

One-dimensional turbulence (ODT): computationally efficient modeling and simulation of turbulent flows

Victoria B. Stephens, David O. Lignell*

Chemical Engineering Department, Brigham Young University, Provo, UT 84602, USA

Abstract

Write this last. About 100 words.

Keywords: turbulence, reacting flows, one-dimensional turbulence

Code Metadata

Nr.	Code metadata description	Please fill in this column
C1	Current code version	1.0
C2	Permanent link to code/repository used for this code version	<i>github.com/BYUignite/ODT</i>
C3	Code Ocean compute capsule	N/A
C4	Legal Code License	MIT
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	C++, Python 3.x, Yaml,
C7	Compilation requirements, operating environments & dependencies	CMake 3.12+, Cantera, Git, Doxygen (optional)
C8	If available Link to developer documentation/manual	N/A
C9	Support email for questions	davidlignellbyu.edu

Table 1: Code metadata (mandatory)

1. Motivation and significance

Turbulent flows characterize the vast majority of fluid flows in practical engineering applications, and simulations of turbulent flows provide re-

*Corresponding author.

Email address: davidlignell@byu.edu (David O. Lignell)

4 searchers with valuable insights into complex systems, particularly reacting
5 turbulent flows such as combustion processes. Turbulence is a complex phe-
6 nomenon that affects the full range of a flow’s length and time scales. As a
7 result, resolving the entire flow field by numerically solving the Navier-Stokes
8 equations of fluid flow, as is done in direct numerical simulations (DNS), re-
9 quires substantial computational resources. DNS is a powerful research tool,
10 but its high computational cost makes it intractable for simulating most
11 practical engineering flows. In order to achieve numerical solutions to prac-
12 tical flow problems, researchers can use alternative frameworks that model
13 turbulence rather than resolving it directly.

14 Large-eddy simulations (LES) address the problem of wide-ranging length
15 and time scales by combining direct resolution of grid-scale quantities, as in
16 DNS, with subgrid modeling of smaller turbulence structures. The more
17 complex the flow, the more modeling is required; for example, a jet flame
18 simulation might require subgrid modeling for the combustion chemistry, ra-
19 diative heat transfer, or soot chemistry in addition to turbulence structures,
20 all of which form a tightly coupled system in which each model interacts
21 heavily with the others. While subgrid modeling makes LES more computa-
22 tionally affordable than DNS, it can introduce empiricism into simulations,
23 which can lead to inaccurate results. Additionally, unresolved quantities are
24 often parameterized in state space with empirical relationships or assumed
25 distributions that lack universal applicability. LES is a valuable simulation
26 tool, but its approach to turbulence modeling can introduce unwanted em-
27 piricism and make errors difficult to isolate and quantify.

28 The one-dimensional turbulence model (ODT) functionally reverses the
29 LES approach, modeling large-scale turbulent advection and directly resolv-
30 ing small-scale flow structures, simulating the full range of length and time
31 scales in a single dimension. Because large-scale structures are much easier
32 to study and model than small-scale structures, ODT mitigates or sidesteps
33 many of the subgrid modeling issues that complicate LES. Previous stud-
34 ies show that ODT can attain accuracy comparable to DNS at a fraction
35 of the computational cost [1, 2], making it an attractive tool for simulating
36 turbulent flows. Because the ODT model is one-dimensional, it is limited to
37 homogeneous or boundary-layer flows, such as jets, wakes, and mixing layers;
38 these types of flows, however, are common in nature and central to turbu-
39 lence research. ODT’s computational efficiency and resolution of a full range
40 of scales make it a valuable tool that complements experimental studies and
41 other simulation tools like DNS and LES.

42 Early applications of ODT focused on homogenous turbulence, wakes, and
43 mixing layers [3, 4, 5]. Later extension to variable-density flows and a spatial
44 downstream coordinate system facilitated its growth and application to more

complex flows, including combustion in jet flames [6, 7, 8, 9, 10, 11, 12, 13], counterflow flames [14], wall fires [15], and sooting flames [1, 16, 17, 18, 19], as well as other particle flows [20, 21, 22, 23]. ODT has also served to complement LES through subgrid modeling studies [24, 25, 26] and has been applied to various other flow configurations such as double-diffusive interfaces [27], Rayleigh-Taylor mixing [28], and stratified turbulence [29]. Most recently, the ODT code was extended to include cylindrical and spherical coordinate systems [30, 31, 32].

During the recent implementation of the cylindrical and spherical model formulations, the ODT code was drastically overhauled and reorganized, resulting in its current configuration. The ODT code presented here is a pared down version of the development code, representing the fundamental aspects of the ODT model and its most reliable functions. The example cases in Section 3 are a representative sample of the ODT code’s capabilities as it is presented here. Future releases will expand this code’s functionality with additional features currently in development.

2. Software description

2.1. Model description

The ODT model is described in detail in the literature [3, 5, 33, 30, 34]; only a brief explanation will be given here. In ODT, turbulent advection is modeled with stochastic processes called eddy events, which punctuate the solution of unsteady, one-dimensional transport equations for mass, momentum, and enthalpy. The ODT code uses a Lagrangian finite-volume formulation for diffusive advancement in which mass stays constant within each grid cell while cell volumes increase or decrease according to cell dilation via an adaptive mesh refinement [34].

