.3: The results from the order-of-accuracy test on the sinusoidal mesh.

4.3.8 Donor Search Performance

As previously discussed, several donor search algorithms have been implemented on the near-body domains. The different methods have been profiled to determine which method should be favoured as the default. In addition to the four meshes used in the verification test, a larger 1000×1000 cell mapped domain was also included. The number of interpolation points that were searched for in each case was approximately constant and is shown in the tables above. For the tests, the ADT method used a leaf size of 8, the bucket array range query method used $N_c/2$ buckets, where N_c is the number of cells, and the inverse map used a constant bucket to cell volume ratio of 5:1. The tests were conducted on an Intel i5-6300HQ (2.3GHz) and the mapped domain was run on a single core. The code was compiled using the GNU compiler suite v5.4.0 with the `-O3` optimisation flag.

The memory requirements for the different methods are shown in Fig. 4.24. All of the methods have been implemented in a memory efficient manner, where the search methods store pointers to a single vector containing each cell's AABB and global index. The figures only show the memory required for the search method (i.e. the search methods internal data structure and the pointer storage), as the storage required for the cell data is the same for all methods. It should be noted that no storage is required for the stencil walk. The results show that the memory requirements for each method are similar, with the bucket array range query method generally having the lowest memory footprint and the ADT method the highest.

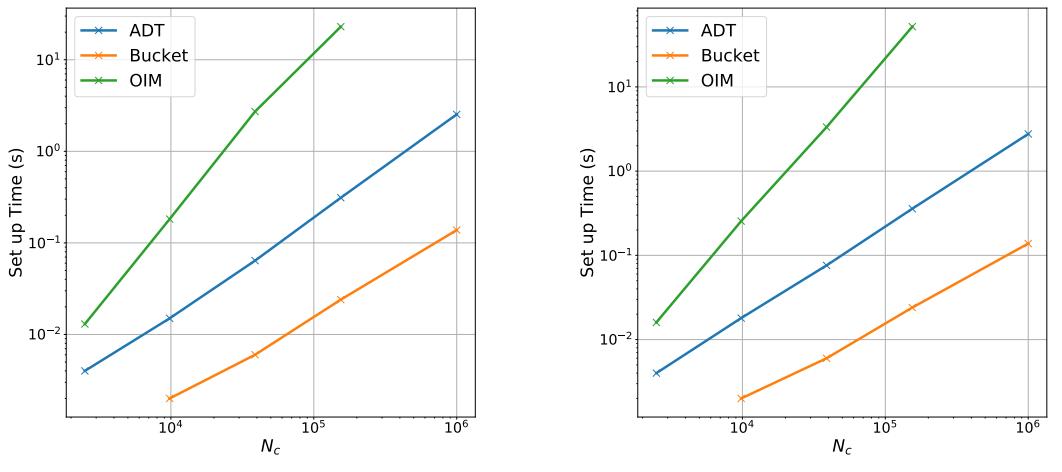


(a) The cylindrical geometry memory usage. (b) The sinusoidal geometry memory usage.

Figure 4.24: The total memory requirements across both processors for the different search methods.

The time required for the set-up of each method is shown in Fig. 4.25, where the set-up time for the stencil walk is zero. It can be seen that the set-up times for the inverse map methods scale poorly with the number of cells; similar results were obtained in

Ref. [140]. This was due to a naive implementation of the set-up algorithm where every bucket was assessed for overlap with every cell. A more efficient algorithm was implemented that reduces the number of overlap tests that are carried out. The new algorithm uses the cell's AABB geometry to reduce the number of bucket overlap tests. This is achieved by mapping the corners of the cell's AABB to bucket indices. The minimum and maximum bucket index in each direction give the index range of the buckets that could overlap. For the OIM, only the buckets within this index range need to be checked for overlap. For the AAIM, the buckets within this range are guaranteed to overlap, so can be added without carrying out an overlap test. This said, the overlap tests have been retained for the AAIM so that the same base class is used by both inverse maps and exact cell overlap tests can be used in future work.



(a) The cylindrical geometry set-up times.

(b) The sinusoidal geometry set-up times.

Figure 4.25: The set-up times for the different methods, tested using an Intel i5-6300HQ processor and GNU v5.4.0 compiler.

The set-up times using the new implementation are shown in Fig. 4.26. It can be seen that an orders-of-magnitude improvement in the set-up time is obtained for both inverse map methods, and the set-up time scales well with the number of cells. To the authors knowledge this is the first time that this more efficient inverse map set-up algorithm has been used in donor search methods and has a major benefit on the efficiency of the methods. The shortest set-up times were given by the bucket based range query method. As expected, the AAIM gave slightly lower set-up times than the OIM due to the simpler overlap tests. The AAIM times could be further improved by removing the explicit overlap tests. The kd-tree set-up times were generally slowest and scaled worst with the number of cells for both geometries.

The search times for each method are shown in Fig. 4.27. In all cases tested, the inverse map methods gave the lowest computational times and were orders-of-magnitude faster than the other methods for the largest meshes. The inverse map methods scaled extremely well with the number of cells, with the search time for the largest mesh

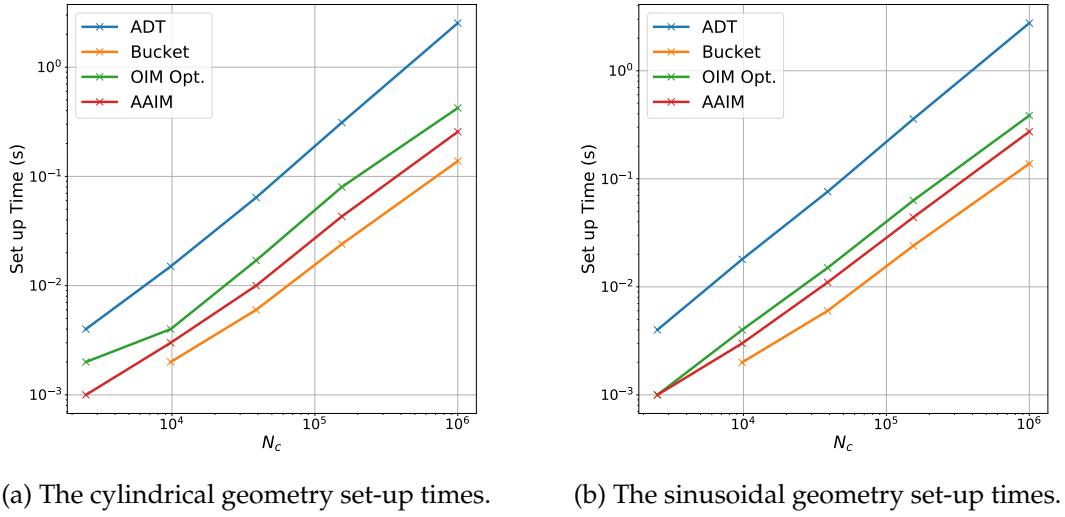


Figure 4.26: The set-up times for the different methods with the optimised inverse map creation, tested using an Intel i5-6300HQ processor and GNU v5.4.0 compiler.

being less than 50% greater than the search time for the smallest mesh in both cases. The two range query methods gave similar performance, but the fastest range query method was dependent on the grid size and shape. Neither range query method scaled particularly well, but the results indicate that the kd-tree method may scale better than the bucket array range query method. The stencil walk performance appears to be highly dependent on the geometry and number of cells, but is outperformed by the inverse map searches in all cases.

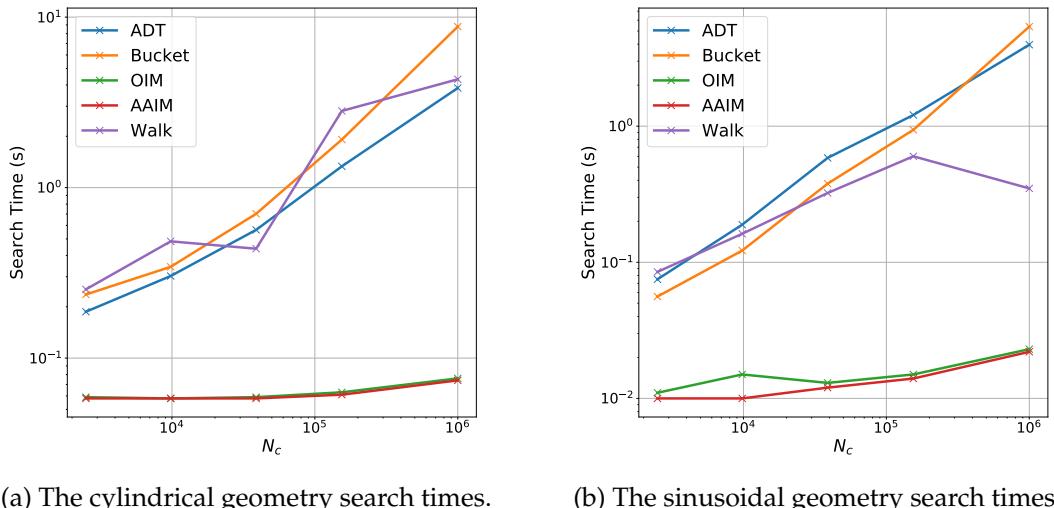


Figure 4.27: The search times for the different methods, tested using an Intel i5-6300HQ processor and GNU v5.4.0 compiler.

The above test cases give a good indication of the performance and scaling of the

different methods. However, the geometries are reasonably simple, and align well with bounding boxes, which could give the inverse map methods an advantage. As such, more complex overset strand mesh geometries were also tested. The overset mesh geometries are shown in Fig. 4.28. The set-up and search time results for these geometries, along with the 393×393 cylinder and sinusoidal mapping cases, are shown in Fig. 4.29, where the times have been normalised by the largest time. Only the *kd-tree* range query method was included as this is the most widely used in the overset literature.

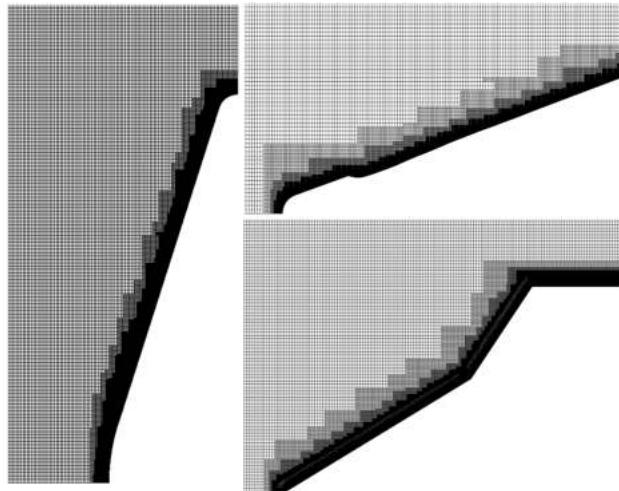
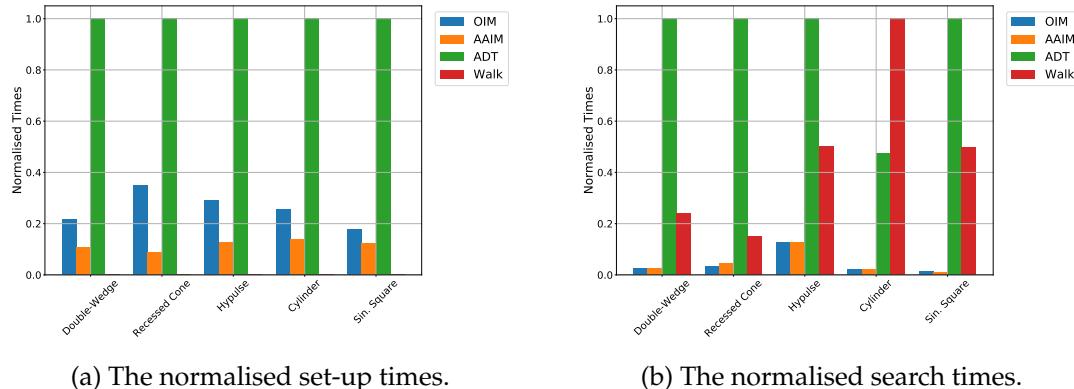


Figure 4.28: Overset strand meshes around a re-entry capsule, recessed cone and double wedge.



(a) The normalised set-up times.

(b) The normalised search times.

Figure 4.29: The donor set-up and search times for a range of mesh geometries, tested using an Intel i5-6300HQ processor and GNU v5.4.0 compiler.

It can clearly be seen that the new AAIM method has the shortest set-up times, and the *kd-tree* method has the slowest set-up times across all geometries. Regarding the search times, both inverse map methods perform similarly well, being orders-of-magnitude faster than the slowest method. The *kd-tree* method is slowest in all cases apart from the cylindrical mesh case, where the stencil walk is slowest. These results are consistent with findings from other authors that have shown auxiliary map

methods give significantly better search performance than tree based methods [112, 140].

In light of the performance results, either of the inverse map methods should be used as the default donor search method. Although they have larger set-up times and memory requirements than the stencil walk method, these are lower than the kd-tree range query method and similar to the bucket array range query method. The inverse map methods outperform all other methods when searching and scale extremely well with the number of cells on the mapped grid. In addition, they will not suffer from the robustness issues that can arise with the stencil walking method in areas where the mapping is highly nonlinear. One final advantage of the OIM method, which isn't explored here, is that it does not have to be regenerated if the near-body mesh is rotated and/or translated through space e.g. in simulations of moving bodies. If the near-body mesh does not deform the inverse map domain can simply be rotated and translated according to the body motion, and the inverse map will still be valid¹.

All of the donor search methods are retained in the software and can be selected by the user.

4.4 Chapter Summary

A novel strand mesh generation algorithm that is better suited to hypersonic flow simulations has been developed. This method maintains orthogonality at the wall, whilst giving a low memory footprint and high mesh smoothness away from the wall. The strand generation method has been integrated with the mapped mesh algorithms detailed in Section 3.2. In addition, a surface deformation algorithm has been tightly coupled to the strand mesh generator. A new overset grid assembly library has been implemented that includes parallel, point-to-point communication, state-of-the-art donor search methods and a novel LSH technique. A number of tests were carried out to verify the donor search and interpolation routines and find the most efficient donor search methods. The inverse map methods, that utilise a novel set-up algorithm, were shown to be fast and robust for a number of different geometries and are recommended as the default method.

The aim of the extensive additions to the AMROC framework presented in the previous two chapters is to enable highly-automated and efficient hypersonic flow simulations using the strand/CAMR paradigm. In the following chapters a range of test cases are used to examine the new strand/CAMR solver's ability to efficiently and accurately predict hypersonic flows, and to simulate deforming surfaces and moving bodies with a high-level of automation.

¹The cell bounding boxes will also need to be transformed in order to carry out bounding box inclusion tests on each point during the search.

Chapter 5

Verification and Validation

One of the primary research questions outlined in the introduction was whether overset methods can accurately predict heat fluxes in hypersonic flow simulations. In this chapter, evidence is gathered to address this question through a number of test cases and comparisons. In addition, the computational performance of the new solver is assessed. The MMS, Blasius boundary layer and overset verification results previously shown provide verification evidence for the individual components of the strand/CAMR solver. This chapter adds further verification and validation evidence for the combined overset solver¹.

An overset order-of-accuracy test is performed, then a vortex shedding test case is used to demonstrate the significantly improved performance of the new strand meshing technique. The nonequilibrium model is validated through comparisons of the experimental and predicted shock stand-off distances in hypersonic flows. The accuracy of the new solver's heat flux predictions are then assessed through comparisons with experimental and single-domain results. Simulations of blunt geometries are used to assess the accuracy of the steady-state heat flux predictions. A shock-shock interaction case is then used to test the overset solver's performance when there are shock structures crossing the overset boundary, impinging on the surface. Previously, there have been very few investigations into the influence of overset meshes on hypersonic heat fluxes, especially in the presences of complex shock structures. As a result, the evidence obtained in this chapter is a key contribution of this thesis.

Verification and validation of the automated surface motion algorithms is then carried out. Two recessing surface test cases are used to assess the surface deformation algorithms. Both recessed surfaces contain concave regions that mesh morphing algorithms common to other hypersonic flow solvers would struggle to accommodate.

¹Additional, blowing wall and compressible flat plate verification studies can also be found in Appendices A.5 and A.6, respectively.

The first surface deformation test case allows for comparisons with experimental results, whilst the second case demonstrates the surface deformation capability in the presence of large surface recession and dynamic shock structures. Finally, the simulation of moving bodies is investigated. The ALE fluxes and equations of motion are verified. Free body simulations of cylinders interacting with an oblique shock are then used to demonstrate the highly-automated shock capturing capabilities of the new solver.

5.1 Overset Order-of-Accuracy Tests

Order-of-accuracy tests were conducted using a time-accurate test case to verify the implementation of the spatial and temporal integration methods, and the overset algorithms. In the test case an isobaric density increase with a Gaussian profile is linearly advected through the domain. The initial density field is given by

$$\rho = \rho_0 + \rho_a \exp\left(-\frac{x^2 + y^2}{R^2}\right), \quad (5.1)$$

where ρ is the density, ρ_a is the user-specified amplitude and R is a user-specified constant that controls the radius of the density bump. The velocity in each dimension is input by the user and held constant over the simulation. Periodic boundary conditions are used for all of the boundaries so that the Gaussian bump can move through any boundary and return to the domain. This test case was selected for the order-of-accuracy tests as the smooth solution means that the MUSCL limiting can be turned off and full second-order-accuracy can be achieved. To allow the second-order-accuracy of the overset and spatial schemes to be verified in this unsteady test case, an explicit second-order-accurate Strong Stability Preserving (SSP) Runge-Kutta scheme was implemented within the overset solver, where overset exchanges are carried out prior to each Runge-Kutta stage.

For the order-of-accuracy tests the bump parameters were set to be $\rho_a = 1.0$ and $R = 0.25$. A domain of $x \in [-1, 1]$ and $y \in [-1, 1]$ was used and the x and y velocities were set to be 1.0. The simulation was run for two time units, resulting in the bump moving through the corners of the boundary and returning to the centre of the domain. The domain consisted of a mapped mesh that was fully embedded into a single-level background mesh. The mapping was similar to that used in the spatial integration and overset verification tests given by Eqs.(3.129) and (3.130). A visualisation of the overset domain and advected density pulse is shown in Fig. 5.1. The observed order-of-accuracy was obtained by uniformly refining the grid from 50×50 cells to 200×200 cells and the time step from 8 ms to 2 ms. AMROC's in-built error estimators were used to compare the numerical solution to the analytic solution

at each time step. The L2-norm of the density error was used as the measure of the numerical accuracy.

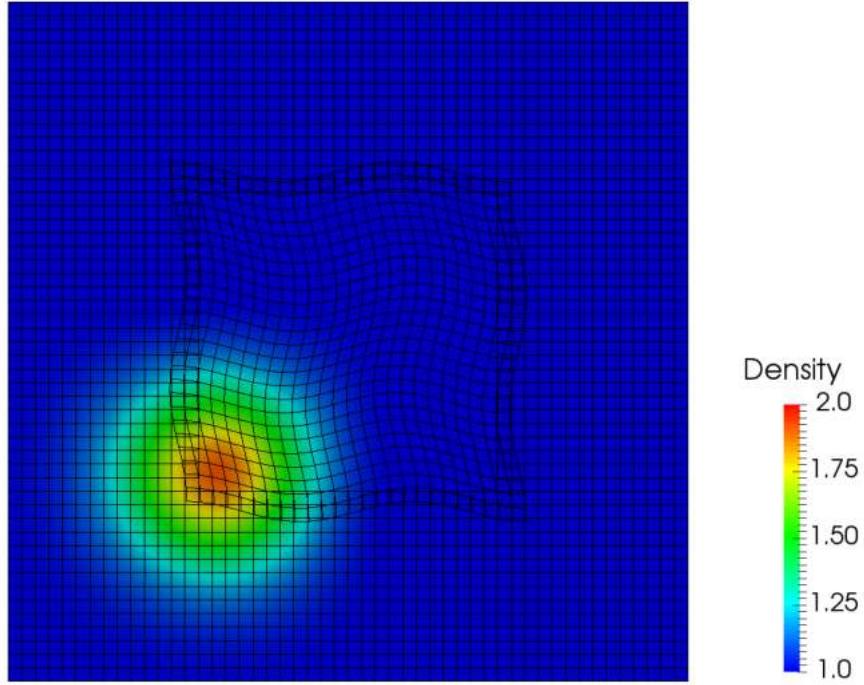


Figure 5.1: The overset domain used for the order-of-accuracy tests, showing the advected Gaussian density bump.

The order-of-accuracy tests were carried out for different overset time-stepping methods. The first used the SSP Runge-Kutta method on both domains (SSP2-SSP2), the second used the forward Euler method on the off-body domain and an unpartitioned backward Euler (FE-BE) method on the near-body domain and the final simulation used the forward Euler method on the off-body domain and the dimensionally partitioned forward/backward Euler method on the near-body domain (FE-F/BE). The results from the order-of-accuracy tests are shown in Fig. 5.2. It can be seen that all of the methods give the expected order-of-accuracy. These results provide excellent verification evidence for the mapped mesh spatial integration, overset routines and time integration methods.

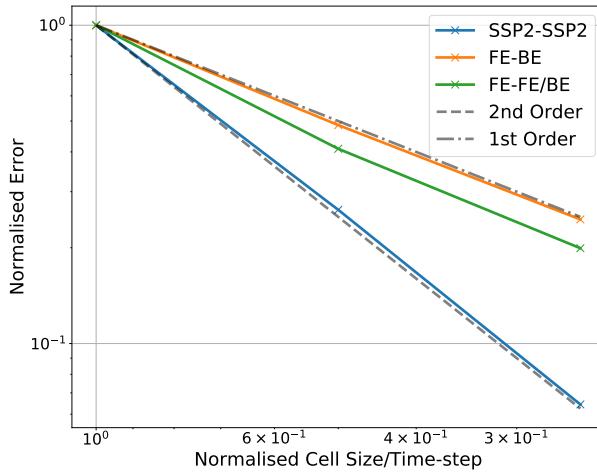


Figure 5.2: The results from the overset order-of-accuracy tests.

5.2 Vortex Shedding Over a Square Cylinder

When using the original strand meshing technique, it was found that the level of smoothing could impact the simulation results. This was because the smoothing resulted in the near-wall cells becoming less orthogonal to the surface [137]. The detrimental effect of smoothing was demonstrated in simulations of low Reynolds number flow over a square cylinder. In these simulations, low levels of smoothing produced the expected regular vortex shedding but high levels of smoothing caused the vortex shedding to become “chaotic” [137]. These simulations were replicated to investigate the behaviour with new strand meshing technique. The new method ensures orthogonality at the surface so should give more consistent results under different levels of smoothing.

The simulation was set up to replicate the case presented in Ref. [137] as closely as possible. An ideal gas model was used and the inflow conditions were set to give a Reynolds number of 250 based on the square side length, $L = 0.01$ m. The freestream conditions were given by $U = 33.92$ m/s and $\rho = 12.73$ g/m³, with a viscosity of 1.73×10^{-5} Pa s. The strand mesh used 70 cells in the wall-normal direction, with strand lengths of $1.25L$ and an initial spacing of $10\ \mu\text{m}$. The square surface contained 60 faces on each side, and a no-slip wall condition was used on the surface of the square. Four near-body meshes were used with the smoothing residual varying between 1×10^{-5} and 1×10^{-8} in factors of 10 (see Fig. 5.3). The off-body domain was extended to a distance of $40L$ in each dimension to minimise the influence of the boundaries on the flow. Static mesh refinement was used in the off-body grid to refine the region around the cylinder and the wake. The domain and a visualisation of the y -velocity is shown in Fig. 5.4. One notable difference in the set-up used here and that

in Ref. [137] is that the results in Ref. [137] were calculated using a three-dimensional solver.

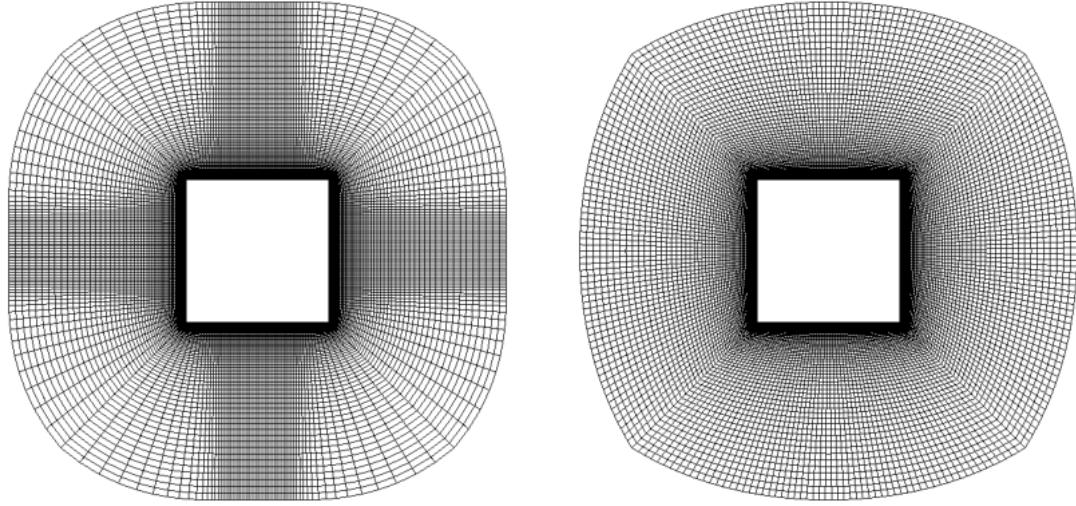


Figure 5.3: The near-body mesh with smoothing residuals of 1×10^{-5} (left) and 1×10^{-8} (right).

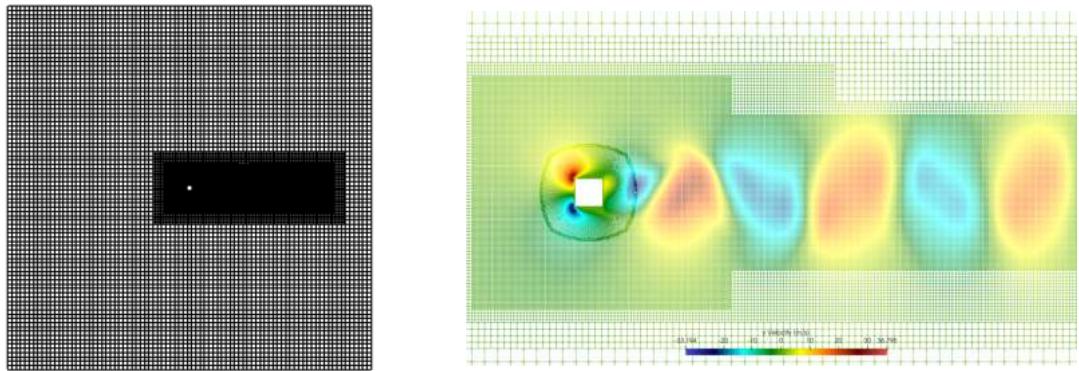


Figure 5.4: The strand/CAMR mesh for the square cylinder simulation (left) and the y -velocity profile showing the vortex shedding (right).

The simulations were run for 0.05 s to allow the unsteady vortex shedding to become established and more than 10 cycles to complete. The frequency of the vortex shedding was calculated using the variation in the lift coefficient over time. The non-dimensional frequency (Strouhal number, $Str = fL/U$) of the lift coefficient oscillations reached an approximately constant value after 0.025 s. The average Strouhal number between 0.025 s and 0.05 s was then calculated. A temporal convergence study was carried out using the strand mesh with a smoothing residual

of 1.0×10^{-6} . The results showed that the lift-coefficient Strouhal number was the same to four significant figures when the CFL number was halved.

Table 5.1 shows the time-averaged Strouhal numbers taken from the AMROC simulations and those taken from Ref. [137] for the different smoothing values. One can see that the published results using the original strand meshing algorithms are highly sensitive to the level of smoothing. In contrast, the AMROC results using the new strand meshing algorithm show very little variation in Strouhal number, even when using a very low smoothing residual of 1.0×10^{-8} . These results clearly demonstrate the benefit of the new strand meshing technique over the previous method. The results give confidence that the gradients in the near-wall region can be accurately captured even when using significant smoothing of the mesh away from the wall. As previously discussed, this will enable accurate assessments of heat fluxes in the hypersonic flow simulations.

Smoothing Res.	1×10^{-5}	5×10^{-6}	1×10^{-6}	1×10^{-7}	1×10^{-8}
Strouhal No. (AMROC)	0.134	0.135	0.135	0.135	0.135
Strouhal No. (Ref. [137])	0.143	0.157	0.163	"chaotic"	N/A

Table 5.1: The Strouhal number as a function of the smoothing residual for the square cylinder simulation.

5.3 Hypersonic Sphere Shock Stand-off

The simulation of spheres in hypersonic flows is often used to validate thermochemical nonequilibrium models by comparing the simulated and experimental shock stand-off distances [220, 300]. This is because the shock stand-off distance depends on the level of thermodynamic and chemical nonequilibrium in the flow. An ideal gas simulation will over-predict the shock stand-off distance as it does not account for chemical reactions or the excitation of the vibrational and electronic modes. Thus, the temperature is over-predicted in an ideal gas simulation. As the pressure is largely unaffected by the thermo-chemistry, the higher post-shock temperature results in a lower density. In turn, this gives a larger shock stand-off distance. Conversely, a flow that reaches equilibrium immediately after the shock will under-predict the shock stand-off distance. This is because all of the energy is transferred from the chemical bonds and translational-rotational modes of the molecules instantaneously. This reduces the temperature behind the shock, increasing the density, resulting in a smaller shock stand-off distance. A flow that is in nonequilibrium will be between these two extremes. If the level of nonequilibrium is accurately modelled, the shock stand-off distance should be in good agreement with experimental data.

A classic set of experimental data that has been used by a number of researchers (for example Refs. [41, 90, 103, 225, 269]) is the data produced by Lobb [165]. In this experiment nylon spheres were fired in a ballistic range at speeds between 4,000 m/s and 6,500 m/s, at four different pressures, giving a unit Reynolds number range of approximately 1×10^6 m $^{-1}$ to 1×10^7 m $^{-1}$. Schlieren images were then used to determine the shock stand-off distance. The strand/CAMR solver was used to conduct viscous simulations of the experiments with a freestream pressure of 1333 Pa (10 mmHg). A five species mixture of air (O_2 , N_2 , O , N , NO) was used with the Park reaction rate constants, the AUSM flux scheme and the Mutation++ Chapman-Enskog LDLT viscosity model. The details of the chemical and thermodynamic models are given in Section 3.1 and Appendix A.1. In addition to validating the thermochemical model, these simulations also verify the axisymmetric governing equations. For comparison, the experiments were also simulated using an ideal gas model with a constant ratio-of-specific-heats of 1.4. Figure 5.5 shows the strand/CAMR mesh in the Mach 11.2 nonequilibrium simulation. One can see that the AMR is able to accurately capture the shock whilst using a coarser mesh away from the shock.

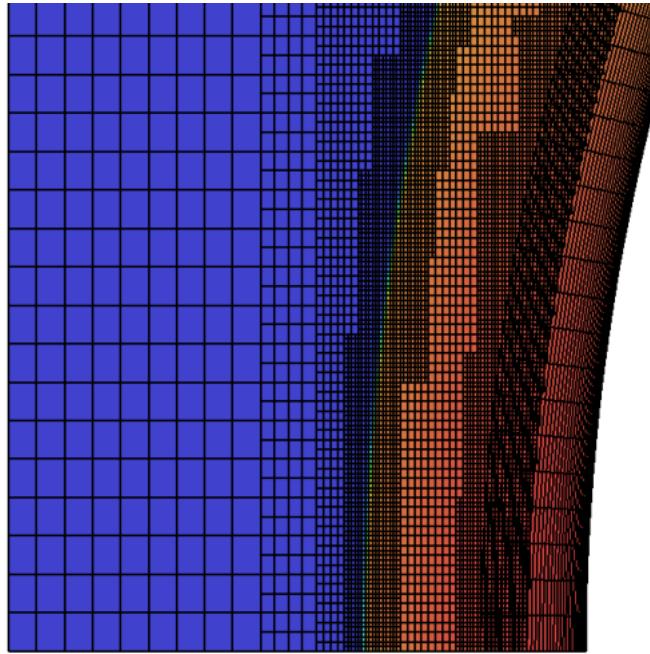


Figure 5.5: The pressure field and strand/CAMR mesh around the sphere in the Mach 11.2 case.

The shock stand-off distances for the ideal and nonequilibrium models are compared with the experimental results and numerical results from Gollan [103] in Fig. 5.6. The results from Ref. [103] were obtained using a single-temperature model, so some differences are to be expected. It can immediately be seen that the ideal gas model drastically over-predicts the shock stand-off distances. The over-prediction becomes larger as the Mach number increases. AMROC's two-temperature air model gives

significantly better agreement with the experimental results. The errors in the AMROC nonequilibrium results are shown in Table 5.2. The largest difference between the simulated and experimental data is less than 11% (less than 7% including the experimental error estimates).

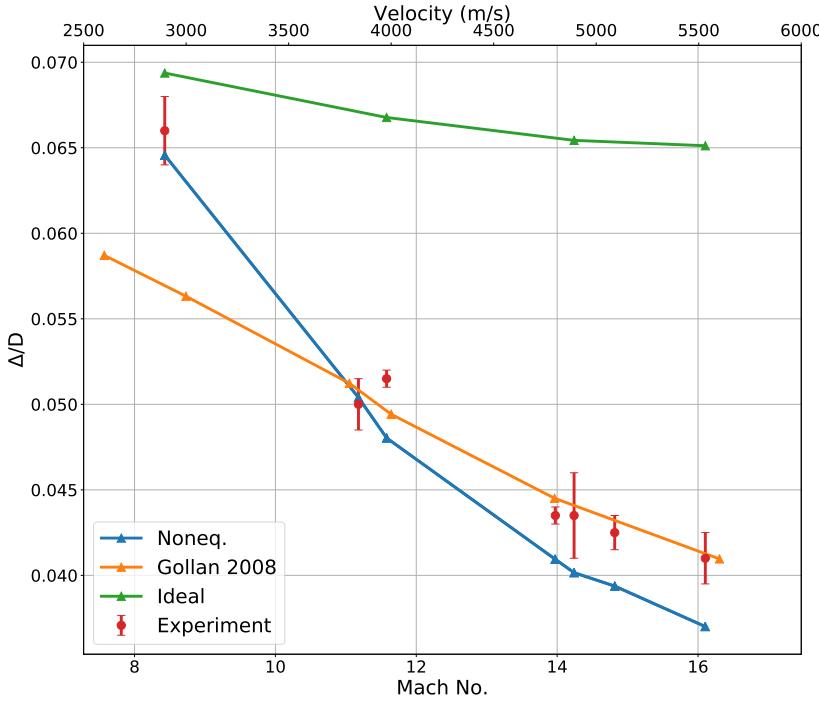


Figure 5.6: A comparison of the normalised shock stand-off data from the experiments and the simulations. The shock stand-off distance is normalised by the sphere diameter (Δ/D).

Mach No.	8.4	11.2	11.6	14.0	14.2	14.8	16.1
Difference (%)	2.22	0.78	7.22	6.24	8.32	7.95	10.79
Err. Diff. (%)	N/A	N/A	6.18	5.02	2.10	5.41	6.73

Table 5.2: The difference between the simulated and experimental shock stand-off distances, with and without the experimental error estimates.

Compared to the simulation results from the literature [103], the current solver appears to give significantly better results at lower Mach numbers, but slightly worse results at higher Mach numbers. Gollan attributes the poor results at the low Mach number to the use of a single-temperature model [103]. The stagnation line temperatures and mass fraction profiles for the low velocity test case are shown in Fig. 5.7. It can be seen that there is negligible chemical nonequilibrium but a high level of thermal nonequilibrium through the shock layer. These results support Gollan's conclusion and clearly demonstrate that accurately accounting for the thermal nonequilibrium is important for predicting the shock stand-off in this case.

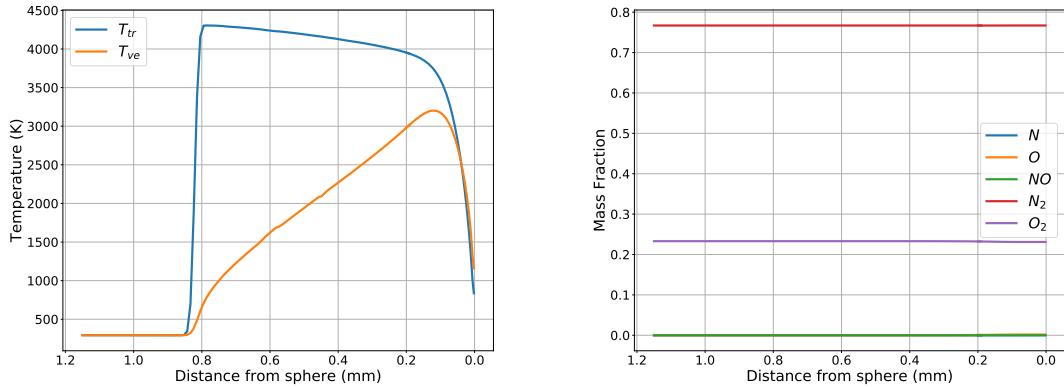


Figure 5.7: A comparison of the stagnation line temperature profiles (left) and the mass fractions (right) in the Mach 8.4 case.

To further illustrate the impact of using an ideal gas model, Fig. 5.8 shows the temperature fields in the ideal and nonequilibrium Mach 16 cases. Figure 5.9 shows a comparison of the temperature profiles along the stagnation line. It can be seen that the shock stand-off distance and translational-rotational temperature are both significantly higher in the ideal gas case. These results clearly demonstrate the need for nonequilibrium models when simulating hypersonic flows.

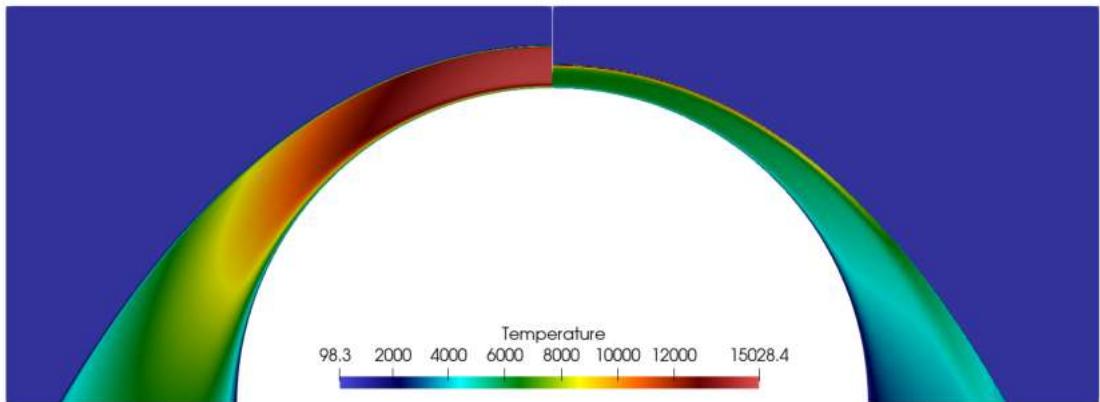


Figure 5.8: A comparison of the temperature field for the ideal gas model (left) and the nonequilibrium air model (right) for the Mach 16.1 case.

Although the nonequilibrium results are significantly more accurate than the ideal gas results, the shock stand-off distance is consistently under-predicted at higher Mach numbers. In Park's review of the two-temperature model [218], the under-prediction of the shock stand-off distance is identified as a common issue. Park suggests that this may be due to differences between the translational and rotational temperatures, which are not accounted for in the two-temperature model. To accurately account for this temperature difference Park suggested that a three-temperature model should be used (T_t , T_r , T_{ve}). This said, the two-temperature model is commonly used for

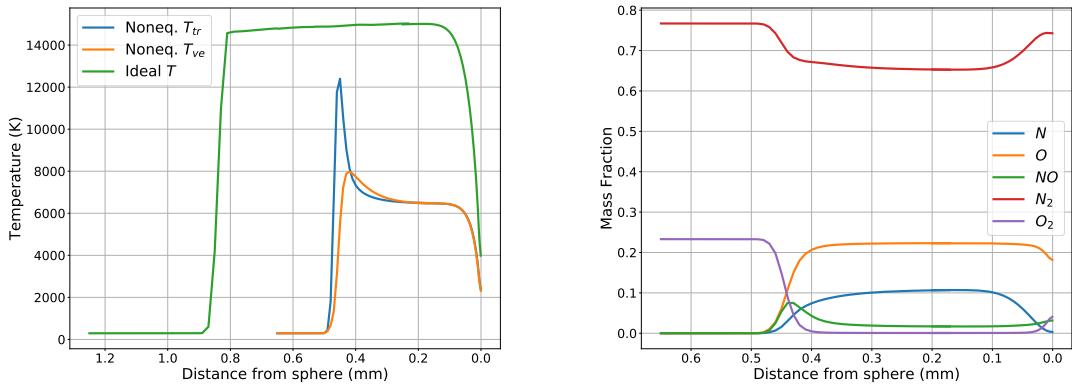


Figure 5.9: A comparison of the stagnation line temperature profiles (left) and the mass fractions in the nonequilibrium simulation (right) for the Mach 16.1 case.

hypersonic flow simulations, where a number of authors have shown that it gives results in good agreement with experiments for a range hypersonic flow regimes [40, 44, 57, 96, 118, 241, 273, 296].

This test case highlights why a nonequilibrium model is required as the ideal gas model is unable to accurately predict the flow features. The nonequilibrium air results are in good agreement with the experimental results, and the observed differences are previously recognised limitations of the two-temperature model. These results provide strong validation evidence for the two-temperature model and the axisymmetric governing equations.

5.4 Blunt Body Heat Flux Predictions

Hypersonic vehicles often use blunt noses and leading edges to increase the shock stand-off distance and reduce the surface heat fluxes [9]. Therefore, it is important that hypersonic CFD solvers are able to accurately predict the heat fluxes experienced by blunt bodies. To validate the new solver's heat flux predictions and examine the influence of the overset methods, two experiments have been simulated. The first simulation is of a cylinder in a high-enthalpy flow, which is analogous to a blunted leading edge. The second simulation is of a spherical capsule experiment, which closely resembles the Apollo re-entry vehicle and is representative of manned re-entry vehicles. The cylinder test case was also used to examine the parallel performance of the new strand/CAMR solver.

5.4.1 Cylinder

An experiment of a 90mm diameter cylinder in a high-enthalpy flow of air was simulated using the new solver. The experiment was conducted in the High Enthalpy Shock Tunnel Göttingen (HEG) facility at the Deutsches Zentrum für Luft- und Raumfahrt (DLR) and measurements were taken of the surface pressure and heat flux. The inflow conditions, given in Table 5.3, were taken from Ref. [61].

T_∞	ρ_∞	U_∞	Y_{N_2}	Y_N	Y_{O_2}	Y_O	Y_{NO}
694 K	3.26 g/m ³	4776 m/s	0.7356	0.0	0.1340	0.07955	0.0509

Table 5.3: Freestream conditions for the HEG cylinder simulation.

The strand mesh was created using a quarter cylinder surface with the strands set to be 2 mm in length. The CAMR domain had a geometry of 75 mm \times 100 mm, with a 150 \times 150 cell base mesh. An isothermal wall condition was used on the cylinder surface with a temperature of 300 K. The simulation used a five species air mixture with the Park reaction rate constants. The simulation was integrated in time until the density residual in the near-body region fell to below 1.0×10^{-8} .

A grid convergence study was carried out to determine the number of cells required for the near-body and off-body mesh and the near-wall grid spacing. Off-body grids with two, three and four levels of refinement were used with near-body grids of 50 \times 35, 100 \times 70 and 200 \times 140 cells, where the near-wall spacings were 4 μm , 2 μm and 1 μm , respectively. The results of the convergence study are shown in Fig. 5.10. One can see that the heat flux results are consistent across the three meshes, but the coarsest mesh does show a slightly lower heat flux at the stagnation point. In light of these results, the intermediate mesh was used for further study.

Figure 5.11 shows the nitrogen field and temperature profiles along the stagnation line. In Fig. 5.11a one can see the coarsening of the AMR away from the stagnation region, where the shock becomes more diffused resulting in a small bump. The streamlines from this area passed far from the surface, so coarsening the mesh in this region gives an efficient off-body domain. The image of the nitrogen field shows the chemical nonequilibrium, as the nitrogen dissociates through the shock layer, before recombining in the cooler boundary layer. The thermal nonequilibrium is shown by the spike in the translational-rotational temperature immediately behind the shock wave that then rapidly falls away as energy is exchanged with the vibrational and electronic modes.

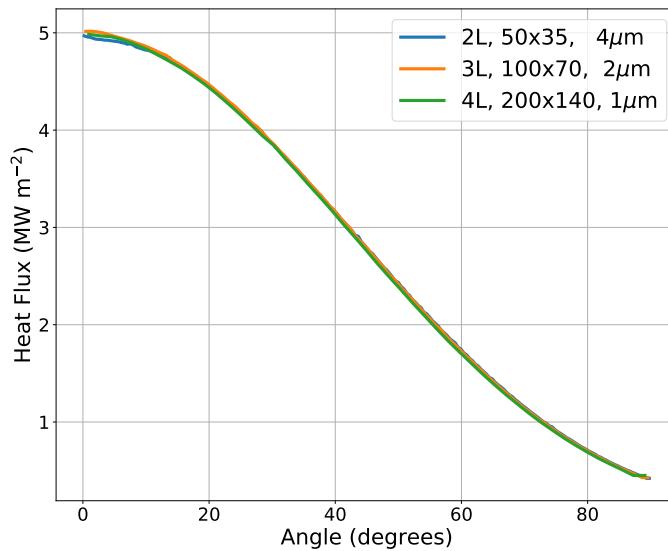
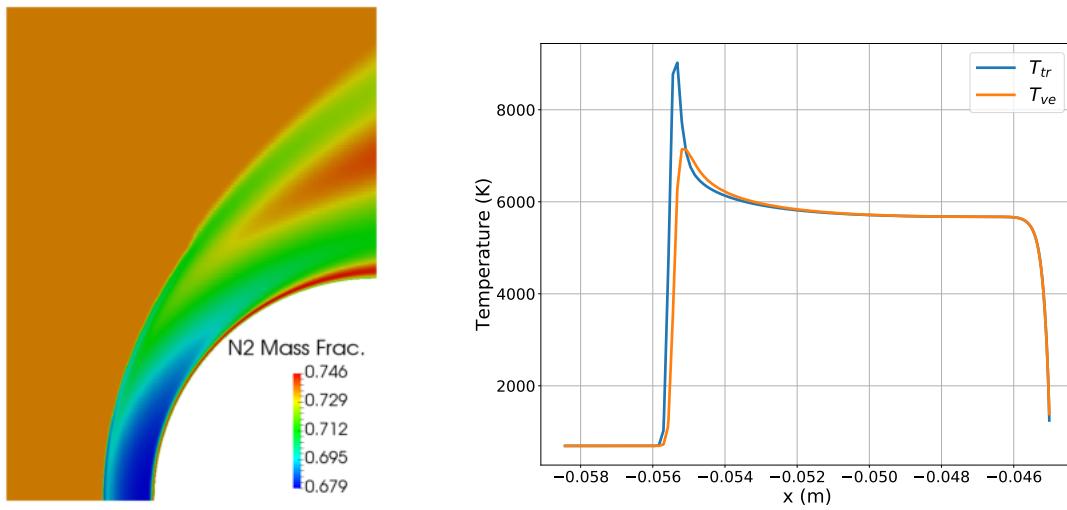


Figure 5.10: The results from the mesh convergence study.



(a) The molecular nitrogen field.

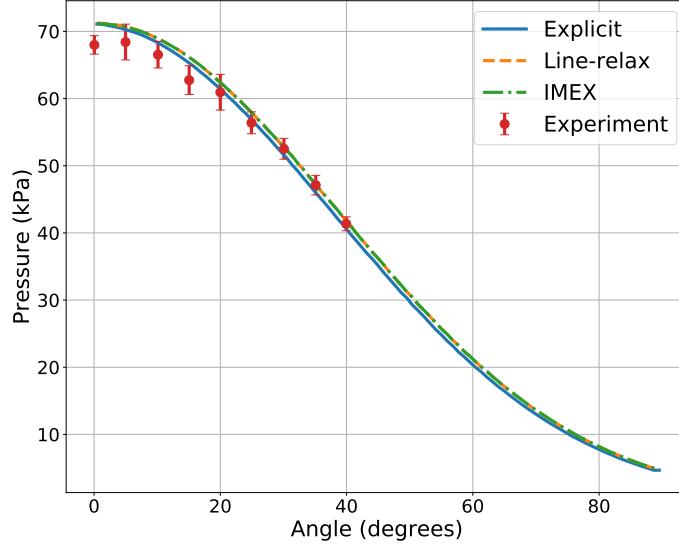
(b) The temperature along the stagnation line.

Figure 5.11: Visualisations of the thermochemical nonequilibrium in the cylinder test case.

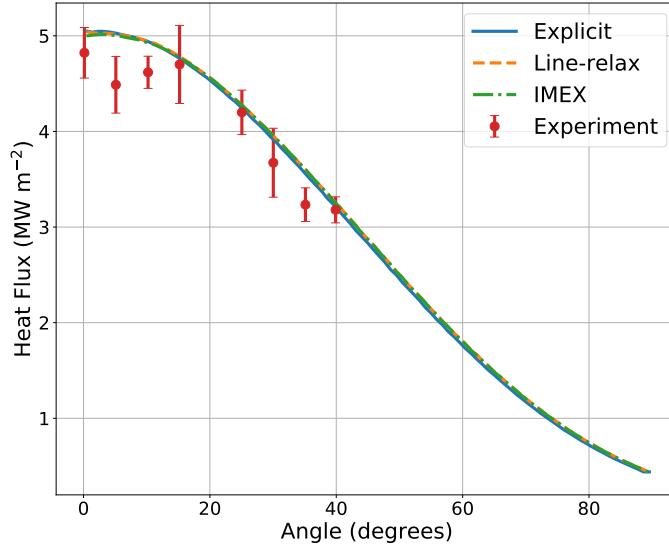
5.4.1.1 Experimental Comparison

The strand/CAMR pressure and heat flux results were compared with the experimental results to validate the overset heat flux prediction. In this work implicit time-integration methods have been implemented within AMROC, in addition to the overset and mapped mesh routines. These methods can be used in the strand domain to reduce the computational time. Results were obtained using both the line-relaxation method and the IMEX method on the strand domain. In this simulation, the global time-step was limited by the explicit off-body grid, rather than

the stability of the implicit method. As a result, the CFL in the wall-aligned direction was significantly lower than one, so the IMEX method was able to converge. The results using the strand/CAMR solver with the implicit methods are compared with the experimental results and the explicit strand/CAMR results in Fig. 5.12.



(a) Surface pressure.



(b) Surface heat flux.