Transport equations for mass, momentum, and enthalpy in the temporal formulation of ODT take the following generic form, derived from the Reynolds Transport Theorem [35] for a given scalar quantity per unit mass β :

$$\frac{d\beta}{dt} = -\frac{j_{\beta,e}A_{x,e} - j_{\beta,w}A_{x,w}}{\rho V} + \frac{S_{\beta}}{\rho V}. \quad (1)$$

Here, j_{β} is the diffusion flux of scalar β across the cell face area A_x where the subscripts e and w refer to the "east" and "west" faces of the grid cell, respectively. S_{β} is the Lagrangian source term derived from the conservation law for β , ρ represents mass density, and V represents cell volume. In practice, we refer to the left hand term on the right side of Equation 1 as the "mixing term" and the right hand term on the right side of Equation 1 as the "source term". The generic transport equation differs slightly in the spatial

formulation of ODT, but its form is the same, so we omit it here for brevity. The system of ordinary differential equations (ODEs) that results is well behaved at all grid points and in all geometries in their finite-volume forms. For details on transport equation derivation and use in both the temporal and spatial formulations of ODT, see Lignell et al. [30].

Eddy events occur as a Poisson process in accordance with their eddy rates, where a given eddy event of size l and location x_0 has an eddy timescale t and an associated eddy rate $1/t$. Three user-defined ODT parameters control the eddy event process: the eddy rate parameter C scales the rate of occurrence of the eddies; the viscous penalty parameter Z suppresses small eddies; and the large eddy suppression parameter β constrains eddies such that they do not reach over the elapsed simulation time. Sampled eddies that do not fit the defined parameters are rejected and not applied to the domain.

Eddy events modify domain variables using triplet maps, as illustrated for a cylindrical domain in Figure 1. For a region of eddy size l , the domain is copied to create three map images; the three images are then placed back to back with the middle image inverted to maintain continuity, and the composite is reapplied to the domain. This process applies to all transported variables on the domain. Applied properly, the triplet map increases scalar gradients and decreases length scales consistent with the application of turbulent eddies in real flows, conserves all quantities and their statistical moments, and maintains continuity in property profiles. Subsequent eddies in the same region will result in a cascade of scales, and eddy rates depend on eddy size and the local kinetic energy such that they follow turbulent cascade scaling laws.

Eddy events occur concurrently with diffusive advancement via solution of the system of unsteady one-dimensional transport equations. In this way, the ODT code marches in time or space until it reaches its end point. Due to the stochastic nature of eddy events, each ODT simulation, or realization, is different, even when it is provided with the same input parameters. In order to obtain statistically stable data for a given set of parameters, we run many realizations with the same input parameters and time-average them. This is done via post-processing tools, which are provided in the ODT package.

2.2. Software Architecture

The ODT package consists primarily of an object-oriented C++ code responsible for running flow simulation cases and generating data. The package also contains auxiliary data processing and visualization tools, written mostly in Python. The post-processing tools are case-specific but will be addressed generally in Section 2.3.

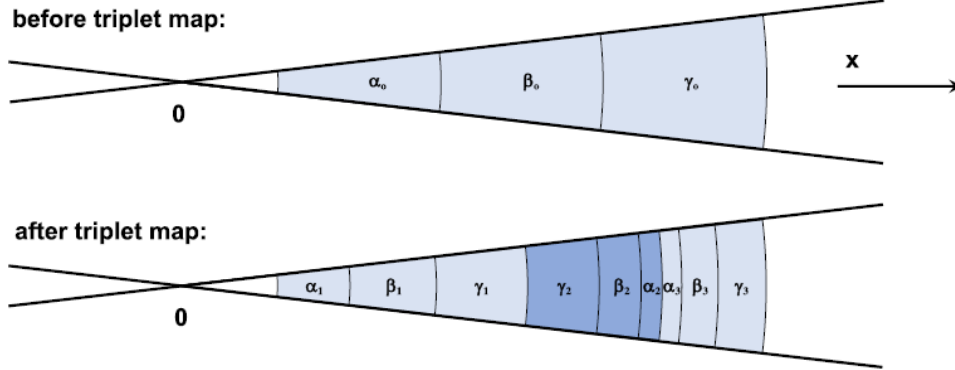


Figure 1: Schematic diagram of a cylindrical triplet map, adapted from [30]. Before the triplet map, the domain contains three grid cells of equal volume, while after the triplet map has been applied, the domain contains nine cells. The nine final cells are labeled according to the cells from which they originated and shaded to indicate that three map images were combined to create the final composite.

Figure 2 illustrates the ODT code’s most important objects and structural features. User inputs are provided to the executable in YAML [36] format via `input.yaml`; the location of the specific `input.yaml` file to be used is determined by the case name and case type specified in the run script. The `main` function defines storage for the main objects, but, once created, the `domain` object is responsible for object initialization as well as variable initialization and storage via a case-specific `domaincase` object.

Three primary objects handle the code’s main functions: the `solver`, `micromixer`, and `eddy` objects. The `solver` coordinates the ODT solution process, marching along the simulation time and invoking diffusive advancement and eddy events when appropriate. The `micromixer` handles diffusive advancement by setting step sizes, interacting with the transported domain variables, and solving the system of ODEs defined by Equation 1 (or its equivalent in the spatial formulation). The `micromixer` includes three solution methods that can be specified in `input.yaml`, each appropriate for various case types: a first-order explicit Euler method (pictured in Figure 2); a first-order semi-implicit method that uses CVODE [37] to advance coupled ODEs in individual grid cells, integrated sequentially; and a second-order Strang splitting method [38] good for treating stiff chemistry. In reacting flow cases, chemical kinetics are handled by Cantera [39], which uses transported variable values—enthalpy and gas species composition, for instance—to specify local scalar values such as gas temperature or density, which can affect flow properties. The `micromixer` is also the code’s primary point of in-

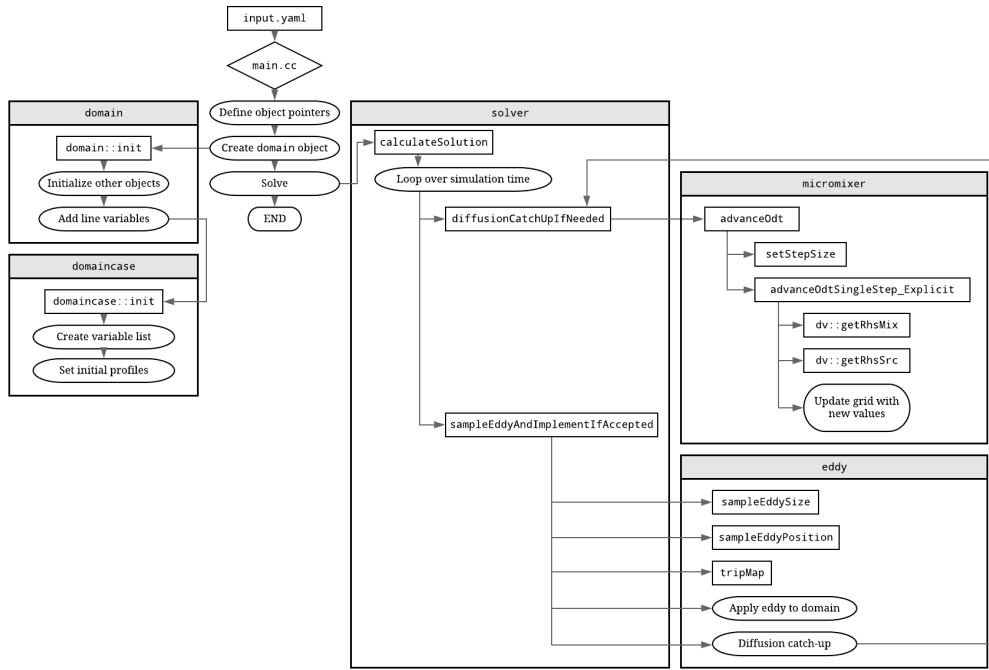


Figure 2: Structural outline of the ODT code, including its major class objects. For brevity, this diagram makes two simplifications. First, it assumes that diffusive advancement in the `micromixer` uses an explicit solution method; in practice, users specify either an explicit, semi-implicit, or Strang split method. Second, it implies that all of the eddy events sampled by the `eddy` object are applied to the domain; in reality, eddy events are filtered through several rejection tests (omitted from this figure for clarity) before acceptance and application to the domain.

145 teraction with the `mesher` object (not pictured in Figure 2), which manages
146 the adaptive grid functions. Finally, the `eddy` object manages eddy event
147 selection and implementation, which proceeds as described in Section 2.1.

148 2.3. Workflow

149 This section outlines the process a user goes through in order to success-
150 fully build the ODT code and run a simulation. For more details, please see
151 the package documentation. ODT is a standalone, self-contained package,
152 and users interact with its files primarily via the command line rather than
153 a graphical interface.

154 Within the main download package, several directories organize the ODT
155 code. The `source`, `build`, and `run` directories contain the ODT source code,
156 compilation tools, and run scripts, respectively. The `input` directory contains
157 subdirectories corresponding to several possible case types, each populated
158 with an appropriate input files. The `data` directory, initially empty, holds
159 the raw data files and runtime information generated by ODT, as well as
160 post-processed data files generated from within the `post` directory. Finally,
161 the `doc` directory contains documentation optionally generated by Doxygen
162 [40] during the build process.

163 2.3.1. Building ODT

164 The ODT build process is automated with CMake. First, the user nav-
165 igates to the `build` directory and edits the CMake configuration file. The
166 CMake configuration file specifies Cantera’s location and must be changed
167 to reflect the local installation location. Note that the user must also specify
168 which chemical mechanism the code will compile with. In order to run sim-
169 ulations with different chemical mechanisms, the code must be recompiled,
170 including the CMake configuration files, between simulations, whereas this
171 is not required when other variables are changed. This is a known inconsis-
172 tency that we plan to address in future code releases. Once the configuration
173 settings are updated, the user runs CMake to apply the changes.

174 YAML is required in order to process input files. If it has not yet been
175 installed, it must be built and installed at this point. For convenience, the
176 ODT package automates this process. YAML need only be installed once
177 for a given instance of the ODT package. Rebuilding the ODT code with
178 CMake does not affect the YAML installation.

179 Once CMake and YAML have been prepared, the user can build the ODT
180 code with the `make` command. There is no associated `install` command that
181 needs to be run after the `make` command. The most common errors that occur
182 during the build process concern incorrect file paths or incomplete installation
183 of required packages. See the build documentation for troubleshooting help.

184 Once the code is built, the user may optionally build a local copy of the
185 documentation via Doxygen, which must be previously installed. This step is
186 not required in order to run the ODT code. As an alternative, documentation
187 can be accessed via `README` files within the code or at the code repository
188 wiki.

189 2.3.2. *Input files*

190 User-modified input files are located within the `input` directory, which
191 contains subdirectories that correspond to various case types that can be run
192 with ODT. At minimum, a case's subdirectory must contain an `input.yaml`
193 file, but may contain other files or subfolders with supplementary informa-
194 tion.