Figure 5.12: A comparison of the experimental and simulated surface results in the HEG cylinder test case when using different time-integration methods.

The figures show that excellent agreement is obtained between the different time-integration methods. The AMROC results agree extremely well with the experimental data, with the majority of pressure and heat flux results falling within

the experimental uncertainty. This provides excellent validation evidence and demonstrates that the strand/CAMR solver is able to accurately compute stagnation region surface heat fluxes in high-enthalpy flows.

The IMEX method was able to reduce the computational time by approximately 26% compared to the line-implicit method. This was largely due to a reduction in the Jacobian creation time as only the wall-normal direction was integrated implicitly. This clearly demonstrates the benefit of the strand based IMEX method when the CFL is limited by the explicit off-body region.

5.4.1.2 Single Domain Comparisons

Comparisons with single-domain simulations were used to assess the impact of the overset mesh on the results and the computational efficiency of the strand/CAMR method. The experiment was simulated using a single-domain body-fitted mesh in AMROC. To ensure a fair comparison between the two solvers the respective mesh spacings at the shock and at the wall were set to be equal. This required the body-fitted mesh to have a mesh spacing of $2 \mu\text{m}$ at the wall, to match the near-body strand mesh, and $125 \mu\text{m}$ at the shock, to match the off-body SAMR mesh. To capture the entire shock region the body-fitted mesh extended to 55 mm from the surface. The body-fitted mesh was created using a hyperbolic tangent stretching function, and required 500 cells in the wall-normal direction in order to give the required resolution. This resulted in a total cell count of 49,500. In addition to the single-domain AMROC results, single-domain, two-temperature results from Ref. [61] were also used for comparison.

It is noted that the total cell count for the body-fitted mesh could be reduced by modifying the mesh to concentrate cells at the shock and reduce the number of cells outside of the shock layer. However, the focus of this research is on highly automated meshing techniques, and this would require significantly more user intervention. An adaptive body-fitted mesh could be used to maintain high-levels of automation for this test case. However, such a meshing strategy would be more limited than an overset strategy in other situations, for example in the mesh deformation and moving body simulations that are presented later in this chapter.

A comparison of the flow fields and meshes in the shock region for the two AMROC simulations is shown in Fig. 5.13. One can see that the shock locations in the two simulations are extremely similar. The surface pressure and heat flux results from both AMROC simulations are compared with the simulated results from Ref. [61] in Fig. 5.14. One can see that excellent agreement is obtained between the two AMROC simulations. These results indicate that the use of an overset mesh has a negligible

impact on the surface heating results for blunt body flows. This is an important result for building confidence in the overset heat flux predictions.

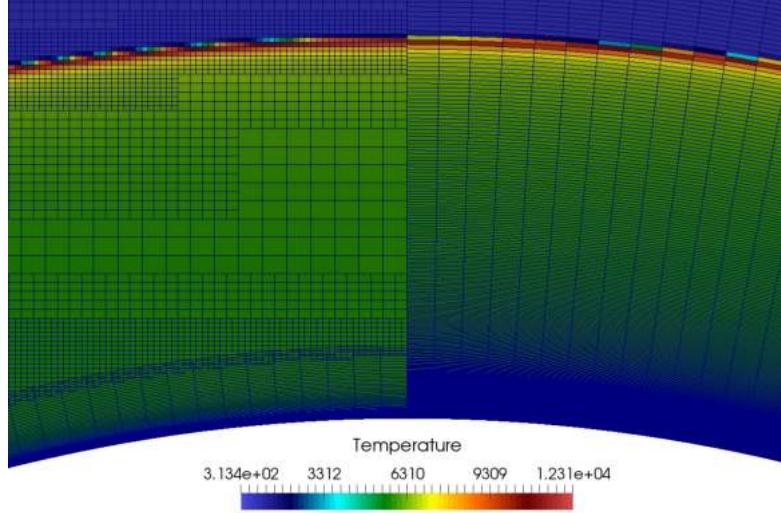


Figure 5.13: The body-fitted and strand/CAMR mesh in the stagnation region.

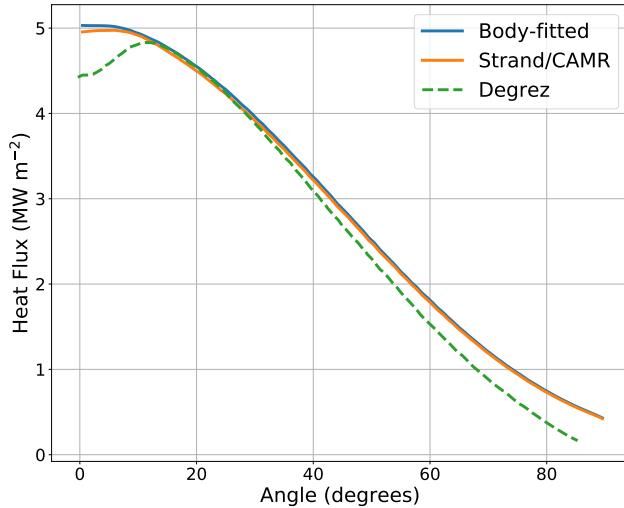


Figure 5.14: A comparison of the surface heat flux in the HEG cylinder simulation when using single and overset domains.

The AMROC heat flux results compare reasonably well with the results from Ref. [61]. However, there are some differences in the stagnation region, and smaller differences on the upper section of the cylinder. The single-domain AMROC results clearly show the discrepancy is not due to the use of overset methods, but more likely due to differences in numerical methods and models. For example, the simulations in Ref. [61] used a three-dimensional domain and the centre-line results were used for comparison. In addition, different species production rates and thermodynamic relaxation models were used in the two solvers.

The computational time for the strand/CAMR simulation was 5.5 times faster than the computational time for the body-fitted mesh. There are two main reasons that contribute to the larger simulation time for the body-fitted mesh. Firstly, the AMR enables a significantly more efficient spatial discretisation. The strand/CAMR mesh had approximately 38,000 cells in the final adaptive mesh, which is a reduction in the cell count of more than 20% compared to the body-fitted mesh. The AMR enables the off-body SAMR mesh to be coarser in the smooth regions away from the shock, as shown in Fig. 5.13. The second reason for the reduced computational time is that implicit time-stepping is used for the entirety of the body-fitted mesh, whereas it is restricted to the near-body region of the strand/SAMR mesh. Whilst the maximum stable time-step was larger for the body-fitted grid as no explicit time-stepping was used, this was offset by the additional cost of constructing and solving a significantly larger linear system at each time step: the overset simulation integrated 6,930 cells implicitly, whereas the body-fitted simulation integrated 49,500 cells implicitly.

5.4.1.3 Parallel Performance

The parallel scalability of the overset solver was assessed using this test case. The test was conducted on a refined mesh with 13,860 near-body cells and approximately 91,000 off-body cells at the end of the simulation. Both domains are parallelised using MPI, where each domain uses its own MPI group. The near-body domain used a fixed domain decomposition where each processor's domain stretched from the wall to the overset boundary, meaning that each strand was on only one processor. The off-body domain used the dynamic load-balancing algorithms previously implemented within AMROC [64]. The computational speed-up is plotted in Fig. 5.15 for the total simulation time (the minimum number of cores is four - two for each domain). The overset parallel decomposition uses an ad-hoc method, where the number of processors on each domain must be specified at the start of the simulation. For larger total core numbers, this ad-hoc split can be better balanced. For example, with 32 cores, the fastest simulation time was obtained with 11 off-body processors and 23 near-body processors, whereas for the four core simulation both domains used two processors. This is the reason why above ideal speed up is obtained between four and eight cores, where the eight core simulation used three off-body and five near-body processors.

The ad-hoc load balancing can have a large impact on the computational cost of the overset simulations. The computational time of the overset routines is dominated by the collective communication required to maintain synchronisation between the domains. When the load balancing between the near-body and background domains is uneven, one domain can spend a significant length of time waiting for the other domain to communicate. Two strategies can be used for the overset synchronisation,

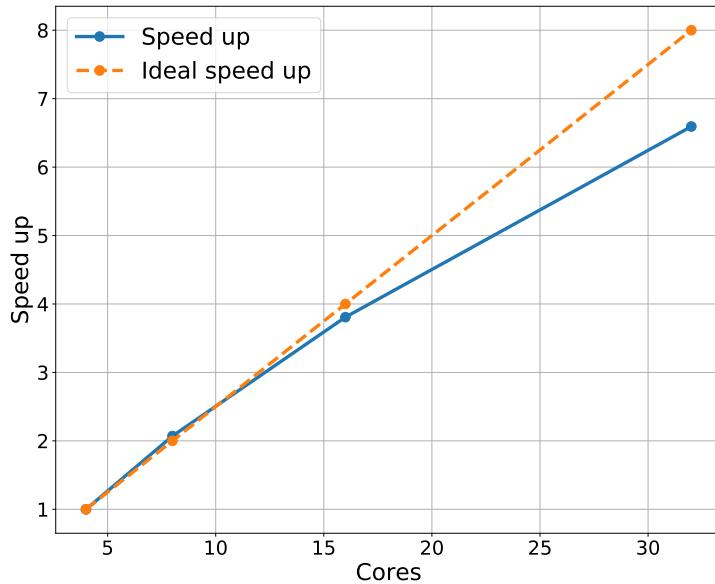


Figure 5.15: The strong scaling results for the overset solver.

where the strategy used influences the amount of time spent synchronising the domains.

The first strategy is the multi-synchronisation strategy. When using this strategy, collective communication is required prior to each coarse-level AMR time step to synchronise the time step across the domains. This is referred to as the “Time Sync”. A second synchronisation is required prior to each fine-level AMR time step, where the overset boundary conditions are exchanged. This is referred to as the “Mesh Sync”. The multi-synchronisation method is graphically depicted for a well-balanced simulation in Fig. 5.16. An important point to note is that the two domains both have multiple points where they may have to wait. In the illustration there is a waiting period for the near-body mesh between the time-step synchronisation at the start of the coarse-level time step and the mesh synchronisation prior to the first fine level update. This is due to the time required for the SAMR domain to complete the coarser-level integration, hole cutting, AMR re-gridding and parallel re-distribution operations. In contrast the SAMR domain must wait for the near-body integration to complete after the first, second and final iterations on the finest SAMR level.

The advantage of the multi-synchronisation method is that the overset exchange is time-accurate and the AMR domain can adapt at every fine-level iteration. The disadvantage of the multi-synchronisation strategy is that the near-body domain must wait for the coarser-level integration, hole-cutting, re-gridding and redistribution to complete on the SAMR domain between each Time Sync and each Mesh Sync (see Fig. 5.16). The re-gridding and redistribution operations are communication intensive, so do not scale well with the number of off-body processors. Therefore, the waiting

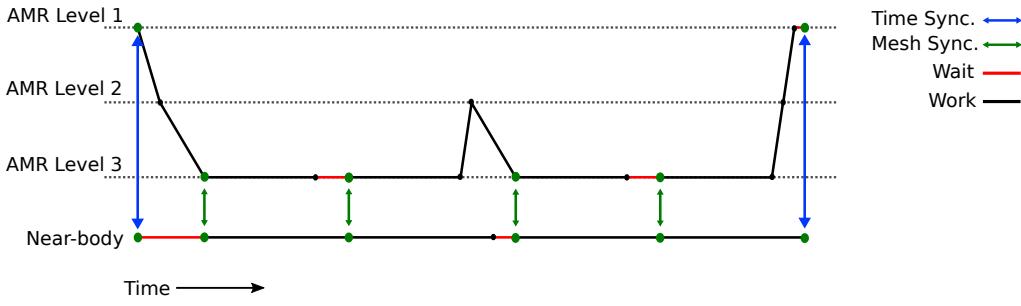


Figure 5.16: An illustration of the synchronisation in a balanced overset SAMR simulation.

time on the near-body mesh reaches a minimum. This is seen in the timing results for the above simulation, which are shown in Fig. 5.17 and Table 5.4. It can be seen that the background waiting times can be removed almost entirely, but the near-body waiting times do not fall below 2,000 seconds. Table 5.4 shows that this minimum is due to the Mesh Sync waiting time, which is the time the near-body mesh spends waiting for the off-body SAMR routines to complete. It is notable that the synchronisation dominates the overset time. The remaining overset routines (communication, donor search interpolation etc.) only make up a tiny fraction of the total computational cost, less than 0.4% in all cases shown here.

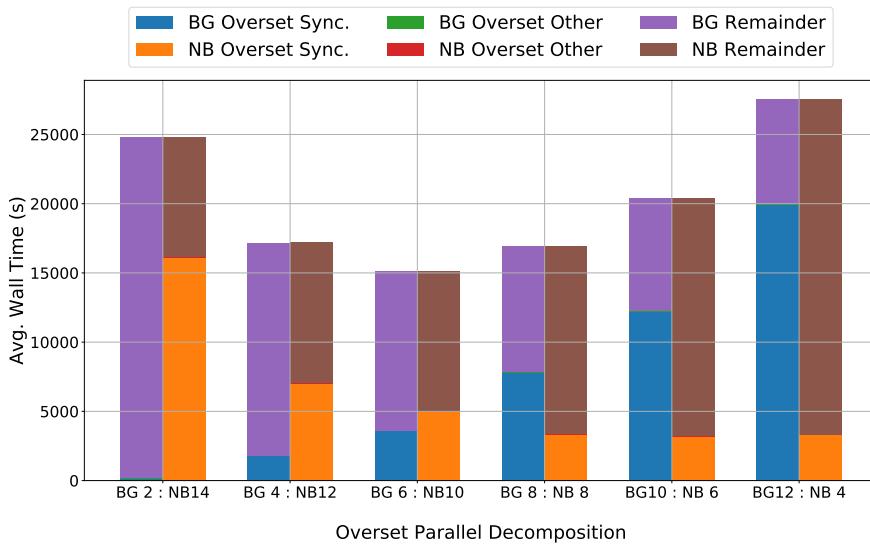


Figure 5.17: The proportion of the overset time relative to the total simulation time when using different overset parallel decompositions.

The results shown in this section are typical of those seen in other simulations conducted in this work where the multi-synchronisation strategy was used. When the overset parallel decomposition is well balanced, the overset time, including synchronisation, is approximately 20-30% of the total simulation time.

BG:NB Proc.	2:14	4:12	6:10	8:8	10:6	12:4
BG Time Sync. (s)	29.7	487.9	913.6	1,537.8	2,625.3	4,149.4
BG Mesh Sync. (s)	48.4	829.2	1,777.4	3,608.2	6,760.1	11,521.0
NB Time Sync. (s)	1,018.0	0.34	0.33	0.28	0.33	0.26
NB Mesh Sync. (s)	12,086.8	5,495.8	4,010.2	2,186.2	2,555.9	2,579.9

Table 5.4: The synchronisation times for the HEG test case using different overset parallel decompositions and the multi-synchronisation method.

The second strategy is the single-synchronisation strategy, where both the time step and the overset boundary conditions are synchronised at the start of each coarse-level AMR iteration. This means that the near-body domain does not have to wait between the Time Sync and the Mesh Sync. A depiction of the single-synchronisation method using a well-balanced distribution is shown in Fig. 5.18. It can be seen that nearly all of the waiting periods have been eliminated.

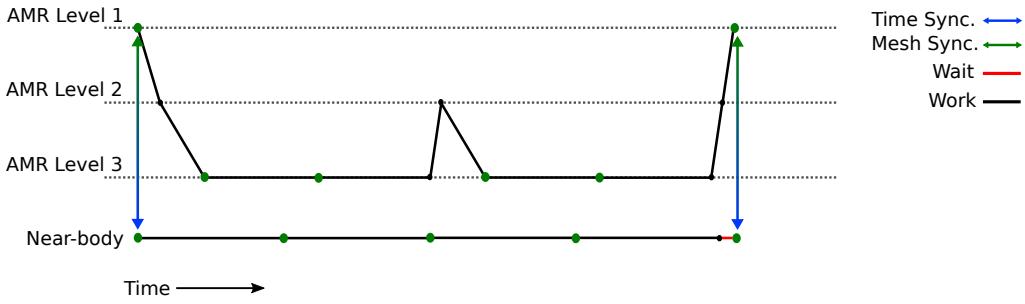


Figure 5.18: An illustration of the single-synchronisation method.

The timing results using the single-synchronisation method are shown in Fig. 5.19 and Table 5.5 for the three most well-balanced cases. For comparison, the most well balanced multi-synchronisation case (BG6:NB10) is also shown. The results clearly show that the single-synchronisation simulation times are all lower than the multi-synchronisation case, even when the domains are less well balanced. In addition, it can be seen that the waiting times for the near-body domain can be almost completely removed. For the most balanced single-synchronisation case, BG6:NB10, the total overset time has been reduced to 13% of the total time on the near-body domain and 5% of the total time on the background domain. This is typical of other single-synchronisation simulations where the overset times can be reduced to approximately 5-10% on both domains.

Although the single-synchronisation method reduces the overset waiting times, there are a number of disadvantages. When using this strategy the AMR re-gridding and parallel redistribution on the background mesh can only take place at the start of each coarse-level iteration. When using multiple levels or large refinement factors this

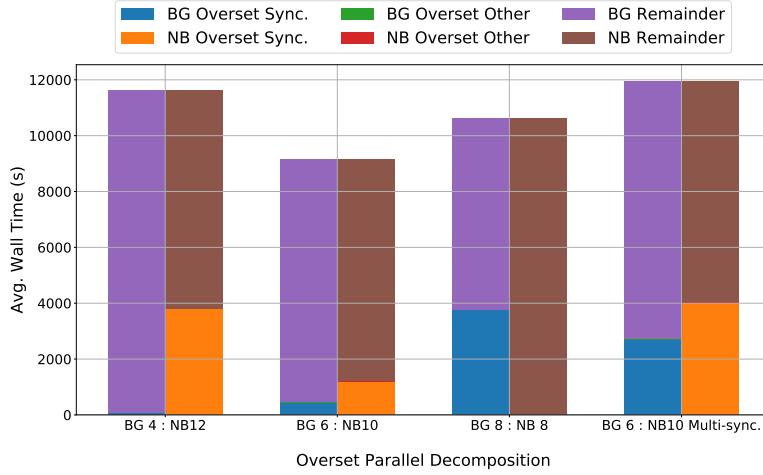


Figure 5.19: The proportion of the overset time relative to the total simulation time when using different overset parallel decompositions and the single-synchronisation method.

BG:NB Proc.	4:12	6:10	8:8	6:10 (Multi.)
BG Time Sync. (s)	72.9	443.7	3756.7	913.6
BG Mesh Sync. (s)	0.05	0.04	0.04	1,777.4
NB Time Sync. (s)	3794.5	1200.2	1.783	0.33
NB Mesh Sync. (s)	0.04	0.04	0.06	4,010.2

Table 5.5: A comparison of the synchronisation times for the HEG test case using different overset parallel decompositions.

could cause the background grid to remain stationary for a number of fine-level updates. In turn, this could lead to issues with the AMR if the location of the shock changes whilst the grid is static. Another disadvantage is that this method results in the overset boundary conditions on both domains staying constant throughout each coarse level SAMR time step, which reduces the temporal coherency between the domains. That said, for some steady-state simulations these effects may be negligible.

For this steady-state test case, the single-synchronisation method was used and shown to give extremely similar surface heat flux and pressure results to those from the multi-synchronisation method. However, considering the disadvantages, care must be taken when using the single-synchronisation method. The single-synchronisation method was not used in test cases where time-accuracy was required, or if the near-body grid was deforming or moving. The single-synchronisation method can be specified as a run-time option, but the default is the multiple-synchronisation method that allows for fine-level re-gridding and time-accurate simulations. Whilst this does incur a penalty in simulation time it ensures the overset routines and AMR are accurate and robust.

One issue that is not considered in the above analysis is that the number of cells in the

background domain is changing as the background mesh adapts. This will lead to the waiting times on each domain changing through the simulation and can make it difficult to obtain well-balanced overset parallel decompositions using static processor allocations. The ability to dynamically redistribute the processors between the background and near-body domains could further reduce the overset synchronisation times. It would also remove the need to empirically find the best processor distribution for each simulation. However, the minimum near-body waiting time imposed by the gap between the time and mesh synchronisation in time-accurate simulations will remain.

Another parallelisation strategy that is not considered here, but could be implemented in future work, is to distribute every overset domain across all processors. This would require each overset domain to be integrated consecutively but would enable the dynamic load balancing to be used on all domains. This could significantly reduce the amount of time that processors are idle. The disadvantage of this approach is that the communication cost on domains with a low cell count could become significant. For example, the background domain could have 100 times more cells than the near-body domain, but both domains would be split across the same number of processors. This would mean that the number of near-body cells on each processor is 100 times fewer than the number of background cells on each processor. In turn, this could lead to high communication overheads on the near-body domain.

Finally, the number of cells on each processor has an influence on the proportion of time spent in the overset routines. Using more cells per processor generally reduces the proportion of time spent performing overset synchronisation as more time is spent in the numerical integration. This said, speed-ups are obtained when using more processors (see Fig. 5.15), even though the proportion of time spent in overset synchronisation is reduced.

5.4.2 Spherical Capsule

An experimental investigation of high-enthalpy flow over a spherical capsule was simulated with the new solver. The aim was to verify and validate the axisymmetric heat flux predictions for the two-dimensional strand/CAMR solver. The experimental set-up, the measured surface heat flux and pressure results, and simulation results from DPLR are all taken from Ref. [173]. The experimental freestream conditions are shown in Table 5.6 and the dimensions of the spherical capsule are shown in Fig. 5.20a.

T_∞	ρ_∞	U_∞	Y_{N_2}	Y_N	Y_{O_2}	Y_O	Y_{NO}
522 K	1.59 g/m ³	4168 m/s	0.735	0.0	0.171	0.0289	0.0655

Table 5.6: Freestream conditions for the spherical capsule simulation.

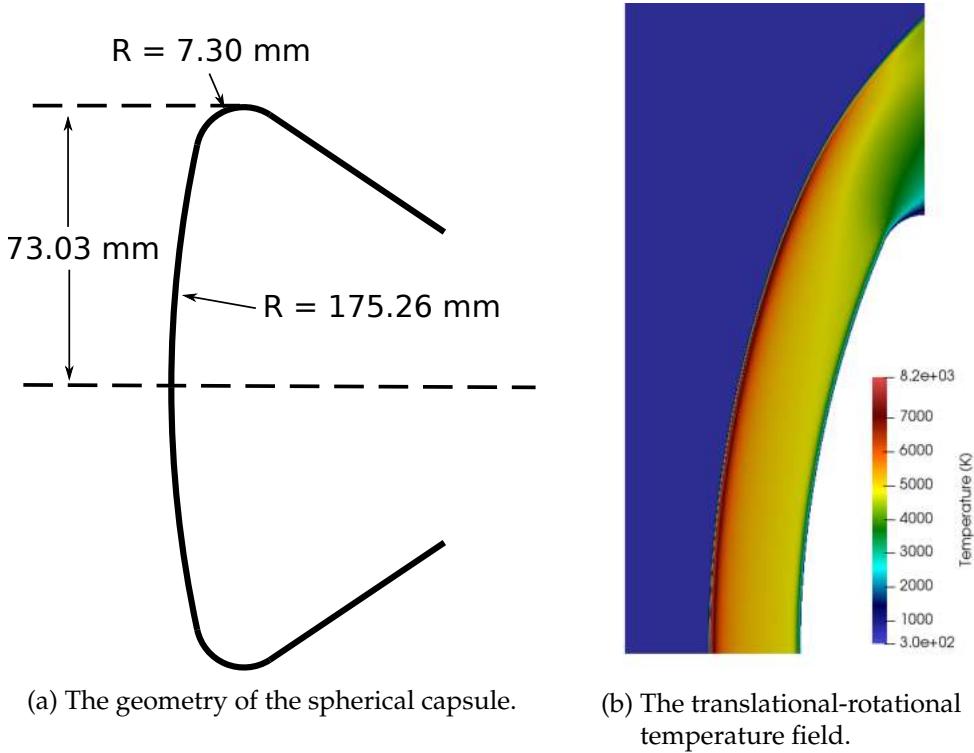
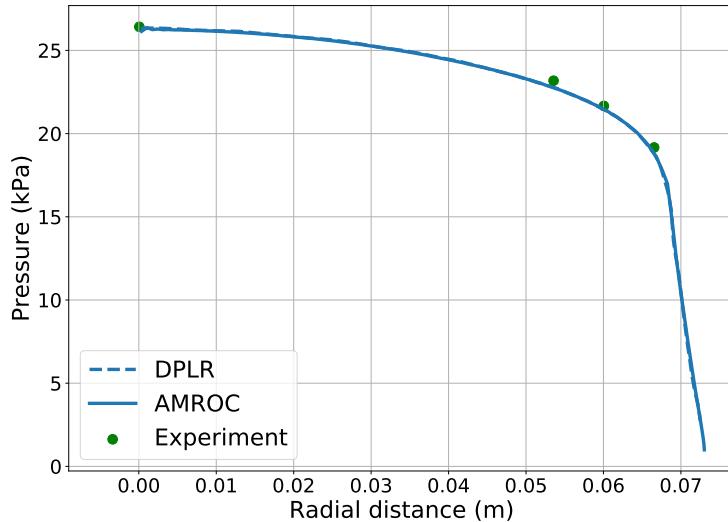


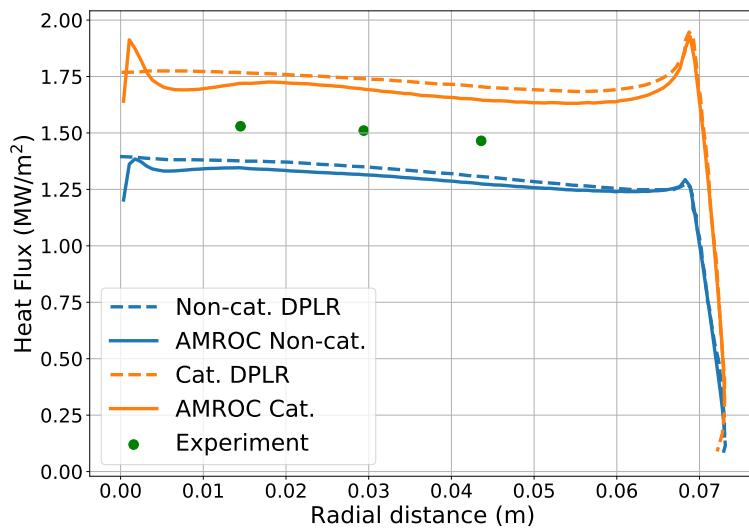
Figure 5.20: The spherical capsule geometry and flow field.

The strand mesh used 115 cells along the surface, where each strand contained 61 nodes and the length of each strand was set to be 3 mm. No smoothing was applied for the inner strands and the outer strands used a smoothing residual of 1×10^{-7} . The background CAMR domain had dimensions $x \in [-0.05, 0]$ m and $y \in [0, 0.125]$ m, with a base mesh of 110×125 cells and three levels of refinement, each with a refinement factor of two. The AUSM flux scheme was used with the Blottner/Euler/Wilke transport model. The simulation was run with both a fully catalytic and non-catalytic isothermal boundary with a temperature of 300 K. The line-implicit method was used for the flux calculations and the single-synchronisation method was used for the overset exchange. The temperature field for the simulation is shown in Fig. 5.20b.

The AMROC results are plotted alongside the experimental results and the DPLR results in Fig. 5.21. The pressure results from AMROC are in excellent agreement with the experimental and numerical results from the literature. The AMROC heat flux results are in excellent agreement with both the non-catalytic and catalytic results from DPLR. However, both sets of simulation results under- and over-predict the experimental heat flux for the non-catalytic and catalytic results, respectively. These results suggest that a boundary condition which models some recombination rather than full recombination would accurately predict the experimental heat fluxes. Finally, an oscillation can be seen in the AMROC heat flux results close to the axis. After extensive investigations it is believed that this is a numerical effect caused by the



(a) Surface pressure.



(b) Surface heat flux.

Figure 5.21: A comparison of the experimental and simulated surface measurements in the spherical capsule test case.

axisymmetric fluxes and source terms. Firstly, the axisymmetric source term and viscous fluxes are both functions of $1/y$. Therefore, this term can become very large near to the symmetry boundary, which may introduce or amplify numerical errors close to the stagnation point. Secondly, oscillations at the stagnation point are not seen for the cylinder case (e.g. Fig. 5.12), which uses the two-dimensional governing equations. Finally, extensive code reviews were unable to find any errors in the implementation of the axisymmetric terms.

In summary, the axisymmetric heat flux results are in good agreement with the published results. This test case provides verification and validation evidence for the

axisymmetric flow predictions with the new solver. In addition, this test case gives a strong indication that the use of overset methods does not adversely impact the surface heating results for blunt bodies when using an axisymmetric formulation.

5.5 Shock-Shock Interaction Test Case

The above cases give confidence that the overset methods have little impact on the hypersonic heat flux predictions for blunt body cases. However, many hypersonic vehicles of interest have more complex geometries that can result in shock structures impinging on the surface of the body. This can create areas of intense heating, so it is important that they can be accurately simulated using the new solver. These cases are more challenging for overset methods as a discontinuity is crossing the overset boundary. In this section a test cases is used to examine the influence of the overset method on surface predictions when discontinuities cross the overset boundary.

The seminal investigations of Edney [77] into the interaction of a bow shock with an oblique shock showed that the interactions can be classified into distinct types. The shock-interaction type, and subsequent flow structures, depends on the strength and angle of the intersecting shocks. The Edney Type IV interaction is of particular importance when designing a TPS as it can lead to regions of extremely high heating. Figure 5.22 shows a Type IV interaction. It can be seen that there is a triple point at the intersection of the upper bow shock and oblique shock. This results in an oblique transmitted shock and a shear layer with a subsonic region above and supersonic region below. There is a second triple point where the transmitted shock from the first triple point meets the bow shock from the lower surface. This results in a second transmitted shock and shear layer being created. The resulting shock structure creates a supersonic jet bounded by shear layers above and below. This jet reaches almost to the surface of the blunt body, where it terminates in a shock wave. The high temperatures behind the termination shock greatly enhanced the heating rates at this point on the surface.

In this test case a Type IV shock-shock interaction is simulated using the strand/CAMR solver and two different meshes. The first mesh uses a very thin strand mesh close to the surface. The majority of the shock structure remains outside of the strand mesh, but the supersonic jet crosses the overset boundary and impinges on the surface. The second mesh uses a large near-body strand mesh so that the entire shock interaction is within a single domain. Previous investigations have shown the results of Type IV interaction simulations can be highly sensitive to the spatial discretisation [59, 289]. As a result, this test case is likely to highlight any adverse effects of using an the overset mesh to study complex shock structures.

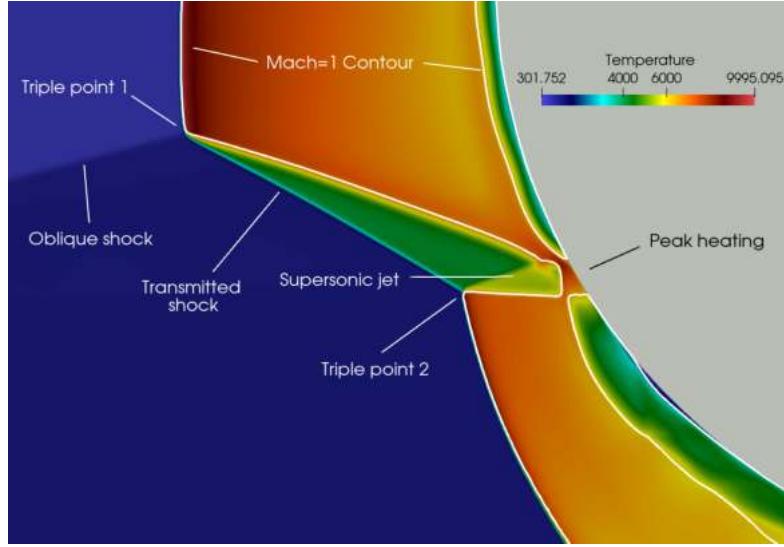


Figure 5.22: A schematic of the Edney Type IV shock structure.

The simulation is based on the experiments conducted by Wieting and Holden [286]. In these experiments a flat plate was used to generate an oblique shock that then interacted with the bow shock of a cylinder, creating a Type IV shock interaction. This case has been used by other authors to validate shock-shock interaction heat flux predictions [31, 147, 297, 303]. The inflow conditions above and below the oblique shock were taken from [303] and are given in Table 5.7. The value y/D is the height above the centreline that the shock would meet the centre of the cylinder, normalised by the cylinder diameter, $D = 76.12$ mm. For all simulations a five species mixture of air was used.

T_1	ρ_1	M_1
111.56 K	3.08 g/m ³	8.03
T_2	ρ_2	M_2
238.04 K	10.24 g/m ³	5.25
Plate Angle	Shock Angle	y/D
12.5 deg.	18.11 deg.	0.2073

Table 5.7: The freestream conditions for the Type IV shock-interaction case [303]. The subscript 1 indicates the free-stream conditions and the subscript 2 the post-oblique-shock conditions.

Initial simulations showed that the shock interaction leads to a mildly unsteady flow, where the location on the shock impingement moved up and down by one or two degrees. This is in agreement with other experimental and numerical investigations of Type IV interactions [235, 289, 297, 303]. Therefore, for this simulation, the heat fluxes results were obtained by averaging the heat flux over approximately 10 characteristic flow times, once a quasi-steady flow had been established. The experimental data given in Ref. [286] is normalised by the values obtained on the cylinder in the same

free-stream conditions, but without a shock interaction. Thus, strand/CAMR simulations with no interactions were used to obtain these values.

The simulations were carried out using the HLLC flux scheme, the Blottner/Euler/Wilke transport equations and the IMEX method was used in the strand domain. A mesh convergence study was conducted using the strand/CAMR solver and the results are shown in Fig. 5.23. Following the convergence study, a large strand mesh was then created that attempted to match the cell sizes of the strand/CAMR solver. A best attempt was made, but this was not possible to achieve in practice. As a consequence, the CAMR mesh is finer than the single-domain mesh at the bow-shock-oblique-shock interaction, but coarser closer to the body. In addition, the growth rate of the small and large strand meshes differ. The final meshes used in this test case are shown in Fig. 5.24 and a visualisation of the large strand mesh flow field is shown in Fig. 5.25.

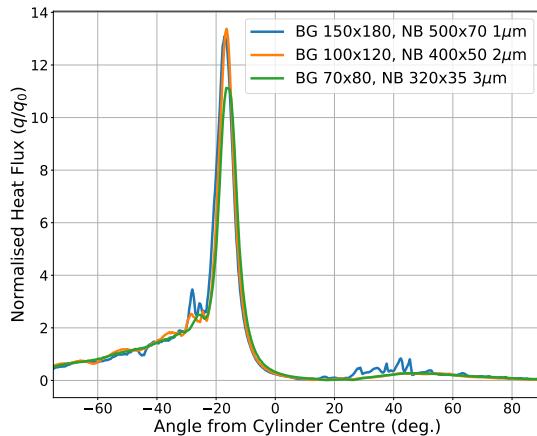


Figure 5.23: The results of the Type IV shock interaction spatial convergence study.

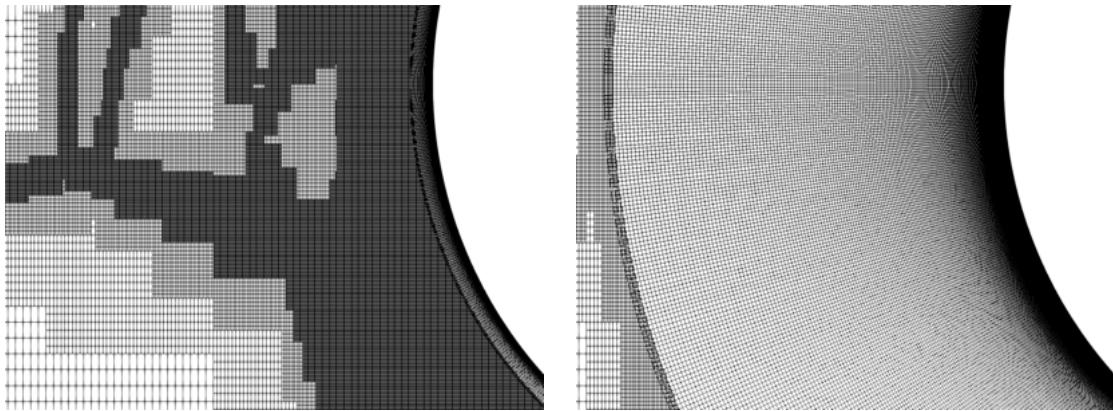


Figure 5.24: The small (left) and large (right) strand meshes used in the Type IV interaction test case.

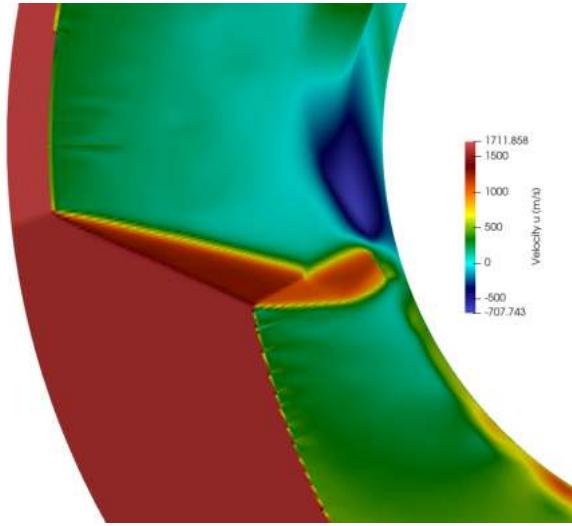


Figure 5.25: The x -velocity field for the large strand mesh. The entire shock interaction is contained within the strand domain.

A comparison of the experimental, strand/CAMR and single-domain results is shown in Fig 5.26. It can be seen that the three sets of results are in excellent agreement. The location of the shock impingement in the two meshes is within 0.95 deg. and the peak heat-fluxes are within 3.1%. Considering the differences in the meshing strategy and cell sizing this is excellent agreement. Both sets of simulation results are in good agreement with the experimental peak heat flux magnitude and location, but do appear to give slightly narrower peaks than the experimental results. For both sets of numerical results the pressure peak is in a very similar location and agrees well with the peak location of the experimental results. The width of the peak pressure region is also in very good agreement with the experimental results. That said, the peak is notably higher in the simulations than in the experimental data. This is the same for both meshes, so is not caused by the overset mesh. In addition, this higher peak is in agreement with previous numerical results from the literature [297, 303].

This test case is a particularly stressing test case for the overset solver, as discontinuities and shear layers cross the overset boundary. The results show that the overset predictions are in excellent agreement with the single-domain and experimental results. To the best of the author's knowledge this is the first time that impact of using an overset mesh on heating predictions for a Type IV interaction has been studied in detail. The results give confidence that the strand/CAMR method is able to accurately predict heat fluxes in flows that include complex shock structures.

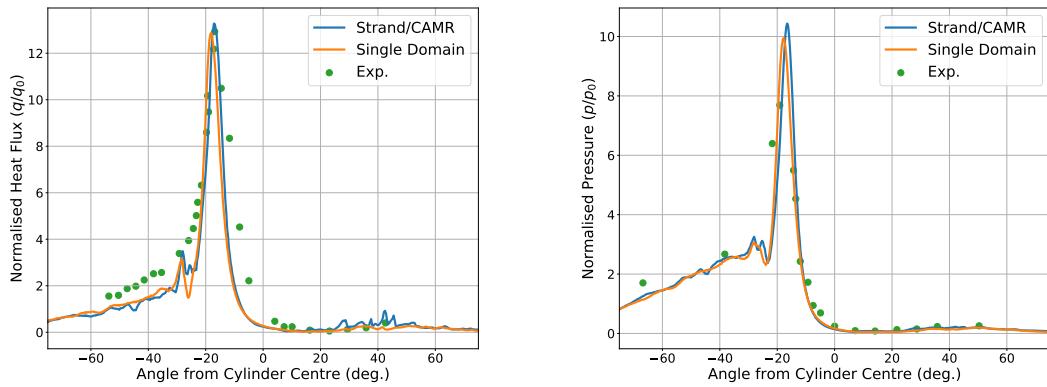


Figure 5.26: A comparison of the surface heat flux (left) and pressure (right) results from the single domain and strand/CAMR simulations and the experiment [286].

5.6 Automated Surface Mesh Deformation

Two simulations of recessing surfaces were used to demonstrate the highly-automated surface mesh deformation enabled by the new solver. The first simulation uses a simple geometry, where the results can be compared against experimental results from the literature. The second simulation includes a much more complex surface deformation that results in dynamic shock structures.

5.6.1 Recessing Plate

An experimental investigation of an ablative plate in an arc-heater flow was simulated using the new solver. In the experiment a blunted wedge was used, where the nose of the wedge was manufactured in copper and did not change shape during the experiment. Behind the nose, a Phenolic Impregnated Carbon Ablator (PICA) plate was fitted to the upper surface of the wedge and the recession of the PICA led to a concave region downstream of the nose. This concave region could cause difficulty for mesh morphing methods that assume a convex surface. In this test case the ablation was not simulated, but the initial and final surfaces measured in the experiment were used as inputs to the mesh motion algorithm.

In order to simulate an arc-heater flow field using a two-dimensional strand/CAMR solver, the arc-heater nozzle and the test-box were simulated separately. The nozzle simulation used the axisymmetric formulation of the governing equations and the outflow from the nozzle was used as an inflow to a separate two-dimensional test-box simulation that included the wedge. In reality, the nozzle flow over the wedge is likely to be three-dimensional but this strategy has been shown to give a good approximation of the centre-line flow over the wedge [101, 102]. As the primary

purpose of this test case is to assess the mesh motion and the automated mesh generation using a convex surface, the two-dimensional simulation was deemed adequate.

The nozzle was simulated from the throat and the inflow conditions are based on those reported for the laminar flow experiment, Run 4-A2, in Ref. [102]. In this experiment the stagnation pressure was measured as 492 kPa and CFD calibration was used to determine the centreline enthalpy at the nozzle throat as 17.9 MJ kg^{-1} . The fluid was a mixture of air and argon (with an argon concentration of 6.5%) and was assumed to be in thermochemical equilibrium at the throat. The outflow data from the nozzle simulation was used as an inflow in the test-box simulation. The inflow location was on the section of the test-box where the nozzle exit positioned. Above the nozzle inflow, a wall boundary condition was imposed on the remainder of the left-hand boundary. A composite image of y -velocity field in the nozzle and test-box is shown in Fig. 5.27. One can see that the flow field between the two domains is continuous and flow features created by nozzle are accounted for within the test-box domain. For example, the expansion at the join between the nozzle and the nozzle extension travels smoothly into the test-box domain.

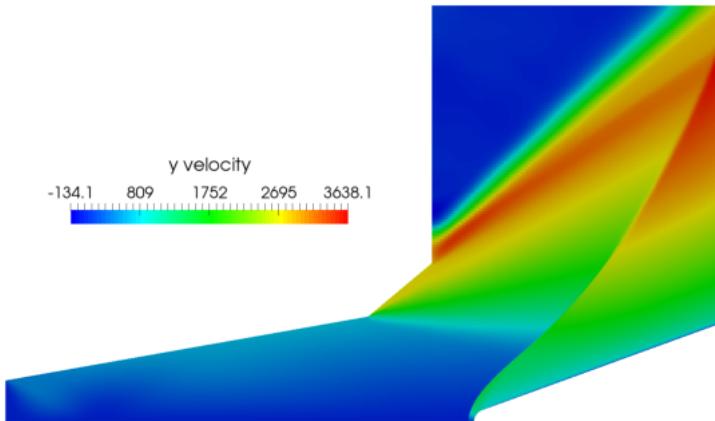


Figure 5.27: The y -velocity field in the nozzle and test section of the arc-heater surface recession test case.

The strand mesh on the wedge used two surfaces as inputs: the initial unablated surface and the final ablated surface. The recession depth for the ablated surface was taken from the measured surface recession results for the laminar flow case, which are given in Ref. [101]. The surfaces were constructed using 198 nodes and the strands were 3 mm in length, with an initial spacing of $2.5 \mu\text{m}$ and 71 nodes along their length. The CAMR domain used a 150×150 cell base mesh and five levels of refinement were used in order to obtain comparable cell sizes at the overset boundary.

The automated surface mesh motion is depicted in Fig. 5.28. This figure shows the unablated, intermediate and ablated geometries in the nose region of the wedge where

the recession was greatest. The minimum number of time steps for the strand motion was specified as 500 and the maximum motion CFL was 0.5. Other than these two parameters and the two surface geometries, no other user input was required for the mesh motion.

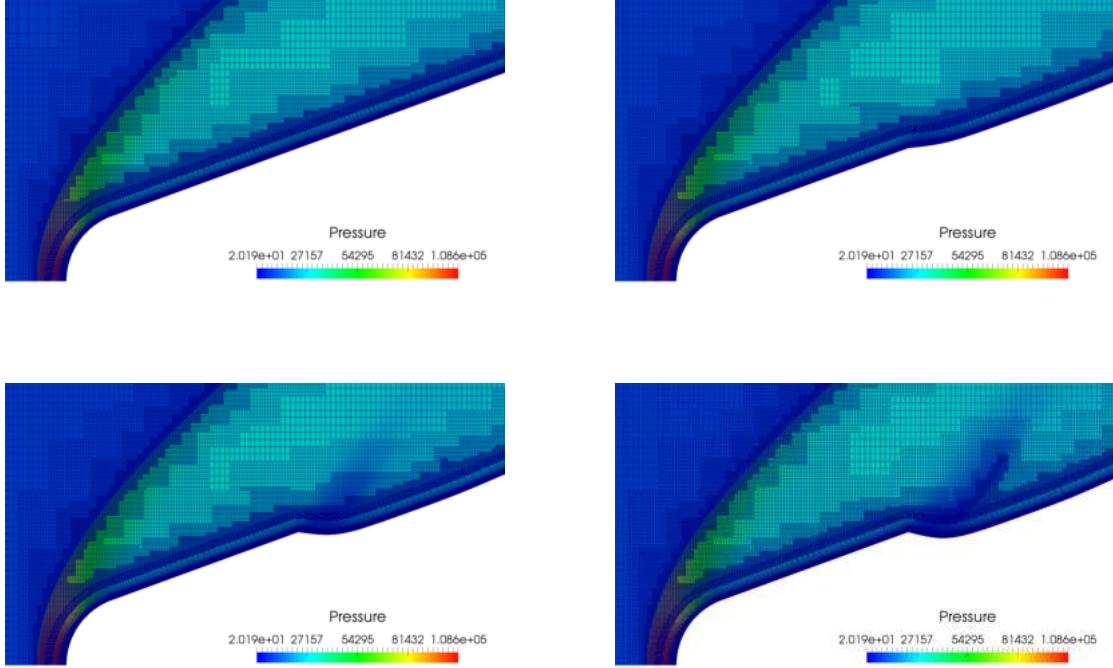


Figure 5.28: The automated mesh motion for in the arc-heater wedge test case.

The pressure fields and AMR levels for the unablated and ablated surfaces are shown in Fig. 5.29. It can be seen that the AMR is able to resolve both the bow shock and the reattachment shock for the ablated surface. Automatically creating a mesh that enables such high resolution of both shocks would be difficult to achieve with mesh motion/morphing techniques that have previously been used in ablation simulations. The simulations in Ref. [101] used a script and external meshing software to update the mesh as the surface recessed. It is likely that this script was geometry specific. In contrast, the mesh deformation algorithm used here is not specific to a given geometry and required no additional scripts or external software. In addition the mesh motion algorithms allow for a large amount of mesh deformation, as will be shown in the next sections.

Finally, the heat flux results for the unablated and ablated surfaces, and the experimental calibration plate are shown in Fig. 5.30a. The temperature field around the ablated surface is shown in Fig. 5.30b. For the unablated case the heat flux is slightly over-predicted for the first two experimental data points, but agrees well with the final data point. Considering the two-dimensional approximation of the test-box flow field, the agreement is reasonable. For the ablated case, there is a spike in the heat flux as the flow expands at the end of the nose, compressing the boundary layer.

Behind the nose the separation of the flow is indicated by a drop in the heat flux, and the subsequent reattachment is shown by a second spike in the heat flux. Downstream of the reattachment, the heat flux is notably higher for the ablated case.

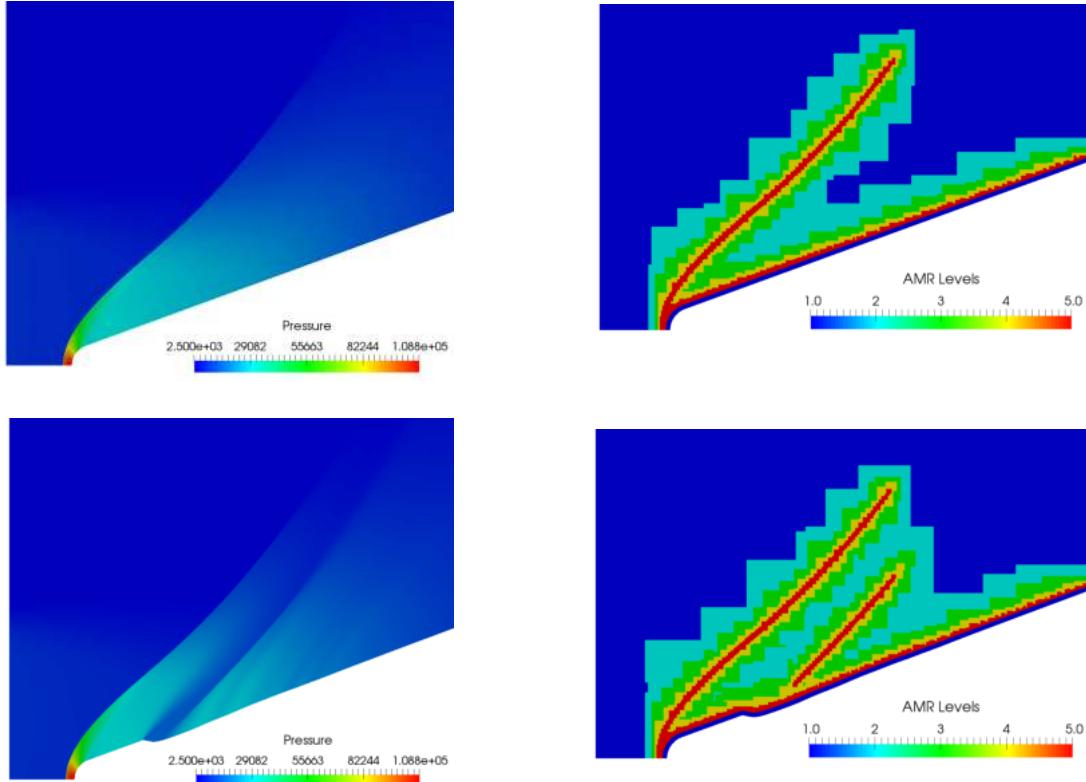
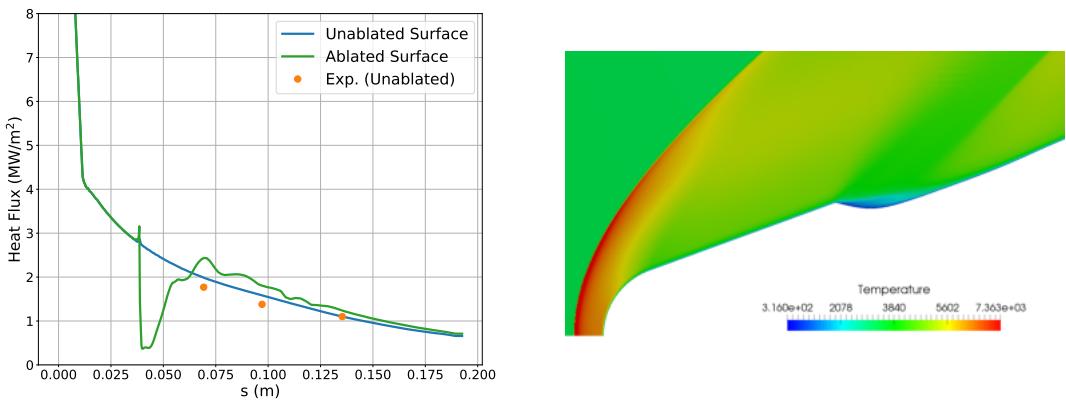


Figure 5.29: The pressure field and mesh refinement levels for the unablated and ablated surfaces.