195 The `input` directory also contains the `gas_mechanisms` subdirectory, which
196 contains chemical mechanism files that can be used in reacting flow cases.
197 For cases in which a chemical mechanism is not required, the `not_used.xml`
198 mechanism file is specified in `input.yaml`. Note that the chemical mech-
199 anism chosen for the case and specified in the input file must match the
200 mechanism specified in the CMake configuration file during the ODT build
201 process.

202 Input files contain simulation parameters in a human-readable format
203 parsed by YAML. Not all of the parameters in an input file may be used
204 in a given simulation. Within `input.yaml`, parameters are organized into
205 sections, several of which are common to all input files. Details about in-
206 dividual parameters, including usage and typical values, are covered in the
207 documentation. Prior to running a simulation, users must select and modify
208 the appropriate `input.yaml` file to reflect the desired simulation conditions.
209 The default values present in the input files represent general parameters
210 that may be used to run a successful simulation of that case type.

211 2.3.3. *Running ODT*

212 To run a simulation, users must then navigate to the `run` directory, which
213 contains the `odt.x` executable and several possible run scripts. The simplest
214 option is `runOneRlz.sh`, which runs one realization of ODT in the specified
215 configuration. In this run script, the user must alter two variables near the
216 top of the file: `inputDir`, which specifies which input directory and files to
217 use; and `caseName`, which provides a name for the simulation and the data
218 files it outputs. The `runManyRlz.sh` script runs many realizations in serial,
219 one after the other; it differs from `runOneRlz.sh` only in that the user must
220 also alter the `nRlz` variable, which specifies the number of realizations to
221 run. To run the simulation with either `runOneRlz.sh` or `runManyRlz.sh`,
222 save the run script and execute it at the command line. Users will see some

output on the command line, but no data, which is instead output to the **data** directory.

ODT simulations can also be run in parallel using MPI. Two run scripts, **slrmJob.sh** and **slrmJob_array.sh**, are configured using SLURM [41], a common workload manager used for massively parallel computing resources. This allows many ODT realizations to run in parallel rather than in serial, reducing overall simulation time. Individual realizations are independent and do not affect one another, but users must take care with case names and input file changes to ensure that individual realizations or entire cases are not overridden accidentally.

2.3.4. Data files and post-processing

For a given simulation, data is output to the **data** directory, which contains subdirectories for each simulation, specified by the **caseName** variable in the run script. Figure 3 illustrates the structure of the **data** directory and the locations of files within it. Each case folder is subdivided into **input**, **runtime**, **data**, and **post**. The **input** folder contains a copy of the input files used for the simulation, **runtime** contains runtime output information, **data** contains the raw data files, and **post** contains post-processed data files once they are generated.

To use post-processing tools, navigate to the **post** directory. Within the **post** directory, data processing tools are organized by case type, which is specified in **input.yaml** and determines case-specific variables and simulation setup parameters (refer to the **domaincase** object in Figure 2). Each set of post-processing tools is different, but may contain some combination of Python scripts (often coordinated by a **driver.py** file), experimental data files for comparison and plot generation, or other supplementary files. Post-processed data and generated plots are deposited in the **data** directory, within the appropriate **caseName/post** folder. For more information on using the provided post-processing tools, please refer to the documentation.

3. Example Cases

3.1. Pipe Flow

First, we present an incompressible pipe flow simulation using the temporal, cylindrical ODT formulation. Results for three different friction Reynolds numbers ($Re_\tau = 550, 1000, 2000$) are compared to DNS results from El Khoury et al. [42] ($Re_\tau = 550, 1000$) and Chin et al. [43] ($Re_\tau = 2000$) for a pipe diameter of $D = 2.0$ m and flow density of $1.0 \text{ kg}\cdot\text{m}^{-3}$. Friction velocity values of $1 \text{ m}\cdot\text{s}^{-1}$ ($Re_\tau = 550, 1000$) and $2 \text{ m}\cdot\text{s}^{-1}$ ($Re_\tau = 2000$) were assumed and used to calculate the mean pressure gradient driving the flow. Using

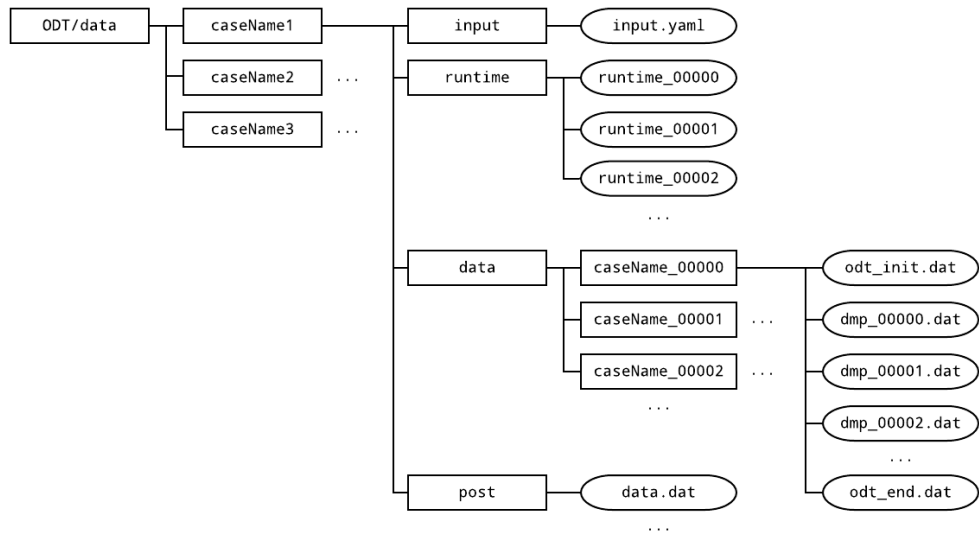


Figure 3: Data folder structure and file locations. Boxes indicate folders and ovals indicate files. For a given simulation, **caseName** will be replaced with the case name specified in the run script. The number attached to **caseName** folder names and to **runtime** data files indicates the realization number. The number attached to **dmp** files indicates the output time step, or "dump" time, of the data file. These correspond to the order of the time steps listed in **input.yaml**.

261 initial conditions with uniform velocity profiles, simulations were run until a
 262 state of developed flow was achieved, at which point data were gathered until
 263 statistical convergence for the root mean square (RMS) velocity difference
 264 from the mean profiles occurred.

265 The simulations were performed with ODT parameters $C = 5$ and $Z =$
 266 350 for the temporal ODT formulation. The values of C and Z were adjusted
 267 to give good agreement of the ODT results compared to the DNS. Schmidt
 268 et al. [25] showed that higher Z results in the buffer-layer being located
 269 further from the wall, and increasing C results in a lower slope of the mean
 270 streamwise velocity in the log-layer.

271 RESULTS AND PLOTS GO HERE

272 3.2. *Non-reacting Jet*

273 Here, we present ODT simulation results for a non-reacting round, tur-
 274 bulent jet compared to the experimental data of Hussein et al. [44]. The
 275 jet consists of air issuing into air through a 1 in (0.0254 m) diameter duct
 276 with a uniform exit velocity of $56.2 \text{ m}\cdot\text{s}^{-1}$ and a reported Reynolds num-
 277 ber of 95,500. The ODT simulations use this diameter and velocity with a
 278 kinematic viscosity of $1.534 \cdot 10^{-5} \text{ m}^2\text{s}^{-1}$, resulting in a Reynolds number
 279 of 93,056. The initial velocity profile in the ODT simulations is a modified
 280 top-hat profile in which a hyperbolic tangent function of width $\delta = 0.1D$ is
 281 used on either side of the jet to smooth the transition between the jet and
 282 the free stream. In the spatial formulation of ODT, the streamwise velocity
 283 must be positive everywhere on the line, so a small minimum velocity of
 284 $v_{min} = 0.1 \text{ m}\cdot\text{s}^{-1}$ is specified and added across the entire velocity profile.

285 ODT simulations were performed with parameters $C = 5.25$, $\beta_{LES} = 3.5$,
 286 and $Z = 400$. The value of Z is the same as the spatial simulations in [15],
 287 and the values of C and β_{LES} were adjusted to give good agreement with
 288 the experimental data. Note the close agreement of the C and Z parameters
 289 here to the optimal values used for the pipe flow simulations ($C = 5$ and
 290 $Z = 350$). This illustrates a level of robustness in the ODT parameters
 291 and suggests that intermediate values could be successfully applied in both
 292 configurations.

293 1024 independent ODT realizations were performed and results were en-
 294 semble averaged. All quantities are normalized consistent with jet similarity
 295 scaling. Downstream locations are normalized by the jet diameter D , and ra-
 296 dial locations are normalized by $(y - y_0)$, where y is the downstream location
 297 and $y_0 = 4D$ is the virtual origin used in [44].

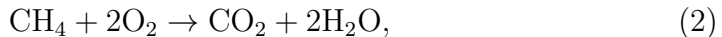
298 RESULTS AND PLOTS GO HERE

299 3.3. Jet Flame

300 ODT is uniquely suited for reacting flow simulations. Here, we present
 301 illustrative ODT simulation results of a round, turbulent jet flame based on
 302 and compared to the experimental DLR-A flame of Meier et al. [45]. This
 303 canonical flame configuration has been used extensively to study and validate
 304 turbulent combustion models [46, 47, 48, 49, 50, 51].

305 The DLR-A fuel stream is mixture of 22.1% CH₄, 33.2% H₂, and 44.7%
 306 N₂ (by volume) that issues into dry air via a nozzle with an inner diameter
 307 of 8 mm at a mean exit velocity of 42.2 m·s⁻¹. The coflow air stream issues
 308 from a concentric nozzle 140 mm in diameter at a velocity of 0.3 m·s⁻¹. The
 309 reported jet Reynolds number is 15,200.

Previous ODT studies of turbulent jet flames have used the temporal
 planar formulation, but the spatial cylindrical formulation developed recently
 [30] more closely matches the experimental configuration. This simulation
 uses the experimentally reported velocity profiles and jet dimensions. In
 the non-reacting case, a small minimum velocity was added uniformly to
 the velocity profile; no such addition is required here because of the slow-
 moving coflow air stream that issues alongside the reacting jet. The fuel
 was diluted with N₂ in the experimental flame to minimize radiative heat
 losses, and radiation is ignored in the simulation. This flame has a low
 Reynolds number, and the combustion chemistry proceeds quickly. The ODT
 simulation transports the chemical species O₂, N₂, CH₄, H₂, H₂O, and CO₂.
 We assume that reactions proceed to the products of complete combustion
 and apply simple, fast reaction rates according to the following chemical
 equations:



310 These assumptions are not reasonable for the DLR-A flame, but they allow
 311 us to illustrate ODT in a reacting jet configuration with variable properties
 312 and heat release, which is the primary purpose of this example case. More
 313 complex combustion reaction mechanisms are available within the source
 314 code and can be accessed by changing the appropriate input file parameter.