(a) Comparison of surface heat flux results.

(b) Temperature field.

Figure 5.30: The surface heat flux and temperature field for the arc-heater wedge test case.

5.6.2 Recessing Nose-tip

The recession rate of a thermal protection system is strongly influenced by the state of the boundary layer. A turbulent boundary layer results in higher heat fluxes and shear stresses, and hence a higher rate of recession. When the state of the boundary layer changes on a vehicle's surface this can result in differential recession rates between the laminar and turbulent regions. This was observed during the Passive Nosetip Technology (PANT) programme, where extensive studies of ablating nose-tips were conducted [249, 293]. Figure 5.31 shows an example of a nose-tip shape that arose during experiments. The concave shape was attributed to transition from laminar flow in the stagnation region to turbulent flow close to the sonic point [249]. The shape is characterised by a flat section close to the stagnation point, where the flow is laminar. This is followed by a gouge near the sonic point, caused by the transition to turbulence, and a "scalloped" region downstream of the transition, where the boundary layer is turbulent [293].



Figure 5.31: An ablated graphite nose-tip, in a transitional flow regime (redrawn from Ref. [293]).

Researchers found that the nose-tip shape shown in Fig. 5.31 could lead to the development of an unsteady shock structure. This was experimentally investigated using shape stable, blunted double-cone geometries by Abbott *et al.* [1]. The experiments showed that the unsteadiness was caused by the growth of the separated region at the double-cone intersection. The separated region would grow until the bow shock moved far enough from the surface that the mass trapped in the separated region could escape over the shoulder of the upper cone [1]. This would reduce the size of the separated region shifting the shock structure towards the surface. The process was seen to repeat periodically, resulting in a "pulsating" flow pattern. The oscillating forces exerted on the body due to the pulsating flow raised concerns that it could lead to structural issues if it were to occur on a hypersonic vehicle [1].

The aim of this test case is to demonstrate the automated surface motion and mesh refinement that is enabled by the strand/CAMR solver. A hypothetical experiment, similar to those conducted in the PANT program, is simulated with the new solver. In the experiment a nose-tip is exposed to an arc-heater flow and ablates in such a way to produce the shape shown in Fig. 5.31. As the ablation time-scales are much longer than the flow time-scales, a pulsating flow could develop on the nose-tip as it ablates. To accurately simulate such an experiment, the solver must be able to automatically update the mesh throughout the surface shape change, from the unablated to the ablated shape. Then, the mesh must be able to adapt to accurately capture the unsteady separation region and shock structure that arises with the new shape. This is a simulation of a purely hypothetical, but representative, experiment where the aim is to fully demonstrate the new solver's capabilities.

The flow field in the arc-heater simulation is based on the DLR L3K arc-heater. The L3K simulation was conducted using a similar strategy to that used in the recessing wedge simulation above. First, the nozzle was simulated and then the outflow from the nozzle was used as the inflow to the arc heater test section. The test piece was placed at 0.1 m from the nozzle exit in order to give a nominal heat flux of 10 MW m^{-2} . To validate the computational set-up, a simulation of a forward facing cylinder was conducted as this is a typical configuration used with the DLR arc-heater [146]. The cylinder surface used an isothermal boundary condition, with the temperature set as 350 K, as per Ref. [146]. The species concentrations were set to give the equilibrium composition at 350 K, using the catalytic wall boundary condition. The simulation was run using the AUSM flux scheme and a five species mixture of air. The flow field for the validation case is shown in Fig. 5.32 and heat flux results are shown in Fig. 5.33. One can see that the heat flux is in good agreement with the expected 10 MW m^{-2} heat flux in the stagnation region. It can be seen that there are significant heat flux contributions from both the diffusive and conductive terms.

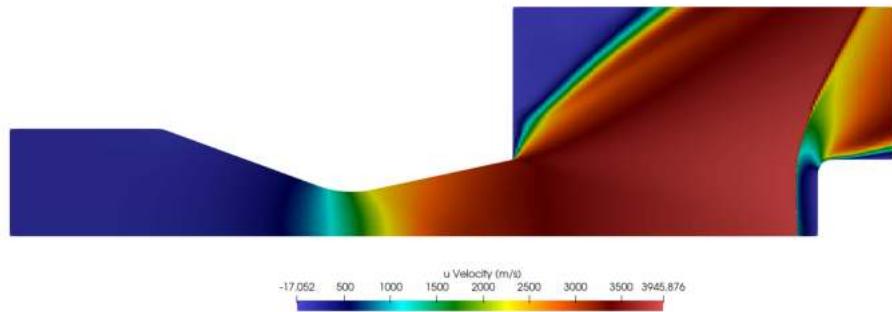


Figure 5.32: The x -velocity field in the DLR L3K simulation when using a cylindrical test piece.

A nose-tip was created based on the IRV-2 vehicle, with the geometry taken from Ref. [151]. The nose-tip radius was 0.01905 m and the cone angle 9 degrees. The

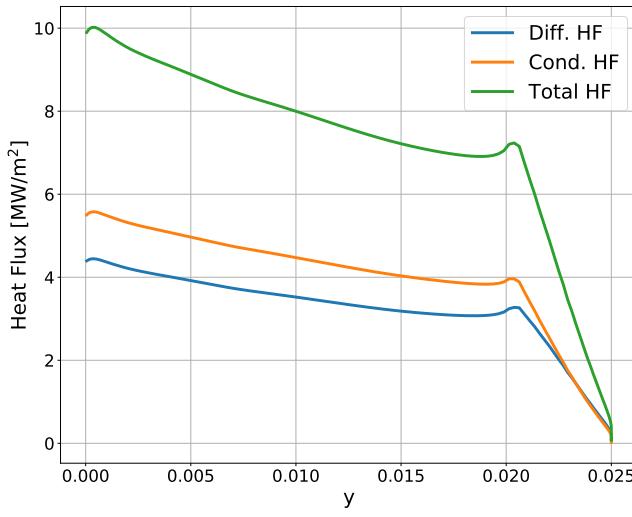


Figure 5.33: The surface heat flux results for the cylindrical test piece in the DLR L3K arc heater.

ablated shape was based on the parameterised shapes used in the PANT experiments [1]. The shape is specified by a nose radius for the first cone, R_n , first cone angle, θ_1 , first cone length, L_1 , fillet radius, R_f , second cone angle, θ_2 , shoulder radius, R_s , and test piece diameter, D , as shown in Fig. 5.34.

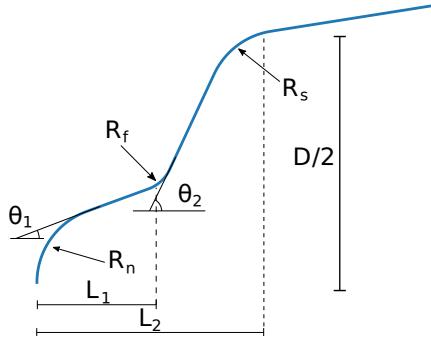


Figure 5.34: The parameters used to define the ablated nose-tip shape.

Currently, there is no method in the solver to predict ablation. Therefore, the geometric parameters were chosen to give a profile qualitatively similar to that obtained in the ablation experiments detailed in Ref. [293]. The parameters used in the simulations are given in Table 5.8. These parameters are also likely to give an unsteady flow based on the results from the experimental studies detailed in Ref. [1]. Finally, the surface of the second cone was given a sinusoidal profile in order to mimic the scalloping that can result from turbulent flow ablation and to further test the automated mesh generation algorithms. The ablated experimental surface and the two surfaces used in the simulation (unablated and ablated) are shown in Fig. 5.35.

$D/2$ (m)	L_1/D	R_n/D	θ_1 (deg)	θ_2 (deg)	R_f/D	R_s/D
0.01905	0.25	0.15	20	65	0.075	0.15

Table 5.8: The parameters used to define the ablated nose-tip shape.

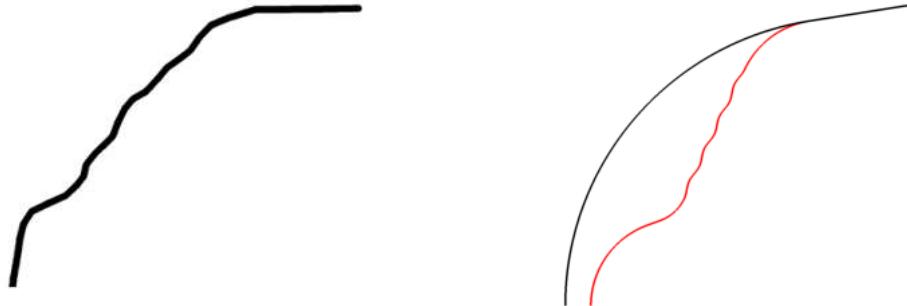


Figure 5.35: The upper half of the ablated nose-tip image redrawn from Ref. [293] (left) and the unablated and ablated nose-tip surfaces used in the simulations (right).

The strand mesh around the original nose-tip shape was created using 51 nodes on each strand, with a wall spacing of $1\ \mu\text{m}$ and strand length of 1 mm. The outer node smoothing residual was set to be 1×10^{-7} and the inner growth vector was not smoothed. The background domain included the entire test box and used a 100×70 base mesh with five levels of refinement, each with a refinement factor of two. First, the unablated shape was run until the flow became steady. Then the surface was updated in the same manner as in the wedge test case, where the surface mesh was moved from the initial surface to the final surface during the simulation, without having to re-initialise the flow field. The rate of surface deformation is not linked to the flow, but is controlled by the size of the ghost cells on the background domain and the grid motion CFL (see Section 4.2.4). Figure 5.36 shows the stages of the mesh motion from the original nose-tip shape to the ablated nose-tip shape. The only inputs required by the user were the initial surface, final surface and the surface deformation CFL (set to be 0.75). One can see that the mesh smoothly changes between the initial and final surface.

The pulsating flow is shown in Fig. 5.37. One can see that the dynamic mesh adaptation is able to accurately track the shock structures as they evolve. This test case clearly demonstrates the new solver's ability to automatically generate meshes around recessing surfaces with a large surface deformation, whilst resolving the dynamic shock structures. The level of deformation in this case is greater than any ablative cases that could be found in the literature. The combination of recession and dynamic shock capturing exhibited in this test case is a significant advance on previous automated meshing capabilities that have been applied to hypersonic flow simulations.

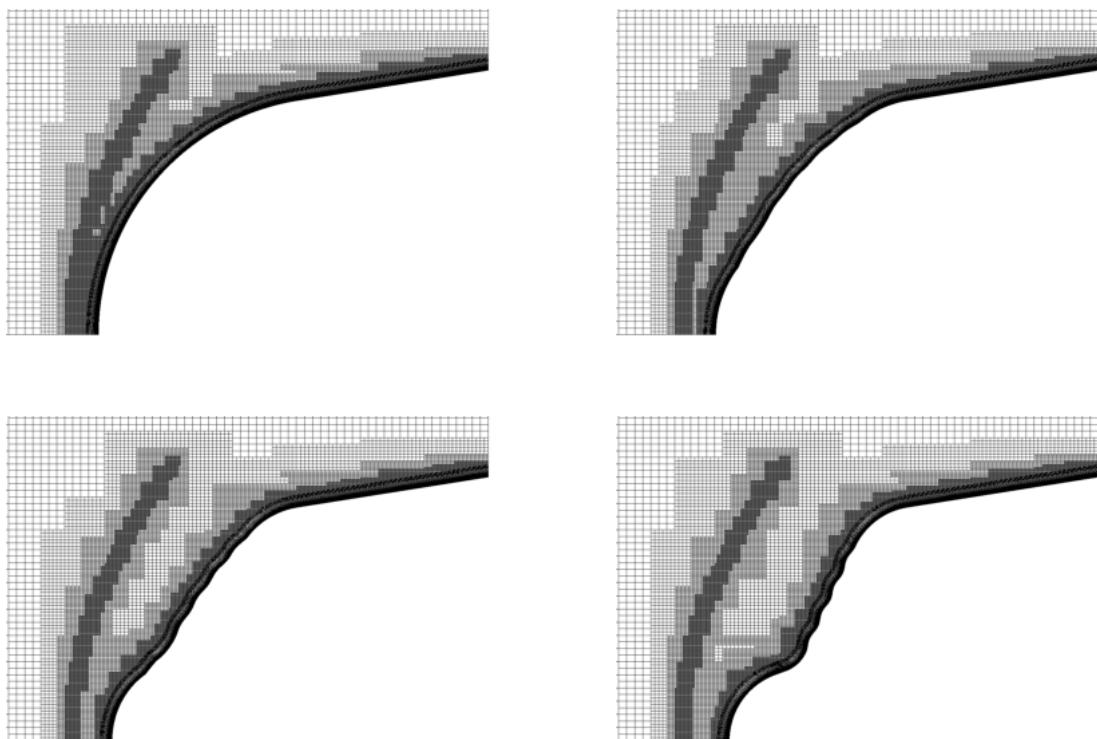


Figure 5.36: The automated mesh motion for the ablating nose-tip.

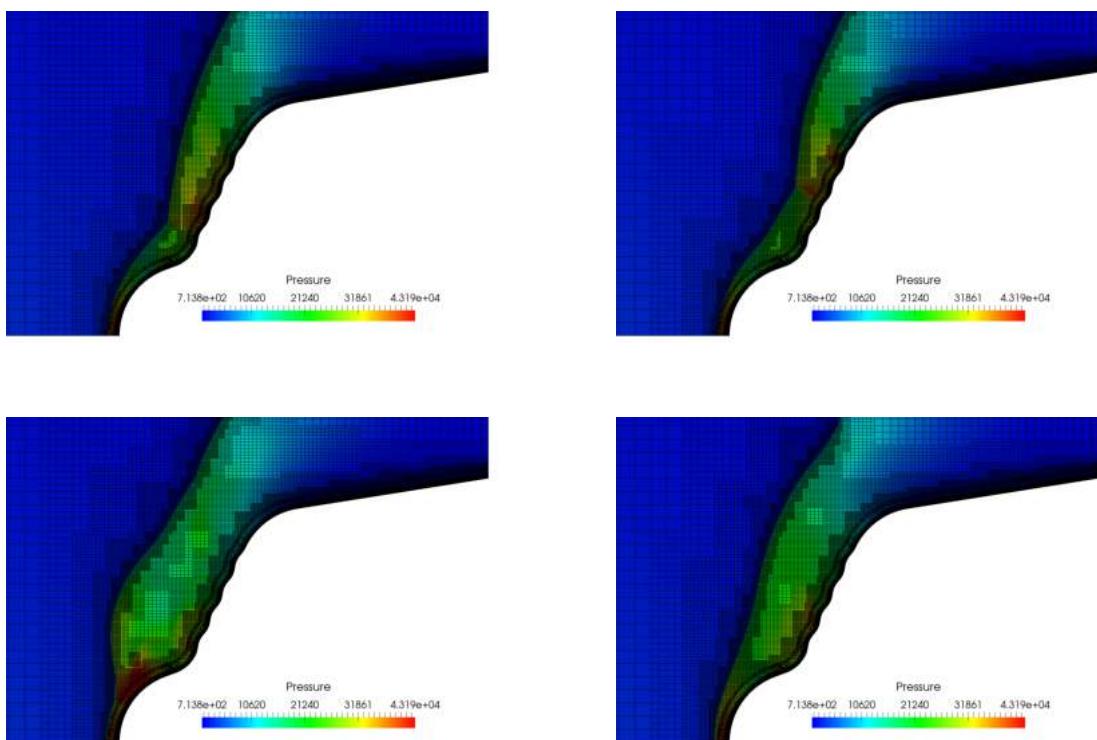


Figure 5.37: The automated mesh adaptation captures the pulsating shock structure.

5.7 Free-body Cylinder Simulations

One of the key advantages of using overset meshes is that they are well suited to moving body simulations. By combining overset methods with CAMR techniques the evolving shock structures can also be automatically resolved. In this work, free-body simulations have been conducted using the strand/CAMR solver, where the motion of the bodies is determined by the forces and moments acting on the body. The aim of this test case is to demonstrate the strand/CAMR solver's ability to simulate bodies in motion, whilst resolving the resulting dynamic shock structures.

5.7.1 Force Integration and ALE Flux Verification

The motion of the body is determined by integrating the equations of motion using the following process:

1. For each coarse-level AMR time step, the body is held in a constant position².
2. The forces and moments acting on the body are evaluated at the end of each coarse-level time step.
3. Before the next time-step, the equations of motion are integrated in time and the results are used to update the position of the body, the near-body strand mesh and the background hole-cut.

In this two-dimensional solver, three degrees of freedom are allowed: rotation about the z -axis and translation in the x - and y -directions. The strand mesh is updated at each time step by simply rotating and translating the surface vertex locations and growth vectors according to the motion of the body. As such, the more computationally costly smoothing and clipping procedures only need to be computed during the initialisation of the strand mesh.

To verify the integration of the forces over time, a simulation of a rectangular body with a fixed x -velocity, angular velocity and force in the y direction was conducted. The body's motion through the domain is depicted in Fig. 5.38 and a comparison between the computed and analytic position and body angle are shown in Fig. 5.39. It can be seen that the mesh dynamically adapts around the falling body and the analytic and computed solutions are in excellent agreement.

²The body position could be updated after each fine-level time step to improve the time-accuracy of the simulation. This was not implemented for these demonstration simulations.

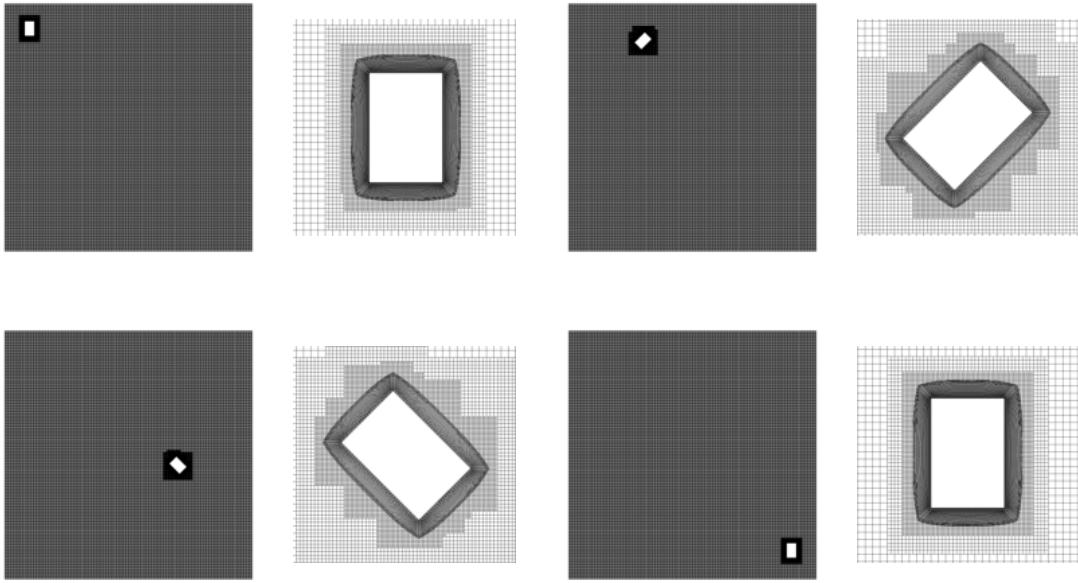


Figure 5.38: The motion of the rectangular body and strand mesh through the background Cartesian domain.

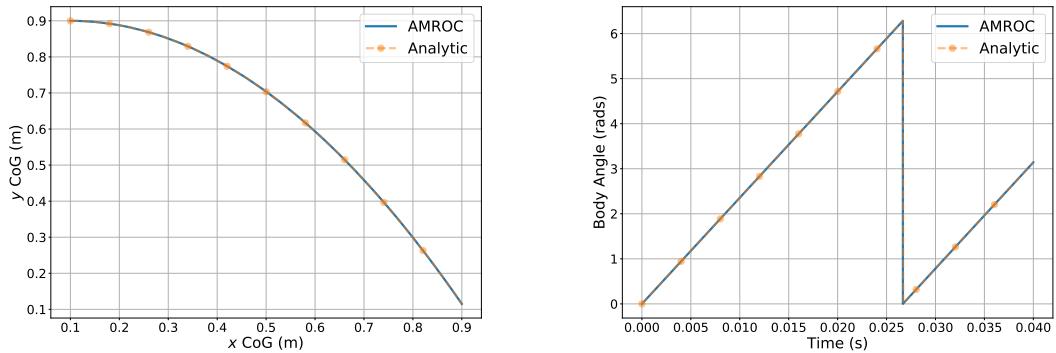


Figure 5.39: A comparison between the analytic and computed position and body angle for the free-body force integration test.

In addition to the force integration, the ALE formulation of the fluxes must be used to account for the mesh motion in the flux calculation. To verify the implementation of the ALE fluxes, inviscid flow around a half cylinder was computed using a strand/CAMR grid. One simulation used a fixed mesh with a positive flow velocity, and the other a moving mesh with zero flow velocity. This effectively changes the frame of reference from an Eulerian frame, where the body is fixed and the fluid is moving past the body, to a Lagrangian frame, where the body is moving through the fluid. The surface pressure results and velocities are compared in Fig. 5.40. Excellent agreement is obtain for the surface pressure results, and the velocity fields are equal and opposite.

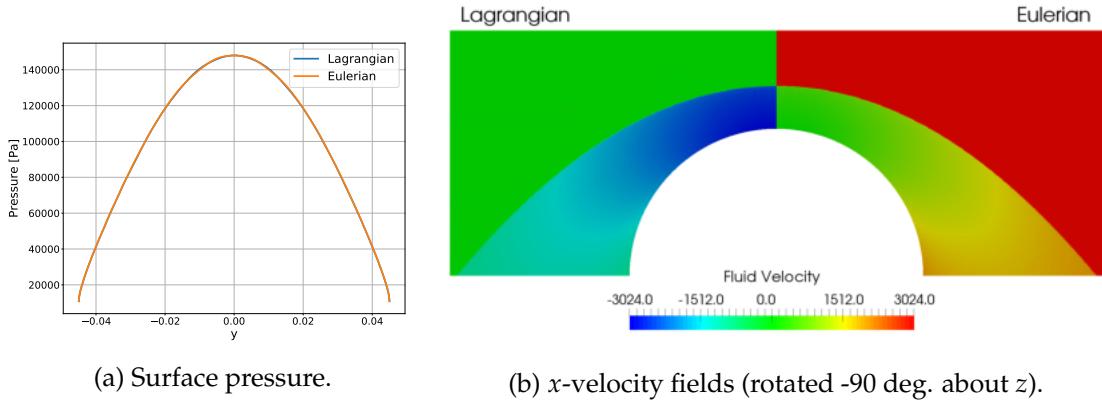


Figure 5.40: A comparison of results obtained when using an Eulerian or Langrangian reference frame.

5.7.2 Cylinder-Ramp Interactions

Inviscid simulations of two cylinders interacting with an oblique shock created by a 10 degree ramp were carried out. The first simulation used a 20 mm diameter circular cylinder, with the centre of the cylinder positioned 30 mm above the ramp. The second simulation used an elliptical cylinder with a semi-major axis of 25 mm and a semi-minor axis 15 mm with the centre of the cylinder positioned 35 mm above the ramp. For both simulations a strand mesh was used for the near-body mesh and the stationary ramp was created using the embedded boundary method within AMROC. The surface of the cylinders each contained 350 nodes, and the strand meshes used 3 mm strands with 35 cells in the wall-normal direction. The off-body domain had external dimensions of 460×250 mm and a base mesh of 450×250 cells with three levels of refinement, each with a refinement factor of two.

The freestream flow of five-species air was initialised at Mach 8, with a temperature of 300 K and a density of 3.26 gm^{-3} . The cylinders were fixed for 0.3 ms to allow the flow field to initialise. They were then allowed to translate and rotate freely through the domain under the influence of the aerodynamic and gravitational forces ($g = 9.81 \text{ ms}^{-2}$). The mass of the cylinders was set to be 33.2 g/m so that there would be a large amount of motion relative to the ramp surface in the 5 ms simulation time. Figures 5.41 and 5.42 show the mesh and temperature field in the domain at different times in the two simulations. It can be seen that the elliptical cylinder rotates away from the surface and has a lower y -velocity than the circular cylinder. This resulted in the elliptical cylinder interacting with the oblique shock for a greater proportion of the simulation time. The elliptical cylinder also had a faster x -velocity and reached the outflow boundary before the end of the simulation time.

For the simulation of the bodies in motion, no additional user input was required for the mesh generation other than the time that the motion should start. These results clearly demonstrated that the strand/CAMR technique is able to simulate free-body

dynamics whilst maintaining high resolution of the shock structures as they develop through the simulation.

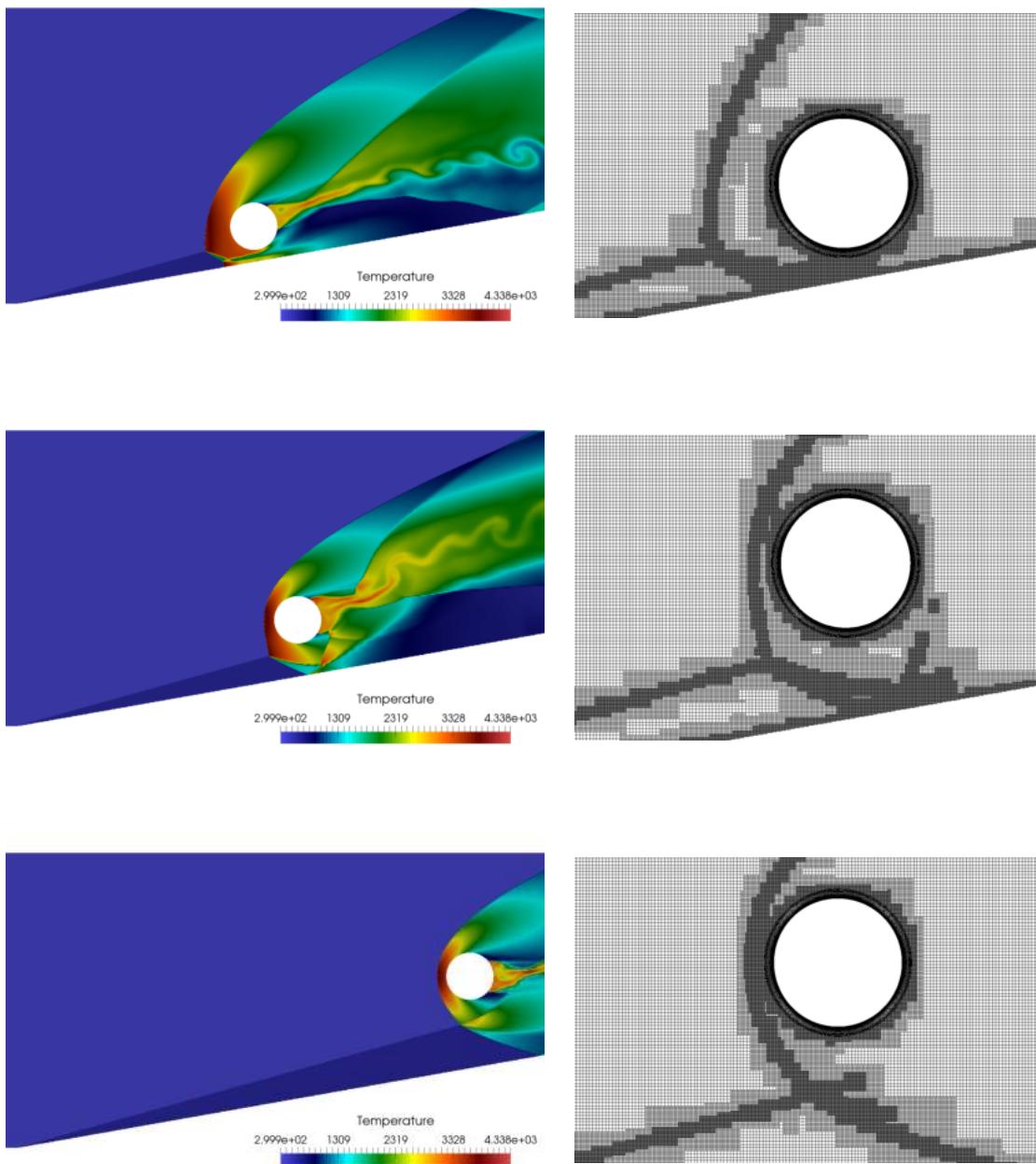


Figure 5.41: The translational-rotational temperature fields (left) and the mesh in the region of the cylinder (right) for the free-body simulation after 0 ms, 2 ms and 4.5 ms of motion.

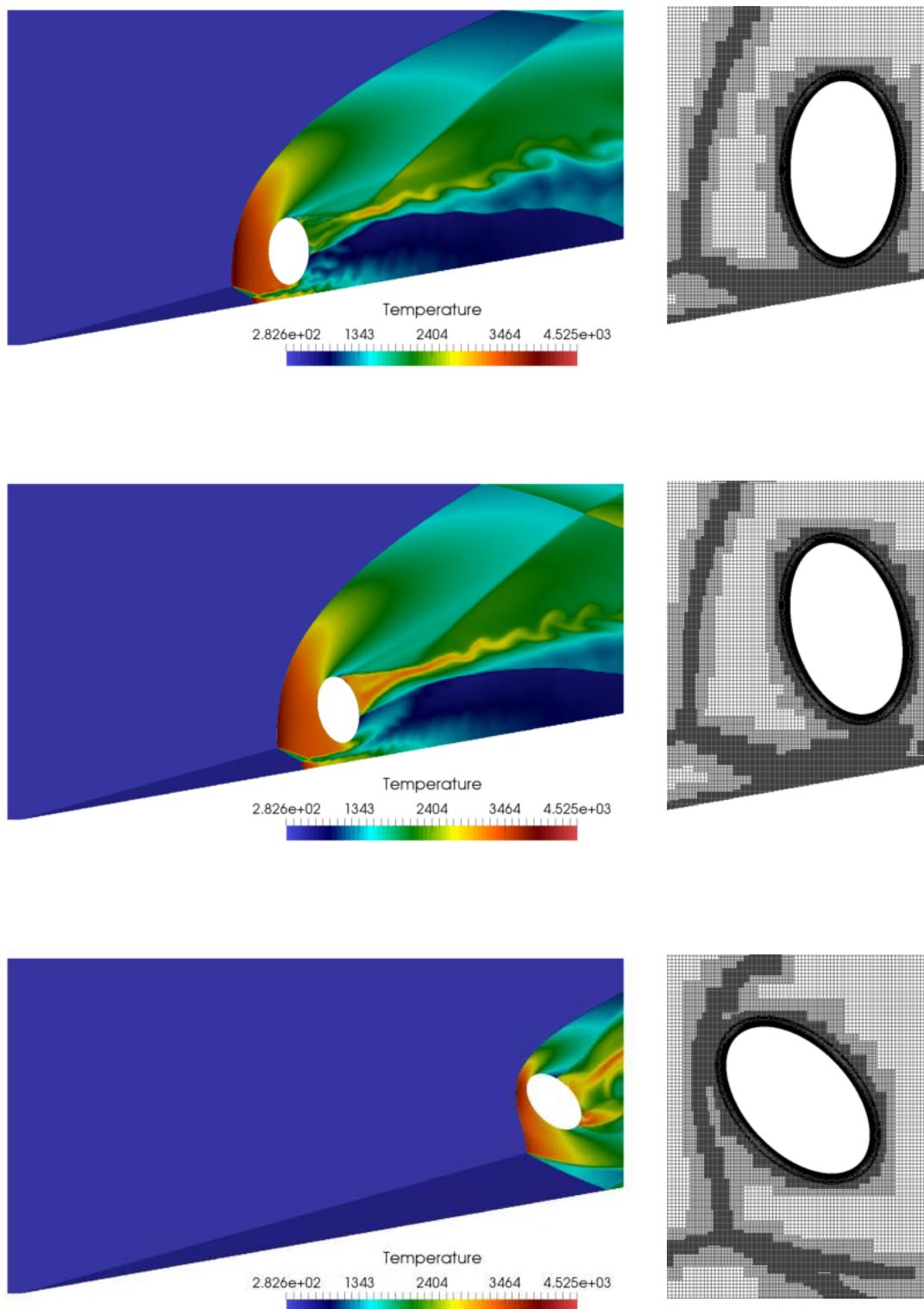


Figure 5.42: The translational-rotational temperature fields (left) and the mesh in the region of the elliptical cylinder (right) for the free-body simulation after 0 ms, 2 ms and 4 ms of motion.

5.8 Chapter Summary

The new solver has been thoroughly verified and validated in this chapter. The overset results are in good agreement with experimental and single domain results for both simple blunt body flows and for more complex shock-shock interaction cases. Previously, there have been very few investigations into the influence of overset methods on hypersonic heating results. The results obtained here are an important contribution as they demonstrate that overset methods can give accurate surface predictions in a range of scenarios. In addition, the mesh deformation and mesh motion algorithms have been verified and demonstrated. This is the first time that the strand method has been applied to surface deformation problems. The results clearly demonstrate that the strand/CAMR paradigm is well suited to automated meshing of deforming surfaces. Finally, it was shown that the new solver is able to automate complex simulations of bodies in motion.

In this chapter and the previous chapters a wide range of verification and validation evidence have been obtained for the new solver. This gives confidence that the solver is implemented correctly and able to give accurate predictions. Having built this base, the solver can now be used to simulate hypersonic flows that have not previously been investigated, whilst benefiting from the efficiency and automation enabled by the strand/CAMR paradigm.

Chapter 6

Unsteady Flow Predictions

In this chapter, the strand/CAMR solver is used to carry out simulations of complex flow phenomena. First, a double-wedge experiment is simulated using the new solver. Simulations of this experiment have previously been conducted, so the set-up itself is not new. However, this is the first time that overset methods and AMR have been used in simulations of this experiment. The shock structures are analysed in detail and the heat flux results are compared with those from the experiment and previous simulations. This provides additional verification and validation evidence for the simulation of dynamic shock structures, large separation regions and Shock-Boundary-Layer Interactions (SBLIs) using the strand/CAMR method.

The solver is then used for two novel numerical investigations. The first builds on the Type IV shock interaction simulation from Section 5.5. The area of extreme, localised heating seen in this case is likely to lead to differential recession of an ablative TPS. To simulate this, small recessed areas of increasing depths are created on the cylinder surface. The effect the recessed region has on the flow field is investigated. The second set of simulations investigates the unsteady flow seen in double-cone simulations and experiments, which were previously discussed in Section 5.6.2. There has recently been a renewed interest in the relationship between double-cone geometries and the resulting flow patterns [125, 238]. The effect of nose bluntness on the unsteady flow patterns has not previously been investigated numerically. Thus, the flow over different blunted double-cone surfaces is simulated, enabling a greater understanding of unsteady double-cone flows.

6.1 Unsteady Double-Wedge Flow

Double-wedge configurations are commonly used to assess a solver's ability to accurately capture shock-shock and shock-boundary-layer interactions. Around such

geometries the interaction of the shocks from the upper and lower surfaces results in a transmitted shock that impinges on the upper surface. This shock impingement creates an adverse pressure gradient, resulting in a large separation region at the inner corner of the wedge surfaces. The separation region then creates additional shock waves, further complicating the shock structure.

The heat flux experienced by a double-wedge is dependent on the complex interaction between the shocks and the separation region. The heat flux is lower in the separated region and peaks at the point where the transmitted shock impinges on the upper surface. Consequently, accurate prediction of the surface heat flux requires accurate simulation of the various shocks, the separation region and the shock-boundary-layer interactions. The AMR available in the new solver can be used to efficiently resolve the dynamic shock structures and accurately simulate the shock interactions. This test case can also be used to build confidence that the overset method can be applied to a wide range of geometries, where there may be multiple discontinuities passing through the overset boundary.

6.1.1 Simulation Set-up

An experimental case of a 30-55 degree double-wedge in a flow of pure nitrogen was simulated using the new solver. The experiment was carried out in the Hypervelocity Expansion Tunnel at the University of Illinois at a Mach number of 7.14 and stagnation enthalpy of 8 MJ kg^{-1} . The geometry, inflow conditions and experimental results are taken from Ref. [267] and the freestream is assumed to be in thermochemical equilibrium. The double-wedge geometry is shown in Fig. 6.1 and the inflow conditions are given in Table 6.1. This experiment has been simulated by various authors [14, 115, 227], used in a code-to-code comparison exercises [149], and recently analysed in detail by Reinert *et al.* [226]. Whilst this set-up has been simulated by others, this is the first time it has been numerically investigated using AMR and overset methods. The existence of previous results allows for additional verification evidence to be obtained for the new solver.

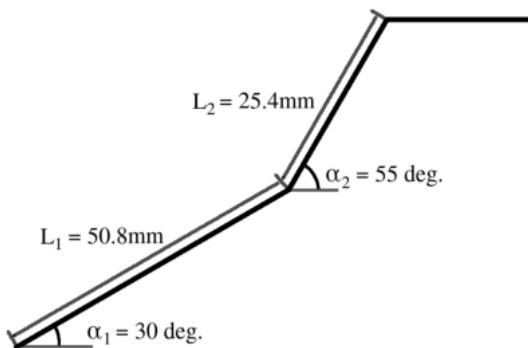
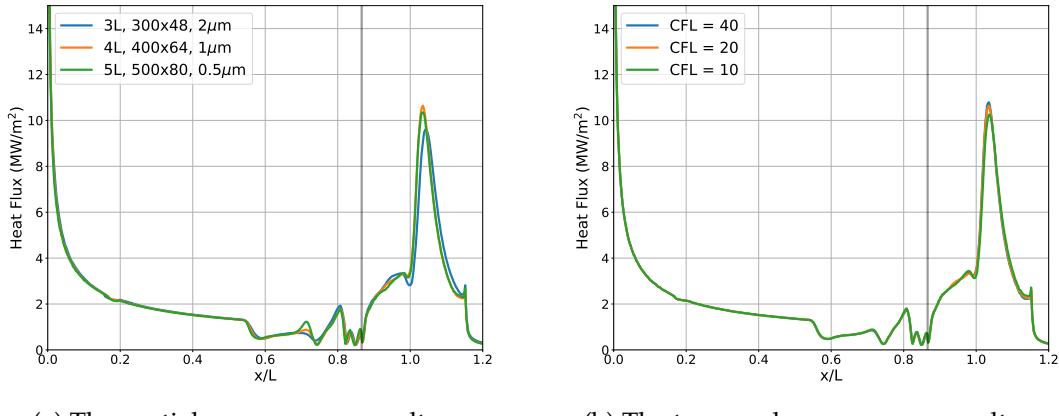


Figure 6.1: The double-wedge geometry.

T_∞	ρ_∞	U_∞	Y_{N_2}	Y_N
710 K	3.8 g/m ³	3812 m/s	1.0	0.0

Table 6.1: The freestream conditions for the double-wedge simulation.

Following a grid convergence study (see Fig. 6.2a), a strand mesh was created with 400 strands and 65 nodes on each strand. The length of each strand was set to be 1.5 mm and the initial spacing at the wall was set as $1.0\ \mu\text{m}$. The outer region of the strand mesh was smoothed to a residual of 1.0×10^{-7} . No smoothing was used in the inner region to ensure orthogonality at the wall. The CAMR domain was 90 mm \times 110 mm in size, with a base mesh of 135×135 cells and four levels of refinement, each with a refinement factor of two. The simulation was run using a thermochemical nonequilibrium two species nitrogen model with Park's reaction rate constants, the AUSM flux scheme and the Blottner/Euler/Wilke transport model. On the near-body domain implicit time-stepping was used, with the line-relaxation solver and a linear solver termination criteria of 1×10^{-7} . The results of a time convergence study are shown in Fig. 6.2b where it can be seen that a CFL number of 40 gives results that are converged in time.



(a) The spatial convergence results. (b) The temporal convergence results.

Figure 6.2: The convergence study results for the double-wedge simulation at 120 μs . The grey line shows the intersection of the cones.

6.1.2 Results and Discussion

6.1.2.1 Shock Structure

The unsteady nature of the flow leads to a dynamically evolving shock structure that changes over the duration of the simulation. The AMR enabled excellent resolution of the shock structures throughout. Images of the pressure field and the mesh for the entire domain are shown in Fig. 6.3. The temperature field and mesh in the shock

interaction region is shown in Fig. 6.4. One can see that the AMR provides an efficient spatial discretisation, where the mesh is refined for the various shocks and coarsened in smoother regions of the flow. This process was highly automated, with pressure and density gradients used as the refinement criteria. The high resolution of the flow features enables accurate predictions and detailed assessments of the double-wedge shock structures.

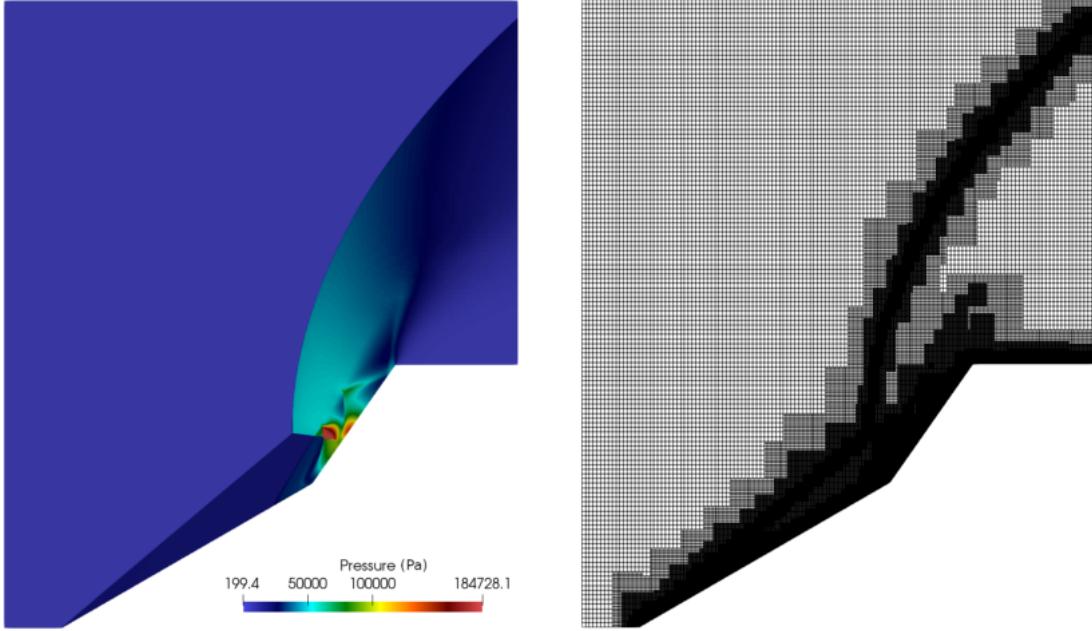


Figure 6.3: The pressure field (left) and strand/CAMR mesh (right) for the double-wedge simulation at $75 \mu\text{s}$.

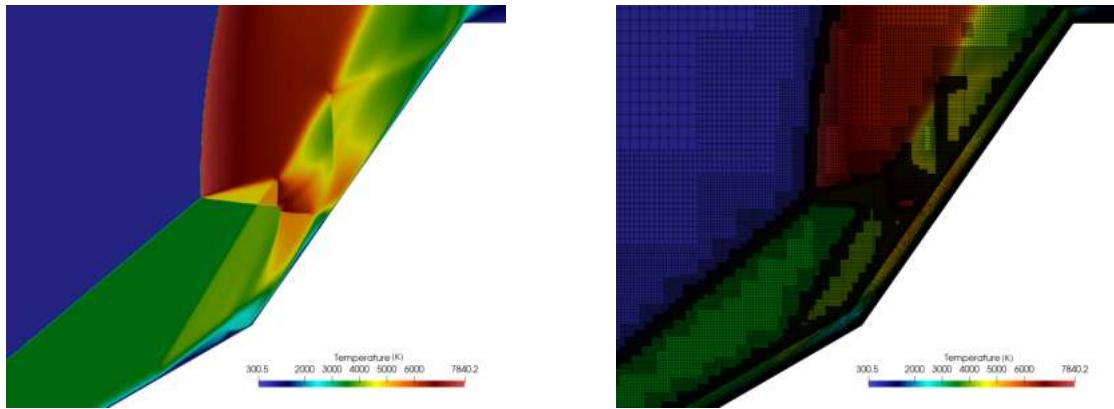


Figure 6.4: The translational-rotational temperature field (left) and the temperature field over-layed with the strand/CAMR mesh (right) at the double-wedge intersection.

Enhanced views of the pressure and x -velocity fields at the corner of the double-wedge are shown in Fig. 6.5 and a schematic of the shock structure is shown in Fig. 6.6. The attached oblique shock from the leading edge of the first wedge interacts

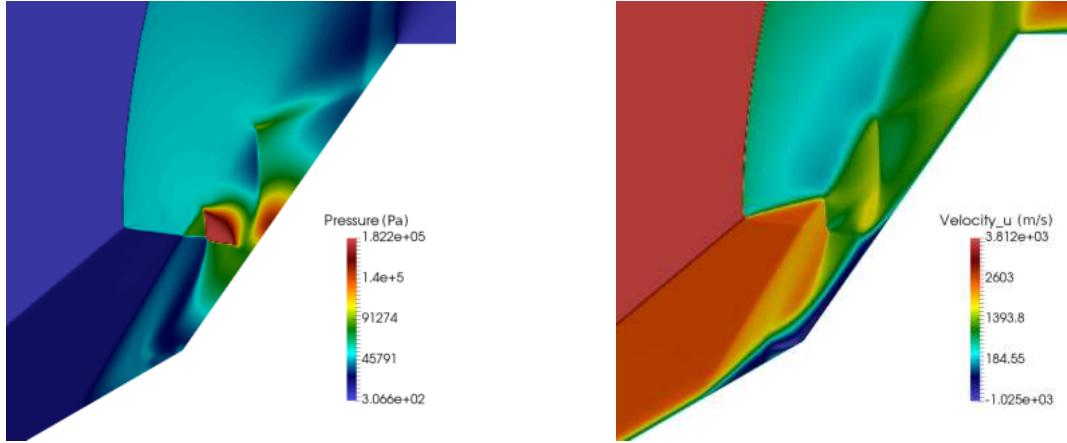


Figure 6.5: The pressure field (left) and the x -velocity field (right) at the corner of the double-wedge.

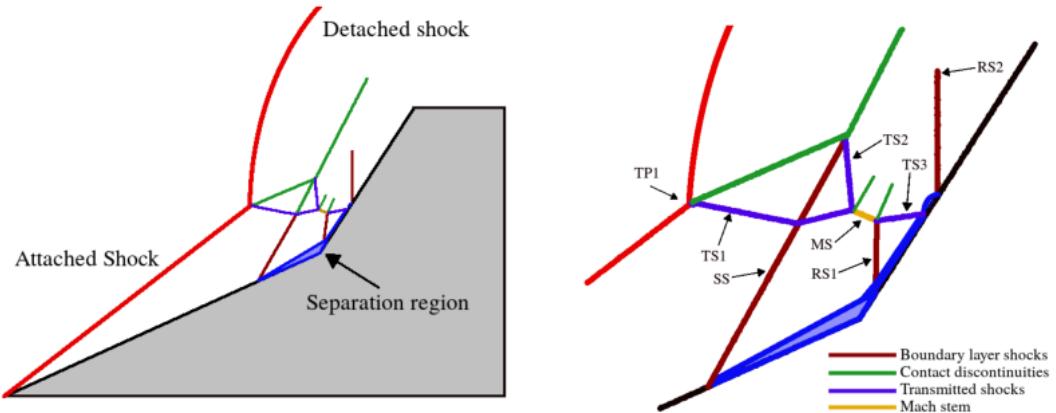


Figure 6.6: The shock structure in the double-wedge simulation at $75 \mu\text{s}$.

with the detached bow shock from the upper wedge. This creates a triple point (TP1) and a transmitted shock (TS1). TS1 is connected to the reattachment shock (RS1) by a Mach stem, which has a triple points at each end, leading to two more transmitted shocks (TS2 and TS3). This is similar to what is observed in a Type V shock interaction. The upper transmitted shock (TS2) impinges on the contact discontinuity and the lower transmitted shock (TS3) impinges on the surface.

Between the two transmitted shocks, downstream of the Mach stem, there is a small region of subsonic flow. At the point where TS3 impinges on the surface there is another smaller separation region, which creates a second reattachment shock (RS2). Early in the simulation, the separation shock (SS) interacts with RS1 and TS3 below the Mach stem. As the separation region grows SS moves closer to TP1, passing through the Mach stem and then interacting with TS1 above the Mach stem. The latter scenario is shown in the figures. Throughout this period TP1 moves away from the upper surface as the bow shock grows. This causes the impingement of the

transmitted shock on the upper surface to move upstream as well, towards the double-wedge intersection.

The structure described above breaks down as the separation region grows. This causes SS to become stronger and move closer to TP1, changing the angle of TS1. In addition, the angle of RS1 becomes more aligned with the upper wedge. As TS1 and RS1 become more aligned the Mach stem disappears. This occurs at approximately $90 \mu\text{s}$. At this point TS1 still impinges on the surface of the upper wedge (after a number of interactions) and the impingement is still moving upstream as the bow shock grows. After approximately $120 \mu\text{s}$ SS interacts with the oblique shock from the lower wedge. This changes the angle of the oblique shock and causes TP1 to move upwards and downstream. In turn, this causes the transmitted shock to impinge higher on the upper wedge surface. Images of the shock structure after $90 \mu\text{s}$ and $130 \mu\text{s}$ are shown in Fig. 6.7. The shock structure continues to become more complex as the separation region grows and smaller re-circulation regions form within it, each generating there own shocks. The growth in the separation region moves the triple point between the separation shock and the bow shock upwards. As a result, the shock impingement at the final simulation time has moved close to the the shoulder of the upper wedge.

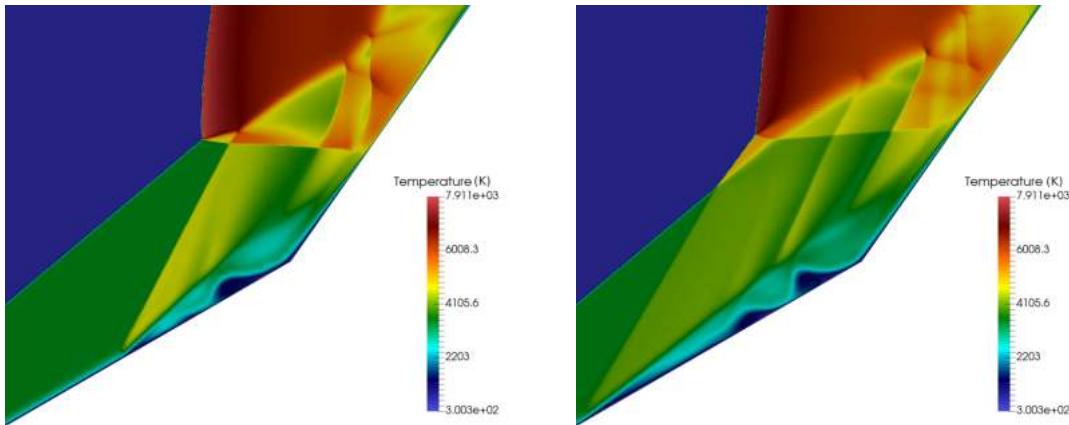


Figure 6.7: The temperature field at the corner of the double-wedge after $90 \mu\text{s}$ (left) and $130 \mu\text{s}$ (right).

Previous numerical studies have shown differences in the shock structure when comparing two- and three-dimensional simulations. This is likely caused by the development of three-dimensional flow features in the separation region, or by edge effects, due to the relatively narrow double-wedge test piece [115]. Differences in the bow shock have been observed after $100 \mu\text{s}$ [115, 149], where the two-dimensional simulation gives a larger shock stand-off distance. Similarly, Reinert *et al.* [226] found that the flow field could only be considered two-dimensional up to $85 \mu\text{s}$. After this time, the separation region in the two-dimensional simulation grows larger than in the three-dimensional simulation. This results in the separation shock moving further upstream in the two-dimensional simulation. Both of these differences will influence

the location of the shock impingement and therefore the heat flux on the upper surface.

6.1.2.2 Reference Result Comparison

The results obtained using the new solver are compared against experimental and numerical results from the literature. Experimental results are available for the time-averaged heat flux on the double-wedge surface [267]. The total experimental time was $242 \mu\text{s}$ and the response time of the heat flux gauges was $1 \mu\text{s}$. The time-averaged simulated heat flux for different time-periods is shown in Fig. 6.8. A start time of $10 \mu\text{s}$ is used to allow for difference in the start up of the simulation the physical test. Previous authors have used a variety of time intervals to calculate the time-averaged results. Thus, time-averaged results are shown for end times of 120 , 180 and $240 \mu\text{s}$. The results show some sensitivity to the time-averaging window, but the simulated results pass through many of the experimental measurements in all cases.

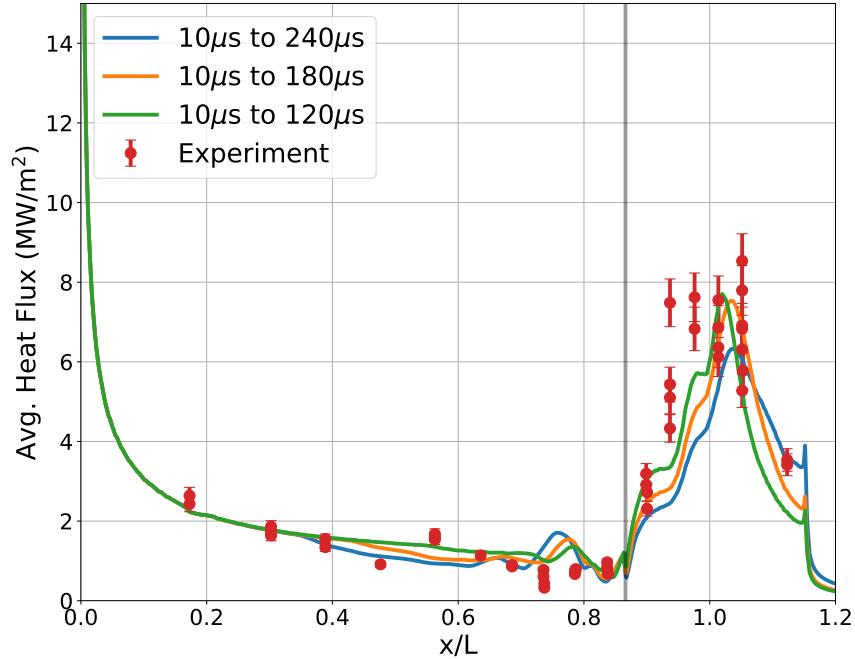


Figure 6.8: The time-averaged heat flux for the double-wedge simulation.

The results up to $180 \mu\text{s}$ are in better agreement than those that include the entire simulation time. This could be due to the use of a two-dimensional domain, as previous computational studies have shown that there are three-dimensional effects, which become more pronounced at later times. As discussed above, the shock stand-off distance and extent of the separation region are greater in two-dimensional cases. These differences will have an influence on the location of the impingement on

the upper surface and will likely shift the location of the shock impingement downstream.

The heat flux results are shown in Fig. 6.9 alongside two-dimensional simulation results from Ref. [115] at $60\ \mu\text{s}$ to $240\ \mu\text{s}$. The results from Ref. [115] were produced using a different solver and a static, single-domain mesh. One can see that remarkable agreement is obtained at $60\ \mu\text{s}$, and excellent agreement at $120\ \mu\text{s}$. At these times the heat fluxes are highest due to the strength of the shock impinging on the surface. These results indicate that the overset mesh has little impact on the results when the shock impingement is strongest.

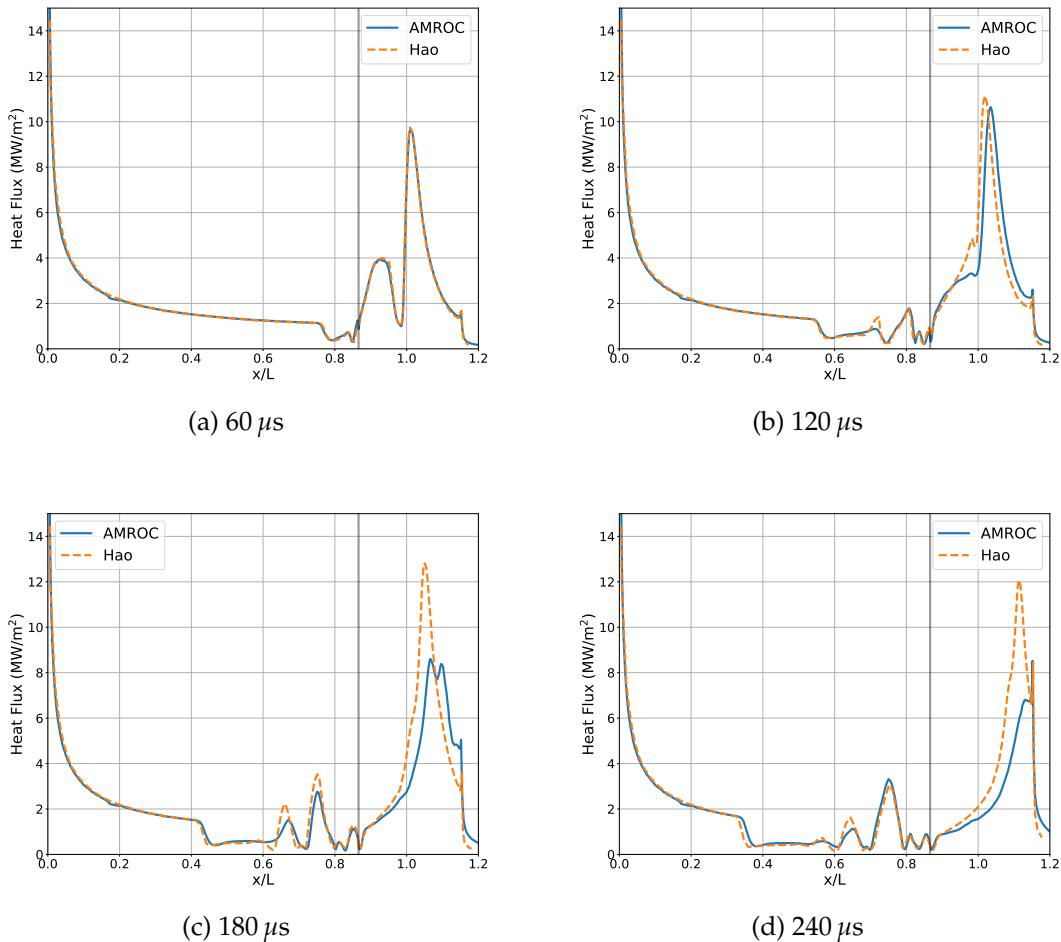


Figure 6.9: A comparison of the heat flux results at different simulation times with single domain results from Ref. [115]. The grey line shows the intersection of the cones.

At all simulation times the heat fluxes in the separated region upstream of the wedge intersection are in excellent agreement. It can be seen that the location of the start of the separated region and the location of the peaks in heat flux within this region are very similar in both simulations. The small, secondary separation bubbles cross the

overset boundary so these results indicate that the overset method has limited impact on the development of separated regions.

At simulation times of $180 \mu\text{s}$ and $240 \mu\text{s}$, it can be seen that there are differences in the location and magnitude of the heating caused by the shock impingement. It is possible that this is due to differences in the complex shock interactions in the off-body region. These interactions dictate the direction and strength of the transmitted shocks and determine the location of the shock impingement. An example of this is shown in Fig. 6.10. The numerical Schlieren from the AMROC simulation is shown at $240 \mu\text{s}$ and points have been overlayed indicating the locations of high density gradients in the numerical Schlieren in Ref. [115]. It can be seen that the oblique shock, separation and reattachment shocks are in very good agreement. However, the location of the triple point between the bow shock and the separation shock is in a different location. This influences the shock impingement location, leading to the difference in the surface heating results. The location of the triple point depends on the position of the bow shock and the interactions between the separation shocks and the oblique shock. For example, it can be seen that the direction of the primary separation shock changes slightly following its interaction with the shock from the first small separation bubble. The difference in shock positions in this simulation and those in Ref. [115] may be partly due to the AMR's ability to resolve the complex shock structures throughout the simulation.

Larger differences at later simulations times are expected to some extent, as the differences in the models are integrated over time and will propagate in a process analogous to error propagation. In addition to the differences in the meshing strategy, there are several differences in the numerical methods used, including in the time integration, the flux scheme, the transport property calculations and the thermochemical coupling. Previous code-to-code comparisons of the same simulation have shown larger differences in results than those seen here [149]. As such, the agreement between the results is relatively good: there is excellent peak heat flux agreement at earlier times and excellent agreement in the fore-body separation at all times.

6.1.2.3 Numerical Methods Comparison

As previously discussed, the simulation of this complex, unsteady flow is likely to highlight differences in the numerical methods. Thus, this simulation was used to examine the influence of different methods on the heat flux results at a simulation time of $120 \mu\text{s}$.

The simulations were run using different time-stepping methods: explicit, unpartitioned implicit backward Euler (BDF1) and dimensionally partitioned IMEX

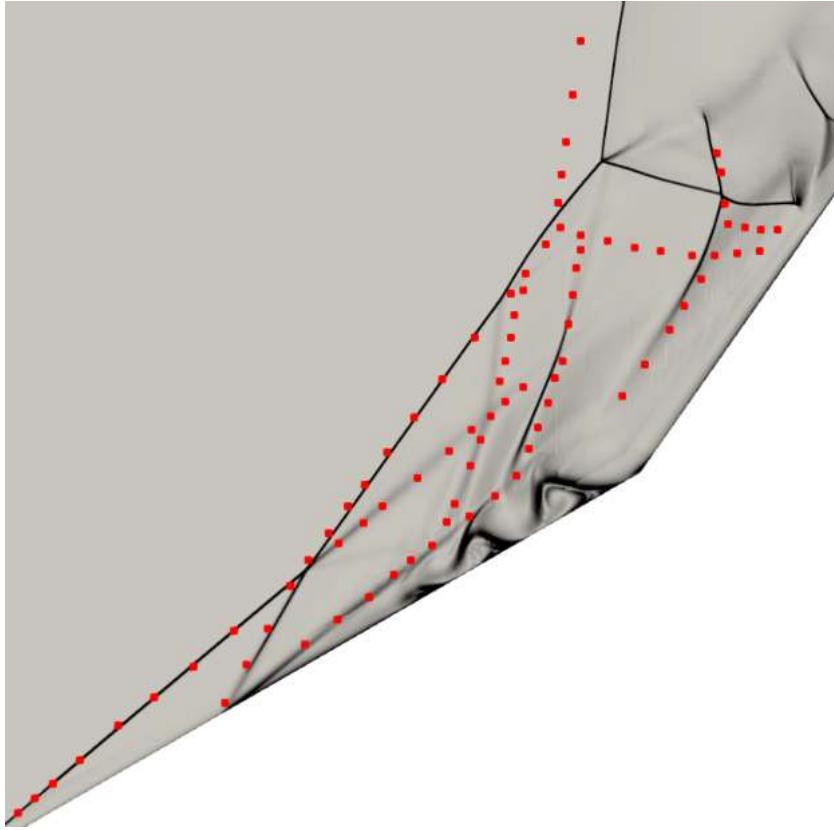


Figure 6.10: The numerical Schlieren at $240 \mu\text{s}$ from AMROC overlaid with red points indicating the positions of shocks in the Schlieren image in Ref. [115].

methods. When using the IMEX method, a CFL number of 40 in the wall-normal direction gave a CFL number of approximately 0.2 for the explicit partition in the wall-tangential direction. The fully explicit method used a CFL number of 0.75 for both the near-body and off-body domains. Using these CFL numbers resulted in the explicit method carrying out approximately 50 times more iterations than the implicit methods. The heat flux results are shown in Fig. 6.11. The heat flux for the explicit method is only shown at the $60 \mu\text{s}$ due to the extremely large computational time. It can be seen that all of the methods are in very good agreement. There are small differences in the magnitude of the peak heat flux, but the location of the peak heat flux, size of the separated region and the location and magnitude of the smaller peaks in heat flux within the separation region are all in excellent agreement. These results provide further verification evidence for the use of implicit methods in time-accurate simulations.

The computational times for the different methods are shown in Table 6.2. All of the simulations were run using 40 processors. For the explicit method the total time up to $60 \mu\text{s}$ was doubled and rounded to the nearest thousand in order to obtain an estimate for the time required to reach $120 \mu\text{s}$. The speed-up enabled by the implicit method is clearly demonstrated by this test case, and the IMEX method is shown to give the

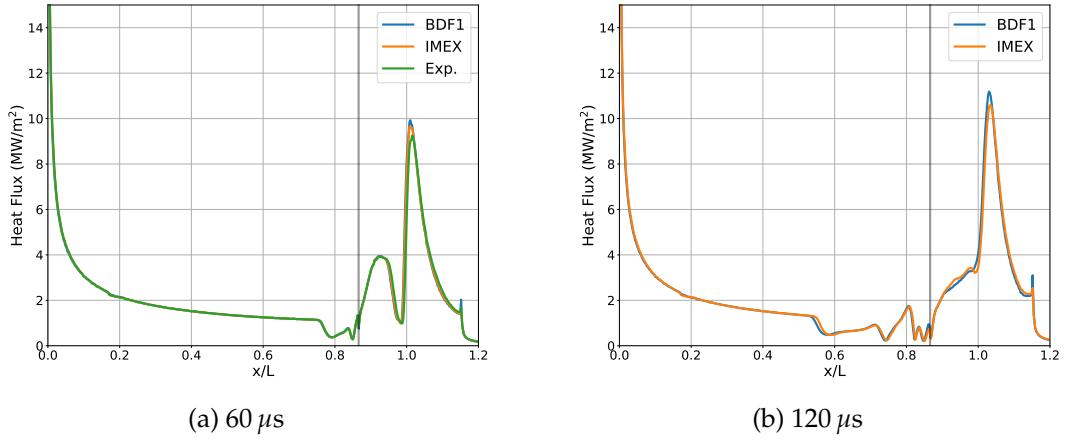


Figure 6.11: A comparison of the heat flux results at different simulation times using different time integration methods. The grey line shows the intersection of the cones.

fastest computational time. The table shows that the difference in run time between the IMEX method and the backward Euler method is largely due to the reduction in the Jacobian creation time. This is because the fully implicit method requires a block penta-diagonal Jacobian, whereas the IMEX method only requires a block tri-diagonal Jacobian. There is also a more modest reduction in the linear solver time as the direct, block tri-diagonal solver can be used along each strand in the IMEX method, which is faster than the line-relaxation solver. For both implicit methods, the Jacobian creation dominates the total time. Comparing the IMEX and explicit methods, the time spent in the time-integrator is reduced by a factor of approximately 17.1. The total speed-up is greater at 21.3 as the explicit method requires significantly more iterations. Additional iterations add extra cost as various operations such as parallel redistribution and synchronisation, source integration, overset exchange and setting boundary conditions are carried out once per each iteration. The significant speed-up obtained relative to the explicit method justifies the considerable effort spent implementing the implicit time integration routines and linear solvers.

	Jac. creation (s)	Lin. solve (s)	Time integ. total (s)	Simulation total (s)
BDF1	9,858.4	911.8	13,079.4	19,389.2
IMEX	4,719.2	514.0	7,957.7	15,928.3
Explicit	N/A	N/A	≈ 136,000	≈ 340,000

Table 6.2: The wall times for the different time integration methods.

The results when using the AUSM and HLLC flux schemes are compared in Fig. 6.12. Very little difference can be seen in the results. Flow fields visualisations show minor differences in the shock resolution with the HLLC giving slightly better defined discontinuities. However, overall, the results indicate that the flux schemes perform

very similarly for this simulation. These results add further verification evidence for the inviscid flux scheme implementation.

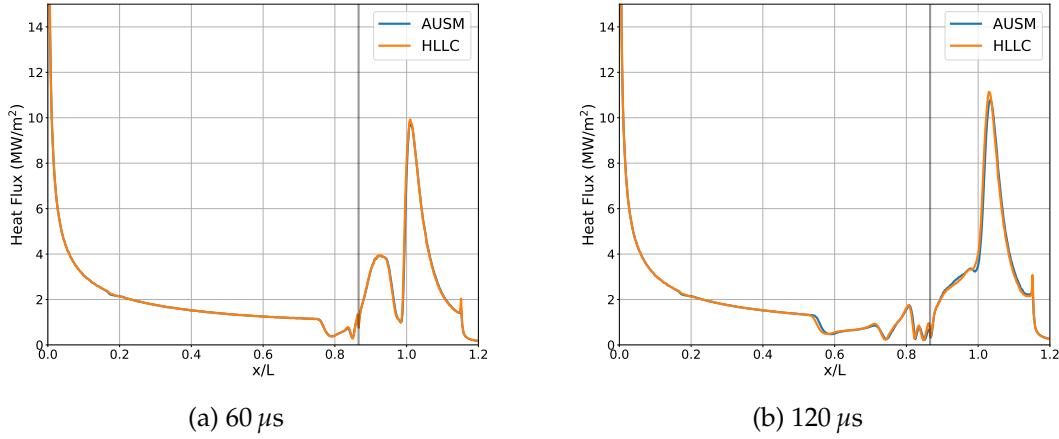


Figure 6.12: A comparison of the heat flux results at different simulation times using the AUSM and HLLC flux schemes.

6.1.3 Summary

The new strand/CAMR solver has been used to simulate an experiment of hypersonic flow over a double-wedge. The AMR was able to efficiently capture the dynamic shock structures, enabling a detailed assessment of the flow features. The surface heating results were then compared with experimental results and simulation results from the literature. The agreement obtained with the reference results gives confidence that the overset solver can give accurate heating predictions for complex flows with shock-boundary-layer interactions. To the author's knowledge, this is the first time that overset heating results have been obtained for flows with multiple shocks crossing the overset boundary. These results build further confidence that the strand/CAMR method can be used for a wide range of geometries with complex shock structures.

6.2 Type IV Shock Interaction with Surface Recession

Type IV shock-shock interactions can result in extremely high heating rates that are confined to a very small area. This was shown in Section 5.5, where simulations and experiments of a Type IV interaction with a cylinder showed that the peak heating was over 12 times the heating experienced by a cylinder alone. Other experiments have shown similar results with heating amplification factors of over 25 being observed [235, 286]. This intense localised heating is likely to lead to differential ablation rates if a vehicle is using an ablative TPS. This could result in cavities being created in the surface. An example of this was seen in the X15 flight tests, where a hole was created in an external pylon as a result of intense localised heating caused by a shock interaction [9].

There have been very few previous investigations into the effect of cavities on a Type IV interaction. The only example that could be found was an experimental and numerical investigation in Ref. [297]. In this study a large, square cavity was created in the centre of a cylinder, and the cylinder was placed into a hypersonic flow. An oblique shock was created using a flat plate, resulting in a Type IV shock interaction between the cylinder bow shock and the oblique shock. The study in Ref. [297] showed that the introduction of a cavity can alter the unsteady flow structures. However, the shape, location and size of the cavity differs from what would be expected when there is intense localised heating on an ablative TPS. The geometry in Ref. [297] was based on previous work that showed square cavities could reduce the aerothermal loads at the stagnation point [78, 236]. Therefore, a set of simulations has been carried out to study the effect that differential ablation could have on the flow structures. To the authors knowledge this is the first investigation of Type IV interactions with non-square, off-centre cavities, akin to those expected on an ablative leading edge.

6.2.1 Simulation Set-up

The simulations used the same set-up that is described in Section 5.5. The intermediate grid was used for all simulations as it gave almost identical results to the refined grids (see Fig. 5.23). Compared to the finest grid, the larger near-wall cell size allows for much larger time steps when using time-accurate explicit schemes. Three different surfaces have been studied, where a recessed region has been created that is located close to the peak heat flux location shown in Fig. 5.26. The recession was created using a sine wave of different depths. The width of the sine wave was set to be 15 degrees of the cylinder arc, with the minimum depth centred at -18 degrees. The depth of the sine wave was set to be $0.025R$, $0.05R$ and $0.1R$, where R is the cylinder

radius. The temperature field from the previous simulation and the three surfaces are shown Fig. 6.13.

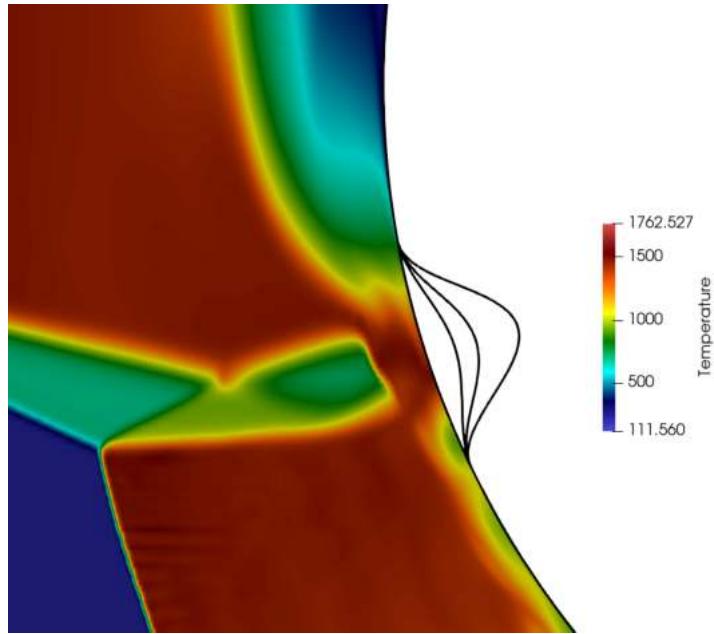


Figure 6.13: The temperature field in the baseline, smooth cylinder case and the three recessed surfaces, with recession depths of $0.025R$, $0.05R$ and $0.1R$

The previous quasi-steady strand/CAMR simulation from Section 5.5 was restarted and the automated mesh motion was used to recess the surface. This demonstrated the robustness of the new mesh deformation algorithms, which were found to be stable even when shock structures crossed the overset boundary. The resulting strand/CAMR meshes close to the recessed regions are shown in Fig. 6.14. Four levels of AMR were used to efficiently resolve the dynamic flow features and an example of the AMR is shown in Fig. 6.15. It can be seen that the shock structures and a pressure disturbance travelling through the subsonic region above the shock interaction is captured by the AMR.

For each surface the simulation was run for 0.5 ms using implicit time-stepping. This allowed the transient features caused by the initial surface recession to reduce and a new quasi-steady state to become established. Each surface was then run for a further period of time using the second-order-accurate RK-SSP2 time-stepping method to give time-accurate results. The results presented below are from this period.

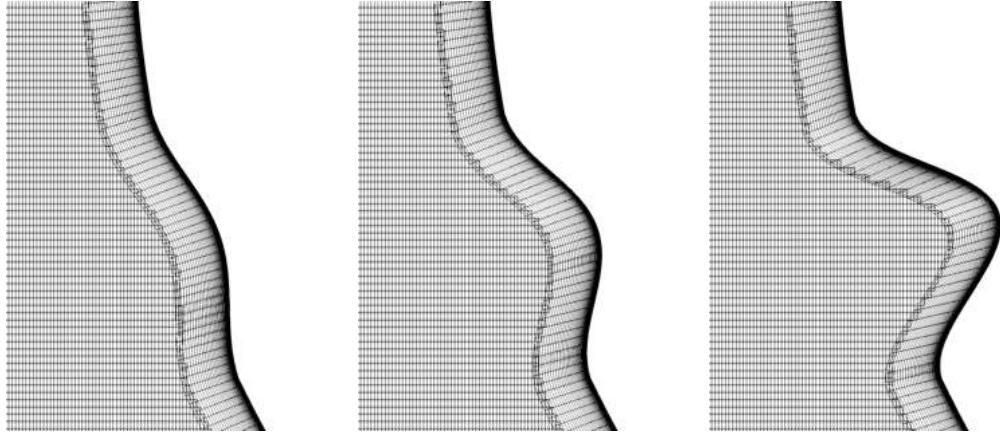


Figure 6.14: The strand/CAMR mesh close to the recessed regions in the shock interaction test case. The automated surface deformation was used to move between these surfaces without re-initialising the flow field.



Figure 6.15: An example of the stand/CAMR mesh in the $0.1R$ case

6.2.2 Results and Discussion

6.2.2.1 $0.025R$ Geometry

For the $0.025R$ case small scale oscillations in the flow are observed and are shown in Fig. 6.16. In Fig. 6.16a the jet impinges at the lower edge of the recessed region. The mass flow into the curved, recessed region leads to the development of a small vortex above the supersonic jet (Fig. 6.16a and Fig. 6.16b), as well as an outward growth in

the separation region above the recessed area (Fig. 6.16c). These features move the upper bow shock away from the surface of the cylinder and the supersonic jet downwards by a small amount (Fig. 6.16d). As the supersonic jet is no longer impinging on the curved region of the cylinder, the separation region reduces in size (Fig. 6.16e). This results in the bow shock moving back towards the body and the jet moving upwards, back towards the recessed region. The cycle then repeats.

A pressure-time map for the $0.025R$ case is shown in Fig. 6.17. In the pressure-time map the oscillations can clearly be seen as cyclical peaks in the pressure below the recessed region. The oscillations are centred around approximately -27 degrees, compared with -17 degrees for the baseline case. Spectral analysis showed that the dominant frequency of the oscillations is approximately $f = 25,000$ Hz, giving a Strouhal number $Str = fD/U_1$ of 1.12.

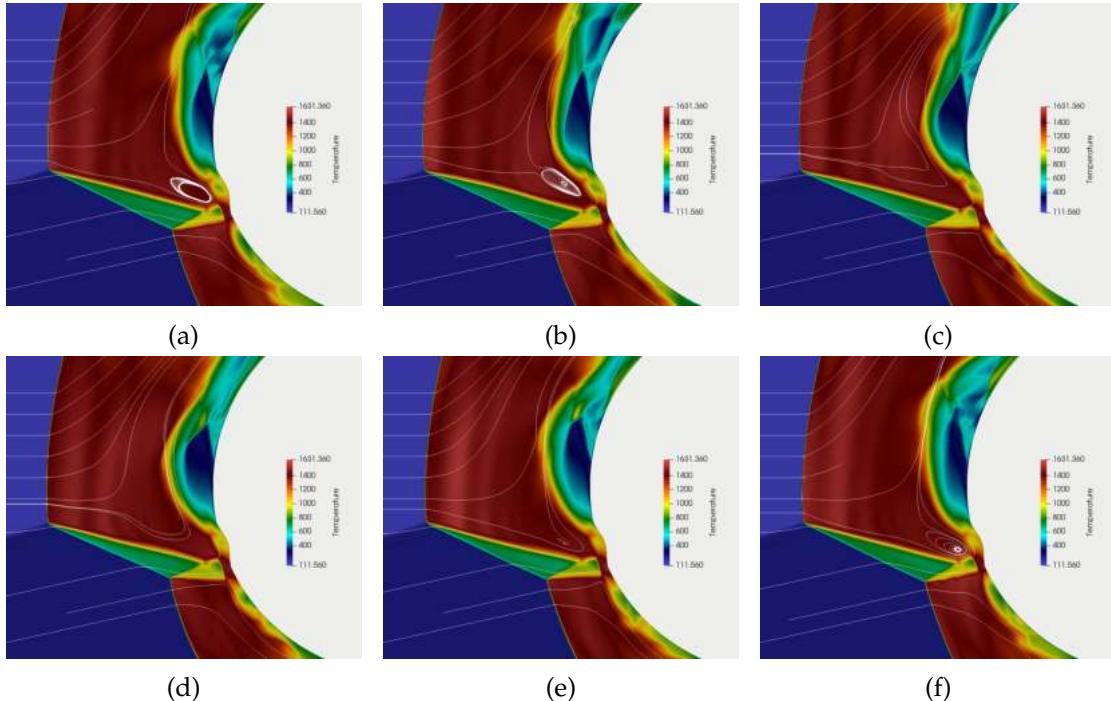


Figure 6.16: The temperature field and streamlines for one cycle of the oscillations seen in the $0.025R$ recession case. The time between the images is $8 \mu\text{s}$.

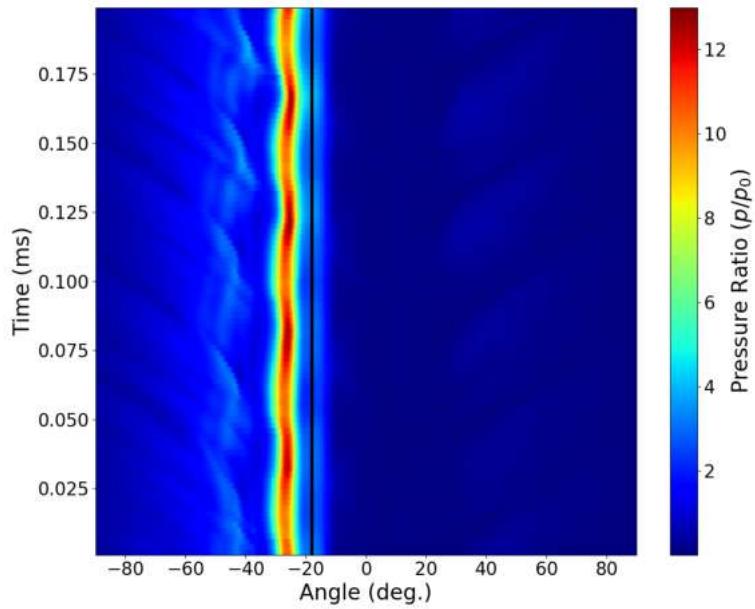


Figure 6.17: The pressure-time map for the $0.025R$ case. The black line indicates the centre of the recessed region.

6.2.2.2 $0.05R$ Geometry

For the $0.05R$ case, much larger, lower frequency oscillations are seen in the flow field. The mechanism driving the oscillations is very similar to that described above for the $0.025R$ case. However, the changes in the flow features are much more pronounced and the frequency of the oscillations is considerably lower.

Figure 6.18 shows one cycle of the unsteady flow. The greater curvature leads to a far greater outward growth in the separation region above the recessed area (Figs. 6.18b to 6.18d). In turn this causes a much greater rise in pressure in the subsonic region and the development of a larger vortex between the separation region and the supersonic jet. As a result, the motion of the bow shock away from the surface is much more pronounced. The larger changes behind the upper bow shock exaggerate the change in shape and direction of the supersonic jet, where the jet is moved much further downwards, away from the recessed region. This change in jet location leads to a collapse in the separation region and a reduction in the pressure in the upper subsonic region (Figs. 6.18e to 6.18g). As a result the bow shock moves back towards the surface and the supersonic jet towards the recessed region (Figs. 6.18h to 6.18i). At this point the cycle then repeats.

A pressure-time map for the $0.05R$ case is shown in Fig. 6.19. The motion in the pressure peak is much greater than in the $0.025R$ case varying between approximately -24 and -34 degrees. The pressure peak is also notably higher, giving a peak value of $p/p_0 > 16$ compared with approximately $p/p_0 \approx 12$ in the $0.025R$ case. Another point to note is that the time scale is much longer. This simulation was run for a total of

0.6 ms in order to capture two oscillations in the unsteady cycle. Spectral analysis gave a dominant frequency of approximately $f = 4,500$ Hz, giving a Strouhal number of 0.20. The differences in results in the above two cases show that the depth and curvature of the recessed region can have a large impact on the flow features.

The flow pattern observed in this case is very similar to the “low-frequency up-down oscillation” described in Ref. [297]. This flow pattern resulted from the impingement of the jet on the lower edge of the cavity leading to a pressure rise above the supersonic jet. In turn, this caused the bow shock to move forward and the shock impingement to move downwards in a similar manner to that observed here. These experimental results give confidence that the simulated results obtained for this case are physically realisable.

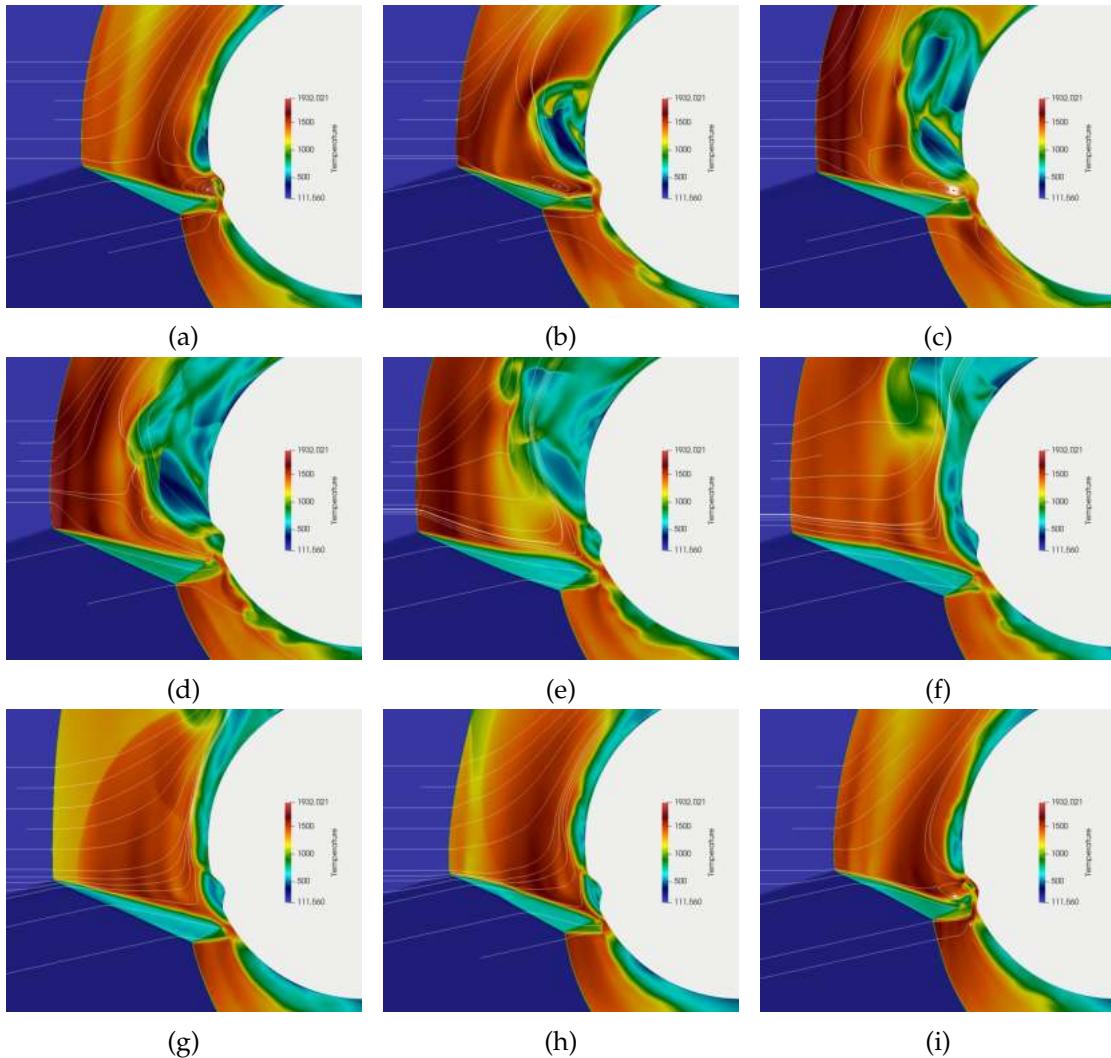


Figure 6.18: The temperature field and streamlines for one cycle of the large scale oscillations seen in the $0.05R$ recession case. The time between the images is $30 \mu\text{s}$.

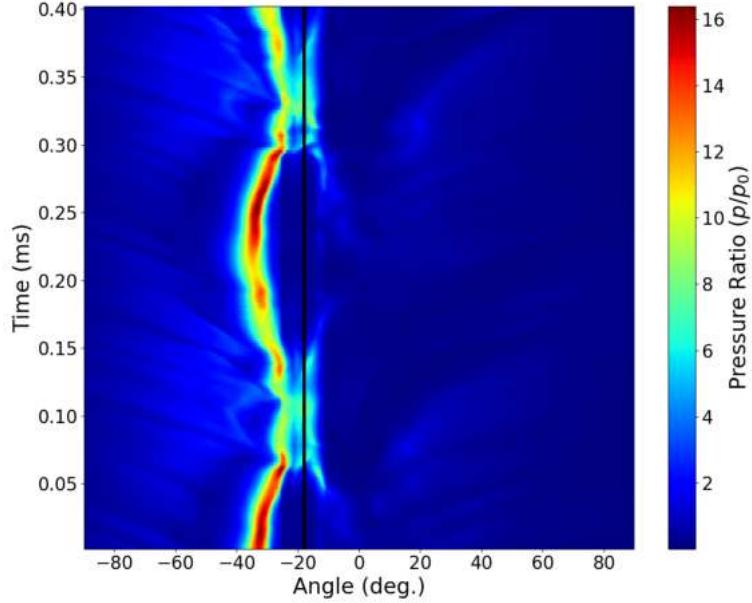


Figure 6.19: The pressure-time map for the $0.05R$ case. The black line indicates the centre of the recessed region.

6.2.2.3 $0.1R$ Geometry

In the $0.1R$ case a different flow structure is observed. One cycle of the oscillating flow structure is shown in Fig. 6.20. In contrast to the above two cases, the jet impinges on the upper edge of the recessed area. The jet moves into the recessed area, leading to a pressure increase in the recessed region and a small growth in the separated region above (Figs. 6.20b and 6.20c). The pressure rise in the recessed region forces the jet out of the recessed region and causes it to curve upwards, resulting in an Edney Type IVa interaction. At the same time the pressure disturbances propagate through both subsonic regions causing both bow shocks (above and below the shock interaction) to move outwards by a small amount. The upward curvature of the jet allows the pressure in the recessed region to reduce. The jet then moves back towards the recessed region, and the bow shocks move towards the surface, starting the cycle again.

The pressure-time map in Fig. 6.21 illustrates this process. It clearly shows the jet impingement on the upper edge of the recessed region and the pressure rise within the recessed region. The dominant frequency is found to be approximately 20,000 Hz giving a Strouhal number of 0.90. This is in-between the first two cases.

These simulation results are in excellent agreement with the “high-frequency backward-forward” oscillations seen in the experimental and numerical results in Ref. [297]. The same mechanism is described with the jet curving upwards and both the upper and lower shocks moving forwards and backwards. Again, this gives confidence that the mechanism observed in these simulations is physically realisable.

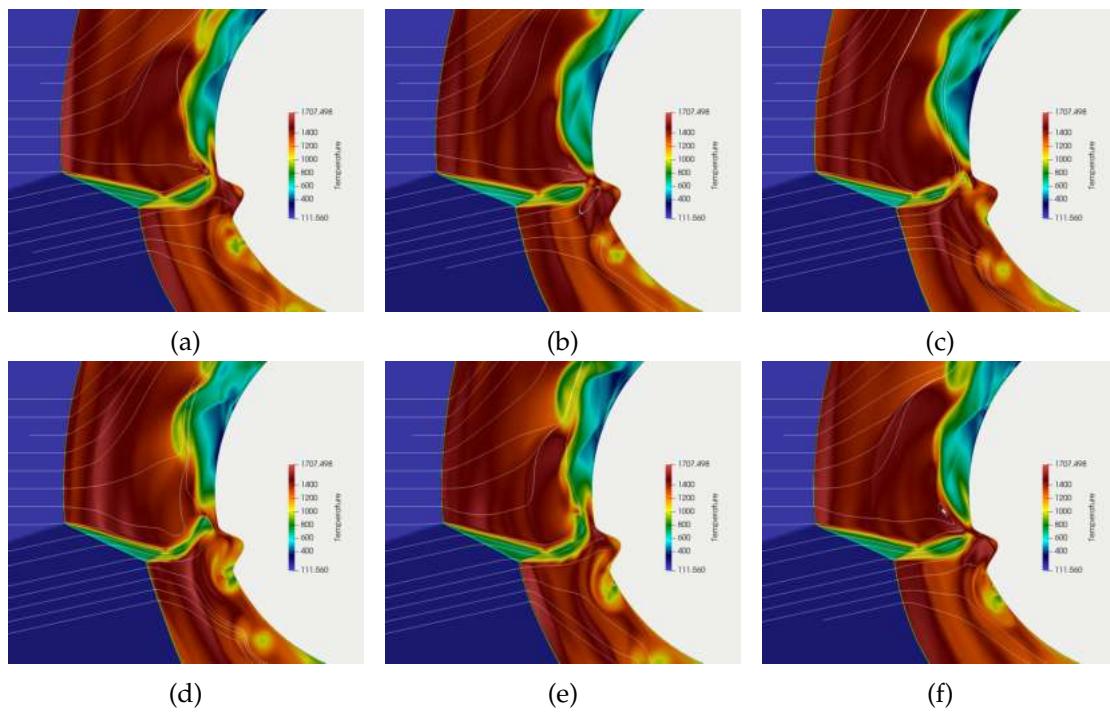


Figure 6.20: The temperature field and streamlines for one cycle of the oscillations seen in the $0.1R$ recession case. The time between the images is $10 \mu\text{s}$

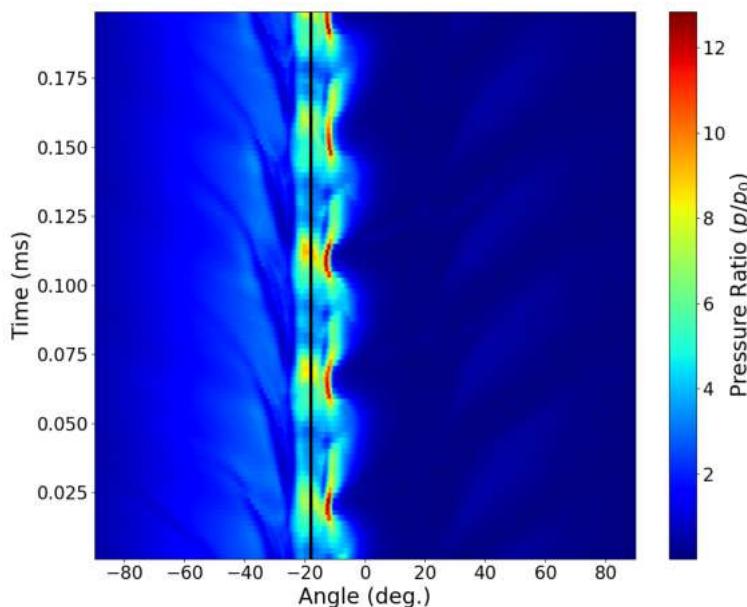


Figure 6.21: The pressure-time map for the $0.1R$ case. The black line indicates the centre of the recessed region.

6.2.2.4 Discussion

The above results show that the introduction of a recessed region to a Type IV interaction can have a profound effect on the flow structures. In all cases here and in Ref. [297] the location of the shock impingement moved relative to the baseline case and unsteady interactions with the cavity were seen. The introduction of a cavity appears to encourage unsteady, oscillating flows, which vary in amplitude and frequency depending on the cavity shape and shock impingement location.

The $0.05R$ case resulted in large-amplitude, low-frequency upward-downward oscillations in the shock structure, whilst the $0.1R$ case led to low-amplitude, high-frequency motion. Both of these modes were previously observed in the experimental investigations in Ref. [297], where a cavity of a different size and shape to this study was used. In the experiment the change between the two oscillating modes was obtained by moving the location of the shock interaction. When the shock impingement was positioned high on the cavity the low-amplitude, high-frequency, backward-forward mode was observed. When the shock impingement was lowered the large-amplitude, low-frequency, upward-downward mode was observed. In contrast, the change in mode in these simulation was caused by changes in the recession depth and, therefore, the curvature of the recessed region. The width and location of the recessed region were constant for all simulations. In addition, a new low-amplitude, high-frequency upward-downward mode was observed in this work in the $0.025R$ case. The $0.025R$ case had similar flow features to the $0.05R$ case, but they were of much lower intensity and higher frequency. This appears to be due to the lesser curvature of the recessed region in the $0.025R$ case.

The results from this study and from Ref. [297] indicate that the interaction of Type IV shock pattern and a cavity can lead to two types of oscillating flow. The first is where the jet moves from the lower edge of the cavity to below the cavity. The motion of the jet is accompanied by a forward-backward motion of the upper bow shock. This mode could be described as a Type IV \leftrightarrow (III) mode, as the interaction moves towards a Edney Type III interaction, but does not reach it. The above results show that the shape of the recessed region influences the amplitude and frequency of the oscillations in this mode.

A second mode is seen when the jet impinges on the upper side of the cavity. In this case, the pressure rise in the cavity causes the jet to curves upwards. In addition, there is a forward-backward motion in both the upper and lower bow shocks. This mode could be described as a Type IV \leftrightarrow IVa mode. High-frequency, low-amplitude oscillations were observed for this mode, both in this work and in Ref. [297]. However, this does not preclude the existence of a low-frequency, high-amplitude variation of this mode. In the Type IV \leftrightarrow (III) mode, the shape of the recessed region had a clear

influence on frequency and magnitude of the oscillations. Therefore, a different shape may lead to different behaviour of the Type IV \leftrightarrow IVa mode as well.

The ablation of a realistic TPS is likely to lead to different surface recession geometries to the idealised ones studied here. Consequently, coupled fluid-ablative-material simulations should be conducted to better understand the interaction between an ablative TPS and a Type IV shock structure. The unique capabilities of the solver developed in this work provide a foundation that can be built on to study this complex, multi-physics problem.

6.2.3 Summary

Three different geometries have been simulated to explore the effect that surface recession has on a Type IV shock interaction. Three distinct oscillating flow patterns were observed for the three geometries as the jet interacted with the recessed region. These flow patterns, along with those observed in Ref. [297] appear to have two basic modes, namely a Type IV \leftrightarrow (III) mode and a Type IV \leftrightarrow IVa modes. The mode, frequency and amplitude were shown to be strongly influenced by the shape of the recessed region. The flow conditions and recession location were the same for all geometries and only the recession depth changed, leading to the observed variation in the flows. Further study of more realistic recession shapes should be conducted using a fully coupled solver to better understand the consequences of a Type IV interaction impinging on an ablative TPS.

6.3 Influence of Nose Bluntness on Unsteady Double-Cone Flows

The recessing nose-tip simulation in Section 5.6.2 demonstrated how certain double-cone geometries can create dynamic shock structures. There has been relatively little investigation of this phenomena since the experiments of Abbott *et al.* [1] in the 1970's. Since then, most studies have focussed on the unsteadiness seen in spiked conical geometries, rather than double-cone geometries. Recently, however, there has been a renewed interest in unsteady double-cone flows with experiments being conducted by Sasidharan and Duvvuri [238] and a numerical study by Hornung *et al.* [125]. Both studies investigated how non-dimensional geometric parameters influenced the unsteadiness of the flow and both found that the double-cone flow could be steady or unsteady depending on the geometry used. These studies, and all other previous studies that the author is aware of (apart from those of Abbott *et al.* [1]) have used sharp conical geometries.

Abbott *et al.* found that the tip radius had "some influence on the stability of the flow" when investigating unsteady double-cone flows [1]. In addition, blunt noses are more realistic as sharp conical geometries experience high heat fluxes in hypersonic flows. Consequently, sharp noses are likely to ablate to become blunted or will be blunted by design to reduce the heat flux. The influence of nose bluntness on the unsteadiness has not been numerically studied previously and has only been studied experimentally by Abbott *et al.* [1]. Thus, the impact of nose bluntness on the unsteadiness of the flow needs to be better understood.

The aim of this study is to investigate how nose bluntness influences the unsteadiness of the double-cone flow. The new solver is well suited to this test case, as the AMR enables efficient resolution of the dynamic shock structures, whilst the strand mesh accurately captures the boundary layer development. In addition, the newly implemented overset RK-SSP2 time-integration method can be used to give overall second-order-accurate simulations in space and time.

6.3.1 Double-Cone Flows

The nature of the double-cone flow is determined by the interaction of the shocks from the two cones and the separation region. The shock structure and the resulting flow depend on the flow conditions and the double-cone geometry. In the previous double-cone studies discussed above, the geometry has been parameterised in different ways, leading to different non-dimensional parameters being assessed. In this work the parameterisation of Abbott *et al.* [1] is used, where the parameters are

shown in Fig. 6.22¹. This parameterisation was selected as it already incorporates nose bluntness, whilst being similar in some aspects to the parameters used by Sasidharan and Duvvuri [238].