315 This simulation uses ODT parameters $C = 20$, $\beta_{LES} = 17$, and $Z = 400$.
 316 The values of C and β_{LES} were adjusted to give good agreement with the
 317 experimental data, and the value of Z is the same as it was for the non-
 318 reacting jet in Section 3.2. 1024 independent flow realizations were performed
 319 in parallel and the results ensemble averaged. Downstream distance y and
 320 radial position r are normalized by the jet diameter D .

321 RESULTS AND PLOTS GO HERE

322 4. Impact

323 Questions to answer in this section (from SoftwareX template)

- 324 1. How can new research questions be pursued with this software?
- 325 • possibility of parametric studies (much harder with DNS/LES/RANS)
 - 326 • study of late-flame soot and radiation interactions, soot emissions
 - 327 as smoke
 - 328 • comparative radiation model studies?
- 329 2. How does the software improve pursuit of existing research questions?
- 330 • late-flame behavior becomes easier to study
 - 331 • validation of LES subgrid models
 - 332 • soot stuff, especially late in the flame (because soot moves slowly
 - 333 compared to gas species and therefore short simulation times like
 - 334 in DNS aren't enough to study it effectively)
- 335 3. How does the software change the daily practice of its users?
- 336 • cases take hours or days rather than weeks using supercomputer
 - 337 resources
 - 338 • test cases can be run on local computers (unlike something like
 - 339 DNS) and as background tasks without disrupting other tasks
 - 340 • ODT as a tool complements other approaches, can cover blind
 - 341 spots and be used in validation
- 342 4. How widespread is the software? Who uses it? (Within and outside of
- 343 intended research area and/or group.)
- 344 • BYU group
 - 345 • JCH at Sandia
 - 346 • Chalmers group in Sweden (Marco Fistler, etc.)
 - 347 • German university group (Heiko Schmidt, Juan Media, Marten
 - 348 Klein, etc.)
 - 349 • TO DO: find other groups who have used or currently use ODT
- 350 5. How is the software used in commercial settings (if any)? Has it led to
- 351 creation of spin-off companies?
- 352 • No commercial use (I think).

353 5. Conclusion

354 Write this part next to last

355 6. Conflict of Interest

356 We wish to confirm that there are no known conflicts of interest associated
357 with this publication and there has been no significant financial support for
358 this work that could have influenced its outcome.

359 Acknowledgements

360 This work was supported in part by the National Science Foundation
361 under Grant No. CBET-1403403.

362 References

- 363 [1] D. O. Lignell, G. C. Fredline, A. D. Lewis, Comparison of one-
364 dimensional turbulence and direct numerical simulations of soot for-
365 mation and transport in a nonpremixed ethylene jet flame 35 (2) (2015)
366 1199–1206. doi:10.1016/j.proci.2014.05.046.
- 367 [2] A. W. Abboud, C. Schulz, T. Saad, S. T. Smith, D. D. Harris, D. O.
368 Lignell, A numerical comparison of precipitating turbulent flows between
369 large-eddy simulation and one-dimensional turbulence 61 (10) (2015)
370 3185–3197. doi:10.1002/aic.14870.
- 371 [3] A. R. Kerstein, One-dimensional turbulence: model formulation and ap-
372 plication to homogeneous turbulence, shear flows, and buoyant stratified
373 flows 392 (1999) 277–334. doi:10.1017/S0022112099005376.
- 374 [4] A. R. Kerstein, T. D. Dreeben, Prediction of turbulent free shear
375 flow statistics using a simple stochastic model 12 (2) (2000) 418–424.
376 doi:10.1063/1.870319.
- 377 [5] A. R. Kerstein, W. T. Ashurst, S. Wunsch, V. Nilsen, One-dimensional
378 turbulence: vector formulation and application to free shear flows 447
379 (2001) 85–109. doi:10.1017/S0022112001005778.
- 380 [6] T. Echekki, A. R. Kerstein, T. D. Dreeben, J.-Y. Chen, ‘one-
381 dimensional turbulence’ simulation of turbulent jet diffusion flames:
382 model formulation and illustrative applications 125 (3) (2001) 1083–
383 1105. doi:10.1016/S0010-2180(01)00228-0.
- 384 [7] J. C. Hewson, A. R. Kerstein, Stochastic simulation of transport and
385 chemical kinetics in turbulent $\text{co}/\text{h}_2/\text{n}_2$ flames 5 (4) (2001) 669–697.
386 doi:10.1088/1364-7830/5/4/309.