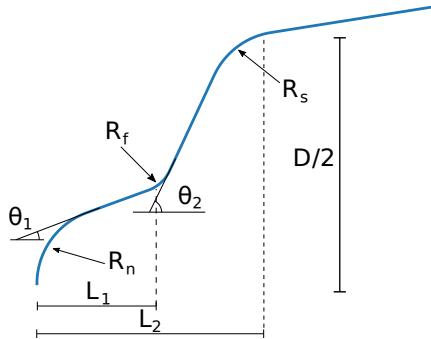


Figure 6.22: The parameters used to define the ablated nose-tip shape.

For a typical double-cone geometry a shock develops on both the fore-cone and the rear-cone. The interaction of these shocks depends on the length of the fore-cone, and the two cone angles. For small L_1/D , and large rear-cone angle, θ_2 , the fore-cone can sit entirely within the subsonic region of the detached shock from the rear-cone. This results in a trivial steady flow structure.

When L_1 is larger than the shock stand-off distance of the rear-cone shock, the oblique shock from the fore-cone interacts with the rear-cone shock. For low values of θ_1 this results in a triple point where they intersect, and a transmitted shock that impinges on the surface. The shock impingement creates an adverse pressure gradient and a separated region, resulting in a separation shock that can interact with the oblique shock from the fore-cone. The result of the shock interactions is an Edney Type IV flow pattern, where a supersonic jet (a.k.a. shock train) is bounded by the subsonic flow behind the rear-cone shock above and the subsonic flow in the separation region below. This is shown in Fig. 6.23. For sharp cones, it has been shown that the resulting flow can either be steady or unsteady, depending on the cone angles and ratio of fore-cone to rear-cone length.

Sasidharan and Duvvuri [238] and Abbott *et al.* [1] used two classifications to distinguish between the different types of unsteady flow. In both studies a label of “pulsating” was used for one unsteady mechanism. Pulsating flows occur when vorticity accumulates in the separation region [125]. This causes the separation region to grow significantly, changing the shape of the separation shock and moving the triple point upwards towards the rear-cone shoulder. The separation region grows until the supersonic jet passes over the shoulder of the rear cone. At this point the mass that has accumulated in the separation region is able to escape. As a result, the shock structure collapses and the cycle begins again [125, 238].

¹In this case the corner and shoulder radii are set to be zero as it was found that they had no discernible impact on the flow [1].

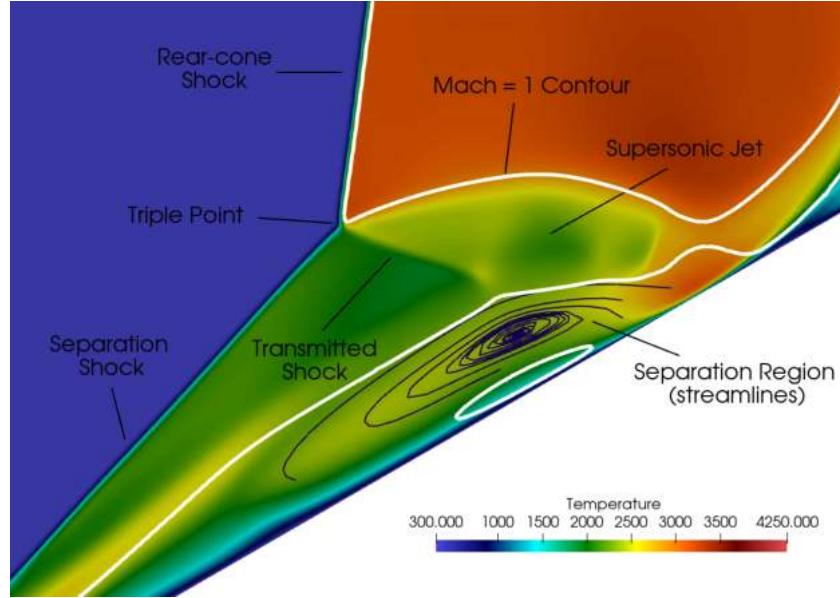


Figure 6.23: An example of the Type IV shock interaction commonly seen in double-cone flows.

The second unsteady category was labelled “oscillating” by Sasidharan and Duvvuri [238]. The oscillating pattern was described as a small amplitude, high frequency motion of the separation shock. The motion was attributed to instabilities in the shear layer between the separation region and the supersonic flow above the separation region. Abbott *et al.* [1] had similar category labelled “steady separated”. Contrary to the “steady” name given to this flow pattern, the description appears to be similar to the oscillating pattern described by Sasidharan and Duvvuri [238]. The authors describe the steady separated pattern as having a large separated region bounded by a shear layer that reattaches close to the shoulder. In addition, example pressure readings for steady separated flows show oscillations, while flow schematics show some motion in the shock structure.

Hornung *et al.* [125] found that the double-cone flow was steady when the rear-cone angle was less than the detachment angle, i.e. $\theta_2 < \theta_{2d}$. In this scenario, the supersonic jet extends beyond the rear-cone shoulder and the mass in the separated region is able to escape. Therefore, the unsteady growth and collapse of the separation region does not occur. It has also been shown that the fore-cone angle influences the unsteadiness. A larger angle results in a stronger shock and higher post-shock pressure. This changes the strength of the interaction and lowers the adverse pressure gradient created by the shock impingement, inhibiting the growth of the separated region. Thus, above a certain value of θ_1 the flow becomes steady.

A recent study into the influence of nose bluntness on steady double-cone flows showed that increasing bluntness leads to an over-expansion behind the nose, reducing the pressure behind the fore-cone shock [114]. In addition, the shear stress in

the region of attached flow behind the nose is reduced compared to a sharp cone. For a steady flow this led to differences in the size of the separation region. As the unsteadiness is closely linked to the behaviour of the separation region it is likely that nose bluntness will have an impact on the unsteadiness of double-cone flows.

6.3.2 Simulation Set-Up

6.3.2.1 Flow Conditions and Geometries

An unsteady, sharp nose, case from Ref. [125] was used as the baseline for this study. The A' flow conditions were used (see Table 6.3) and cone angles $\theta_1 = 30$ deg. and $\theta_2 = 70$ deg. were selected. The hypotenuse of each cone was set to be equal at a length of 0.05 m. This set-up was found to give an unsteady flow in Ref. [125]. In addition, a small bluntness of $R_n = 100 \mu\text{m}$ was given to the nose as a plausible manufacturing tolerance.

T_∞	ρ_∞	U_∞
300 K	2.25 g/m ³	2720 m/s

Table 6.3: Freestream conditions for the ideal gas double-cone simulations.

The blunt nose radii were selected based on an idealised ablation of the sharp nose geometry. In the ablated geometry the fore-cone is recessed, reducing the length of L_1 , and then joined to the axis by a circular section. Three different ratios of R_n/L_1 were chosen: $R_{n,1}/L_1 = 0.2$, $R_{n,2}/L_1 = 0.5$ and $R_{n,3}/L_1 = 1.0$. In addition to the “ablated” double-cone geometries, three geometries with a constant L_1 were created, with the same nose radii. This is so that the effects of nose bluntness can be separated from the effects of reducing L_1 . Full details of the geometries, including the sharp cone, are given in Table 6.4. It can be seen that the two classes of blunted geometries either have a constant base diameter D (ablated shapes) or a constant fore-cone length L_1 . All of the geometries are shown in Fig. 6.24.

Name	R_n (mm)	$D/2$ (mm)	L_1 (mm)	θ_1 (deg.)	θ_2 (deg.)	R_n/L_1
$R_{n,0}$	0.010	71.98	43.30	30	70	0.00231
$R_{n,1}$ const. D	7.22	71.98	36.08	30	70	0.2
$R_{n,2}$ const. D	14.43	71.98	28.87	30	70	0.5
$R_{n,3}$ const. D	21.65	71.98	21.65	30	70	1.0
$R_{n,1}$ const. L_1	7.22	76.15	43.30	30	70	1/6
$R_{n,2}$ const. L_1	14.43	80.32	43.30	30	70	1/3
$R_{n,3}$ const. L_1	21.65	84.48	43.30	30	70	0.5

Table 6.4: The parameters used to define the blunted double-cones.

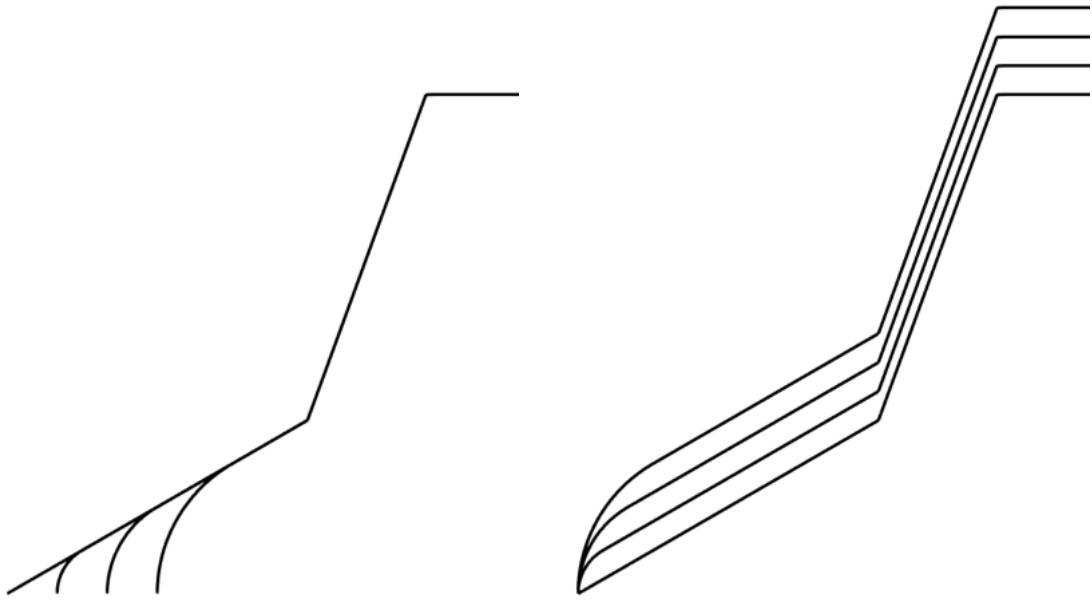


Figure 6.24: The sharp cone surface shown alongside the constant D (ablated) geometries (left) and constant L_1 geometries (right). Images are to scale.

6.3.2.2 Computational Mesh

A grid convergence study was carried out to ensure that the unsteady mechanisms were being accurately captured. In Ref. [125] it was shown that relative coarse grids compared to those required for heat flux predictions could accurately capture the unsteady flow patterns. Three grids were created for the grid refinement study, all of which used five levels of refinement and a lower near-wall spacing than those used in Ref. [125]. The details of the different grids are given in Table 6.5. The grid convergence study was carried out using the $R_{n,1}$ and $R_{n,2}$ constant D geometries. The RK-SSP2 time-integration method was used, with a constant CFL number of 0.7.

Name	Background Coarse Size	Near-body Size	Near-wall spacing
Coarse	50×75	330×30	$20 \mu\text{m}$
Mid	75×100	470×40	$15 \mu\text{m}$
Fine	100×150	600×60	$10 \mu\text{m}$

Table 6.5: The computational meshes used in the mesh refinement study.

In this study the unsteadiness of the flow is assessed in a number of ways: the dynamic flow features are qualitatively compared using Schlieren and streamline plots, the pressure-time maps from each surface are examined, the axial forces are calculated and the frequency of the oscillations/pulsations are assessed using spectral analysis. For all of the above cases it was shown that the Mid and Fine grids gave results that were in excellent agreement after start-up periods of slightly different

lengths. For example, the Mid and Fine pressure-time maps for $R_{n,2}$ are shown in Fig. 6.25. Plots of the axial force coefficient against time and the power spectrum plots for all three grids are shown in Fig. 6.26. The pressure and force results have been shifted in time due to minor differences in the start-up period, but it can be seen that the Mid and Fine grids give extremely similar results once the pulsating flow pattern is established. A similar set of results were obtained for the $R_{n,1}$ constant D case.

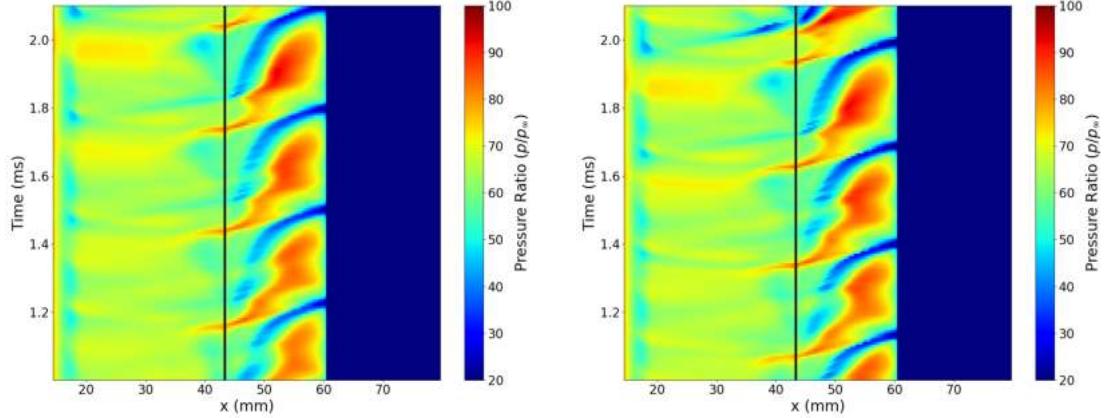


Figure 6.25: The pressure-time maps for the Mid (left) and Fine (right) grids in the grid convergence study for the $R_{n,2}$ constant D geometry.

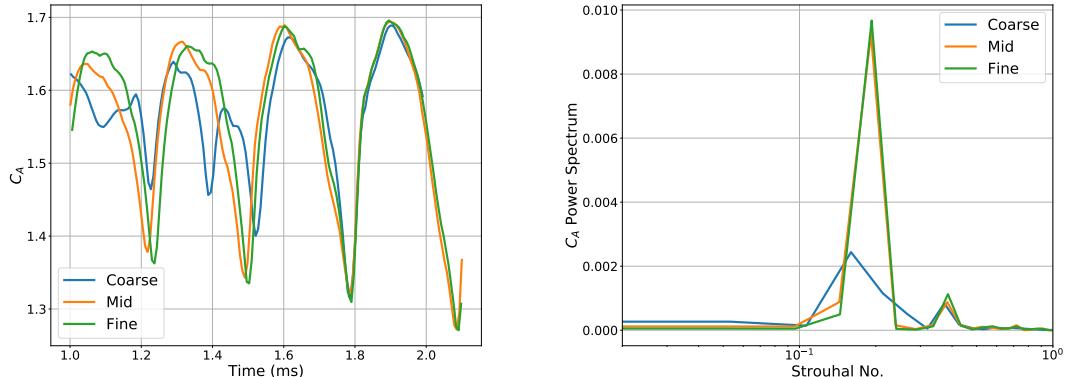


Figure 6.26: The inviscid axial force coefficient plotted against time (left) and power spectrum (right) of the axial force coefficient in the grid convergence study.

Considering the above results, the Mid grid was used for all of the following simulations. Even when using the Mid grid a high temporal resolution is obtained, where the time-steps were generally less than 5 ns when using a CFL number of 0.7. In addition, the spatial resolution is high, especially in the off-body region where five levels of grid refinement were used to capture the dynamic shock structures. As a consequence, each simulation required approximately 5,000 CPU hours and were run on either 40 or 80 processors.

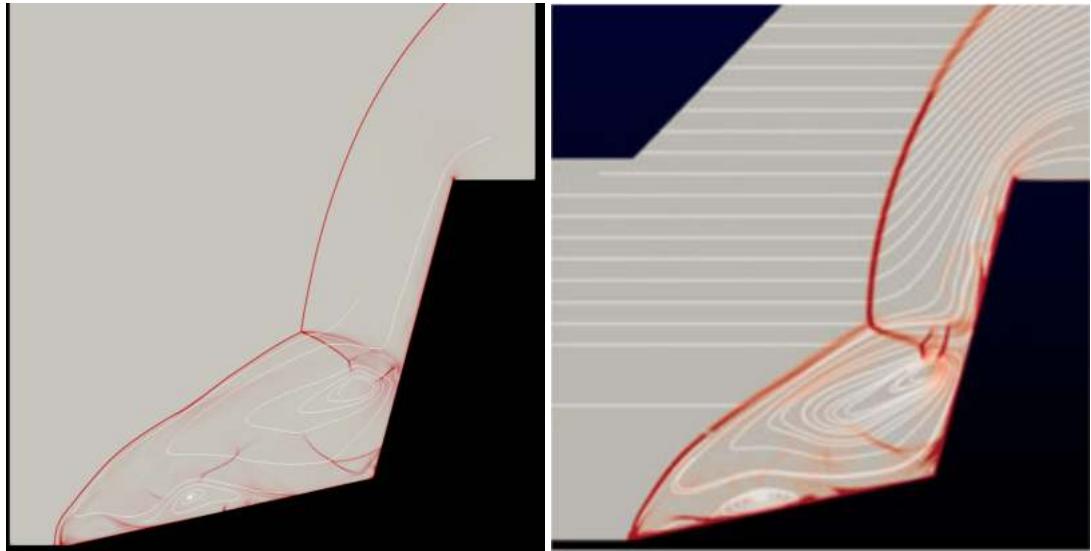


Figure 6.27: A comparison of the numerical Schlieren images from the AMROC simulation (left) and from Ref. [125] (right). (Image reproduced inline with copyright license.)

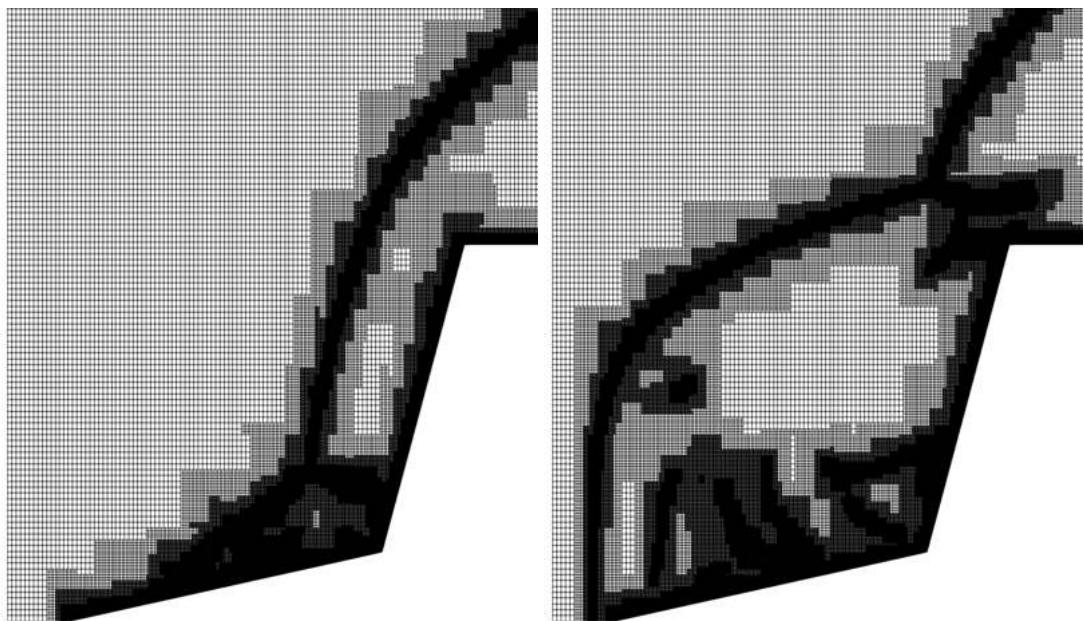


Figure 6.28: The AMR mesh at two different times in the pulsation cycle.

Following the grid convergence study, qualitative comparisons with the results from Ref. [125] were made. A numerical Schlieren image from AMROC is shown in Fig. 6.27, alongside one from Ref. [125]. It can be seen that the shock structures and streamlines in the two images are in very good agreement. It is notable that the AMR in the new solver enables significantly better resolution to be obtained. In the AMROC image the off-body shocks and shock interactions are far better defined. A depiction of the AMR grid at two different times in the pulsation cycle is shown in Fig. 6.28.

6.3.2.3 Automated Meshing Demonstration

The automated mesh motion algorithms can be further demonstrated using this test case. An example is shown in Fig. 6.29, where the flow field was initialised and the automated mesh motion routines were used to change from the sharp to the $R_{n,2}$ constant D geometry. One can see that the mesh smoothly transitions between the two geometries. Again, minimal user input was required: only the initial and final mesh, and the start time of the motion. This is further evidence of the robustness of the automated mesh motion. It shows that the new technique is able to incorporate large geometry changes that lead to significant differences in the off-body shock structure. The level of automation for the combination of surface deformation and shock capturing is beyond what has previously been shown in the hypersonic literature.

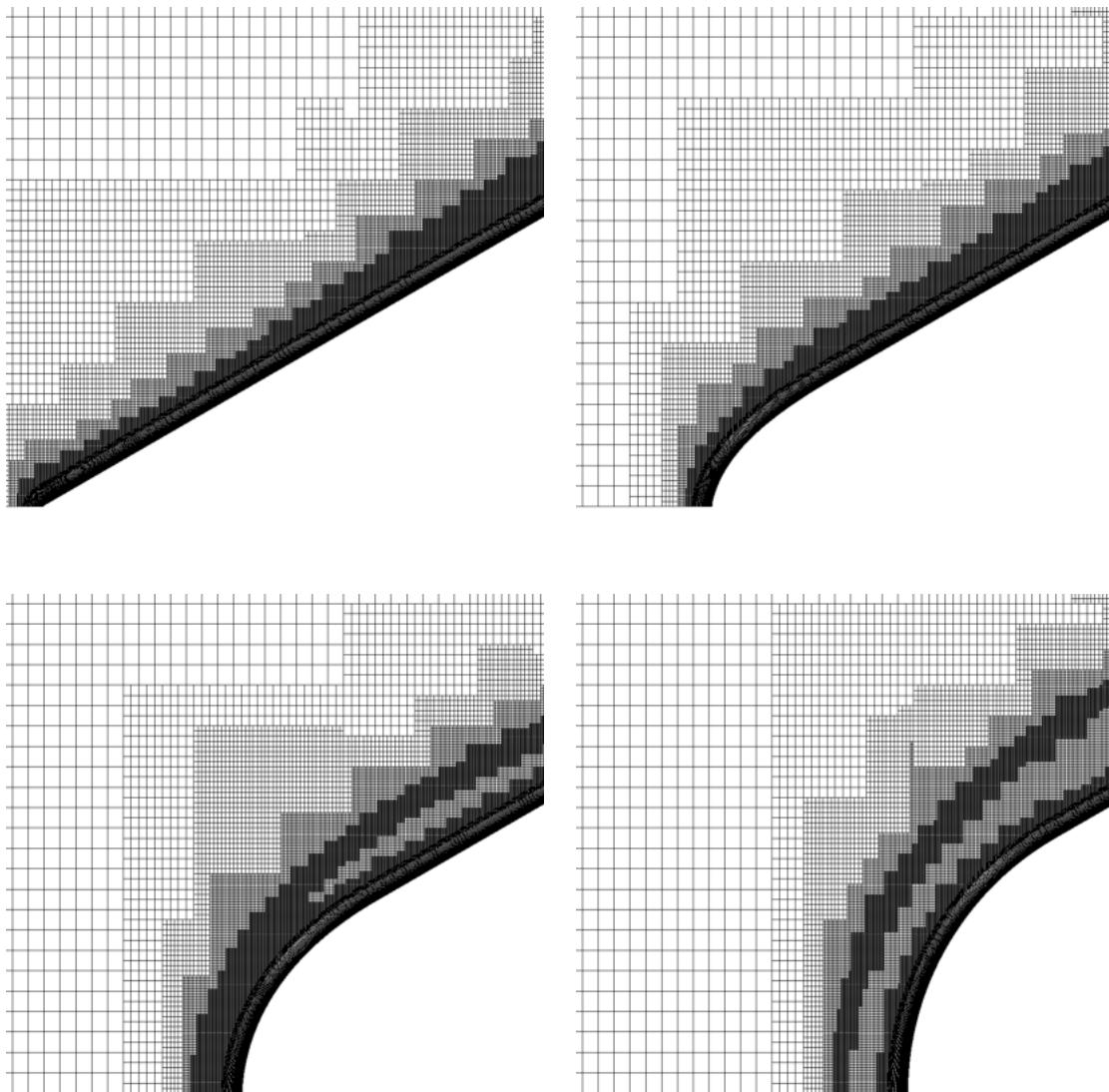


Figure 6.29: The automated mesh motion for an ablating double-cone geometry.

6.3.3 Results and Discussion

In this sub-section the results from the simulations are presented and discussed. First, the sharp cone results are presented, which demonstrate the unsteady flow features previously seen in pulsating double-cone flow investigations. Then, the results from the blunt nose geometries are presented. These are separated according to nose radius and the constant D and constant L_1 results are compared. Finally, the effect of nose bluntness on the flow is discussed in more detail.

6.3.3.1 Sharp Nose

For the sharp nose case, a pulsating flow pattern was established after an initialisation period. One oscillation of the pulsating flow pattern is shown in Fig. 6.30. In Fig. 6.30a, it can be seen that a large separation region is present, where the separation region reattaches approximately half way along the rear cone. A weak Type IV interaction between the separation shock and the rear-cone shock can be seen. Between Figs. 6.30a and 6.30d the size of the separation bubble increases and the Type IV interaction becomes stronger, with the supersonic shock train becoming clearly visible. This process is caused by a feedback loop between the shock interactions and the separation bubble:

1. The adverse pressure gradient at the reattachment point causes the separation region to grow upstream on the fore-cone.
2. The upstream growth of the separation region changes the shape of the separation shock, making it more curved and reducing the separation shock angle close to the triple point.
3. The reduction in the angle of the separation shock reduces the shock strength close to the triple point, lowering the total pressure loss over the separation shock and increasing the Mach number in the supersonic jet.
4. This results in an increase in the pressure at the point the supersonic jet interacts with the rear-cone surface.
5. The increased adverse pressure gradient leads to further upstream growth in the separation region.

The above cycle is broken in Figs. 6.30e and 6.30f, where the growth of the separated region causes the shock train to move above the shoulder of the second cone. As the shock train passes over the shoulder the mass in the separation bubble is able to escape. This leads to a reduction in the size of the separated region and a collapse of the shock structure, as shown in Figs. 6.30g to 6.30i. At this point the process repeats.

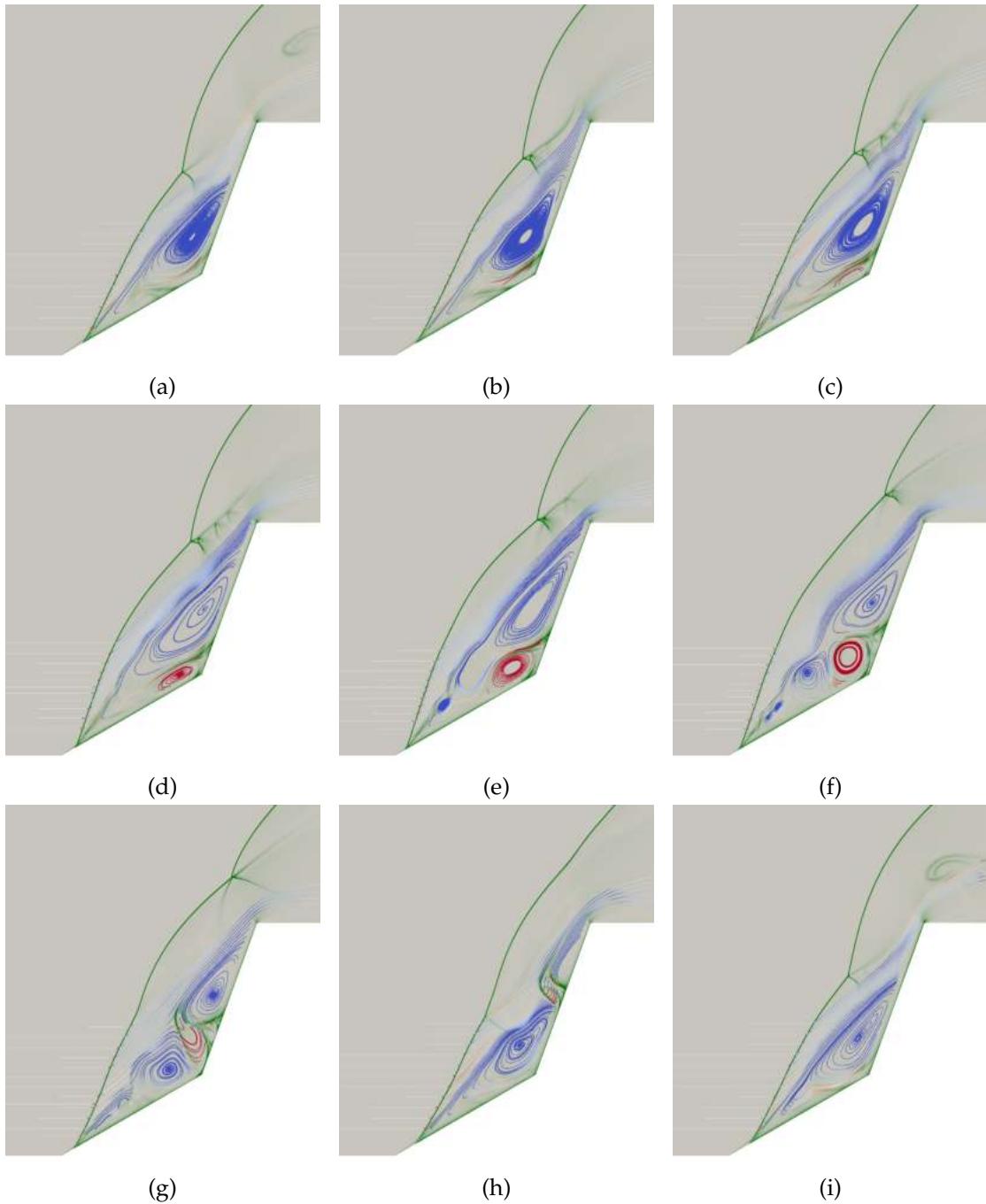


Figure 6.30: The numerical Schlieren and streamlines for approximately one unsteady oscillation for the sharp cone geometry. The streamlines are coloured by vorticity, with red representing positive and blue negative (anti-clockwise and clockwise rotation, respectively).

A pressure time map for the sharp cone is shown in Fig. 6.31. The pressure map clearly shows the motion and increase in strength of the peak pressure location on the rear-cone surface. In addition, the forward growth of the separation region on the fore-cone can clearly be seen.

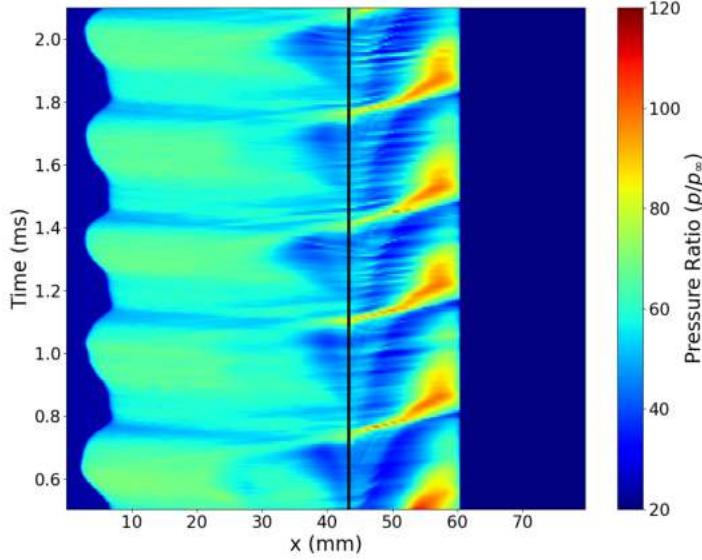


Figure 6.31: The pressure-time map for the sharp cone surface. The black line represents the double-cone intersection.

The axial force acting on the cone is used to quantify the unsteadiness of the flow. The inviscid axial force coefficient, C_A is plotted against the simulation time in Fig. 6.32, along with the Power Spectral Density (PSD) plot for C_A . The value of C_A is calculated using the pressure force acting on the double-cone surface, the free-stream conditions and the double-cone base area, $\pi D^2/4$. For all geometries the same y axis will be used in the C_A plots to aid qualitative comparisons. The C_A PSD is calculated for all geometries using the data between 0.5 and 2.1 ms, giving bin width 625 Hz. The square root of the area under the PSD gives the Root-Mean-Square (RMS) of the C_A power, allowing the power to be compared between the different double-cone geometries. The PSD is plotted against the Strouhal number $Str = \omega L_2/U_\infty$, where ω is the frequency of the force oscillation. For this case, the power spectrum results show that the dominant Strouhal number is 0.0693 (giving a dimensional frequency of approximately 3,200 Hz). The C_A power RMS for this geometry is 0.167.

In addition to the PSD results, the structural effects of the force oscillations can be approximated by considering the system as a driven harmonic oscillator. Assuming a periodic sinusoidal variation in the driving force, the amplitude of the displacement, A , for a driven harmonic oscillator is given by,

$$A = \frac{F_0}{\sqrt{(\omega_0^2 - \omega^2)^2 + b^2\omega^2}}, \quad (6.1)$$

where F_0 is the force amplitude, b is the damping constant, ω_0 is the natural frequency of the system. Taking the worst case scenario, where $\omega = \omega_0$ leads to

$$A = \frac{F_0}{b\omega}. \quad (6.2)$$

Given the above, it is useful to calculate a non-dimensional value of F_0/ω to compare across the different geometries. In this case, it can be approximated by $\Delta C_A/Str$, where ΔC_A is the maximum difference in C_A . This is displayed in the title of the C_A plot and for this case is 8.315.

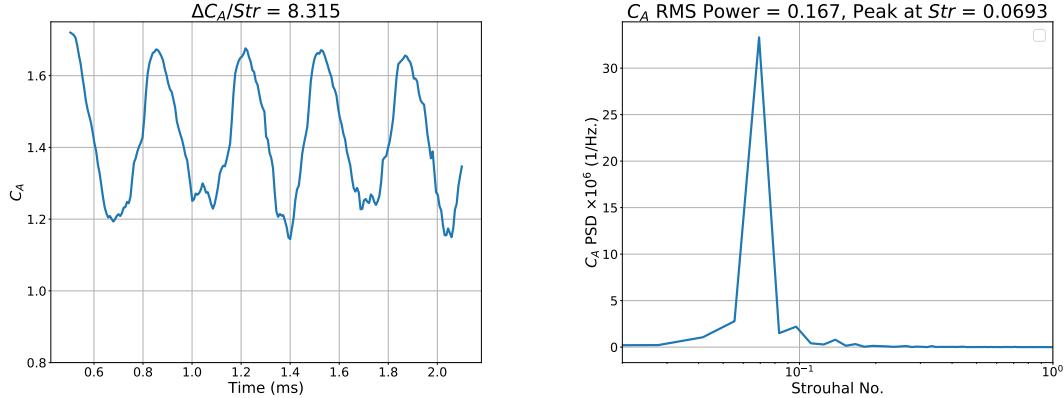


Figure 6.32: The inviscid axial force coefficient plotted against time (left) and PSD (right) of the axial force coefficient for the sharp cone case.

6.3.3.2 Blunt Nose $R_n = R_{n,1}$

For both the constant D and constant L_1 geometries an unsteady, pulsating flow pattern was established. One cycle for the constant D geometry is shown in Fig. 6.33 and one cycle for the constant L_1 is shown in Fig. 6.34. The behaviour between the two geometries is very similar, and different from the sharp cone unsteadiness shown above.

In the first image of each cycle it can be seen that there is a small separation region close to the nose, as well as a separated region on the rear-cone. This is different to the sharp geometry, where the separated region was much larger at the start of each cycle. The small separation region on the fore-cone results in a separation shock, which has a Type IV interaction with the shock from the rear-cone. The supersonic jet impingement falls on the fore-cone, whereas for the sharp cone case the impingement was always on the rear-cone.

In both geometries, the separation region on the fore-cone grows towards the double-wedge intersection in the next two images. At the same time, the separation region on the rear-cone reattaches as the vortex moves over the shoulder. The growth of the separation bubble moves the Type IV interaction towards the corner. The separated region stays relatively thin until the supersonic jet impingement reaches the corner of the double-wedge. After this point the separated region starts to grow rapidly. This is likely caused by the increased pressure gradient due to the interaction

of the supersonic jet with the steeper rear-cone. As the supersonic jet impinges on the rear-cone a strong termination shock creates a region of high pressure.

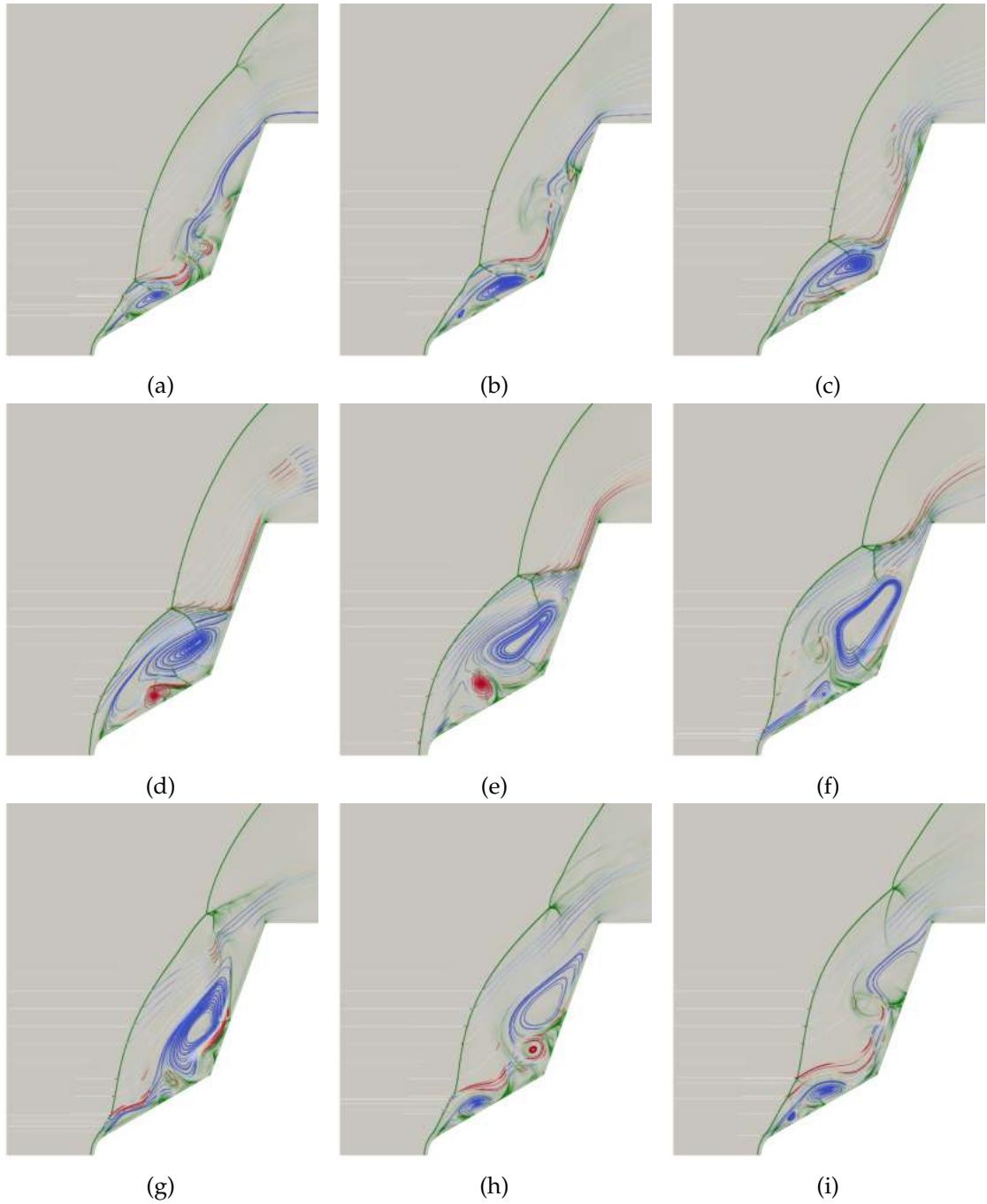


Figure 6.33: The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,1}$, constant D geometry. The streamlines are coloured by vorticity, with red representing positive and blue negative (anti-clockwise and clockwise rotation, respectively).

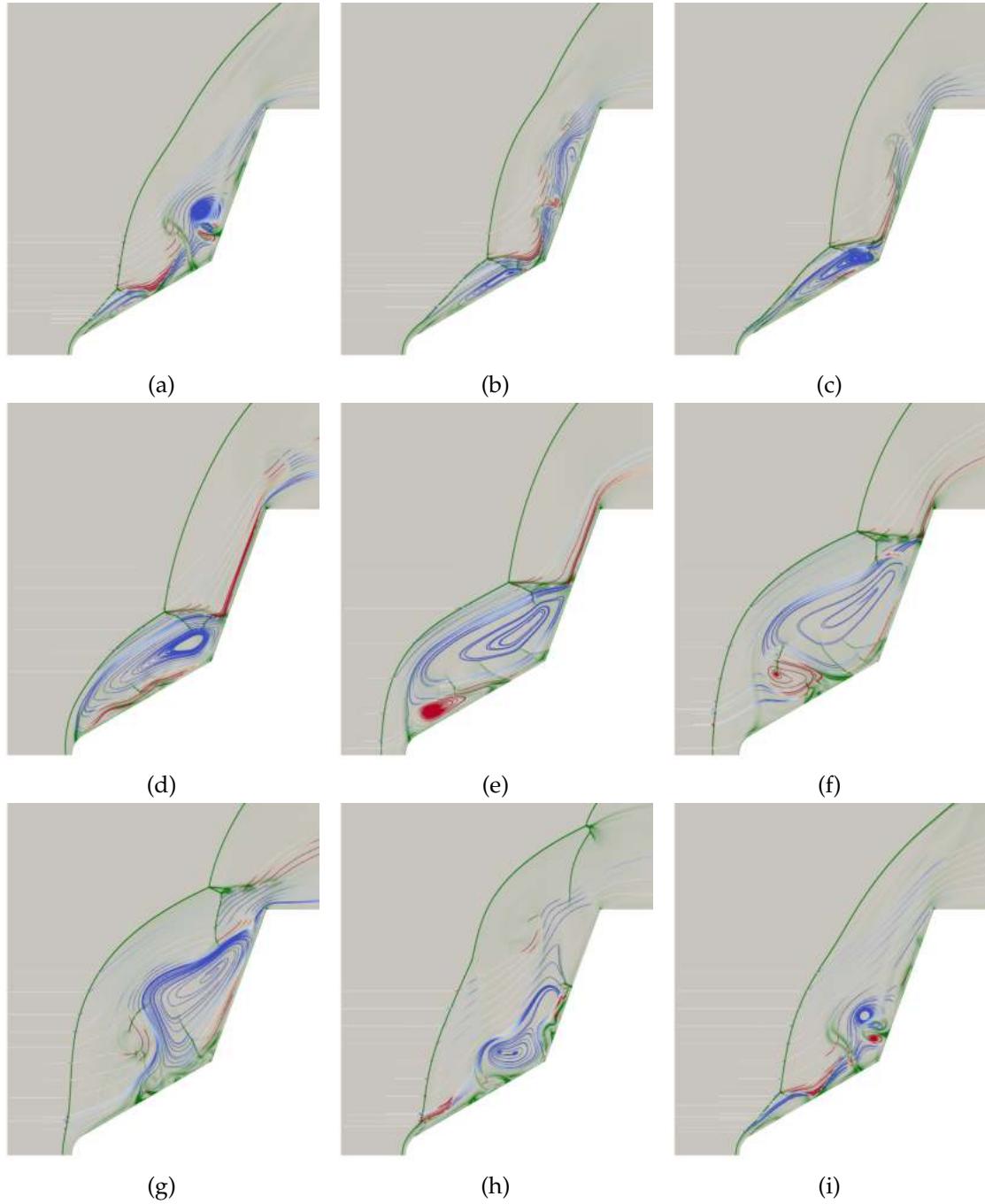


Figure 6.34: The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,1}$ constant L_1 geometry. The streamlines are coloured by vorticity, with red representing positive and blue negative (anti-clockwise and clockwise rotation, respectively).

The feedback loop between the forward growth of the separation region and the strength of the adverse pressure gradient is the same as in the sharp cone case described above. In this case it is visibly clearer and appears stronger: the separation region grows to such an extent that the flow over the whole fore-cone separates and the bow shock extends beyond the nose. This is more pronounced in the constant L_1

case, where there is a substantial forward motion of the shock on the nose, as shown in Fig. 6.34f. This is distinct from the sharp cone case where the front section of the fore-cone remains attached throughout. During the growth of the separated region the shock train moves upwards and becomes more clearly defined in both cases, with a termination shock becoming visible close to where it impinges on the surface. Eventually the shock train moves over the shoulder of the rear-cone and the mass in the separated region is able to escape. The shock structure then collapses and the pulsation cycle restarts.

The pressure-time maps for both surfaces are shown in Figs. 6.35. The impingement of the Type IV supersonic jet can clearly be seen as a pressure peak on the fore-cone of both geometries. In addition, the motion of the separation region beyond the nose can be seen as an increase in pressure at the nose. Neither of these features are observed in the sharp cone pressure-time map (see Fig. 6.31). The axial force plots and PSD plots for both surfaces are shown in Figs. 6.36 and 6.37, respectively. It can be seen that the constant L_1 geometry gives a much greater variation in the axial force than the constant D case and the sharp cone case. The Strouhal number is smaller for the constant L_1 geometry compared to the constant D geometry, but both Strouhal numbers are greater than for the sharp cone case.

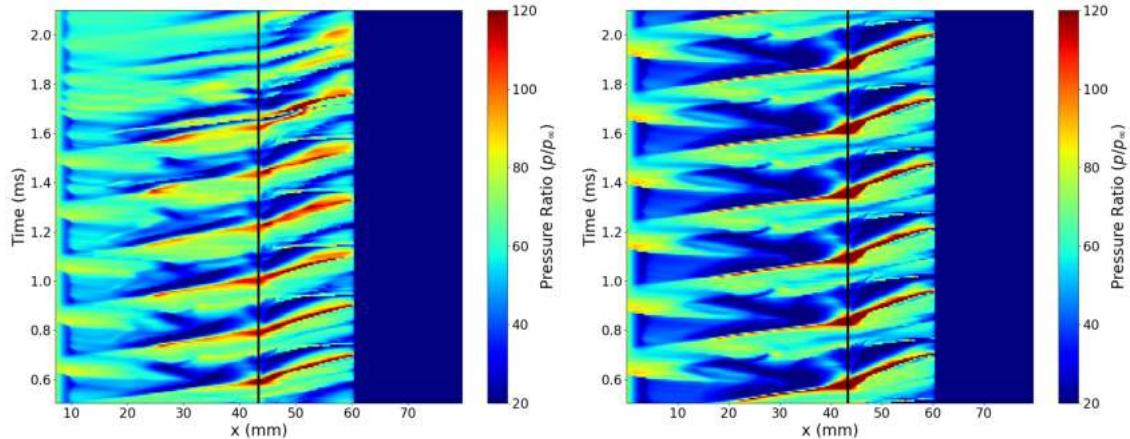


Figure 6.35: The pressure-time map for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,1}$.

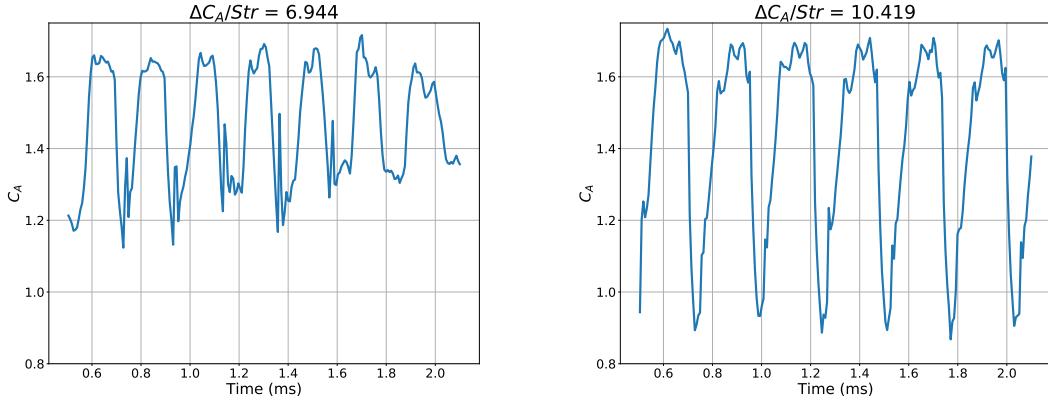


Figure 6.36: The axial force coefficient plotted against time for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,1}$.

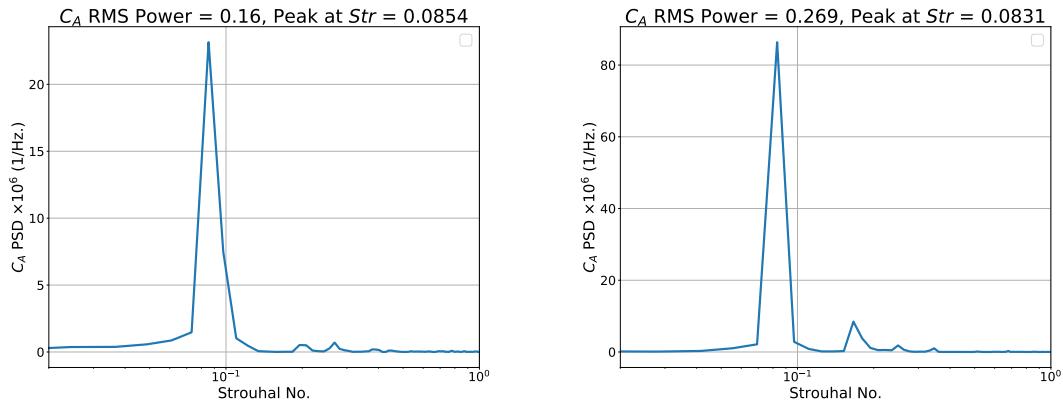


Figure 6.37: The axial force PSD plot for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,1}$.

6.3.3.3 Blunt Nose $R_n = R_{n,2}$

A pulsating flow pattern was established for both the constant D and constant L_1 geometries at $R_n = R_{n,2}$. One cycle of the flow is shown in Fig. 6.38 and Fig. 6.39 for the constant D and constant L_1 geometries, respectively. The pressure-time maps for both geometries are shown in Fig. 6.40. The axial force and axial force PSD plots are shown in Figs. 6.41 and 6.42, respectively.

For the constant D geometry there is a notable difference in the size of the initial separation bubble and the angle of the separation shock when compared to the $R_{n,1}$ constant D case above. The flow structure is more similar to that shown by the sharp cone case, where there is a large separated region throughout. However, compared to the sharp cone case the Type IV interaction is less well defined and the shock train does not become as obvious. In addition, the pressure-time map shows a lower peak pressure than the sharp cone and constant L_1 cases. Finally, the separated region does not extend past the nose as it did in the $R_{n,1}$ case.

The constant L_1 pattern also shows a large separation bubble at the early stage of each cycle. However, the upstream extent of the separation bubble becomes larger than in the constant D case. The larger L_1 enables the shock from the nose to reach a lower angle before it interacts with the rear-cone shock. This reduces the shock strength close to the triple point, lowering the total pressure loss and increasing the Mach number in the supersonic jet. In turn, this leads to a higher peak pressure on the rear-cone face, as shown in the pressure-time map. Similar to the sharp cone case and $R_{n,1}$ case, a feed back loop between the peak pressure on the rear-cone and the separation region growth is established.

The axial force coefficient plots show that the force oscillation in the constant D case appears to be increasing in magnitude and period as the simulation time increases. This is also shown in the pressure-time map for the constant D case. The constant L_1 case appears to have a noisier axial force plot than the constant D case, where the variation both within and between the oscillations is larger. The Strouhal number for the two geometries are in reasonable agreement, and much closer to the sharp cone case than the $R_{n,1}$ geometry.

This test case shows clearer differences between the constant D and constant L_1 geometries than in the $R_{n,1}$ case. This indicates that the length ratios between the R_n , D and L_1 have an important effect on the flow structure.

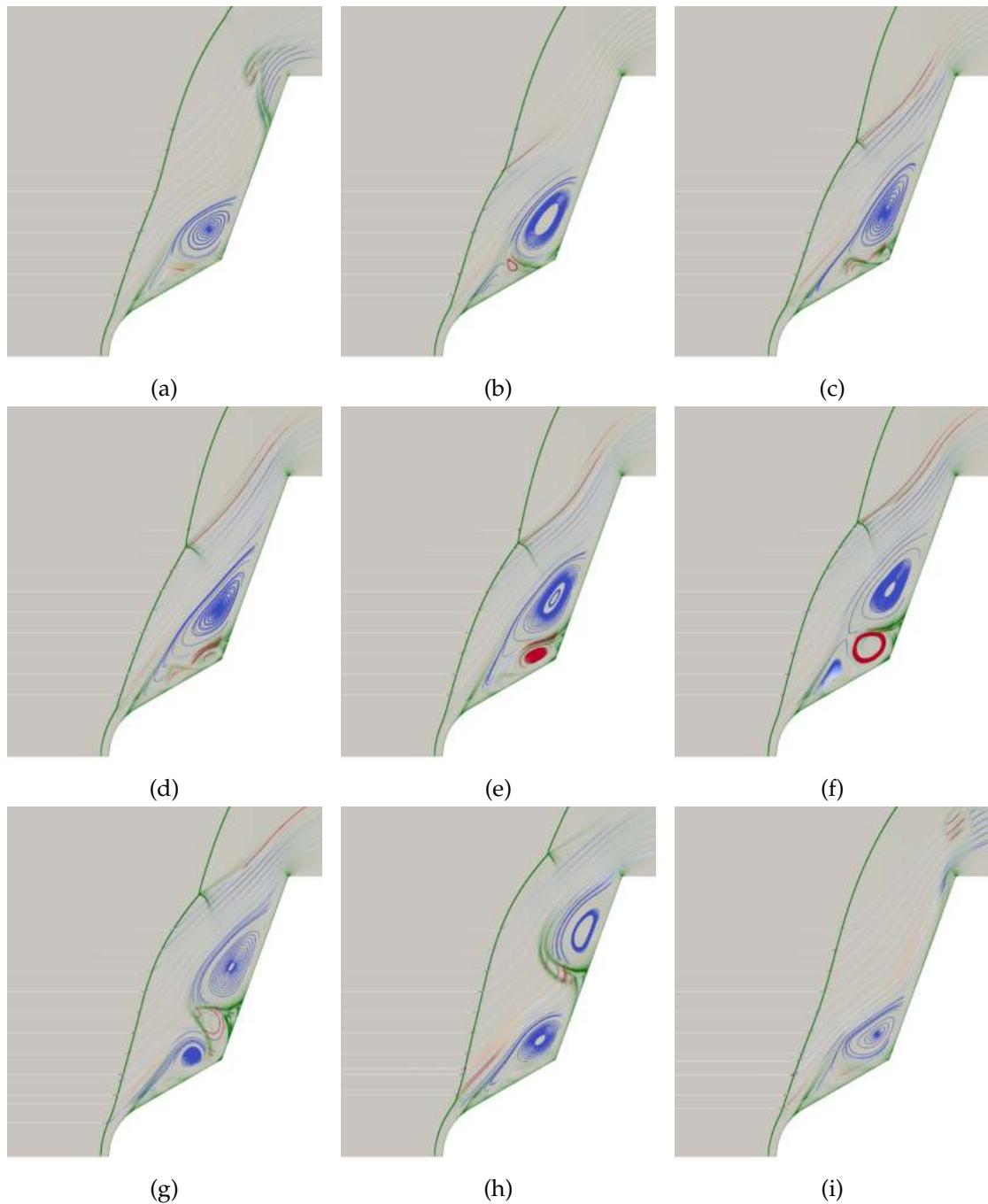


Figure 6.38: The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,2}$, constant D geometry. The streamlines are coloured by vorticity, with red representing positive and blue negative (anti-clockwise and clockwise rotation, respectively).

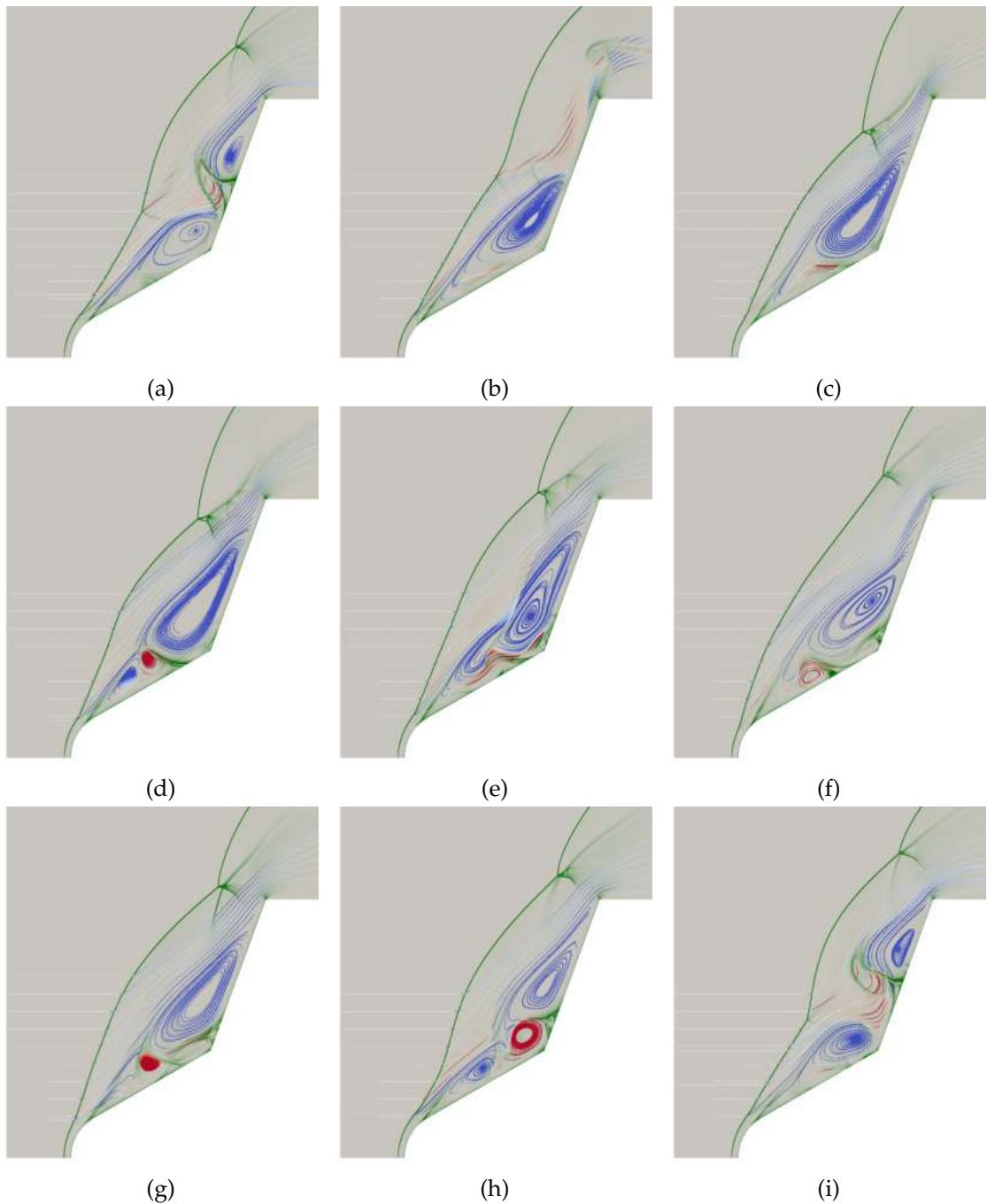


Figure 6.39: The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,2}$ constant L_1 geometry. The streamlines are coloured by vorticity, with red representing positive and blue negative (anti-clockwise and clockwise rotation, respectively).

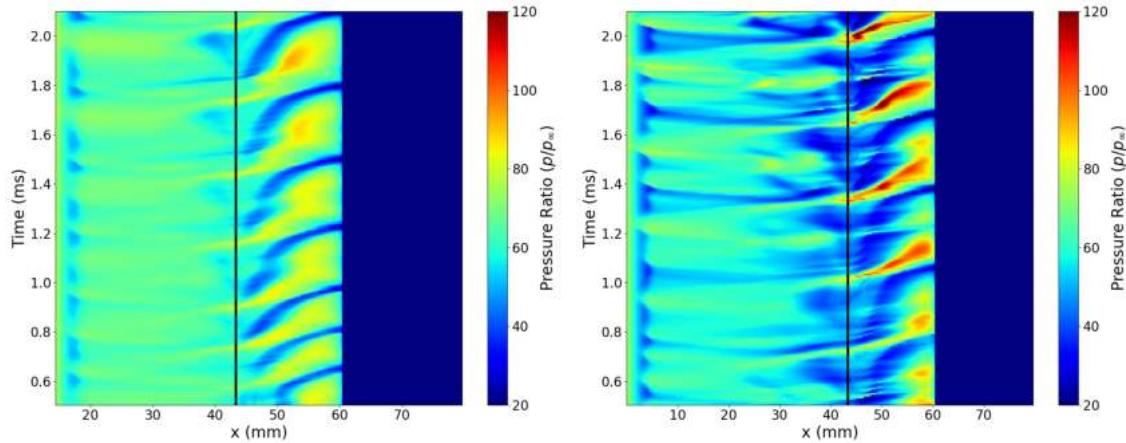


Figure 6.40: The pressure-time map for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,2}$.

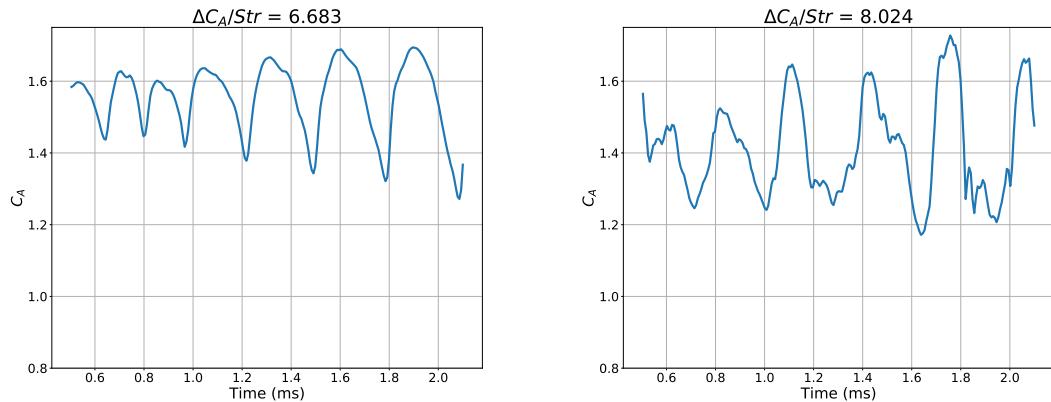


Figure 6.41: The axial force coefficient plotted against time for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,2}$.

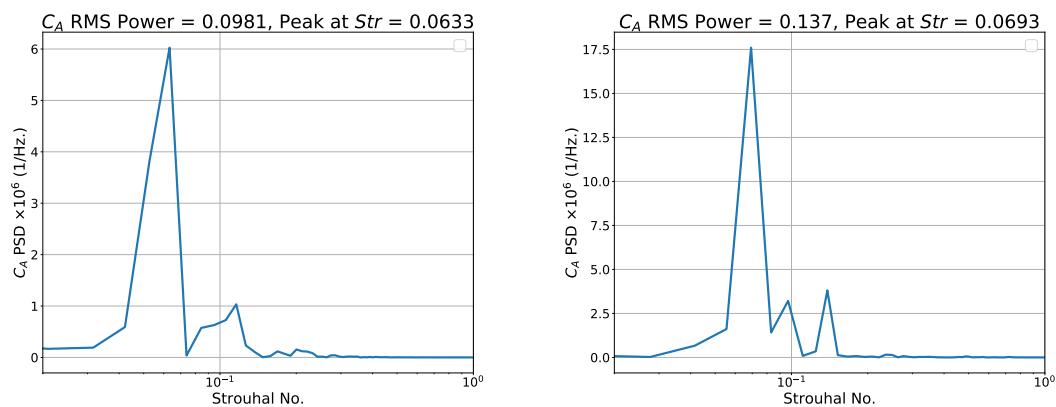


Figure 6.42: The axial force PSD plot for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,2}$.

6.3.3.4 Blunt Nose $R_n = R_{n,3}$

For the $R_{n,3}$ cases the flow is distinctly different from the above cases. The constant D case produces a steady flow and the shock structure for this case is shown in Fig. 6.43. It can be seen that the shock from the rear-cone and the shock from the nose merge together with no notable shock interaction. The separated region extends from the end of the nose to approximately half way along the rear-cone. This structure is maintained throughout the simulation time. The resulting pressure-time map and axial force plots are shown in Figs. 6.45 and 6.46, respectively, and it can be seen that there is very little variation over the simulation time.

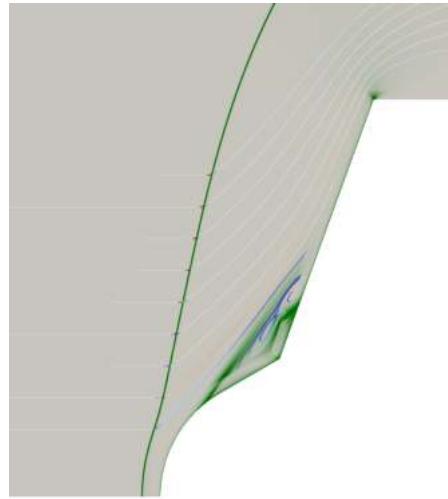


Figure 6.43: The numerical Schlieren and streamlines for the steady case, $R_{n,3}$, constant D geometry. The streamlines are coloured by vorticity, with red representing positive and blue negative (anti-clockwise and clockwise rotation, respectively).

The constant L_1 case displays an unsteady flow pattern, but the level of unsteadiness is greatly reduced. For this case the flow separates immediately behind the nose and the separation region extends almost to the shoulder of the rear-cone. Throughout each cycle there are multiple interacting vortices in the separation region. A Type IV interaction can be seen moving above and below the rear-cone shoulder but the motion is constrained to the upper area of the rear-cone. This is shown in the pressure-time map where a modest cyclical change in the pressure can be seen close to the shoulder. The location of the shock impingement means that the separation region only needs to expand by a small amount before mass is able to escape over the shoulder. This results in a smaller variation in axial force through each cycle. The Strouhal number of 0.0693 is the same as in the sharp cone and $R_{n,2}$ constant L_1 cases.

The flow pattern for the constant L_1 case closely matches the description of “steady separated” flow given by Abbott *et al.* [1]. Schematics of this flow showed a small shock interaction near the rear-cone shoulder as seen in the simulations. It is difficult

to say whether the flow pattern falls into the oscillating or pulsating category of Sasidharan and Duvvuri [238]. The shock remains approximately in the same position, but the oscillations are of a similar frequency to the pulsating cases and are driven by the shock interaction rather than instabilities in the shear layer.

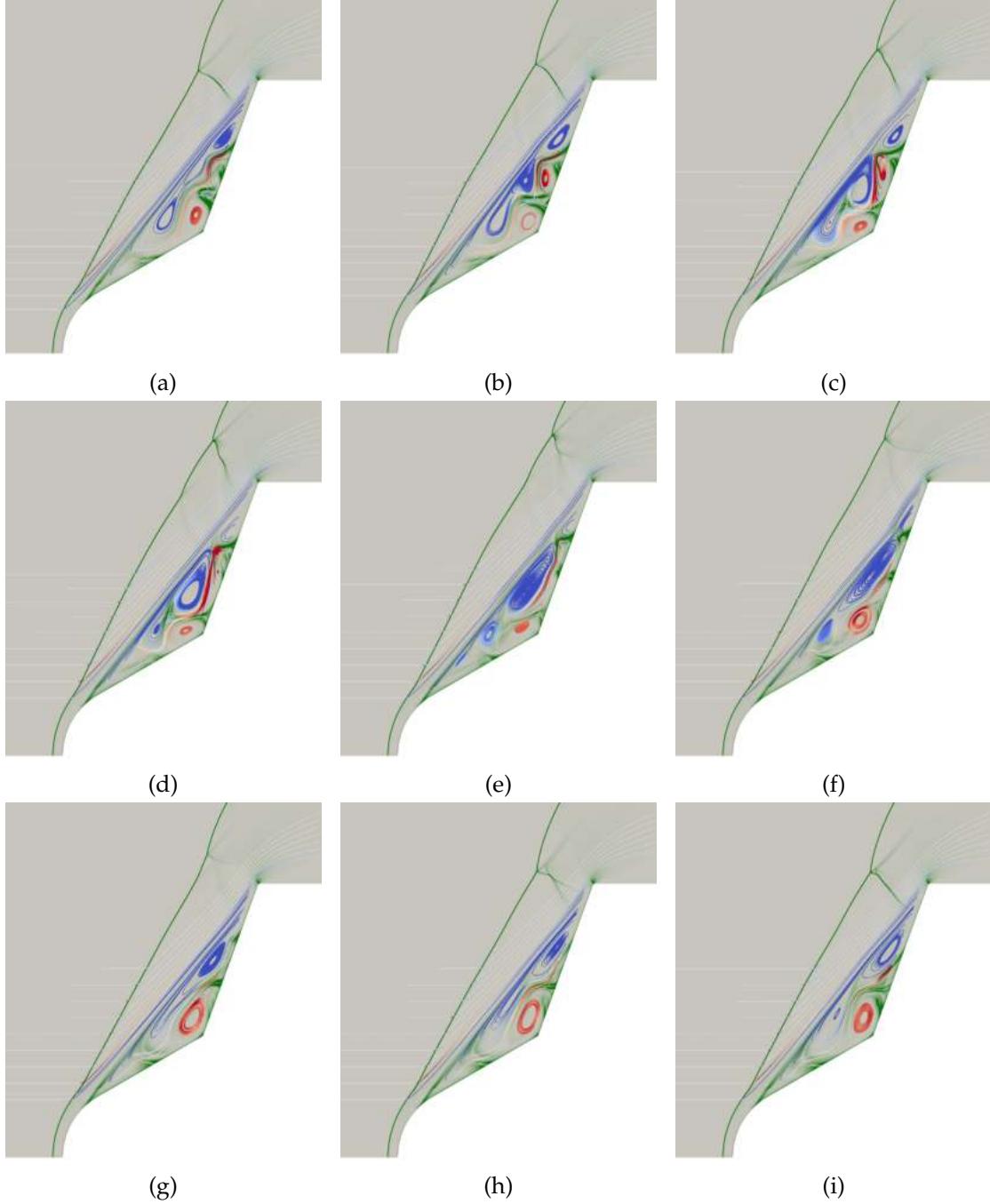


Figure 6.44: The numerical Schlieren and streamlines for approximately one unsteady oscillation for the $R_{n,2}$ constant L_1 geometry. The streamlines are coloured by vorticity, with red representing positive and blue negative (anti-clockwise and clockwise rotation, respectively).

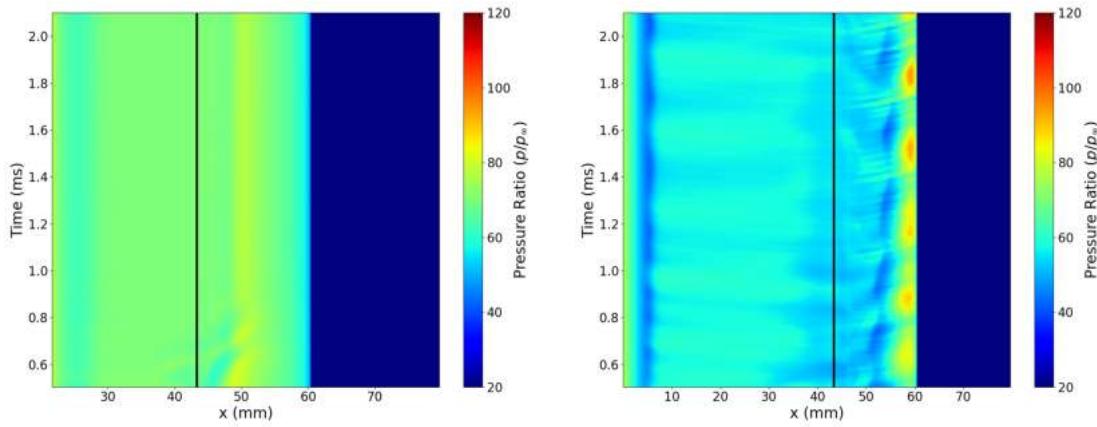


Figure 6.45: The pressure-time map for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,3}$.

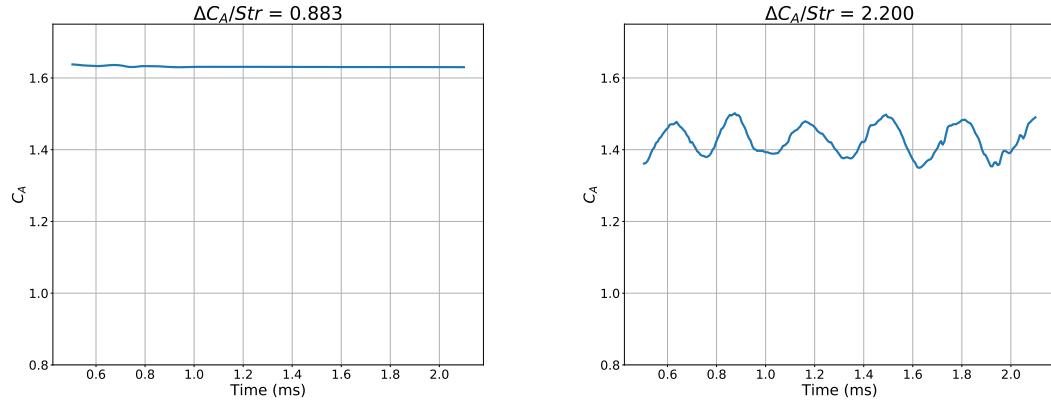


Figure 6.46: The axial force coefficient plotted against time for the constant D geometry (left) and constant L_1 geometry (right), with $R_n = R_{n,3}$.

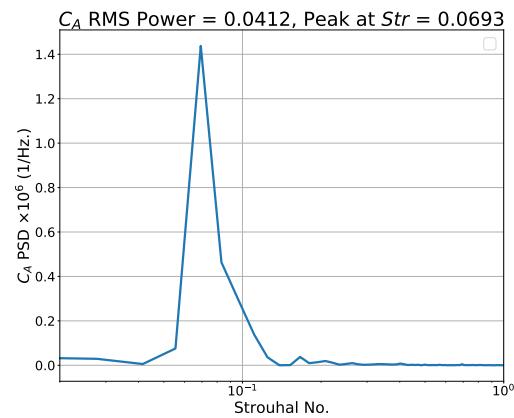


Figure 6.47: The axial force coefficient PSD plot for the constant L_1 geometry, with $R_n = R_{n,3}$.

6.3.3.5 Discussion

The above results clearly show that nose bluntness has a marked effect on unsteady double-cone flows. Simulations of cones with $R_n = R_{n,1}, R_{n,2}, R_{n,3}$ were carried out to better understand the influence that nose bluntness has on the flow.

Plots of the surface pressure and the shear stress coefficients for the different cones are shown in Fig. 6.48. It can clearly be seen that the nose bluntness reduces both the pressure and shear stress behind the nose. The reduction in pressure is caused by an over-expansion behind the nose, whilst the reduction in shear stress is attributed to the entropy layer [9, 114, 129]. The entropy layer is created by the strong shock at the blunt nose generating more entropy than the weaker shock over the conical body. The entropy layer can extend much further downstream than the blunt nose. Figure 6.49 depicts the entropy over the sharp and $R_{n,1}$ cones. It can be seen that the post-shock entropy is uniform for the sharp cone case. For the $R_{n,1}$ cone, a region of higher entropy extends past the nose before being partially “swallowed” by the boundary layer. When the boundary layer is contained within the entropy layer the boundary layer edge conditions are modified. Inside the entropy layer there is a lower Mach number, higher temperature and lower density at the edge of the boundary layer. This results in a thicker boundary layer, lower unit Reynolds number and a reduction in shear stress [9, 129].

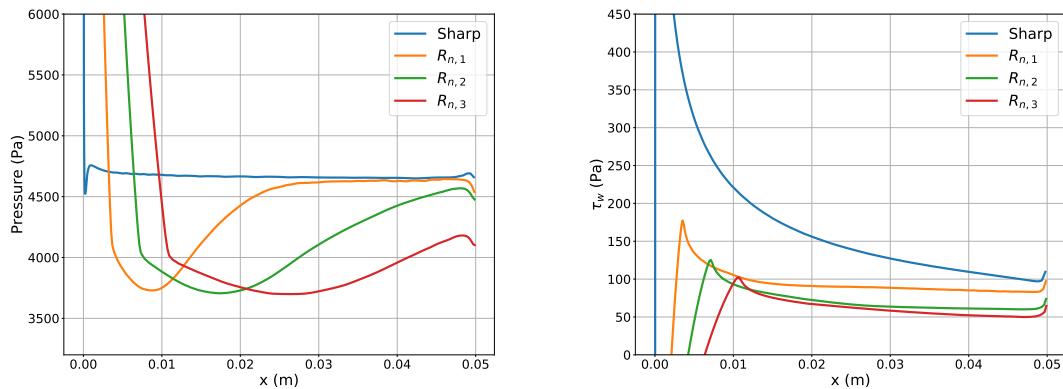


Figure 6.48: The pressure (left) and shear stress (right) on cones with different nose radii.

Triple-deck theory [261] provides a theoretical framework for the analysis of self-separating flows. In this method the flow before the separated region is broken down into three decks: a viscous, incompressible deck close to the wall, inviscid rotational flow in the middle deck and inviscid irrotational flow in the outer deck.

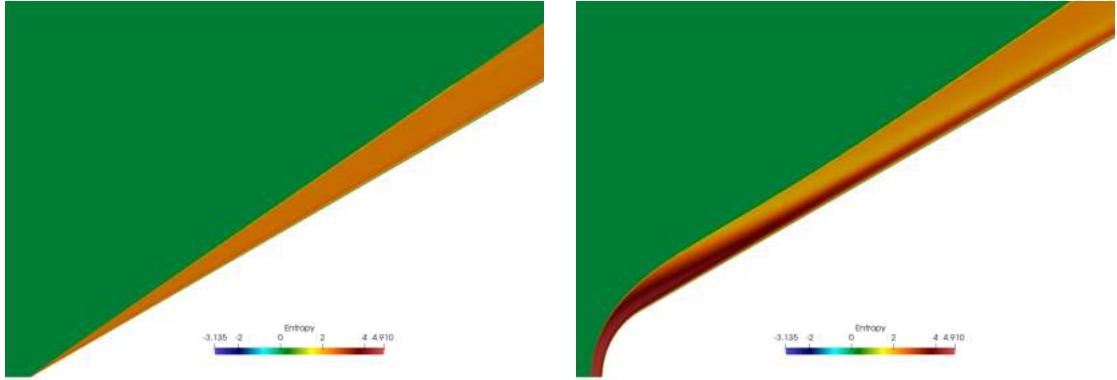


Figure 6.49: The entropy field for the sharp cone (left) and $R_{n,1}$ cone (right). Calculated using $S = [\gamma/(\gamma - 1)] \ln(T/T_\infty) - \ln(p/p_\infty)$.

This theory, and its later extensions [199, 262], provides the following relationship between the length of the separation region and the flow conditions [114]:

$$L_s \propto \left(\frac{\mu_w T_w^{1/2}}{\tau_w^2} \right)_1 \frac{(p_3 - p_2)^{3/2}}{p_1^{1/2}}. \quad (6.3)$$

The subscript 1 represents the conditions upstream of separation, 2 the conditions behind the separation shock and 3 the conditions at the reattachment point.

The relationship in Eq. (6.3) indicates that a reduction in the shear stress and the pressure in front of the separation region will increase the size of the separation region. Figure 6.48 shows that both pressure and shear stress reduce on the fore-cone as the nose radius increases. This allows the separation bubble to grow more easily, leading to the feedback loop with the shock structure that increases p_3 , as described above.

In addition to higher entropy, the nose bluntness leads to additional vorticity in the entropy layer. As demonstrated by Hornung *et al.* [125], the accumulation of vorticity helps drive the growth of the separation region. It was shown in Ref. [125] that the vorticity could be generated by a no-slip boundary condition on a sharp cone or, in an inviscid simulation, by a small blunt nose. Both simulations produced a similar pulsating unsteadiness. The increase in vorticity created by the blunt nose is therefore likely to contribute to the growth of the separation region. For the $R_{n,1}$ simulations this may account for the larger, more rapid growth of the separation region and the higher frequency of the pulsation cycle.

Previous work has shown that the angles and length ratios between the two cones have an important influence on the resulting flow [1, 125, 238]. The above results and analysis demonstrate that nose bluntness must also be considered, and indicate that two non-dimensional parameters govern the changes caused by nose bluntness: R_n/L_1 and L_1/D . An additional simulation was carried out using $R_n/L_1 = 0.1$ and a constant D geometry to better understand this relationship. The non-dimensional

values and unsteadiness metrics for all geometries are given in Table 6.6. It can be seen that the $R_n/L_1 = 0.1$ constant D geometry gives large unsteadiness metrics and a high frequency. The observed flow pattern was very similar to that in $R_{n,1}$ cases, with the shock interaction starting on the fore-body and the separation region expanding beyond the nose.

Name	R_n/L_1	L_1/D	R_n/D	C_A Power	$\Delta C_A/Str$	Str_{L_2}
$R_{n,0}$	0.00231	0.3005	0.000694	0.1668	8.315	0
$R_{n,1}$ const. D	0.2	0.2506	0.05012	0.16	6.944	0.01385
$R_{n,2}$ const. D	0.5	0.2005	0.1002	0.09815	6.683	0.02771
$R_{n,3}$ const. D	1	0.1504	0.1504	0.001617	0.8829	0.04156
$R_{n,1}$ const. L_1	0.1667	0.2843	0.04738	0.2689	10.42	0.05541
$R_{n,2}$ const. L_1	0.3333	0.2695	0.08984	0.1373	8.024	0.06927
$R_{n,3}$ const. L_1	0.5	0.2562	0.1281	0.04123	2.2	0.08312
$R_{n,0.5}$ const. D	0.1	0.2707	0.02707	0.2592	13.63	0.09697

Table 6.6: The non-dimensional length parameters and unsteadiness metrics for the blunted double-cone geometries.

This investigation shows that the variation in unsteadiness with nose bluntness can be explained in terms of changes in R_n/L_1 and L_1/D . As R_n/L_1 is increased by a small amount the unsteadiness is increased as well. This is due to the pressure expansion and entropy layer accommodating additional growth in the separation region.

However, the blunt nose effects are small enough that the flow is initially attached behind the nose, and L_1/D is large enough that the separation shock is at an oblique angle when it interacts with the rear-cone shock. As R_n/L_1 is increased further, and L_1/D is reduced, the unsteadiness reduces as well. This is because the increase in the blunt nose effects leads to separation immediately behind the nose, which dampens the unsteady feedback loop. In addition, the shock interaction is weakened as the separation shock is at a larger angle when it interacts with the rear-cone shock. This helps to reduce the adverse pressure gradient that drives the growth in the separation bubble. When L_1/D is small enough the flow becomes steady as the whole fore-body region is in subsonic flow. A larger R_n/L_1 will mean that this point is reached at a lower L_1/D as the larger nose will generate a larger region of subsonic flow on the fore-body.

It is noted that the above two parameters can be combined into a single value of R_n/D . This value was used by Abbott *et al.* [1], but means that explicit dependence on L_1 is lost. Plots showing the variation in the C_A power RMS with R_n/L_1 and R_n/D are shown in Fig. 6.50. It can be seen both values show the trend described above, where unsteadiness is first increased and then reduced. As such, more data is required to better understand the relative importance of the different non-dimensional parameters.

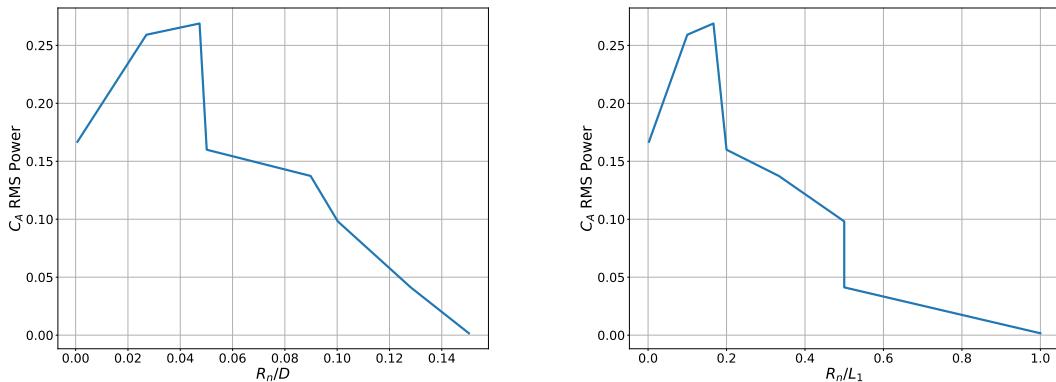


Figure 6.50: The C_A power RMS as a function of R_n/D (left) and R_n/L_1 (right).

6.3.4 Summary

A study has been carried out to investigate the influence of nose bluntness on the unsteady forces experienced by a double-cone. The influence of nose radius has not previously been studied and the results indicate that it can have a marked effect on the forces experienced by the double-cone.

Large differences in the forces (relative to the sharp nose case) were seen for both small and large values of nose bluntness. For small bluntness the growth of the separation region extended beyond the nose, and the frequency of the oscillations increased. In some cases the forces were increased considerably. For the larger bluntness the forces were generally reduced, and the flow was steady when the nose bluntness was large and L_1/D was small.

Simulations of simple blunt cone geometries showed that nose bluntness reduces the pressure behind the nose and introduces an entropy layer. Both these effects accommodate greater growth of the separation region. In addition, nose bluntness changes the location and strength of the shock interaction between the separation shock and the rear-cone shock. A smaller separation shock angle at the triple point tends to increase the adverse pressure gradient at the reattachment point. The combination of these effects help to explain the variation in flow features with nose bluntness.

The results obtained in this work indicate that the influence of nose radius should be considered when designing double-cone geometries. This is especially important for ablating geometries as it was shown that a change from a sharp nose to a nose with small bluntness could lead to large changes in the power and frequency of the oscillatory forces.

6.4 Chapter Summary

In this chapter novel numerical investigations have been carried out using the strand/CAMR solver. This was the first time that the double-wedge experiment of Swantek and Austin [267] has been simulated using AMR or overset domains. The AMR was able to efficiently resolve the shock structures and the overset results were in good agreement with the reference results. A Type IV interaction with a locally recessed surface was then simulated. The results showed that the shape of the recessed region had a large impact on the unsteadiness of the flow field. Two modes of oscillation were identified, where the mode, frequency and amplitude of the oscillations was found to be dependent on the shape of the recessed region. Finally a detailed study into the influence of nose bluntness on unsteady double-cone flows was carried out. The results showed that the nose radii could have a marked impact on the flow features and forces. When using an ablating TPS the nose radius is likely to change over time. Consequently, an understanding of how the oscillatory forces change with bluntness is important in order to avoid resonance and aero-structural issues. This chapter has demonstrated how the new solver can be used to obtain novel insights into previously un-investigated flow phenomena whilst using efficient, automatically generated grids.

Chapter 7

Conclusion

7.1 Summary and Contributions

The primary aim of this research was to determine whether a strand/CAMR solver could produce accurate hypersonic heat flux predictions and enable automated mesh generation. To answer this question a new hypersonic strand/CAMR solver has been developed within the AMROC framework. This required a number of extensions to the existing algorithms and additional components to be implemented. These significant developments added over 40,000 lines of code to AMROC’s C++ and FORTRAN77 libraries. The new solver was then used for a range of simulations to determine whether the strand/CAMR method enables accurate, automated simulations of ablating hypersonic vehicles. This work has resulted in a number of contributions to the wider domain as well as contributions to AMROC’s capabilities.

7.1.1 AMROC Extensions

The two-temperature model was implemented within an AMROC patch integrator to account for the thermochemical nonequilibrium that is observed in hypersonic flows. The implementation of the fluid model required additional inviscid and viscous flux schemes that account for the separate species and energy equations, as well as source terms to account for the chemical reactions and internal energy transfer. The state-of-the-art Mutation++ library was used for the calculation of the thermodynamic properties, source terms and transport properties. This contribution to the AMROC framework enables accurate simulations of thermochemical nonequilibrium flows.

The spatial integration used within AMROC was modified to enable simulations on two-dimensional mapped grids. This required modification of the inviscid and viscous flux schemes and the MUSCL reconstruction. The new spatial integration

methods were verified using the Method of Manufactured Solutions (MMS). This was the first published use of MMS for a thermochemical nonequilibrium flow solver. It was also the first time this method was applied within the AMROC framework and it has since been used to verify other patch integrators. Finally, this was the first application of mapped mesh methods within AMROC and they have since been used by other researchers in the group [221].

7.1.2 Implicit and Implicit/Explicit Time Integration

Implicit time-integration methods have been implemented to increase the time step on the highly-stretched strand grids. Each implicit update requires the creation and solution of a large, sparse linear system. The Jacobian in the LHS matrix is constructed using the complex-step method. This method was found to give accurate derivatives, outperforming the finite-difference method by orders of magnitude. Methods to efficiently solve the linear system on quasi-structured strand meshes were investigated. A range of solvers have been implemented and tested including dimensionally split and line-implicit methods that take advantage of the strand mesh data layout. The solution of the linear system across multiple processors is achieved using an additive Schwarz/block-Jacobi method. Testing of the different solvers showed that the most efficient method was highly dependent on the flow. For high Reynolds number, supersonic flows the line-implicit method was found to be most efficient. These tests are a contribution to the wider literature as it is the first time that the performance of different linear solution methods, that take advantage of the strand grid structure, have been tested.

Various simulations showed that the implicit methods can significantly reduce computational times relative to explicit methods. The addition of implicit time-integration methods is a major contribution to the AMROC framework. Without this addition, many of the viscous strand computations shown in the results sections would have been computationally intractable.

7.1.3 Novel Strand Mesh Generation Algorithm

The extensions to the spatial and time integration methods enabled the solution of the governing equations on mapped, body-fitted grids. The mapping was integrated with a strand mesh generator to enable high levels of automation. A significant contribution of this work is a novel strand mesh generation algorithm that retains a low memory footprint and is suitable for high Reynolds number flow simulations. The new strand algorithm connects an inner and outer growth vector with a circular arc in order to ensure mesh orthogonality at the wall and high smoothness away from the wall. This new method enables default values to be used in most scenarios,

increasing the level of automation. A vortex shedding test case showed that results using the new method were largely independent of the level of smoothing. This is a significant improvement over the original strand mesh technique, which was highly sensitive to the user-input smoothing residual.

7.1.4 Overset Library with Novel Donor Search Algorithms

A new overset library was implemented to join the off-body CAMR domain and the near-body strand domain. The hole-cutting is performed using the highly flexible CPT method of Mauch [182], where the level surface is specified using the strand mesh structure. Efficient, point-to-point inter-processor communication is established by comparing overlapping bounding boxes that contain each processor's donor cells and receptor locations. The number of receptors exchanged (and thus the number of donor searches) is minimised by using spatial partitioning techniques based on OBBs and oriented auxiliary grids.

Several near-body donor search methods were implemented and tested. These tests led to a novel inverse map set-up algorithm, which significantly reduces the set-up times of the inverse map methods. Using this method, the inverse map methods are recommended as the default: they have competitive set-up times, the lowest search times, and both the search and set-up times scale extremely well with the number of cells. They were also found to be extremely robust in practical applications. With the new set-up algorithm, the inverse map methods were found to give significant performance improvements over the more commonly used ADT method. This thesis also introduces a novel spatial hashing technique that almost completely removes the need for near-body donor searches when using a stationary near-body mesh. The number of searches is significantly reduced even when the off-body CAMR domain is adapting or being dynamically redistributed across processors.

The overset grid assembly algorithms were verified using order-of-accuracy tests based on the L_∞ norm. These tests showed that the interpolation to receptor locations from the near-body domain was approximately second-order-accurate for all receptors in the off-body domain.

7.1.5 Overset Heat Flux Predictions in Hypersonic Flows

One of the primary contributions of this work is a detailed assessment of the influence of overset methods on hypersonic heat flux predictions. To the authors knowledge this has not previously been studied in detail. The solver was able to accurately predict the heat fluxes in high-enthalpy, blunt body test cases, where the AMR accurately captured the bow shock in the off-body region and the boundary layer was

well resolved by the near-body strand mesh. Comparisons with single-domain simulations and data taken from the literature showed that overset methods have very little impact on the results for blunt body cases. The results also showed that the strand/CAMR method can reduce the computational time compared with single domain methods. This is because the off-body region can be integrated in time using explicit methods, reducing the number of cells that must be integrated using the more expensive implicit methods.

More challenging shock-shock and shock-boundary-layer-interaction test cases were then considered, where discontinuities crossed the overset boundary and impinged on the surface. An Edney Type IV shock interaction case and a double-wedge test case were used to assess the influence of the overset mesh on the surface predictions when discontinuities cross the overset boundaries. The results from the strand/CAMR solver were in good agreement with single-domain results and/or results from the literature in both cases. In summary, it has been demonstrated that the strand/CAMR method can be used for engineering heating predictions, even in cases involving shock-boundary-layer interactions.

7.1.6 Strand/CAMR Surface Deformation

The second major contribution of this thesis is the demonstration of a highly-automated surface deformation method. Test cases with deforming surfaces showed that the strand/CAMR solver is able to update the mesh as the surface recesses, with very little user input required. This confirms that the strand/CAMR method is suitable for highly-automated simulations of ablating or flexible TPSs, where the external geometry of the body will change throughout flight. In addition, the ability to perform overset simulations of bodies in motion was demonstrated using a case with significant changes in the bodies' positions and the shock structures.

7.1.7 Novel Unsteady Flow Simulations

Finally, the new solver was used to perform two novel flow investigations. First, the influence of a locally recessed region on a Type IV interaction was studied. Three surfaces were created based on the idealised recession of an ablating surface. The three surfaces led to three different oscillating flows showing how the recession shape has a large influence on the unsteady Type IV interaction. The flow structures were in good agreement with previous experimental findings [297], where a different geometry was used. As a result, the oscillatory motion caused by a cavity was categorised into two unsteady modes: the Type IV \leftrightarrow (III) and Type IV \leftrightarrow IVa modes. The simulations showed that the mode, frequency and amplitude of the oscillations depended on the recessed surface shape.

The second novel flow study investigated the impact of nose bluntness on unsteady, double-cone flows. The results showed that the over-expansion around the blunt nose and the entropy layer effects could greatly influence the unsteadiness of the flow and the forces acting on the double-cone. Small bluntness resulted in greater changes in the size of the separation region. In turn this created larger variations in the forces acting on the body than for the sharp-nose case. As bluntness was increased further the force variations were reduced relative to the sharp cone case with the flow becoming steady or “steady separated” for the largest nose radius. These results indicate that ablation of a double-cone nose could greatly impact the power and frequency of the force oscillations. Consequently, engineers should consider the nose bluntness when designing ablating experimental models and flight vehicles.

In summary, this work has shown that the strand/CAMR paradigm gives accurate results for a range of hypersonic flows and enables high levels of automation in the presence of dynamic shock structures and changes to a vehicle’s geometry. Therefore, strand/CAMR solvers could be a useful tool in the design of future hypersonic vehicles and TPSs.

7.2 Possible Future Research

Whilst this research has demonstrated the ability of a strand/CAMR solver to obtain accurate heat flux results in hypersonic flows, the solver has a number of limitations. These limitations could be addressed in future research to create a solver that is more flexible and better suited to the design of practical vehicles.

7.2.1 Fluid-Material Coupling

In this research it has been shown that the strand/CAMR method can be used for highly-automated surface deformation simulations. For the demonstrations in this work the surface was deformed between two predetermined geometries, as the recession is not predicted by the solver. Future work could couple the flow solver to an ablative material response solver that could accurately predict the surface recession. The implementation could follow previous examples in the literature [51, 151, 179, 206] in order to couple the two solvers. Most examples in the literature use a loose coupling strategy, where steady-state fluid simulations are coupled to time-accurate material simulations. However, if the coupling routines are combined with robust, time-accurate implicit methods then tighter coupling of the fluid and material solvers could be investigated in future research.

7.2.2 Extension to Three Dimensions

In this research, the solver was limited to two dimensions and future research could extend the solver to three dimensions. Three-dimensional strand/CAMR solvers have been demonstrated in the literature, but extension to three dimensions will require a number of additions and modification to the current method. The extension of the off-body AMROC patch integrator to three dimensions should be relatively simple, due to the structure, Cartesian mesh, and the use of face normal flux functions.

However, a new strand mesh solver will need to be written as the structured layout of the mesh will be lost. The new algorithms should take advantage of the quasi-structured nature of the strand mesh, where the cells in the wall-normal direction can still be treated as one-dimensional structured lines. This will enable efficient memory layouts and simple construction of line-implicit and dimensionally-split methods. The layers in the strand mesh will need to be treated as two-dimensional unstructured domains and utilise the necessary unstructured algorithms.

The strand mesh generation algorithms will need to be modified for three-dimensional surfaces. That said, many of the routines can be re-used without change. As discussed in Section 4.2, the strand mesh algorithm developed in this work already utilises three-dimensional vector algebra to calculate the locations of the nodes on the strand. In addition, many of the strand generation algorithms have been implemented using data types that accept the number of dimensions as a user-input template parameter and could therefore be re-used in a three-dimensional solver. However, there are a number of areas within the strand generation where two dimensions are assumed, such as in the strand intersection algorithms.

Finally, the overset algorithms will also need to be extended to three dimensions. Many of the algorithms developed in this work can be re-used, as they have also been written with the number of dimensions as a template parameter. For example, the OBB creation and overlap tests, spatial partitioning methods, range-based and inverse map search algorithms, and point-to-point communication set-up should work in three dimensions without any modification (although they have only been tested in two dimensions). In addition, the previously implemented SAMR interpolation and CPT hole-cutting algorithms have already been thoroughly tested in three dimensions [54, 66]. However, the point-in-cell tests on the mapped domain assume a quadrilateral cell, and the mapped interpolation routines and stencil walking algorithm assume a two-dimensional structured mesh. Clearly, the extension of the strand/CAMR solver to three dimensions is a significant undertaking and various unforeseen issues will need to be resolved.

7.2.3 Overset Grid Assembly

The current solver only allows for a single near-body domain. In future work the solver could be modified to allow for multiple near-body domains. The solver would need to account for a variable number of near-body domains when reading in data, determining the time step, performing the updates and outputting information. The overset grid assembly will also need to be extended. The off-body AMROC solver is already capable of handling multiple holes in the domain, but hole cutting on the near-body domains must be implemented to remove cells that are within solid bodies and to minimise the number of cells that overlap with other near-body domains. The spatial partitioning algorithms developed in this work could be used to perform the near-body to near-body communication set-up and hole cutting. Point-to-point communication could be set up between the near-body domains using the same OBB overlap tests that are used for the near-body to background grid communication set-up. The Cartesian inverse map method could be used to generate hole-maps [188] and rapidly determine which cells are within solid bodies, outside of solid bodies or require further overlap tests. Once the cells that lie within other solid bodies have been cut, overlapping near-body cells could be cut using implicit hole-cutting techniques to reduce the size of the near-body domains. Implicit hole-cutting enables a high-level of automation by cutting overlapping near-body cells according to grid metrics such as cell size or wall distance. In a strand grid, the node index could be used as a metric to determine which near-body cell should be removed from the simulation.

Further enhancements to the overset methods developed in this work could be investigated. Research could be conducted into how the overset method can be made conservative. For example, one can interpolate fluxes to overset boundary faces [53, 58, 268], rather than interpolating conserved variables to ghost cells. This may reduce some of the discrepancies that were shown between overset and single-domain results in cases where discontinuities crossed the overset boundary.

Currently, the overhead associated with the overset routines can account for 25-30% of the computational time when using time-accurate methods¹. The overset methods would benefit from the implementation of dynamic load balancing where the proportion of processors used by the off-body and near-body grids can be modified during a simulation. This would remove the need to manually find the best parallel overset decomposition and better balance the work-load as the dynamic AMR evolves. In addition, investigations could be carried out into how computation and non-blocking overset communication could be better overlapped.

¹For steady-state simulations the single-synchronisation method can be used to reduce the overset times to less than 10% of the total time.

7.2.4 Implicit Time Integration

There are a number of ways that the implicit time-stepping methods implemented within this work could be used as a basis for future research. Firstly, the calculation of the Jacobians using the complex-step method was shown to dominate the computational time of the implicit update (see Section 3.3.3). Therefore, the implementation of analytic derivatives would likely reduce the computational time significantly. The implementation of the analytic derivatives could be verified using the complex-step Jacobians. Secondly, the implicit methods are currently designed for a uniform (i.e. single-level) mesh, but future research could investigate how these implicit methods could be extended for AMR simulations. This would enable implicit time-stepping to be used on both the near-body and off-body domains, where the overset nature of the grid can be accounted for using the additive Schwarz method [248].