- [8] J. C. Hewson, A. R. Kerstein, Local extinction and reignition in nonpremixed turbulent $\text{co}/\text{h}_2/\text{n}_2$ jet flames 174 (5-6) (2002) 35–66. doi:10.1080/713713031.
- [9] D. O. Lignell, D. S. Rappleye, One-dimensional-turbulence simulation of flame extinction and reignition in planar ethylene jet flames 159 (9) (2012) 2930–2943. doi:10.1016/j.combustflame.2012.03.018.
- [10] N. Punati, J. C. Sutherland, A. R. Kerstein, E. R. Hawkes, J. H. Chen, An evaluation of the one-dimensional turbulence model: Comparison with direct numerical simulations of co/h_2 jets with extinction and reignition 33 (1) (2011) 1515–1522. doi:10.1016/j.proci.2010.06.127.
- [11] A. Abdelsamie, D. O. Lignell, D. Thévenin, Comparison between odt and dns for ignition occurrence in turbulent premixed jet combustion: safety-relevant applications 231 (10) (2017) 1709–1735. doi:10.1515/zpch-2016-0902.
- [12] D. O. Lignell, V. B. Lansinger, A. R. Kerstein, A cylindrical formulation of the one-dimensional turbulence (odt) model for turbulent jet flames, in: AIChE Annual Meeting 2017, American Institute of Chemical Engineers, 2017.
- [13] B. Goshayeshi, J. C. Sutherland, Prediction of oxy-coal flame stand-off using high-fidelity thermochemical models and the one-dimensional turbulence model 35 (3) (2015) 2829–2837. doi:10.1016/j.proci.2014.07.003.
- [14] Z. Jozefik, A. R. Kerstein, H. Schmidt, S. Lyra, H. Kolla, J. H. Chen, One-dimensional turbulence modeling of a turbulent counterflow flame with comparison to dns 162 (8) (2015) 2999–3015. doi:10.1016/j.combustflame.2015.05.010.
- [15] E. I. Monson, D. O. Lignell, M. A. Finney, C. Werner, Z. Jozefik, A. R. Kerstein, R. S. Hintze, Simulation of ethylene wall fires using the spatially-evolving one-dimensional turbulence model 52 (1) (2016) 167–196. doi:10.1007/s10694-014-0441-2.
- [16] J. C. Hewson, A. J. Ricks, S. R. Tieszen, A. R. Kerstein, R. O. Fox, Conditional-moment closure with differential diffusion for soot evolution in fire, in: Center for Turbulence Research, Proceedings of the Summer Program 2006, Stanford University, 2006.
- [17] J. C. Hewson, A. J. Ricks, S. R. Tieszen, A. R. Kerstein, R. O. Fox, On the transport of soot relative to a flame: modeling differential diffusion

- 422 for soot evolution in fire, in: H. Bockhorn, A. D’Anna, A. F. Sarofim,
423 H. Wang (Eds.), *Combustion Generated Fine Carbonaceous Particles*,
424 KIT Scientific Publishing, 2009, pp. 571–588.
- 425 [18] D. O. Lignell, J. C. Hewson, One-dimensional turbulence simulation:
426 overview and application to soot formation in nonpremixed flames, in:
427 SIAM Conference on Computational Science and Engineering, 2015.
- 428 [19] A. J. Ricks, J. C. Hewson, A. R. Kerstein, J. P. Gore, S. R. Tieszen,
429 W. T. Ashurst, A spatially developing one-dimensional turbulence (odt)
430 study of soot and enthalpy evolution in meter-scale buoyant turbulent
431 flames 182 (1) (2010) 60–101. doi:10.1080/00102200903297003.
- 432 [20] G. Sun, J. C. Hewson, D. O. Lignell, Evaluation of stochastic par-
433 ticle dispersion modeling in turbulent round jets 89 (2017) 108–122.
434 doi:10.1016/j.ijmultiphaseflow.2016.10.005.
- 435 [21] J. R. Schmidt, J. O. L. Wendt, A. R. Kerstein, Non-equilibrium wall
436 deposition of inertial particles in turbulent flow 137 (2) (2009) 233–257.
437 doi:10.1007/s10955-009-9844-8.
- 438 [22] G. Sun, D. O. Lignell, J. C. Hewson, C. R. Gin, Particle dispersion
439 in homogeneous turbulence using the one-dimensional turbulence model
440 26 (10) (2014) 103301. doi:10.1063/1.4896555.
- 441 [23] M. Fistler, D. O. Lignell, A. R. Kerstein, M. Oevermann, Numerical
442 studies of turbulent particle-laden jets using spatial approach of one-
443 dimensional turbulence, in: ILASS-Europe 28th Conference on Liquid
444 Atomization and Spray Systems, 2017.
- 445 [24] S. Cao, T. Echehki, A low-dimensional stochastic closure model for com-
446 bustion large-eddy simulation 9. doi:10.1080/14685240701790714.
- 447 [25] R. C. Schmidt, A. R. Kerstein, S. Wunsch, V. Nilsen, Near-wall les
448 closure based on one-dimensional turbulence modeling 186 (1) (2003)
449 317–355. doi:10.1016/S0021-9991(03)00071-8.
- 450 [26] R. C. Schmidt, A. R. Kerstein, R. McDermott, Odtles: A multi-scale
451 model for 3d turbulent flow based on one-dimensional turbulence mod-
452 eling 199 (13-16) (2010) 865–880. doi:10.1016/j.cma.2008.05.028.
- 453 [27] E. Gonzalez-Juez, A. R. Kerstein, D. O. Lignell, Fluxes across double-
454 diffusive interfaces: a one-dimensional-turbulence study 677 (2011) 218–
455 254. doi:10.1017/jfm.2011.78.