7.2.5 Cut-Cell Comparisons

Cut-cell methods were discussed in Section 2.2.1 as they allow for accurate boundary representations in a CAMR solver, without having to use a near-body mesh. The primary disadvantage of this method is that a large number of cells are required to give adequate boundary layer resolution for arbitrary geometries that may not align with the Cartesian cells, as discussed in Section 2.2.1. However, the benefits of a cut-cell method are that only a single CAMR domain is required and there are very few restrictions on the surface shape and the amount of shape change. Consequently, comparisons between the strand/CAMR and cut-cell approaches in terms of accuracy, computational efficiency and flexibility would be a useful contribution to the wider literature. When using the cut-cell method there is likely to be a severe time-step restriction imposed by the small near-wall cells. As a result, it is possible that implicit methods will need to be implemented on the CAMR domain.

7.2.6 Alternative Thermochemical Models

The current solver provides a framework for exploring the impact of different thermochemical models on hypersonic flow simulations. As discussed in Section 2.1 there are a number of new two-temperature thermochemical models being developed using data from first principle calculations of particle interactions. These models could be implemented within the Mutation++ library and the results compared with the more traditional models used in this work. In addition, alternatives to the two-temperature model could be implemented and tested. For example, the three-temperature model uses a separate electron temperature. Accurate assessment

of the electron temperature is important when the electron density is high and when simulating ablation at high Mach numbers. Finally, other libraries such as KAPPA [37] and PLATO [195] are available that implement different thermochemical models. The accuracy and efficiency of each library in different flow conditions could be assessed.

Appendix A

A.1 Thermochemical Data for Air and Nitrogen

In this work the five species air and two species nitrogen two-temperature models were used. The constants and coefficients used for these mixtures are given below.

Species	h_s^0 (J/mol)	M_s (g/mol)	$\theta_{v,s}$ (K)
N_2	0	28.0	3,408.46
O_2	0	32.0	2,276.98
NO	91,089	30.0	2,759.29
N	472,440	14.0	N/A
O	249,229	16.0	N/A

Table A.1: The species constants used in this work.

Reaction	A_r (cm ³ /mol)	$\eta_{f,r}$	θ_r (K)	$T_{c,f}$	$T_{c,b}$
$N_2 + M \rightleftharpoons 2N + M$	3×10^{22}	-1.6	113,200	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$N_2 + N_2 \rightleftharpoons 2N + N_2$	7×10^{21}	-1.6	113,200	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$N_2 + O_2 \rightleftharpoons 2N + O_2$	7×10^{21}	-1.6	113,200	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$N_2 + NO \rightleftharpoons 2N + NO$	7×10^{21}	-1.6	113,200	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$O_2 + M \rightleftharpoons 2O + M$	1×10^{22}	-1.5	59,360	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$O_2 + N_2 \rightleftharpoons 2O + N_2$	5×10^{21}	-1.5	59,360	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$O_2 + O_2 \rightleftharpoons 2O + O_2$	5×10^{21}	-1.5	59,360	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$O_2 + NO \rightleftharpoons 2O + NO$	5×10^{21}	-1.5	59,360	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$NO + M \rightleftharpoons N + O + M$	5×10^{15}	0.0	75,500	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$NO + NO \rightleftharpoons N + O + NO$	1×10^{17}	0.0	75,500	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$NO + N \rightleftharpoons N + O + O_2$	1×10^{17}	0.0	75,500	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$NO + O \rightleftharpoons N + O + NO$	1×10^{17}	0.0	75,500	$\sqrt{T_{tr}T_{ve}}$	T_{tr}
$N_2 + O \rightleftharpoons NO + N$	5.69×10^{12}	0.42	42,938	T_{tr}	T_{tr}
$O_2 + N \rightleftharpoons NO + O$	2.49×10^9	1.18	4,005.5	T_{tr}	T_{tr}

Table A.2: The constants and rate-controlling temperatures used in the Park two-temperature chemical model.

Vibrator	Partner	$A_{s,r}$	$B_{s,r}$
N_2	N_2	221	0.0290
N_2	O_2	229	0.0295
N_2	NO	225	0.0293
N_2	N	180	0.0262
N_2	O	72.4	0.015
O_2	N_2	134	0.0295
O_2	O_2	138	0.03
O_2	NO	136	0.0298
O_2	N	72.4	0.015
O_2	O	47.7	0.059
NO	N_2	49.5	0.042
NO	O_2	49.5	0.042
NO	NO	49.5	0.042
NO	N	49.5	0.042
NO	O	49.5	0.042

Table A.3: The Millikan and White constants used relaxation model.

Species	A_s	B_s	C_s
N_2	2.68×10^{-2}	0.318	-11.3
O_2	4.49×10^{-2}	-8.26×10^{-2}	-9.20
NO	4.36×10^{-2}	-3.36×10^{-2}	-9.58
N	1.16×10^{-2}	0.603	-12.4
O	2.03×10^{-2}	0.429	-11.6

Table A.4: The species constants used in the Blottner viscosity model.

A.2 Method of Manufactured Solutions Inputs and Results

This appendix contains additional data for the MMS verification study. The tables give the coefficients used in the creation of the analytic solutions and the figures show comparisons between the observed and expected order-of-accuracy.

Variable (ϕ)	ϕ_0	ϕ_x	f_x	a_x^ϕ	ϕ_y	f_y	a_y^ϕ
ρ_{O_2}	0.75	0.1	sin	1.5	0.2	cos	0.75
ρ_N	1.0	-0.15	cos	0.5	0.1	cos	1.25
u	± 900	60	sin	1.5	30	sin	0.75
v	± 850	40	cos	1.0	50	sin	1.25
$e_{O_2}^{ve}$	285,548	20,000	sin	0.75	15,000	cos	1.0
e_N^{ve}	40	5	sin	0.75	-6	sin	1.25
T_{tr}	300	30	cos	1.0	35	cos	0.5

Table A.5: The constants and functions used to create the solutions for the supersonic Euler MMS test cases.

Variable (ϕ)	ϕ_0	ϕ_x	f_x	a_x^ϕ	ϕ_y	f_y	a_y^ϕ
ρ_{O_2}	1.25	0.075	sin	1.0	-0.1	cos	0.75
ρ_N	1.3	-0.1	cos	0.5	0.125	cos	1.25
u	± 105	10	sin	1.5	9	sin	0.75
v	± 95	9	cos	1.0	7.5	sin	1.25
$e_{O_2}^{ve}$	285,548	20,000	sin	0.75	15,000	cos	1.0
e_N^{ve}	40	5	sin	0.75	-6	sin	1.25
T_{tr}	250	30	cos	1.0	-25	cos	0.5

Table A.6: The constants and functions used to create the solutions for the subsonic Euler MMS test cases.

Variable (ϕ)	ϕ_0	ϕ_x	f_x	a_x^ϕ	ϕ_y	f_y	a_y^ϕ
ρ_{O_2}	0.75	0.1	sin	1.0	0.2	cos	0.75
ρ_N	1.0	-0.15	cos	0.5	0.1	cos	1.25
u	70	4	sin	1.5	-12	sin	0.75
v	95	-20	cos	1.75	4	sin	1.25
$e_{O_2}^{ve}$	285,548	20,000	sin	0.75	15,000	cos	1.0
e_N^{ve}	40	5	sin	0.75	-6	sin	1.25
T_{tr}	300	30	cos	1.0	-25	cos	0.5
μ	10	2	sin	-1.75	-0.2	cos	0.9

Table A.7: The constants and functions used to create the solutions for the subsonic Navier-Stokes MMS test case.

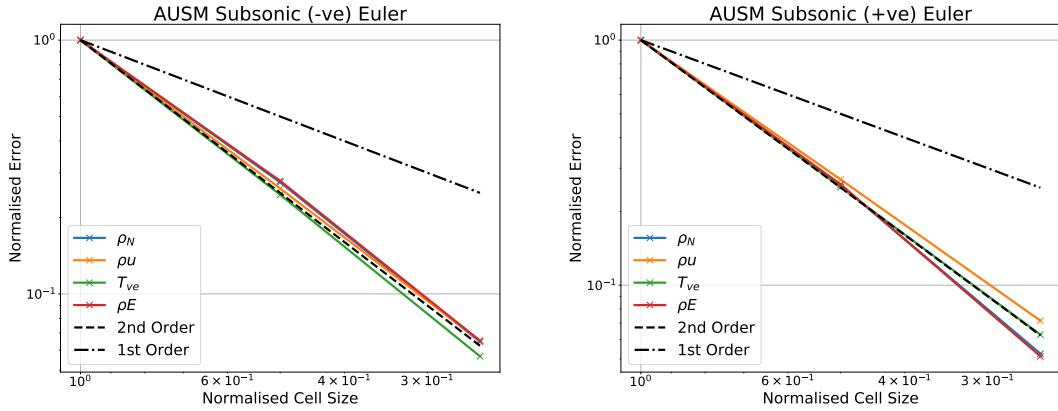


Figure A.1: The normalised error convergence for the subsonic AUSM fluxes, compared with the expected convergence for first- and second-order-accuracy.

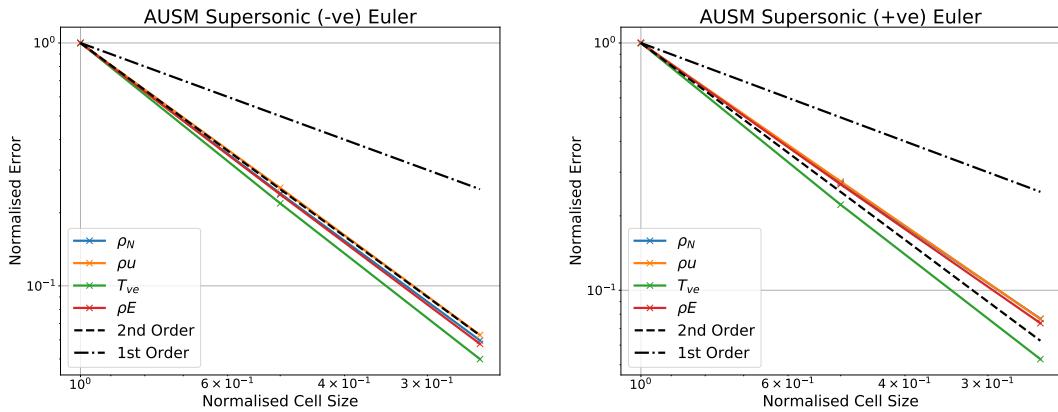


Figure A.2: The normalised error convergence for the supersonic AUSM fluxes, compared with the expected convergence for first- and second-order-accuracy.

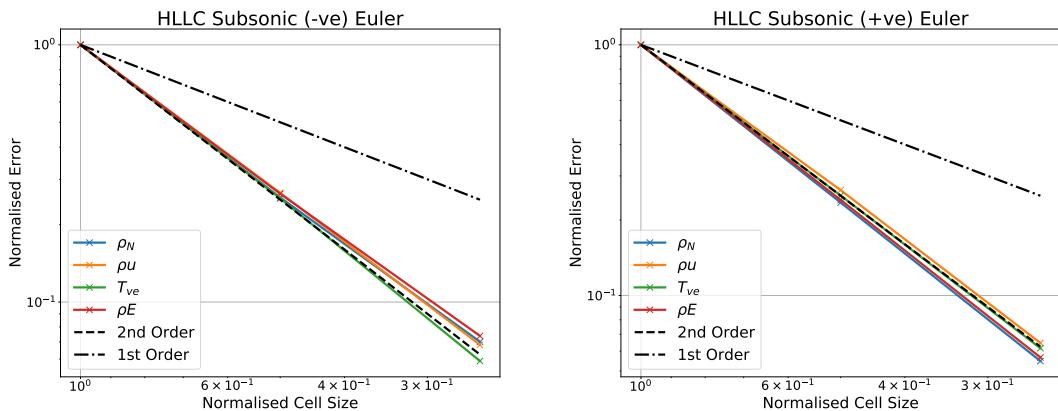


Figure A.3: The normalised error convergence for the subsonic HLLC fluxes, compared with the expected convergence for first- and second-order-accuracy.

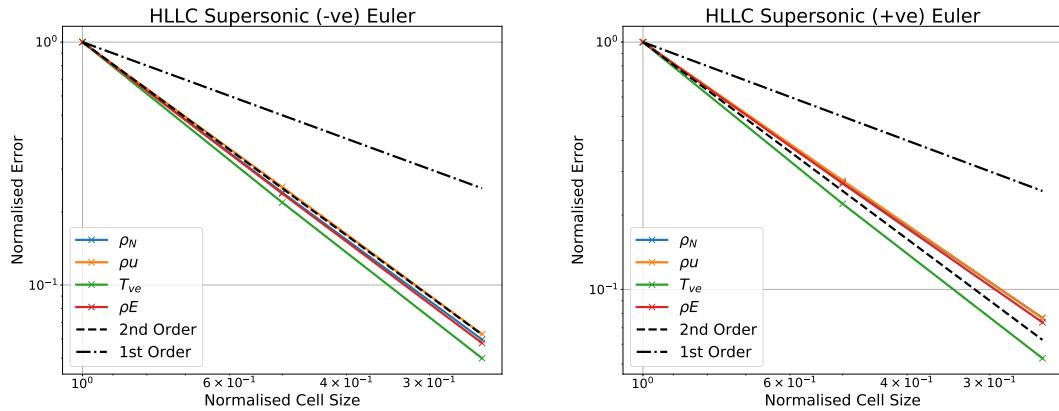


Figure A.4: The normalised error convergence for the supersonic HLLC fluxes, compared with the expected convergence for first- and second-order-accuracy.

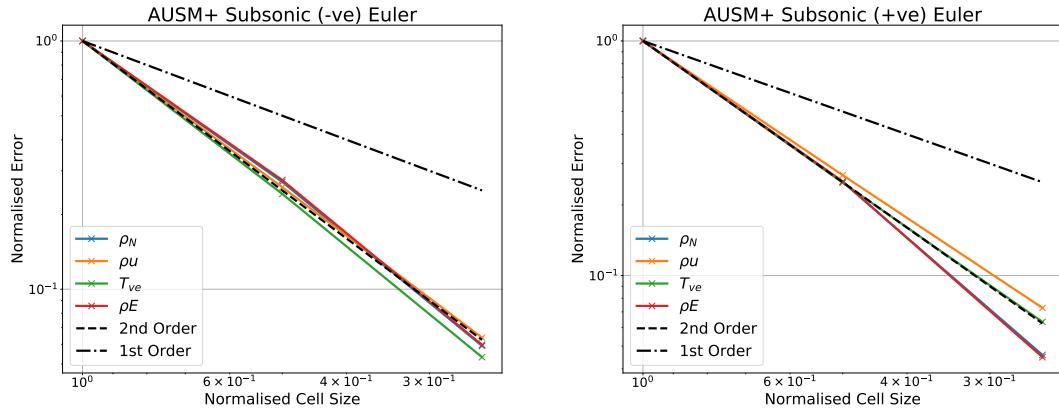


Figure A.5: The normalised error convergence for the subsonic AUSM+ fluxes, compared with the expected convergence for first- and second-order-accuracy.

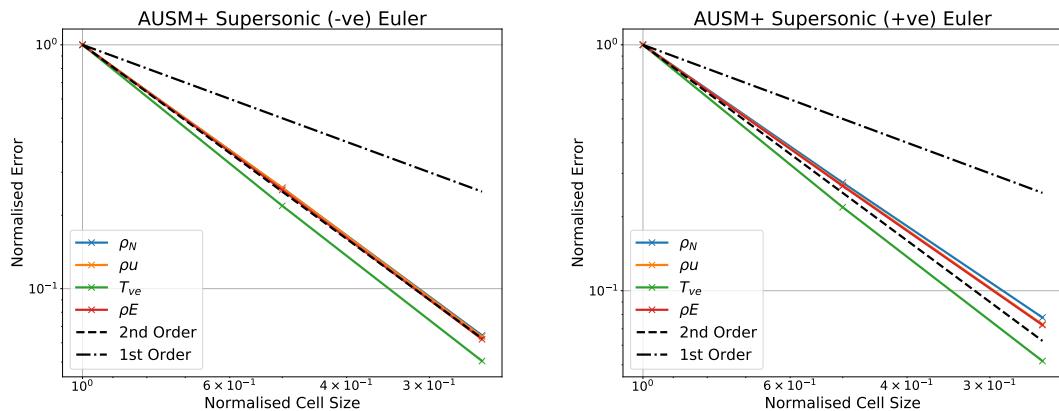


Figure A.6: The normalised error convergence for the supersonic AUSM+ fluxes, compared with the expected convergence for first- and second-order-accuracy.

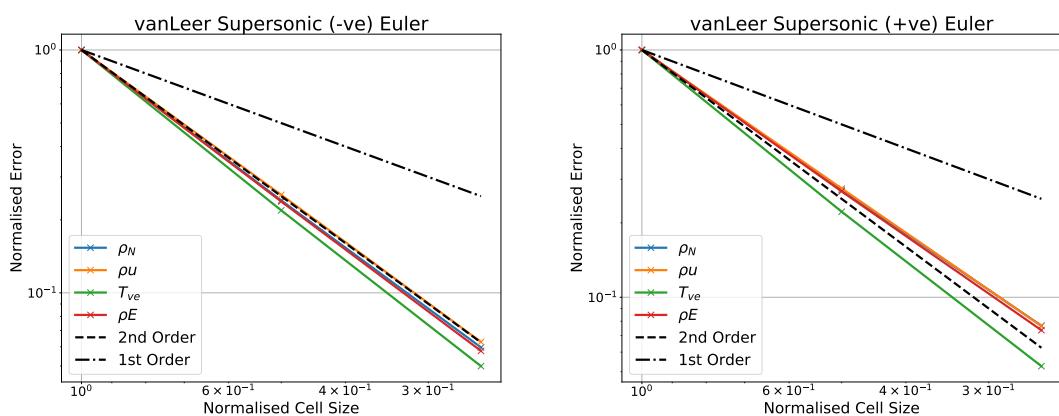


Figure A.7: The normalised error convergence for the supersonic van Leer fluxes, compared with the expected convergence for first- and second-order-accuracy.

A.3 Block Tri-diagonal Solution Method

A block tri-diagonal system is solved when using dimensional splitting or as part of the line-relaxation method. In this case, the LHS matrix A consists of lower, central and upper block diagonals, that are adjacent to each other

$$A = A^L + A^D + A^U. \quad (\text{A.1})$$

This block tri-diagonal system can be solved efficiently and directly using block LU factorisation [281]. In this method, the block tri-diagonal matrix is split into two matrices,

$$A = LU, \quad (\text{A.2})$$

where

$$L = \begin{bmatrix} I & & & \\ L_2 & I & & \\ & \ddots & \ddots & \\ & & L_{N_c-1} & I \\ & & & L_{N_c} & I \end{bmatrix}, \quad (\text{A.3})$$

and

$$U = \begin{bmatrix} D_1 & A_1^U & & \\ & D_2 & A_2^U & \\ & & \ddots & \ddots & \\ & & & D_{N_c-1} & A_{N_c-1}^U \\ & & & & D_{N_c} \end{bmatrix}. \quad (\text{A.4})$$

The matrices L_k and D_k are found using

$$D_1 = A_1^D, \quad (\text{A.5})$$

$$L_k = A_k^L [D_{k-1}]^{-1} \quad \text{for } k = 2, \dots, N_c, \quad (\text{A.6})$$

$$D_k = A_k^D - L_k A_k^U \quad \text{for } k = 2, \dots, N_c. \quad (\text{A.7})$$

The solution vector can then be found using the standard method for LU factorisation, adapted for a block matrix,

$$Ly = \mathbf{b}, \quad (\text{A.8})$$

$$Ux = \mathbf{y}, \quad (\text{A.9})$$

where the forward substitution is given by

$$\mathbf{y}_1 = \mathbf{b}_1 \quad (\text{A.10})$$

$$\mathbf{y}_k = \mathbf{b}_k - L_k \mathbf{y}_{k-1} \quad \text{for } k = 2, \dots, N_c, \quad (\text{A.11})$$

and the back substitution is given by

$$\mathbf{x}_N = [D_{N_c}]^{-1} \mathbf{y}_{N_c}, \quad (\text{A.12})$$

$$\mathbf{x}_k = [D_k]^{-1}(\mathbf{y}_k - A_k^U \mathbf{y}_{k+1}) \quad \text{for } k = (N_c - 1), \dots, 1. \quad (\text{A.13})$$

This block tri-diagonal solution method has been implemented within AMROC using efficient BLAS and LAPACK routines for the matrix inversions and matrix-vector products.

A.4 Point Hash Implementation

In this work LSH was used to significantly reduce the number of unnecessary donor searches on the near-body domain. The hash table used for the LSH was created using the `unordered_map` that is part of the C++ standard library. This requires a hash functor and comparison functor that is used to check for hash collisions. Listing A.1 shows how these functors were implemented.

```

template<int N, typename T>
struct PointHasher{
    std::size_t operator()(const Vector<T,N>& point) const
    {
        // Points within tolerance produce same hash:
        double tol = 1e8;
        // Seed for the combined hash
        std::size_t seed = 0;
        for (unsigned int n=0; n<N; n++){
            std::size_t point_uint = round(point[n]*tol);
            // Taken from boost::hash_combine
            seed ^= point_uint + 0x9e3779b9 + (seed<<6) + (seed>>2);
        }
        return seed;
    }
};

template<int N, typename T>
struct MyPointEqual{
    bool operator()(const Vector<T,N>& point1, const Vector<T,N>& point2) const {
        for (int n = 0; n != N; n++){
            if(fabs(point1[n] - point2[n]) > 1e-8){
                return false;
            }
        }
        return true;
    }
};

```

Listing A.1: The receptor point hash and comparison functors.

A.5 Blasius Boundary Layer Comparison

The overset strand solver was used to simulate an incompressible viscous flow over a flat plate. The results were verified using the analytic Blasius boundary layer solution. This solution is obtained by substituting

$$\xi = x, \quad \eta = y\sqrt{\frac{u_\infty}{\nu x}} \quad (\text{A.14})$$

and the stream function

$$\psi = \sqrt{\nu u_\infty x} f(\eta) \quad (\text{A.15})$$

into the incompressible equations for flow over a flat plate

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \quad (\text{A.16})$$

$$u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = \nu \frac{\partial^2 u}{\partial x^2}, \quad (\text{A.17})$$

$$\frac{\partial p}{\partial y} = 0, \quad (\text{A.18})$$

where y is the distance above the flat plate, x is the distance from the start of the flat plate, u_∞ is the free stream velocity and ν is the kinematic viscosity of the fluid. Using the definitions

$$u = \frac{\partial \psi}{\partial y} \quad (\text{A.19})$$

and

$$v = -\frac{\partial \psi}{\partial x} \quad (\text{A.20})$$

the substitution of Eqs. (A.14) and (A.15) into the momentum equation, Eq. (A.17), results in the ODE

$$f''' + \frac{1}{2} f'' f = 0, \quad (\text{A.21})$$

where ' denotes the derivative of f with respect to η .

Given boundary conditions for $f(0)$, $f'(0)$ and $f''(0)$, Eq. (A.21) can be integrated numerically from $\eta = 0$ to a distance away from the wall. Using Eq. (A.19), the analytic x -velocity boundary layer profile is given by $f' = u/u_\infty$ and therefore the boundary condition at the wall is given by

$$f'(0) = 0. \quad (\text{A.22})$$

Using Eq. (A.20), the y -velocity is given

$$v = \frac{1}{2} \sqrt{\frac{\nu u_\infty}{x}} [\eta f'(\eta) - f(\eta)]. \quad (\text{A.23})$$

Applying the no-slip condition at the wall, $f'(0) = 0$, gives the y -velocity at a distance x from the leading edge as

$$v = -\frac{1}{2} \sqrt{\frac{\nu u_\infty}{x}} f(\eta). \quad (\text{A.24})$$

For a standard no-slip wall, v is equal to zero, giving $f(0) = 0$. However, the boundary condition at $f(0)$ can be used to obtain an analytic velocity profiles for a boundary layer that includes blowing or suction at the wall. Using negative values of $f(0)$ results in wall blowing and positive values results in suction.

A third boundary condition is required for $f''(0)$, but cannot be obtained directly. However, far from the wall, the x -velocity is equal to the free stream velocity so

$$f'(\infty) = 1. \quad (\text{A.25})$$

This condition can be used along with an initial guess of $f''(0)$ to numerically obtain the correct value of $f''(0)$, and thus the analytic boundary layer profile, using the shooting method.

Overset strand simulations were carried out using the mesh shown in Fig. A.8. The strand mesh has been constructed with a small spacing in the x -direction at the start of the no-slip region and a spacing of $10 \mu\text{m}$ in the y -direction along the whole of the 0.3048 m plate. The simulation was carried out using a pure nitrogen flow and run until the boundary layer profile converged. Three different simulations were conducted where the blowing rate at the wall was calculated using $f(0) = 0$, $f(0) = -0.4$ and $f(0) = -0.8$. The Blasius boundary layer equation was solved numerically using a shooting method for the no blowing condition, $f(0) = 0$, and the blowing wall conditions, $f(0) = -0.4$ and $f(0) = -0.8$.

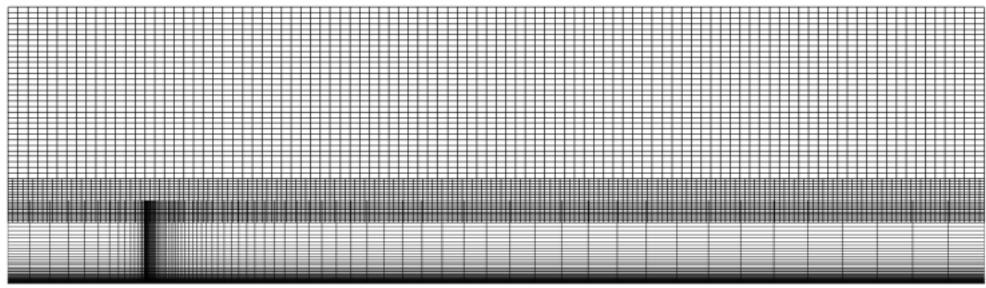


Figure A.8: The overset strand mesh used in the Blasius boundary layer simulations.

A comparison of the velocity profiles at $x = 0.15 \text{ m}$ is shown in Fig. A.9. One can see that excellent agreement is obtained for the no blowing and low blowing case. For the higher blowing case there is slightly larger disagreement, with the AMROC simulation slightly over-predicting the velocity. However, the agreement is still reasonable and the high blowing results are similar to those reported in the literature

[180, 271]. This verifies the implementation of the no-slip and blowing wall boundary conditions within the AMROC solver.

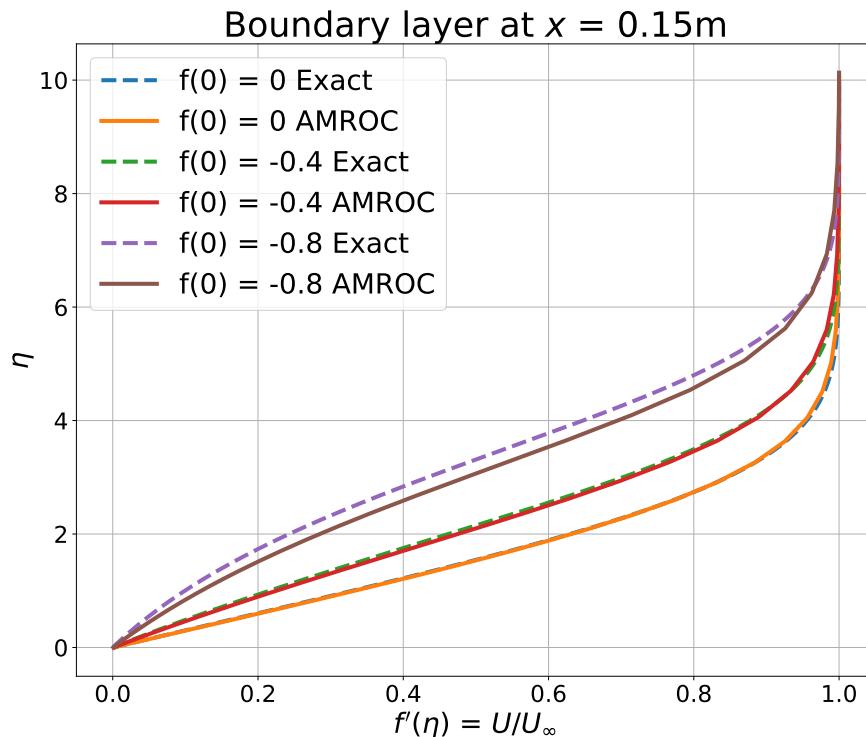


Figure A.9: A comparison between the AMROC strand mesh boundary layer profile and the Blasius boundary layer profile for three different blowing wall conditions.

A.6 Compressible Flat Plate Comparison

Incompressible flow over a flat plate using the strand solver has been compared to the analytic Blasius boundary-layer profile (see Section 3.3.3 and A.5). The profile of a compressible boundary layer is different to an incompressible boundary layer as it includes changes in the temperature and density through the boundary layer. In addition, the boundary layer profile is strongly linked to the wall boundary condition. To verify the isothermal and adiabatic boundary conditions for compressible flow simulations, supersonic flow over a flat plate was simulated using AMROC and a third party solver.

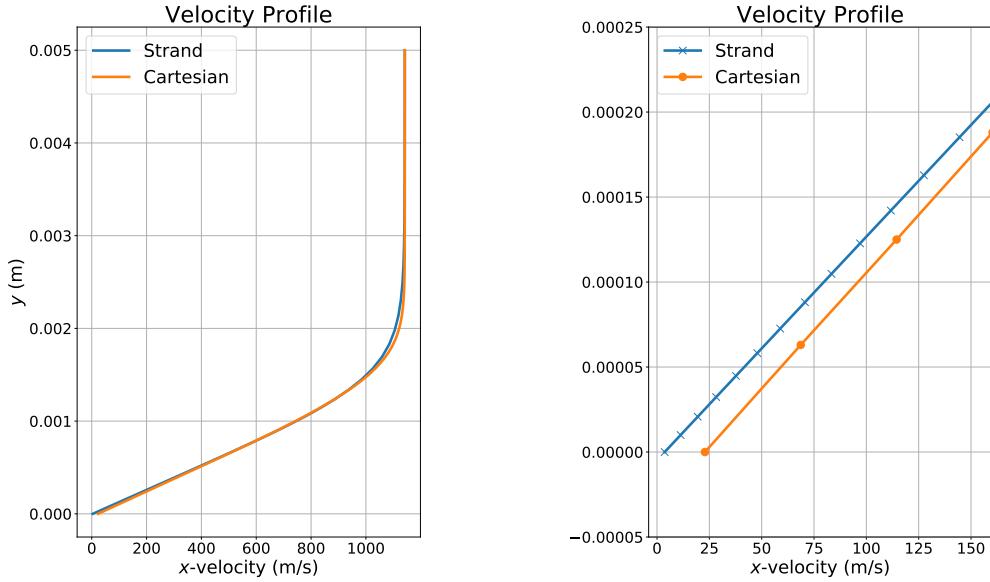
First, a flow of pure nitrogen at Mach 3 was simulated using a strand domain and an unmapped SAMR domain, both of dimensions 0.1 m by 0.25 m. Along the lower boundary a slip region of 0.03 m was included prior to the start of the no-slip wall boundary condition. Both domains used a base mesh of 100 cells in the wall-normal direction and 50 cells in the x -direction. In the SAMR simulation, two levels of refinement were permitted, with each level having a refinement factor of 4.

Pre-specified refinement blocks were used to refine the SAMR mesh in the wall region, giving an initial wall spacing of $62.5 \mu\text{m}$, with a total cell count of approximately 75,000 cells. For the strand mesh, the initial spacing was set to be $10 \mu\text{m}$, and the total cell count was 5,000 cells. The inflow conditions for the simulation are shown in Table A.8 and an adiabatic wall boundary condition was used in both simulations.

Mass Frac. N_2	Mass Frac. N	T_∞	p_∞	U_∞	M_∞
1.0	0.0	350 K	5.0 kPa	1143.68 m/s	3.0

Table A.8: Freestream conditions for the flat plate test case.

A comparison of the boundary layer velocity profiles is shown in Fig. A.10a and a magnified view of the area close to the wall is shown in Fig. A.10b. One can see that the two simulations are generally in good agreement, further verifying the mapped and Cartesian implementations. However, close to the wall the lower resolution of the Cartesian solver is not able to accurately capture the strong gradients, and overpredicts the velocity at the wall when compared to the strand domain. This would likely lead to the Cartesian simulation incorrectly predicting the surface shear stresses and heat fluxes. These results demonstrate the importance of adequately resolving the flow near the wall. The results also show that the strand mesh is able to obtain a higher near-wall resolution with a cell count that is 15 times lower than the SAMR mesh as it is able to stretch away from the wall. This highlights advantage of using a strand mesh in the near-wall region even when the boundary is aligned with the Cartesian domain.



- (a) The boundary velocity layer profile over a flat plate computed using a Cartesian and strand mesh.
 (b) A comparison of the Cartesian and strand boundary layer velocity profile close to the wall.

Figure A.10: A comparison of the boundary layer velocity profile using a strand and Cartesian domain.

In order to compare the AMROC solver with an independent solver, the results from the mapped AMROC computation were compared with results from an extended version of the unstructured solver SU² [76]. This version of SU² also uses the Mutation++ library to determine the nonequilibrium flow variables and contains the AUSM flux scheme. The same mesh used in the AMROC strand mesh simulation was used in the SU² simulation and it was run to steady-state. Both the adiabatic and isothermal boundary conditions were tested in both codes, with the isothermal wall temperature set to 350 K.

A comparison of the velocity and temperature boundary layers at $x = 0.2$ m are shown for both types of boundary condition in Figs. A.11 to A.14. One can see that excellent agreement is obtained between the two codes for both the velocity and temperature boundary layer profiles, for both boundary conditions. One can see that the level of disagreement is never greater than 1% through the whole boundary layer. This is a remarkable level of agreement, considering that the SU² solver is an unstructured dual-mesh solver. Again, this provides excellent verification evidence for the strand mesh and viscous boundary condition implementation within AMROC.

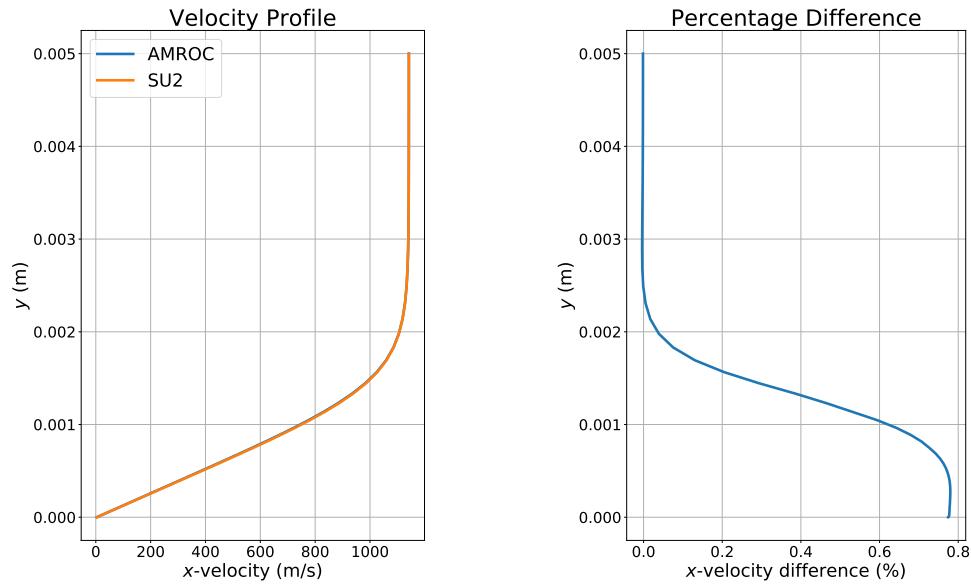


Figure A.11: A comparison of the boundary layer velocity profiles over an adiabatic flat plate, where $M_\infty = 3.0$.

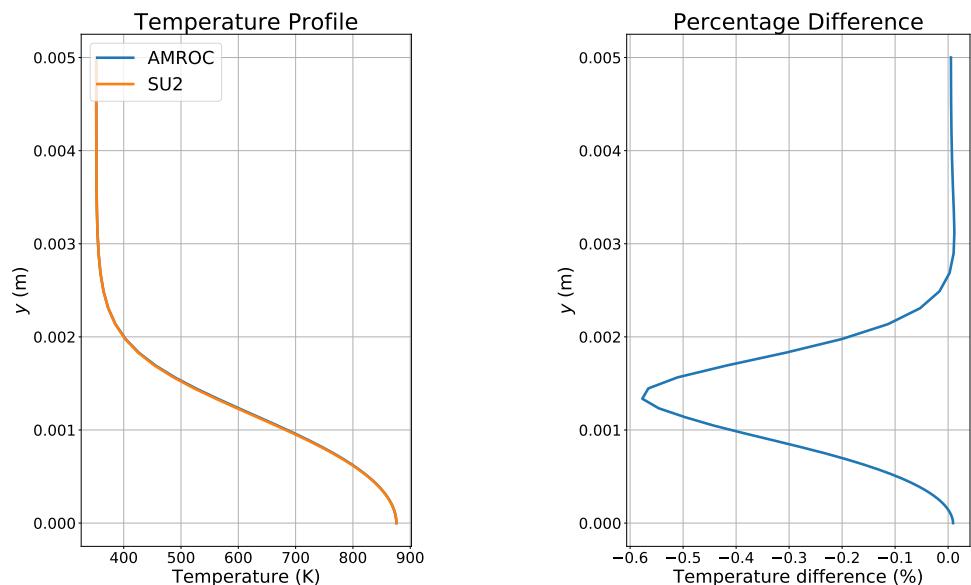


Figure A.12: A comparison of the boundary layer temperature profile over an adiabatic flat plate, where $M_\infty = 3.0$.

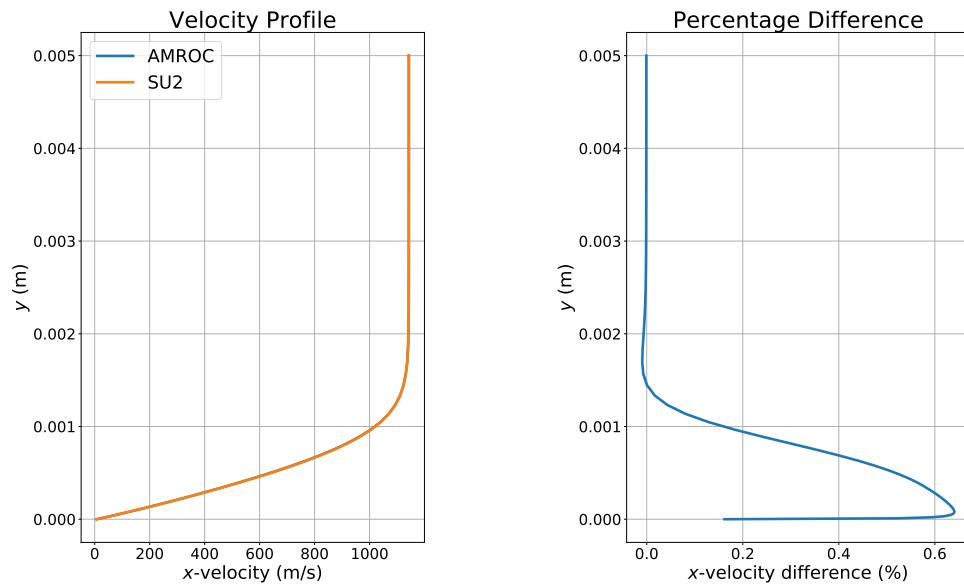


Figure A.13: A comparison of the boundary layer velocity profile over an isothermal flat plate, where $M_\infty = 3.0$.

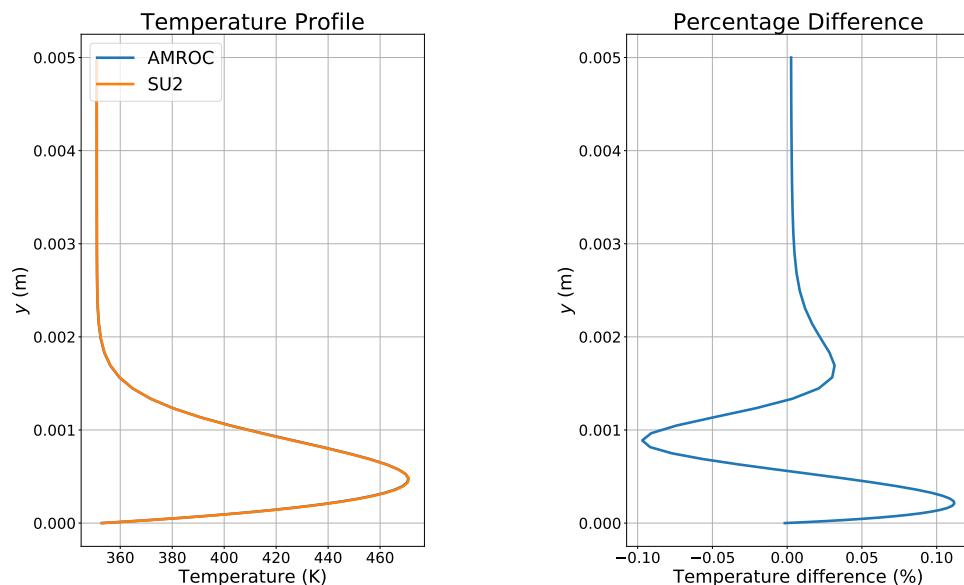


Figure A.14: A comparison of the boundary layer temperature profile over an isothermal flat plate, where $M_\infty = 3.0$.

Glossary

AABB Axis-Aligned Bounding Box.

AAIM Axis-Aligned Inverse Map.

ADT Alternating Digital Tree.

ALE Arbitrary Lagrangian-Euler.

AMR Adaptive Mesh Refinement.

AMROC Adaptive Mesh Refinement in Object-oriented C++.

AUSM Advection Upstream Splitting Method.

BLAS Basic Linear Algebra Subprograms.

CAMR Cartesian Adaptive Mesh Refinement.

CFD Computational Fluid Dynamics.

CFL Courant-Friedrichs-Lowy.

CPT Closest Point Transform.

DLR Deutsches Zentrum für Luft- und Raumfahrt.

DPLR Data Parallel Line Relaxation.

DSMC Direct Simulation Monte Carlo.

ESA European Space Agency.

FHO Forced Harmonic Oscillator.

FSI Fluid-Structure Interaction.

GMRES Generalized Minimal Residual.

HEG High Enthalpy Shock Tunnel Göttingen.

HLL Harten-Lax-van Leer.

HLLC Harten-Lax-van Leer-Contact.

IMEX Implicit/Explicit.

LAPACK Linear Algebra Package.

LAURA Langley Aerothermodynamic Upwind Algorithm.

LeMANS The Michigan Aerothermodynamic Navier-Stokes solver.

LHS Left Hand Side.

LSH Locality Sensitive Hashing.

LU Lower-Upper.

LU-SGS Lower-Upper Symmetric-Gauss-Seidel.

MMS Method of Manufactured Solutions.

MUSCL Monotonic Upstream-centered Scheme for Conservation Laws.

OBB Oriented Bounding Box.

ODE Ordinary Differential Equation.

OIM Oriented Inverse Map.

ORESD Operator Recovery Error Source Detector.

PANT Passive Nosetip Technology.

PICA Phenolic Impregnated Carbon Ablator.

PSD Power Spectral Density.

QCT Quasi-Classical Trajectory.

RBF Radial Basis Function.

RHS Right Hand Side.

RMS Root-Mean-Square.

RRHO Rigid-Rotator Harmonic Oscillator.

SAMR block-Structured Adaptive Mesh Refinement.

SBLI Shock-Boundary-Layer Interaction.

SGS Symmetric Gauss-Seidel.

SSH Schwartz, Slawsky and Herzfeld.

SSP Strong Stability Preserving.

TAMR Tree-based Adaptive Mesh Refinement.

TPS Thermal Protection System.

TVD Total Variation Diminishing.

UAMR Unstructured Adaptive Mesh Refinement.

US3D UnStructured 3D.

WENO Weighted Essentially Non-Oscillatory.

References

- [1] M. J. Abbott, L. Cooper, T. J. Dahm, and M. D. Jackson. Passive Noisetip Technology (PANT) Program: Unsteady flow on ablated nosetip shapes - PANT Series G test and analysis report. Technical Report December, 1973.
- [2] I. V. Adamovich. Three-dimensional analytic model of vibrational energy transfer in molecule-molecule collisions. *AIAA Journal*, 39(10), 2001.
- [3] M. Adams, P. Colella, D. T. Graves, J. N. Johnson, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo software package for AMR applications - design document. Technical Report LBNL-6616E, Lawrence Berkeley National Laboratory, 2019.
- [4] M. Aftosmis. Cart3D documentation.
<https://www.nas.nasa.gov/publications/software/docs/cart3d/>, 2015.
[Online; accessed 1-November-2019].
- [5] M. J. Aftosmis. Solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries. *28th Computational Fluid Dynamics Lecture Series*, page 108, 1997.
- [6] K. Ajmani, W.-F. Ng, and M.-S. Liou. Preconditioned conjugate gradient methods for the Navier-Stokes Equations, 1994.
- [7] H. Alkandry, I. D. Boyd, and A. Martin. Comparison of models for mixture transport properties for numerical simulations of ablative heat shields. In *51st AIAA Aerospace Sciences Meeting*, number January. AIAA, 2013.
- [8] J. J. Alonso, S. Hahn, F. Ham, M. Herrmann, G. Kalitzin, G. Iaccarino, P. Legresley, K. Mattsson, G. Medic, P. Moin, H. Pitsch, J. Schluter, M. Svärd, E. Van der Weide, D. You, and X. Wu. CHIMPS: A high performance scalable module for multi-physics simulations. In *42nd AIA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit*, number July, 2006.
- [9] J. D. Anderson. *Hypersonic And High Temperature Gas Dynamics*. AIAA Education Series, second edition, 2006.

- [10] D. A. Andrienko and I. D. Boyd. Vibrational relaxation and dissociation of oxygen in molecule-atom collisions. In *45th AIAA Thermophysics Conference*, number June, pages 1–19, 2015.
- [11] B. Armaly and K. Sutton. Viscosity of multicomponent partially ionized gas mixtures. In *15th Thermophysics Conference*, page 1495, 1980.
- [12] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton. Implicit-Explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.
- [13] C. W. C. Atkins and R. Deiterding. Simulations of flow in thermochemical nonequilibrium using Adaptive Mesh Refinement. In *7th European Conference on Computational Fluid Dynamics (ECFD 7)*, number 7, 2018.
- [14] M. A. Badr and D. D. Knight. Shock wave laminar boundary layer interaction over a double wedge in a high mach number flow. In *52nd Aerospace Sciences Meeting*, page 1136, 2014.
- [15] C. Ballesteros. A parallel Adaptive Mesh Refinement library for Cartesian meshes. *Phd Thesis*, (August), 2019.
- [16] J. W. Banks, W. D. Henshaw, and D. W. Schwendeman. Deforming composite grids for solving fluid structure problems. *Journal of Computational Physics*, 231(9):3518–3547, 2012.
- [17] J. T. Batina. Unsteady Euler airfoil solutions using unstructured dynamic meshes. *AIAA Journal*, 28(8):1381–1388, 1990.
- [18] F. Ben Ameur and A. Lani. R-Adaptive algorithms for high-speed flows and plasma simulations. *Computer Physics Communications*, 261:107700, 2021.
- [19] J. Benek, P. Buning, and J. Steger. A 3-D chimera grid embedding technique. In *7th Computational Physics Conference*, page 1523, 1985.
- [20] J. A. Benek, J. L. Steger, F. C. Dougherty, and P. G. Buning. Chimera: A grid-embedding technique. Technical Report AEDC-TR-85-64, Arnold Engineering Development Center, 1986.
- [21] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.
- [22] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.
- [23] J. J. Bertin and R. M. Cummings. Critical hypersonic aerothermodynamic phenomena. *Annual Review of Fluid Mechanics*, 38:129–157, 2006.

- [24] F. G. Blottner, M. Johnson, and M. Ellis. Chemically reacting viscous flow program for multi-component gas mixtures. Technical report, Sandia Labs., Albuquerque, N. Mex., 1971.
- [25] E. P. Boden. *An Adaptive Gridding Technique for Conservation Laws on Complex Domains*. PhD thesis, Cranfield University, 1997.
- [26] F. Bonelli, G. Colonna, and G. Pascazio. State-to-state vs multi-temperature models for simulating hypersonic flows. In *International Conference on Flight Vehicles, Aerothermodynamics and Re-entry Missions*, 2019.
- [27] J. Bonet and J. Peraire. An Alternating Digital Tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31(1):1–17, 1991.
- [28] A. Bonfiglioli, R. Paciorri, and A. Di Mascio. The role of mesh generation, adaptation, and refinement on the computation of flows featuring strong shocks. *Modelling and Simulation in Engineering*, 2012(May 2014), 2012.
- [29] M. Born and R. Oppenheimer. Zur quantentheorie der molekeln. *Annalen der Physik*, 389(20):457–484, 1927.
- [30] M. J. Brazell, J. Sitaraman, and D. J. Mavriplis. An overset mesh approach for 3D mixed element high-order discretizations. *Journal of Computational Physics*, 322:33–51, 2016.
- [31] T. Bridel-Bertomeu. Immersed boundary conditions for hypersonic flows using ENO-like least-square reconstruction. *Computers & Fluids*, 215:104794, Jan. 2021.
- [32] K. D. Brislawn, D. L. Brown, G. S. Chesshire, and J. S. Saltzman. Adaptively-refined overlapping grids for the numerical solution of systems of hyperbolic conservation laws. Technical report, Los Alamos National Laboratory, 1995.
- [33] D. L. Brown, S. Chesshire, Geoffrey, W. D. Henshaw, and D. J. Quinlan. Overture: An object-oriented software system for solving partial differential equations in serial and parallel environments. In *Proceedings of Eighth SIAM Conference on Parallel Processing for Scientific Computing*, 1997.
- [34] P. Buchmüller and C. Helzel. Improved accuracy of high-order WENO finite volume methods on Cartesian grids. *Journal of Scientific Computing*, 61(2):343–368, 2014.
- [35] H. J. Bungartz, M. Mehl, T. Neckel, and T. Weinzierl. The PDE framework Peano applied to fluid dynamics: An efficient implementation of a parallel multiscale fluid dynamics solver on octree-like adaptive Cartesian grids. *Computational Mechanics*, 46(1):103–114, 2010.