- [28] E. Gonzalez-Juez, A. R. Kerstein, D. O. Lignell, Reactive rayleigh–taylor turbulent mixing: a one-dimensional-turbulence study 107 (5) (2013) 506–525. doi:10.1080/03091929.2012.736504.
- [29] S. Wunsch, A. R. Kerstein, A model for layer formation in stably stratified turbulence 13 (3) (2001) 702–712. doi:10.1063/1.1344182.
- [30] D. O. Lignell, V. B. Lansinger, J. Medina, M. Klein, A. R. Kerstein, H. Schmidt, M. Fistler, M. Oevermann, One-dimensional turbulence modeling for cylindrical and spherical flows: model formulation and application 32 (4) (2018) 495–520. doi:10.1007/s00162-018-0465-1.
- [31] M. Klein, D. O. Lignell, H. Schmidt, Map-based modeling of turbulent convection: Application of the one-dimensional turbulence model to planar and spherical geometries, in: International Conference on Rayleigh-Benard Turbulence, 2018.
- [32] M. Klein, D. O. Lignell, H. Schmidt, Stochastic modeling of temperature and velocity statistics in spherical-shell convection, in: EGU Conference on Recent developments in Geophysical Fluid Dynamics, 2019.
- [33] W. T. Ashurst, A. R. Kerstein, One-dimensional turbulence: Variable-density formulation and application to mixing layers 17 (2). doi:10.1063/1.1847413.
- [34] D. O. Lignell, A. R. Kerstein, G. Sun, E. I. Monson, Mesh adaption for efficient multiscale implementation of one-dimensional turbulence 27 (3-4) (2013) 273–295. doi:10.1007/s00162-012-0267-9.
- [35] Y. A. Çengel, J. M. Cimbala, Fluid Mechanics, 2nd Edition, Çengel series in engineering thermal-fluid sciences, McGraw-Hill Higher Education, 2010.
- [36] J. Beder, yaml-cpp v0.6.3 (2008).
URL <https://github.com/jbeder/yaml-cpp/>
- [37] A. C. Hindmarsh, R. Serban, D. R. Reynolds, CVODE, https://computing.llnl.gov/sites/default/files/public/cv_guide.pdf (2020).
URL <https://computing.llnl.gov/projects/sundials/cvode>
- [38] G. Strang, On the construction and comparison of difference schemes 5 (3) (1968) 506–517. doi:10.1137/0705041.

- [39] D. G. Goodwin, R. L. Speth, H. K. Moffat, B. W. Weber, Cantera (2018). doi:10.5281/zenodo.1174508.
URL <https://cantera.org/>
- [40] D. van Heesch, Doxygen (2018).
URL <https://www.doxygen.nl/>
- [41] A. B. Yoo, M. A. Jette, M. Grondona, Slurm: simple linux utility for resource management, in: D. G. Feitelson, L. Rudolph, U. Schwiegelshohn (Eds.), Job Scheduling Strategies For Parallel Processing, Vol. 2862 of Lecture notes in computer science, 0302-9743, Springer, 2003, pp. 44–60. doi:10.1007/10968987-3.
- [42] G. K. El Khoury, P. Schlatter, A. Noorani, P. F. Fischer, G. Brethouwer, A. V. Johansson, Direct numerical simulation of turbulent pipe flow at moderately high reynolds numbers 91 (3) (2013) 475–495. doi:10.1007/s10494-013-9482-8.
- [43] C. Chin, J. P. Monty, A. Ooi, Reynolds number effects in dns of pipe flow and comparison with channels and boundary layers 45 (2014) 33–40. doi:10.1016/j.ijheatfluidflow.2013.11.007.
- [44] H. J. Hussein, S. P. Capp, W. K. George, Velocity measurements in a high-reynolds-number, momentum-conserving, axisymmetric, turbulent jet 258 (1994) 31–75. doi:10.1017/S002211209400323X.
- [45] W. Meier, R. S. Barlow, Y.-L. Chen, J.-Y. Chen, Raman/Rayleigh/LIF measurements in a turbulent CH₄/H₂/N₂ jet diffusion flame: experimental techniques and turbulence–chemistry interaction 123 (3) (2000) 326–343. doi:10.1016/S0010-2180(00)00171-1.
URL <https://tnfworkshop.org/data-archives/simplejet/dlrflames/>
- [46] H. Pitsch, Unsteady flamelet modeling of differential diffusion in turbulent jet diffusion flames 123 (3) (2000) 358–374. doi:10.1016/S0010-2180(00)00135-8.
- [47] R. P. Lindstedt, H. Ozarovsky, Joint scalar transported pdf modeling of nonpiloted turbulent diffusion flames 143 (4) (2005) 471–490. doi:10.1016/j.combustflame.2005.08.030.
- [48] H. Wang, S. B. Pope, Large eddy simulation/probability density function modeling of a turbulent ch₄/h₂/n₂ jet flame 33 (1) (2011) 1319–1330. doi:10.1016/j.proci.2010.08.004.

- [49] M. Fairweather, R. M. Woolley, First-order conditional moment closure modeling of turbulent, nonpremixed methane flames 138 (1-2) (2004) 3–19. doi:10.1016/j.combustflame.2004.03.001.
- [50] K. W. Lee, D. H. Choi, Prediction of no in turbulent diffusion flames using eulerian particle flamelet model 12 (5) (2008) 905–927. doi:10.1080/13647830802094351.
- [51] K. W. Lee, D. H. Choi, Analysis of no formation in high temperature diluted air combustion in a coaxial jet flame using an unsteady flamelet model 52 (5-6) (2009) 1412–1420. doi:10.1016/j.ijheatmasstransfer.2008.08.015.

Current executable software version

Ancillary data table required for sub version of the executable software: (x.1, x.2 etc.) kindly replace examples in right column with the correct information about your executables, and leave the left column as it is.

Nr.	(Executable) software meta-data description	Please fill in this column
S1	Current software version	2.1
S2	Permanent link to executables of this version	For example: https://github.com/combogenomics/DuctApe/releases/tag/DuctApe-0.16.4
S3	Legal Software License	MIT
S4	Computing platforms/Operating Systems	Linux, OS X, Microsoft Windows
S5	Installation requirements & dependencies	CMake 3.12+, Cantera, Git, Doxygen (optional)
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	For example: http://mozart.github.io/documentation/
S7	Support email for questions	davidlignell@byu.edu

Table 2: Software metadata (optional)