- [36] C. Burstedde, L. C. Wilcox, and O. Ghattas. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.
- [37] L. Campoli, G. P. Oblapenko, and E. V. Kustova. KAPPA: Kinetic approach to physical processes in atmospheres library in C++. *Computer Physics Communications*, 236:244–267, 2019.
- [38] G. V. Candler. Rate dependent energetic processes in hypersonic flows. *Progress in Aerospace Sciences*, 72:37–48, 2015.
- [39] G. V. Candler, M. D. Barnhardt, T. W. Drayna, I. Nompelis, D. M. Peterson, and P. Subbareddy. Unstructured grid approaches for accurate aeroheating simulations. In *18th AIAA Computational Fluid Dynamics Conference*, number June, 2007.
- [40] G. V. Candler, B. Johnson, Heath, I. Nompelis, P. K. Subbareddy, T. W. Drayna, V. Gidzak, and M. D. Barnhardt. Development of the US3D code for advanced compressible and reacting flow simulations. In *53rd AIAA Aerospace Sciences Meeting*, 2015.
- [41] G. V. Candler and R. W. MacCormack. Computation of weakly ionized hypersonic flows in thermochemical nonequilibrium. *Journal of Thermophysics and Heat Transfer*, 5(3):266–273, 1991.
- [42] M. Capitelli and S. Longo. Collision integrals of high-temperature air species. *Journal of Thermophysics and Heat Transfer*, 14(2), 2000.
- [43] V. Casseau, R. C. Palharini, T. J. Scanlon, and R. E. Brown. A two-temperature open-source CFD model for hypersonic reacting flows, part one: Zero-dimensional analysis. *Aerospace*, pages 1–21, 2016.
- [44] V. Casseau, T. J. Scanlon, and R. E. Brown. Development of a two-temperature open-source CFD model for hypersonic reacting flows. In *20th AIAA International Space Planes and Hypersonic Systems and Technologies Conference*, pages 1–14, 2015.
- [45] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. *International Journal for Numerical Methods in Fluids*, 25(4):475–491, 1997.
- [46] W. Chan. Developments in strategies and software tools for overset structured grid generation and connectivity. In *20th AIAA Computational Fluid Dynamics Conference*, pages 1–19, 2011.
- [47] W. M. Chan and S. A. Pandya. Advances in distance-based hole cuts on overset grids. In *22nd AIAA Computational Fluid Dynamics Conference*, pages 1–19, 2015.

- [48] S. Chapman and T. G. Cowling. *The mathematical theory of non-uniform gases*. Cambridge university press, 1970.
- [49] R. S. Chaudhry and G. V. Candler. Development of a vibration-dissociation model for high-enthalpy air. In *International Conference on Flight Vehicles, Aerothermodynamics and Re-entry Missions*, 2019.
- [50] F. M. Cheatwood and P. A. Gnoffo. User manual for the Langley Aerothermodynamic Upwind Relaxation Algorithm (LAURA). Technical Report April, NASA, 1996.
- [51] Y.-K. Chen, F. S. Milos, and T. Gokcen. Loosely coupled simulation for two-dimensional ablation and shape change. *Journal of Spacecraft and Rockets*, 47, 2010.
- [52] G. Chesshire and W. D. Henshaw. Composite overlapping meshes for the solution of partial differential equations. *Journal of Computational Physics*, 90(1):1–64, 1990.
- [53] G. Chesshire and W. D. Henshaw. A scheme for conservative interpolation on overlapping grids. *SIAM Journal on Scientific Computing*, 15(4):819–845, 1994.
- [54] F. Cirak, R. Deiterding, and S. P. Mauch. Large-scale fluid-structure interaction simulation of viscoplastic and fracturing thin-shells subjected to shocks and detonations. *Computers and Structures*, 85(11-14):1049–1065, 2007.
- [55] C. Colavolpe, F. Voitus, and B. Pierre. RK-IMEX HEVI schemes for fully compressible atmospheric models with advection: Analyses and numerical testing. *Quarterly Journal of the Royal Meteorological Society*, (April):1336–1350, 2017.
- [56] T. F. Coleman and J. J. More. Estimation of sparse Jacobian matrices and graph coloring. *SIAM Journal on Numerical Analysis*, 20(1):187–209, 1983.
- [57] S. R. Copeland. *A Continuous Adjoint Formulation For Hypersonic Flows In Thermochemical Non-Equilibrium*. PhD thesis, Stanford University, 2015.
- [58] J. Crabbill, A. Jameson, and J. Sitaraman. A high-order overset method on moving and deforming Grids. In *AIAA Modeling and Simulation Technologies Conference*, number June, 2016.
- [59] D. D'Ambrosio. A study on Shock Wave-Boundary Layer Interactions in high-speed flows. In *Fourth Symposium on Aerothermodynamics for Space Vehicles*, 2002.
- [60] A. De Boer, A. Van Zuijlen, and H. Bijl. Radial basis functions for interface interpolation and mesh deformation. In *Advanced Computational Methods in Science and Engineering*, pages 143–178. Springer, 2009.

- [61] G. Degrez, A. Lani, M. Panesi, O. Chazot, and H. Deconinck. Modelling of high-enthalpy, high-Mach number flows. *Journal of Physics D: Applied Physics*, 42, 2009.
- [62] R. Deiterding. *Parallel Adaptive Simulation of Multi-Dimensional Detonation Structures*. PhD thesis, 2003.
- [63] R. Deiterding. A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains. *Computers & Structures*, 87:769–783, 2009.
- [64] R. Deiterding. Block-structured adaptive mesh refinement: Theory, implementation and application. *ESAIM Proceedings*, 34:97–150, 2011.
- [65] R. Deiterding, M. O. Domingues, and K. Schneider. Multiresolution analysis as a criterion for effective dynamic mesh adaptation – A case study for Euler equations in the SAMR framework AMROC. *Computers and Fluids*, 205:104583, 2020.
- [66] R. Deiterding, R. Radovitzky, S. P. Mauch, L. Noels, J. C. Cummings, and D. I. Meiron. A virtual test facility for the efficient simulation of solid material response under strong shock and detonation wave loading. *Engineering with Computers*, 22(3-4):325–347, 2006.
- [67] R. Deiterding and S. L. Wood. Predictive wind turbine simulation with an adaptive lattice Boltzmann method for moving boundaries. *Journal of Physics: Conference Series*, 753(8), 2016.
- [68] J. Del Corso, F. Cheatwood, W. Bruce, S. Hughes, and A. Calomino. Advanced high-temperature flexible TPS for inflatable aerodynamic decelerators. In *21st AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, page 2510, 2011.
- [69] C. Desmeuzes, G. Duffa, and B. Dubroca. Different levels of modeling for diffusion phenomena in Neutral and ionized mixtures. *Journal of Thermophysics and Heat Transfer*, 11(1), 1997.
- [70] D. DeZeeuw and K. G. Powell. An adaptively refined Cartesian mesh solver for the Euler equations. *Journal of Computational Physics*, 104(1):56–68, 1993.
- [71] L. F. Diachin, R. Hornung, P. Plassmann, and A. Wissink. Parallel adaptive mesh refinement. In *Parallel processing for scientific computing*, pages 143–162. SIAM, 2006.
- [72] V. Dolean, P. Jolivet, and F. Nataf. *An introduction to domain decomposition methods: algorithms, theory and parallel implementation*. Number August 2018. 2016.

- [73] A. Dubey, A. Almgren, J. Bell, M. Berzins, S. Brandt, G. Bryan, P. Colella, D. Graves, M. Lijewski, F. Löffler, B. O'Shea, E. Schnetter, B. Van Straalen, and K. Weide. A survey of high level frameworks in block-structured adaptive mesh refinement packages. *Journal of Parallel and Distributed Computing*, 74(12):3217–3227, 2014.
- [74] G. Duffa. *Ablative Thermal Protection Systems Modeling*. 2011.
- [75] M. G. Dunn and S.-W. Kang. Theoretical and experimental studies of re-entry plasmas. Technical Report April, NASA CR-2232, 1973.
- [76] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 2016.
- [77] B. E. Edney. Effects of shock impingement on the heat transfer around blunt bodies. *AIAA Journal*, 6(1):15–21, Jan. 1968.
- [78] W. A. Engblom, B. Yuceil, D. B. Goldstein, and D. S. Dolling. Experimental and numerical study of hypersonic forward-facing cavity flow. *Journal of Spacecraft and Rockets*, 33(3):353–359, May 1996.
- [79] F. Esposito, M. Capitelli, and C. Gorse. Quasi-classical dynamics and vibrational kinetics of N + N₂ (v) system. *Chemical Physics*, 257(2-3):193–202, 2000.
- [80] E. Eucken. Ueber das wärmeleitvermogen, die spezifische wärme und die innere reibung der gase. *Phys Z.*, 14, 1913.
- [81] S. Eyi, K. M. Hanquist, and I. D. Boyd. Aerothermodynamic design optimization of hypersonic vehicles. *Journal of Thermophysics and Heat Transfer*, 33(2):392–406, 2019.
- [82] E. Farbar, H. Alkandry, J. Wiebenga, and I. D. Boyd. Simulations of ablating hypersonic vehicles with finite-rate surface chemistry. In *11th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, 2014.
- [83] C. Farhat, C. Degand, B. Koobus, and M. Lesoinne. Torsional springs for two-dimensional dynamic unstructured fluid meshes. *Computer Methods in Applied Mechanics and Engineering*, 163(1-4):231–245, 1998.
- [84] P. Fast and M. J. Shelley. A moving overset grid method for interface dynamics applied to non-Newtonian Hele-Shaw flow. *Journal of Computational Physics*, 195(1):117–142, 2004.
- [85] D. C. D. R. Fernández and D. W. Zingg. High-order compact-stencil summation-by-parts operators for the second derivative with variable coefficients. In *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, pages 1–23, 2012.

- [86] B. A. Freno, B. R. Carnes, and V. G. Weirs. Code-verification techniques for hypersonic reacting flows in thermochemical nonequilibrium. *Journal of Computational Physics*, 425:109752, 2021.
- [87] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. *Computer Methods in Applied Mechanics and Engineering*, 194(48-49):5068–5082, 2005.
- [88] B. Fryxell, K. Olson, P. Ricker, F. X. Timmes, M. Zingale, D. Q. Lamb, P. MacNeice, R. Rosner, J. W. Truran, and H. Tufo. FLASH: An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes. *The Astrophysical Journal Supplement Series*, 131(1):273–334, 2000.
- [89] M. Furudate, B. J. Lee, E. Jeong, and I.-s. Jeung. Numerical simulation of hypervelocity flow on hemisphere in T2 shock tunnel. (January):1–8, 2006.
- [90] M. A. Furudate. Calculation of shock stand-off distance for a sphere in nonequilibrium hypersonic flow. *Journal of Computational Fluids Eng.*, 17(4):69–74, 2012.
- [91] W. C. Gardiner. *Combustion chemistry*. Springer New York etc., 1984.
- [92] A. H. Gebremedhin, F. Manne, and A. Pothen. What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.
- [93] D. Giordano. Impact of the Born-Oppenheimer approximation on aerothermodynamics. *Journal of thermophysics and heat transfer*, 21(3):647–657, 2007.
- [94] V. Giovangigli, B. Graillle, T. Magin, and M. Massot. Multicomponent transport in weakly ionized mixtures. *Plasma Sources Science and Technology*, 19:1–6, 2010.
- [95] P. A. Gnoffo. Computational fluid dynamics technology for hypersonic applications. *AIAA\ICAS International Air and Space Symposium and Exposition: The Next 100 Years*, 2003.
- [96] P. A. Gnoffo and F. M. Cheatwood. User's manual for the Langley Aerothermodynamic Upwind Relaxation Algorithm (LAURA). 1996.
- [97] P. A. Gnoffo, R. N. Gupta, and J. L. Shinn. Conservation equations and physical models for hypersonic air flows in thermal and chemical nonequilibrium. page 58, 1989.
- [98] P. A. Gnoffo, K. J. Weilmuenster, H. H. Hamilton, D. R. Olynick, and E. Venkatapathy. Computational aerothermodynamic design issues for hypersonic vehicles. *Journal of Spacecraft and Rockets*, 36(1):21–43, 1999.

- [99] P. A. Gnoffo and J. A. White. Computational aerothermodynamic simulation issues on unstructured grids. In *37th AIAA Thermophysics Conference*, number July, pages 1–17, 2004.
- [100] S. Godunov and I. Bohachevsky. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Matematicheskij sbornik*, 47(3):271–306, 1959.
- [101] T. Gokcen, Y.-K. Chen, K. A. Skokova, and F. S. Milos. Computational analysis of arc-jet wedge tests including ablation and shape change. In *10th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, 2010.
- [102] T. Gökçen, K. Skokova, J. A. Balboni, I. Terrazas-Salinas, and D. Bose. Computational analysis of arc-jet wedge calibration tests in IHF 6-inch conical nozzle. *47th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, (January):1–16, 2009.
- [103] R. J. Gollan. *The computational modelling of high-temperature gas effects with application to hypersonic flows*. PhD thesis, University of Queensland, 2008.
- [104] T. Gombosi, D. DeZeeuw, C. Groth, and K. Powell. Magnetospheric configuration for Parker-spiral IMF conditions: Results of a 3D AMR MHD simulation. *Advances in Space Research*, 26(1):139–149, 2000.
- [105] S. Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, University of North Carolina, 2000.
- [106] B. Grossman and P. Cinella. Flux-split algorithms for flows with non-equilibrium chemistry and vibrational relaxation. *Journal of Computational Physics*, 168:131–168, 1990.
- [107] C. Groth, E. Costa, and M. E. Biancolini. RBF-based mesh morphing approach to perform icing simulations in the aviation sector. *Aircraft Engineering and Aerospace Technology*, 91(4):620–633, 2019.
- [108] R. N. Gupta, K.-P. Lee, R. A. Thompson, and J. M. Yos. Calculations and curve fits of thermodynamic and transport properties of equilibrium air to 30,000K. Technical report, NASA, 1991.
- [109] R. N. Gupta, J. M. Yos, and R. A. Thompson. A review of reaction rates and thermodynamic and transport properties for the 11-species air model for chemical and thermal nonequilibrium calculations to 30,000K. Technical Report February, NASA, 1989.
- [110] W. G. Habashi, J. Dompierre, Y. Bourgault, D. Ait-Ali-Yahia, M. Fortin, and M.-G. Vallet. Anisotropic mesh adaptation: Towards user-independent,

- mesh-independent and solver-independent CFD. Part I: General principles. *International Journal for Numerical Methods in Fluids*, 32(6):725–744, 2000.
- [111] R. Haimes. MOSS: multiple orthogonal strand system. *Engineering with Computers*, 31(3):453–463, 2015.
- [112] K. Han, Y. T. Feng, and D. R. J. Owen. Performance comparisons of tree-based and cell-based contact detection algorithms. *Engineering Computations: International Journal for Computer-Aided Engineering and Software*, 24(2):165–181, 2007.
- [113] J. Hao, J. Wang, and C. Lee. Assessment of vibration–dissociation coupling models for hypersonic nonequilibrium simulations. *Aerospace Science and Technology*, 67:433–442, 2017.
- [114] J. Hao and C. Y. Wen. Hypersonic flow over spherically blunted double cones. *Journal of Fluid Mechanics*, 896, 2020.
- [115] J. Hao, C. Y. Wen, and J. Wang. Numerical investigation of hypervelocity shock-wave/boundary-layer interactions over a double-wedge configuration. *International Journal of Heat and Mass Transfer*, 138:277–292, 2019.
- [116] A. Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. *Communications on Pure and Applied Mathematics*, 48(12):1305–1342, 1995.
- [117] D. Hash, J. Olejniczak, M. Wright, D. Prabhu, M. Pulsonetti, B. Hollis, P. Gnoffo, M. Barnhardt, I. Nompelis, and G. Candler. FIRE II calculations for hypersonic nonequilibrium aerothermodynamics code verification: DPLR, LAURA, and US3D. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, page 605, 2007.
- [118] M. Haynes. *ANITA Theory Manual*, 2012.
- [119] W. D. Henshaw and G. Chesshire. Multigrid on composite meshes. *SIAM Journal on Scientific and Statistical Computing*, 8(6):914–923, 1987.
- [120] W. D. Henshaw and D. W. Schwendeman. An adaptive numerical scheme for high-speed reactive flow on overlapping grids. *Journal of Computational Physics*, 191(2):420–447, 2003.
- [121] W. D. Henshaw and D. W. Schwendeman. Moving overlapping grids with adaptive mesh refinement for high-speed reactive and non-reactive flow. *Journal of Computational Physics*, 216(2):744–779, 2006.
- [122] W. D. Henshaw and D. W. Schwendeman. Parallel computation of three-dimensional flows using overlapping grids with adaptive mesh refinement. *Journal of Computational Physics*, 227(16):7469–7502, 2008.

- [123] J. O. Hirschfelder. Heat conductivity in polyatomic or electronically excited gases. ii. *The Journal of Chemical Physics*, 26(2):282–285, 1957.
- [124] W. J. Horne and K. Mahesh. A massively-parallel, unstructured overset method for mesh connectivity. *Journal of Computational Physics*, 376:585–596, 2019.
- [125] H. G. Hornung, R. J. Gollan, and P. A. Jacobs. Unsteadiness boundaries in supersonic flow over double cones. *Journal of Fluid Mechanics*, 916:1–23, 2021.
- [126] R. D. Hornung and S. R. Kohn. The use of object-oriented design patterns in the samrai structured amr framework. In *In Proceedings of the SIAM Workshop on Object-Oriented Methods for Inter-Operable Scientific and Engineering Computing*, pages 235–244, 1998.
- [127] D. M. Ingram, D. M. Causon, and C. G. Mingham. Developments in Cartesian cut cell methods. *Mathematics and Computers in Simulation*, 61(3-6):561–572, 2003.
- [128] L. Jameson. AMR vs high order schemes. *Journal of Scientific Computing*, 18(1):1–24, 2003.
- [129] B. John and V. Kulkarni. Effect of leading edge bluntness on the interaction of ramp induced shock wave with laminar boundary layer at hypersonic speed. *Computers & Fluids*, 96:177–190, June 2014.
- [130] A. A. Johnson and T. E. Tezduyar. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering*, 119(1-2):73–94, 1994.
- [131] A. A. Johnson and T. E. Tezduyar. Simulation of multiple spheres falling in a liquid-filled tube. *Computer Methods in Applied Mechanics and Engineering*, 134(3-4):351–373, 1996.
- [132] E. Josyula. Multiquantum state-to-state transitions in hypersonic blunt body flows. In *49th AIAA Aerospace Sciences Meeting*, number January, 2011.
- [133] E. Josyula. *Hypersonic Nonequilibrium Flows: Fundamentals and Recent Advances*. American Institute of Aeronautics and Astronautics, 2015.
- [134] S. J. Kamkar, A. M. Wissink, V. Sankaran, and A. Jameson. Feature-driven Cartesian adaptive mesh refinement for vortex-dominated flows. *Journal of Computational Physics*, 230(16):6271–6298, 2011.
- [135] A. Katz and V. Sankaran. An efficient correction method to obtain a formally third-order accurate flow solver for node-centered unstructured grids. *Journal of Scientific Computing*, 51(2):375–393, 2012.
- [136] A. Katz and A. M. Wissink. Efficient solution methods for strand grid applications. *AIAA Journal*, 52(2), 2014.

- [137] A. Katz, A. M. Wissink, V. Sankaran, R. L. Meakin, and W. M. Chan. Application of strand meshes to complex aerodynamic flow fields. *Journal of Computational Physics*, 230(17):6512–6530, 2011.
- [138] A. Katz and D. Work. High-order flux correction/finite difference schemes for strand grids. *Journal of Computational Physics*, 282:360–380, 2015.
- [139] R. Keppens, M. Nool, G. Tóth, and J. P. Goedbloed. Adaptive mesh refinement for conservative systems: Multi-dimensional efficiency evaluation. *Computer Physics Communications*, 153(3):317–339, 2003.
- [140] M. Khoshnati, G. R. Stuhne, and D. A. Steinman. Relative performance of geometric search algorithms for interpolating unstructured mesh data. *Lecture Notes in Computer Science*, 2879(PART 2):391–398, 2003.
- [141] J. G. Kim and S. M. Jo. Modification of chemical-kinetic parameters for 11-air species in re-entry flows. *International Journal of Heat and Mass Transfer*, 169, 2021.
- [142] K. H. Kim, C. Kim, and O.-h. Rho. Methods for the accurate computations of hypersonic flows. *Journal of Computational Physics*, 174:38–80, 2001.
- [143] M. Kim. Active plasma layer manipulation scheme during hypersonic flight. *Aerospace Science and Technology*, 35(1):135–142, 2014.
- [144] M. Kim, A. Guhlan, and I. D. Boyd. Modeling of electron temperature in hypersonic flows. In *49th AIAA Aerospace Sciences Meeting*, number January, 2011.
- [145] M. Kim, A. Gühan, and I. D. Boyd. Modeling of electron energy phenomena in hypersonic flows. *Journal of Thermophysics and Heat Transfer*, 26(2):244–257, 2012.
- [146] M. K. Kim, B. Esser, U. Koch, and A. Gühan. Numerical and experimental study of high enthalpy flows in a hypersonic plasma wind tunnel: L3K. In *42nd AIAA Thermophysics Conference*, number AIAA Paper 2011-3777, 2011.
- [147] K. Kitamura. Assessment of SLAU2 and other flux functions with slope limiters in hypersonic shock-interaction heating. *Computers & Fluids*, 129:134–145, Apr. 2016.
- [148] O. Knab, H.-H. Frühauf, and E. Messerschmid. Theory and validation of the physically consistent coupled vibration-chemistry-vibration model. *Journal of Thermophysics and Heat Transfer*, 9(2):219–226, 1995.
- [149] D. Knight, O. Chazot, J. Austin, M. A. Badr, G. Candler, B. Celik, D. de Rosa, R. Donelli, J. Komives, A. Lani, D. Levin, I. Nompolis, M. Panesi, G. Pezzella, B. Reimann, O. Tumuklu, and K. Yuceil. Assessment of predictive capabilities for aerodynamic heating in hypersonic flow. *Progress in Aerospace Sciences*, 90(February):39–53, 2017.

- [150] B. Kreiss. Construction a curvilinear grid. *SIAM Journal on Scientific and Statistical Computing*, 4(2):270–279, 1983.
- [151] D. W. Kuntz, B. Hassan, and D. L. Potter. Predictions of ablating hypersonic vehicles using an iterative coupled fluid/thermal approach. *Journal of Thermophysics and Heat Transfer*, 15, 2001.
- [152] E. Kustova, E. Nagnibeda, G. Oblapenko, A. Savelev, and I. Sharafutdinov. Advanced models for vibrational-chemical coupling in multi-temperature flows. *Chemical Physics*, 464:1–13, 2016.
- [153] E. V. Kustova and G. P. Oblapenko. Mutual effect of vibrational relaxation and chemical reactions in viscous multitemperature flows. *Physical Review E*, 93(3), 2016.
- [154] V. Lakshminarayan, J. Sitaraman, and A. Wissink. Sensitivity of rotorcraft hover predictions to mesh resolution in strand grid framework. *AIAA Journal*, 57(8):3173–3184, 2019.
- [155] L. Landau and E. Teller. Theory of sound dispersion. *Physikalische Zeitschrift der Sowjetunion*, 10(34):34–43, 1936.
- [156] G. Lapenta. A recipe to detect the error in discretization schemes. *International Journal for Numerical Methods in Engineering*, 59(15):2065–2087, 2004.
- [157] J.-H. Lee. Basic governing equations for the flight regimes of aeroassisted orbital transfer vehicles. *Thermal Design of Aeroassisted Orbital Transfer Vehicles*, 96:3–53, 1985.
- [158] Y. Lee and J. Baeder. Implicit hole cutting-a new approach to overset grid connectivity. In *16th AIAA Computational Fluid Dynamics Conference*, page 4128.
- [159] S. Li. Comparison of refinement criteria for structured adaptive mesh refinement. *Journal of Computational and Applied Mathematics*, 233(12):3139–3147, 2010.
- [160] M.-s. Liou. A sequel to AUSM: AUSM+. *Journal of Computational Physics*, 129:364–382, 1996.
- [161] M.-S. Liou. Ten years in the making - AUSM-family. Technical Report November, 2001.
- [162] M.-s. Liou. A sequel to AUSM, Part II : AUSM + -up for all speeds. *Journal of Computational Physics*, 214:137–170, 2006.
- [163] M.-S. Liou and C. J. Steffen Jr. A new flux splitting scheme. *Journal of Computational Physics*, 107:23–39, 1993.

- [164] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *Journal of Computational Physics*, 115(4):200–212, 1994.
- [165] K. R. Lobb. Experimental measurement of shock detachment distance on spheres fired in air at hypervelocities. *High Temperature Aspects of Hypersonic Flows*, 14(5):519–527, 1964.
- [166] S. J. Lock, N. Wood, and H. Weller. Numerical analyses of Runge-Kutta implicit-explicit schemes for horizontally explicit, vertically implicit solutions of atmospheric models. *Quarterly Journal of the Royal Meteorological Society*, 140(682):1654–1669, 2014.
- [167] R. Loehner, D. Sharov, H. Luo, and R. Ramamurti. Overlapping unstructured grids. In *39th Aerospace Sciences Meeting and Exhibit*, page 439, 2001.
- [168] M. M. Lopes, R. Deiterding, A. K. F. Gomes, O. Mendes, and M. O. Domingues. An ideal compressible magnetohydrodynamic solver with parallel block-structured adaptive mesh refinement. *Computers and Fluids*, 173:293–298, 2018.
- [169] H. Luo, J. D. Baum, and R. Löhner. On the computation of multi-material flows using ALE formulation. *Journal of Computational Physics*, 194(1):304–328, 2004.
- [170] R. W. MacCormack and G. V. Candler. The solution of the Navier-Stokes equations using Gauss-Seidel line relaxation. *Computers and Fluids*, 17(1):135–150, 1989.
- [171] S. O. Macheret and I. V. Adamovich. Semiclassical modeling of state-specific dissociation rates in diatomic gases. *The Journal of Chemical Physics*, 113(17):7351–7361, 2000.
- [172] S. O. Macheret and J. W. Rich. Nonequilibrium dissociation rates behind strong shock waves: classical model. *Chemical Physics*, 174(1):25–43, 1993.
- [173] M. MacLean, E. Mundy, T. Wadhams, M. Holden, and R. Parker. Analysis and ground test of aerothermal effects on spherical capsule geometries. In *38th Fluid Dynamics Conference and Exhibit*, page 4273, 2008.
- [174] P. MacNeice, K. M. Olson, C. Mobarry, R. De Fainchtein, and C. Packer. PARAMESH: A parallel adaptive mesh refinement community toolkit. *Computer Physics Communications*, 126(3):330–354, 2000.
- [175] T. Magin and A. Bellemans. Calculation of collision integrals for ablation species. In *8th European Conference on Aerodynamics for Space Vehicles*, 2015.
- [176] T. E. Magin. Transport algorithms for partially ionized and unmagnetized plasmas. *Journal of Computational Physics*, 198:424–449, 2004.

- [177] P. V. Marrone and C. E. Treanor. Chemical relaxation with preferential dissociation from excited vibrational levels. *The Physics of Fluids*, 6(9):1215–1221, 1963.
- [178] A. Martin and I. D. Boyd. Strongly coupled computation of material response and nonequilibrium flow for hypersonic ablation. In *41st AIAA Thermophysics Conference*, 2009.
- [179] A. Martin and I. D. Boyd. Mesh tailoring for strongly coupled computation of ablative material in nonequilibrium hypersonic flow. In *10th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, number June, 2010.
- [180] A. Martin and I. D. Boyd. Strongly coupled computation of material response and nonequilibrium flow for hypersonic ablation. *Journal of Spacecraft and Rockets*, 52(1), 2015.
- [181] D. F. Martin and P. Colella. A cell-centered adaptive projection method for the incompressible Euler equations. *Journal of Computational Physics*, 163(2):271–312, 2000.
- [182] S. Mauch. *Efficient algorithms for solving static Hamilton-Jacobi equations*. PhD thesis, California Institute of Technology, 2003.
- [183] B. J. McBride, M. J. Zehe, and S. Gordon. NASA Glenn coefficients for calculating thermodynamic properties of individual species. Technical Report September, 2002.
- [184] J. McBride and A. Reno. Coefficients thermodynamic properties for calculating and transport species of individual species. Technical report, 1993.
- [185] R. Meakin. Moving body overset grid methods for complete aircraft tiltrotor simulations. In *11th Computational Fluid Dynamics Conference*, page 3350, 1993.
- [186] R. Meakin and R. Meakin. On adaptive refinement and overset structured grids. In *13th Computational Fluid Dynamics Conference*, page 1858, 1997.
- [187] R. L. Meakin. A new method for establishing intergrid communication among systems of overset grids. In *10th Computational Fluid Dynamics Conference*, 1991.
- [188] R. L. Meakin. Composite overset structured grids. In J. F. Thompson, B. K. Soni, and N. P. Weatherill, editors, *Handbook of grid generation*, chapter 11. CRC press, 1999.
- [189] R. L. Meakin. Object x-rays for cutting holes in composite overset structured grids. In *15th AIAA Computational Fluid Dynamics Conference*, 2001.
- [190] R. L. Meakin, W. M. Chan, S. A. Pandya, and J. Sitaraman. On strand grids for complex flows. In *18th AIAA Computational Fluid Dynamics Conference*, 2007.

- [191] I. S. Men'shov and Y. Nakamura. Numerical simulations and experimental comparisons for high-speed nonequilibrium air flows. *Fluid Dynamics Research*, 27(5):305–334, 2000.
- [192] R. C. Millikan and D. R. White. Systematics of vibrational relaxation. *The Journal of chemical physics*, 39(12):3209–3213, 1963.
- [193] C. H. Mortensen and X. Zhong. Simulation of second-mode instability in a real-gas hypersonic flow with graphite ablation. *AIAA J.*, 52(August 2014):1–21, 2014.
- [194] F. Moukalled, L. Mangani, and M. Darwish. *The finite volume method in computational fluid dynamics: An Advanced Introduction with OpenFOAM and Matlab*. Springer, 2015.
- [195] A. Munafò, A. Alberti, C. Pantano, J. B. Freund, and M. Panesi. A computational model for nanosecond pulse laser-plasma interactions. *Journal of Computational Physics*, 406:109190, 2020.
- [196] S. M. Murman. Dynamic simulations of atmospheric-entry capsules. *Journal of Spacecraft and Rockets*, 46(4):829–835, 2009.
- [197] D. P. Murray. *A Study of Ice Particle Motion Through a Shock Wave*. PhD thesis, Imperial College London, 2010.
- [198] K. Nakahashi, F. Togashi, and D. Sharov. Intergrid-boundary definition method for overset unstructured grid approach. *AIAA Journal*, 38(11):2077–2084, 2000.
- [199] V. Y. Neiland. Flow behind the boundary layer separation point in a supersonic stream. *Fluid Dynamics*, 6(3):378–384, 1973.
- [200] K. Neitzel and I. D. Boyd. Influence of turbulence modeling on aftbody surface heating prediction for a hypersonic entry capsule. *44th AIAA Thermophysics Conference*, pages 1–12, 2013.
- [201] K. Neitzel, J. G. Kim, and I. D. Boyd. Nonequilibrium modeling of oxygen in reflected shock tube flows. *11th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, (June):1–13, 2014.
- [202] M. Nemec, M. J. Aftosmis, and M. Wintzer. Adjoint-based adaptive mesh refinement for complex geometries. *46th AIAA Aerospace Sciences Meeting and Exhibit*, pages 1–23, 2008.
- [203] M. Nishida and M. Matsumoto. Thermochemical nonequilibrium in rapidly expanding flows of high-temperature air. *Zeitschrift für Naturforschung A*, 52(4):358–368, 1997.

- [204] R. Noack, N. Wyman, G. McGowan, and C. Brown. Dual-grid interpolation for cell-centered overset grid systems. *AIAA Scitech 2020 Forum*, 2020.
- [205] R. W. Noack, D. A. Boger, R. F. Kunz, and P. M. Carrica. Suggar++: An improved general overset grid assembly capability. In *19th AIAA Computational Fluid Dynamics Conference*, 2009.
- [206] I. Nompelis, G. V. Candler, and R. J. Conti. A parallel implicit CFD code for the simulation of ablating re-entry vehicles. In *47th AIAA Aerospace Sciences Meeting*, number January, 2009.
- [207] I. Nompelis, T. Drayna, and G. Candler. A parallel unstructured implicit solver for hypersonic reacting flow simulation. In *Parallel Computational Fluid Dynamics 2005*, pages 389–395. Elsevier, 2006.
- [208] G. P. Oblapenko, E. V. Kustova, K. Hannemann, and V. Hannemann. Assessment of recent thermo-chemical relaxation models using the DLR-TAU code. *AIP Conference Proceedings*, 2132(August), 2019.
- [209] J. Olejniczak and G. V. Candler. Vibrational energy conservation with vibration-dissociation coupling: General theory and numerical studies. *Physics of Fluids*, 7(7):1764–1774, 1995.
- [210] R. Paciorri, M. Onofri, D. Cardillo, E. Cosson, P. Binetti, and T. Walloschek. Numerical assessment of wall catalytic effects on the IXV surface. In *6th European Symposium on Aerothermodynamics for Space Vehicles*, volume 659, page 55, 2009.
- [211] G. E. Palmer and M. J. Wright. Comparison of methods to compute high-temperature gas viscosity. *Journal of Thermophysics and Heat Transfer*, 17(2):232–239, 2003.
- [212] P. Papadopoulos, E. Venkatapathy, D. Prabhu, M. P. Loomis, and D. Olynick. Current grid-generation strategies and future requirements in hypersonic vehicle design, analysis and testing. *Applied Mathematical Modelling*, 23(9):705–735, 1999.
- [213] C. Park. Problems of rate chemistry in the flight regimes of aeroassisted orbital transfer vehicles. In *19th Thermophysics Conference*, page 1730, 1984.
- [214] C. Park. Assessment of two-temperature kinetic model for ionizing air. In *AIAA, 22nd Thermophysics Conference*, volume 1, 1987.
- [215] C. Park. *Nonequilibrium Hypersonic Aerothermodynamics*. Wiley-Interscience, 1990.
- [216] C. Park. Review of chemical-kinetic problems of future NASA missions, I - Earth entries. *Journal of Thermophysics and Heat Transfer*, 7(1):9–23, 1993.

- [217] C. Park. Review of chemical-kinetic problems of future NASA missions. II - Mars entries. *Journal of Thermophysics and Heat Transfer*, 8, 1994.
- [218] C. Park. The limits of two-temperature model. In *48th AIAA Aerospace Sciences Meeting*, number January, pages 1–13, 2010.
- [219] C. Park and S.-H. Lee. Validation of multitemperature nozzle flow code. *Journal of thermophysics and heat transfer*, 9(1):9–16, 1995.
- [220] C. Park and C. Park. Validation of CFD codes for real-gas regime. In *32nd Thermophysics Conference*, page 2530, 1997.
- [221] H. Peng, C. W. Atkins, and R. Deiterding. A solver for simulating shock-induced combustion on curvilinear adaptive meshes. *Computers & Fluids*, 232:105188, 2022.
- [222] N. A. Petersson. An algorithm for assembling overlapping grid systems. *SIAM Journal on Scientific Computing*, 20(6):1995–2022, 1999.
- [223] B. Pincock and A. Katz. High-order flux correction for viscous flows on arbitrary unstructured grids. *International Journal of Mathematics and Computer Sciences*, 30:880–900, 2014.
- [224] K. G. Powell, P. L. Roe, and J. Quirk. Adaptive-mesh algorithms for computational fluid dynamics. In *Algorithmic trends in computational fluid dynamics*, pages 303–337. Springer, 1993.
- [225] A. Prakash, N. Parsons, X. Wang, and X. Zhong. High-order shock-fitting methods for direct numerical simulation of hypersonic flow with chemical and thermal nonequilibrium. *Journal of Computational Physics*, 230(23):8474–8507, 2011.
- [226] J. D. Reinert, G. V. Candler, and J. Komives. Simulations of Unsteady Three-Dimensional Hypersonic Double-Wedge Flow Experiments. *AIAA Journal*, 2020.
- [227] J. D. Reinert, S. Gs, G. V. Candler, and J. R. Komives. Three-dimensional simulations of hypersonic double wedge flow experiments. In *47th AIAA Fluid Dynamics Conference*, page 4125, 2017.
- [228] T. C. Rendall and C. B. Allen. Efficient mesh motion using radial basis functions with data reduction algorithms. *Journal of Computational Physics*, 228(17):6231–6249, 2009.
- [229] P. J. Roache. Verification of codes and calculations. *AIAA Journal*, 36(5):696–702, 1998.

- [230] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.
- [231] S. E. Rogers, N. E. Suhs, and W. E. Dietz. PEGASUS 5: An automated preprocessor for overset-grid computational fluid dynamics. *AIAA Journal*, 41(6):1037–1045, 2003.
- [232] B. Roget and J. Sitaraman. Robust and efficient overset grid assembly for partitioned unstructured meshes. *Journal of Computational Physics*, 260:1–24, 2014.
- [233] B. Roget, J. Sitaraman, V. Lakshminarayan, and A. Wissink. Prismatic mesh generation using minimum distance fields. In *Tenth International Conference on Computational Fluid Dynamics*, 2018.
- [234] B. Roget, J. Sitaraman, V. Lakshminarayan, and A. Wissink. Prismatic mesh generation using minimum distance fields. *Computers and Fluids*, 200:104429, 2020.
- [235] S. R. Sanderson, H. G. Hornung, and B. Sturtevant. The influence of non-equilibrium dissociation on the flow produced by shock impingement on a blunt body. *Journal of Fluid Mechanics*, 516:1–37, Oct. 2004.
- [236] S. Saravanan, G. Jagadeesh, and K. P. J. Reddy. Investigation of missile-shaped body with forward-facing cavity at Mach 8. *Journal of Spacecraft and Rockets*, 46(3):577–591, May 2009.
- [237] G. Sarma. Physico chemical modelling in hypersonic flow simulation. *Progress in Aerospace Sciences*, 36(3-4):281–349, 2000.
- [238] V. Sasidharan and S. Duvvuri. Large- and small-amplitude shock-wave oscillations over axisymmetric bodies in high-speed flow. *Journal of Fluid Mechanics*, 913:R7, Apr. 2021.
- [239] D. Saunders, S. Yoon, and M. Wright. An approach to shock envelope grid tailoring and its effect on reentry vehicle solutions. In *45th AIAA Aerospace Sciences Meeting and Exhibit*, 2007.
- [240] P. Sawicki, R. S. Chaudhry, and I. D. Boyd. Influence of Chemical Kinetics Models on Plasma Generation in Hypersonic Flight. *AIAA Journal*, 60(1):31–40, 2022.
- [241] L. C. Scalabrin. *Numerical Simulation Of Weakly Ionised Hypersonic Flow Over Reentry Capsules*. PhD thesis, University of Michigan, 2007.
- [242] L. C. Scalabrin and I. D. Boyd. Numerical simulation of weakly ionized hypersonic flow for reentry configurations. In *9th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, number June. AIAA, 2006.

- [243] J. C. Schulz, E. C. Stern, S. Muppudi, G. E. Palmer, O. Schroeder, and A. Martin. Development of a three-dimensional, unstructured material response design tool. *AIAA SciTech Forum - 55th AIAA Aerospace Sciences Meeting*, pages 1–17, 2017.
- [244] R. N. Schwartz and K. F. Herzfeld. Vibrational relaxation times in gases (three-dimensional treatment). *The Journal of Chemical Physics*, 22(5):767–773, 1954.
- [245] J. B. Scoggins and T. E. Magin. Development of Mutation++ : MULTicomponent Thermodynamics And Transport properties for IONized gases library in C++. In *11th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*, 2014.
- [246] J. S. Shang and S. T. Surzhikov. Simulating nonequilibrium flow for ablative Earth reentry. *Journal of Spacecraft and Rockets*, 47(5), 2010.
- [247] J. S. Shang and S. T. Surzhikov. Nonequilibrium radiative hypersonic flow simulation. *Progress in Aerospace Sciences*, 53:46–65, 2012.
- [248] A. Sharma, S. Ananthan, J. Sitaraman, S. Thomas, and M. A. Sprague. Overset meshes for incompressible flows: On preserving accuracy of underlying discretizations. *Journal of Computational Physics*, 428:109987, 2021.
- [249] A. B. Shimizu, J. E. Ferrell, and C. A. Powars. Passive Nosetip Technology (PANT) Program: Nosetip transition and shape change tests in the AFFDL 50MW rent arc - Data report. Technical report, 1974.
- [250] J. L. Shinn, J. N. Moss, and A. L. Simmonds. Viscous-shock-layer heating analysis for the shuttle windward-symmetry plane with surface finite catalytic recombination rates. 1982.
- [251] N. Singh and T. Schwartzenruber. Consistent kinetic-continuum dissociation model. II. Continuum formulation and verification. *The Journal of chemical physics*, 152(22):224303, 2020.
- [252] N. Singh and T. E. Schwartzenruber. Nonequilibrium Dissociation and Recombination Models for Hypersonic Flows. *AIAA Journal*, pages 1–16, 2022.
- [253] J. Sitaraman, M. Floros, A. Wissink, and M. Potsdam. Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids. *Journal of Computational Physics*, 229(12):4703–4723, 2010.
- [254] J. Sitaraman, V. K. Lakshminarayan, B. Roget, and A. M. Wissink. Progress in strand mesh generation and domain connectivity for dual-mesh CFD simulations. *55th AIAA Aerospace Sciences Meeting*, m(January):1–23, 2017.

- [255] A. Skillen. *The overset grid method, applied to the solution of the incompressible Navier-Stokes equations in two and three spatial dimensions*. PhD thesis, University of Manchester, 2012.
- [256] R. W. Smith. AUSM(ALE): A geometrically conservative arbitrary Lagrangian-Eulerian flux splitting scheme. *Journal of Computational Physics*, 150(1):268–286, 1999.
- [257] G. Starius. Composite mesh difference methods for elliptic boundary value problems. *Numerische Mathematik*, 28(2):243–258, 1977.
- [258] J. L. Steger and J. A. Benek. On the use of composite grid schemes in computational aerodynamics. *Computer Methods in Applied Engineering*, 64:301–320, 1987.
- [259] J. L. Steger and R. Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. *Journal of computational physics*, 40(2):263–293, 1981.
- [260] E. C. Stern, A. M. Schwing, J. M. Brock, and M. Schoenenberger. Dynamic CFD simulations of the MEADS II ballistic range test model. *AIAA Atmospheric Flight Mechanics Conference*, 2016-Janua:1–27, 2016.
- [261] K. Stewartson and P. Williams. Self-induced separation. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 312(1509):181–206, 1969.
- [262] K. Stewartson and P. G. Williams. On self-induced separation II. *Mathematika*, 20(1):98–108, June 1973.
- [263] J. M. Stone, K. Tomida, C. J. White, and K. G. Felker. The Athena++ adaptive mesh refinement framework: Design and magnetohydrodynamic solvers. *The Astrophysical Journal Supplement Series*, 249(1):4, jun 2020.
- [264] N. E. Suhs, S. E. Rogers, and W. E. Dietz. Pegasus 5: An automated pre-processor for overset grid CFD. In *32nd AIAA Fluid Dynamics Conference*, 2002.
- [265] W. Sutherland. LII. the viscosity of gases and molecular force. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 36(223):507–531, 1893.
- [266] K. Sutton and P. A. Gnoffo. Multi-component diffusion with application to computational aerothermodynamics. *7th AIAA / ASME Joint Thermophysics and Heat Transfer Conference*, 1998.
- [267] A. B. Swantek and J. M. Austin. Heat transfer on a double wedge geometry in hypervelocity air and nitrogen flows. *50th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, (January):1–12, 2012.

- [268] H. Tang, S. Casey Jones, and F. Sotiropoulos. An overset-grid method for 3D unsteady incompressible flows. *Journal of Computational Physics*, 191(2):567–600, Nov. 2003.
- [269] G. Tchuen and Y. Burtschell. *Physico-chemical modelling in nonequilibrium hypersonic flow around blunt bodies*. 2011.
- [270] F. Thivet, M. Perrin, and S. Candel. A unified nonequilibrium model for hypersonic flows. *Physics of Fluids A: Fluid Dynamics*, 3(11):2799–2812, 1991.
- [271] R. A. Thompson and P. A. Gnoffo. Implementation of a blowing boundary condition in the LAURA code. *46th AIAA Aerospace Sciences Meeting and Exhibit*, c:1–11, 2008.
- [272] G. A. Tirsky. Up-to-date gasdynamic models of hypersonic aerodynamics and heat transfer with real gas properties. *Annual Review of Fluid Mechanics*, 25:151–181, 1993.
- [273] S. Tissera. *Assessment of high-resolution methods in hypersonic real-gas flows*. PhD thesis, Cranfield University, 2010.
- [274] J. Tölke, S. Freudiger, and M. Krafczyk. An adaptive scheme using hierarchical grids for lattice Boltzmann multi-phase flow simulations. *Computers and Fluids*, 35(8-9):820–830, 2006.
- [275] O. Tong, A. Katz, Y. Yanagita, and D. Work. Verification and validation of a high-order strand grid method for two-dimensional turbulent flows. *Computers and Fluids*, 154:335–346, 2017.
- [276] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics: A practical introduction*. Springer Science & Business Media, 1999.
- [277] P. Ullrich and C. Jablonowski. Operator-split Runge-Kutta-Rosenbrock methods for nonhydrostatic atmospheric models. *Monthly Weather Review*, 140(4):1257–1284, 2012.
- [278] B. van der Holst and R. Keppens. Hybrid block-AMR in cartesian and curvilinear coordinates: MHD applications. *Journal of Computational Physics*, 226(1):925–946, 2007.
- [279] B. van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *Journal of Computational Physics*, 23:276–299, 1977.
- [280] B. Van Leer. Flux-vector splitting for the Euler equations. In *Eighth international conference on numerical methods in fluid dynamics*, pages 507–512. Springer, 1982.

- [281] J. M. Varah. On the solution of block-tridiagonal systems arising from certain finite-difference equations. *Mathematics of Computation*, 26(120):859, 1972.
- [282] M. Vinokur. On one-dimensional stretching functions for finite-difference calculations. *Journal of Computational Physics*, 50(2):215–234, 1983.
- [283] Y. Wada and M.-S. Liou. A flux splitting scheme with high-resolution and robustness for discontinuities. *AIAA paper*, pages 1–23, 1994.
- [284] X. Wang and X. Zhong. Nonequilibrium and reactive high-speed flow simulations with a fifth-order WENO scheme. In *39th AIAA Fluid Dynamics Conference*, number June, 2009.
- [285] H. Weller, S.-j. Lock, and N. Wood. Runge-Kutta IMEX schemes for the Horizontally Explicit / Vertically Implicit (HEVI) solution of wave equations. *Journal of Computational Physics*, 252:365–381, 2013.
- [286] A. R. Wieting and M. S. Holden. Experimental shock-wave interference heating on a cylinder at Mach 6 and 8. *AIAA Journal*, 27(11):1557–1565, Nov. 1989.
- [287] C. R. Wilke. A viscosity equation for gas mixtures. *The Journal of Chemical Physics*, 18(4):517–519, 1950.
- [288] C. Windisch, B. Reinartz, and S. Müller. H-adaptive simulation of hypersonic flows in thermochemical nonequilibrium. *18th AIAA/3AF International Space Planes and Hypersonic Systems and Technologies Conference 2012*, pages 1–10, 2012.
- [289] C. Windisch, B. U. Reinartz, and S. Müller. Investigation of unsteady Edney Type IV and VII shock–shock interactions. *AIAA Journal*, 54(6):1846–1861, June 2016.
- [290] A. M. Wissink, N. K. Burgess, A. Katz, J. Sitaraman, and R. Haimes. Validation of 3D RANS-SA calculations on strand/Cartesian meshes. In *21st AIAA Computational Fluid Dynamics Conference*. AIAA, 2013.
- [291] A. M. Wissink, A. J. Katz, W. M. Chan, and R. L. Meakin. Validation of the strand grid approach. *19th AIAA Computational Fluid Dynamics Conference*, (June), 2009.
- [292] A. H. C. Wood. *A Parallel Implicit Adaptive-Mesh-Refinement Scheme for Hypersonic Flows with an Equilibrium High-Temperature Equation of State*. PhD thesis, 2008.
- [293] M. R. Wool. Passive NoseTip Technology (PANT) Program: Summary of experimental and analytical results. Technical report, Acurex Corporation, 1975.
- [294] D. Work, O. Tong, R. Workman, A. Katz, and A. M. Wissink. Strand-grid-solution procedures for sharp corners. *AIAA Journal*, 52(7), 2014.

- [295] M. J. Wright, D. Bose, G. E. Palmer, and E. Levin. Recommended collision integrals for transport property computations, Part 1: Air species. *AIAA Journal*, 43(12):2558–2564, 2005.
- [296] M. J. Wright, G. V. Candler, and D. Bose. Data-Parallel Line Relaxation method for the Navier-Stokes equations. *AIAA Journal*, 36(9), 1998.
- [297] F. Xiao, Z. Li, Y. Zhu, and J. Yang. Hypersonic Type-IV shock/shock interactions on a blunt body with forward-facing cavity. *Journal of Spacecraft and Rockets*, 54(2):506–512, Mar. 2017.
- [298] S. Yoon and A. Jameson. Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations. *AIAA Journal*, 26(9):1025–1026, 1988.
- [299] S.-T. Yu. Convenient method to convert two-dimensional CFD codes into axisymmetric ones. *Journal of Propulsion and Power*, 9(3), 1993.
- [300] F. Zander, P. A. Jacobs, U. A. Sheikh, and R. G. Morgan. High speed imaging of spherical shock standoff in hypervelocity flows. *18th Australasian Fluid Mechanics Conference*, (December):5–8, 2012.
- [301] W. Zhang, A. Almgren, V. Beckner, J. Bell, J. Blaschke, C. Chan, M. Day, B. Friesen, K. Gott, D. Graves, M. Katz, A. Myers, T. Nguyen, A. Nonaka, M. Rosso, S. Williams, and M. Zingale. AMReX: a framework for block-structured adaptive mesh refinement. *Journal of Open Source Software*, 4(37):1370, May 2019.
- [302] J. Z. X. Zheng, C. P. T. Groth, and D. Street. Block-based adaptive mesh refinement finite-volume scheme for hybrid multi-block meshes. *Seventh International Conference on Computational Fluid Dynamics*, pages ICCFD7–2504, 2012.
- [303] X. Zhong. Application of essentially nonoscillatory schemes to unsteady hypersonic shock-shock interference heating problems. *AIAA Journal*, 32(8):1606–1616, Aug. 1994.
- [304] A. L. Zubitsker, J. McQuaid, C. Brehm, and A. Martin. Development and verification of a mesh deformation scheme for a three dimensional ablative material solver. In *AIAA SCITECH 2022 Forum*, page 1285, 2022.
- [305] J. L. Ziegler, R. Deiterding, J. E. Shepherd, and D. I. Pullin. An adaptive high-order hybrid scheme for compressive, viscous flows with detailed chemistry. *Journal of Computational Physics*, 230(20):7598–7630, 2011.