

# SoftwareX

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

<b>Manuscript Number:</b>	SOFTX-D-20-00063R2
<b>Article Type:</b>	Original software publication
<b>Section/Category:</b>	Mathematical and Physical Sciences
<b>Keywords:</b>	turbulence; reacting flows; one-dimensional turbulence; Simulation
<b>Corresponding Author:</b>	David Lignell Brigham Young University Provo, Utah UNITED STATES
<b>First Author:</b>	David Lignell
<b>Order of Authors:</b>	David Lignell Victoria Stephens, BS
<b>Abstract:</b>	One-dimensional turbulence (ODT) is an accurate and computationally efficient technique for modeling turbulent flows. ODT has been applied to a wide range of flow problems including reaction, multiphase, differential diffusion, heat release, buoyancy, and wall flows. Applications include use as a standalone model and as a closure for large-eddy simulation (LES). Its strength lies in the ability to capture a full range of turbulent length and time scales. The ODT model is strongly coupled with its implementation, complicating its formulations. We present a modern, open-source, object- oriented C++ implementation of ODT. This code can be used as a starting point to understand, apply, and extend the ODT model, enabling its further application to turbulent flow research.
<b>Suggested Reviewers:</b>	Tony Saad, PhD Assistant Professor, University of Utah tony.saad@utah.edu  Randall J McDermott, PhD NIST randall.mcdermott@nist.gov  Yuxin Wu, PhD Tsinghua University wuyx09@mail.tsinghua.edu.cn
<b>Response to Reviewers:</b>	

# Authors Response to Editor Comments

- Editor comments are listed. Author responses are shown in blue text.

When I read the paper, I got confused on whether ODT is the model or the software. It seems like you use ODT to mean multiple things.

Look at the first sentence of the abstract and then the second last sentence. (also you say ODT is a technique and in other places, it is a model)

My understanding is that the software is an implementation of the ODT model in C++.

You need to make the distinction clearer, possibly calling the software a different name.

The content of the paper is fine but I think this ambiguity needs to be cleared up.

This is a fair point, and your understanding is correct that we are presenting a software implementation of the ODT model in C++. One-dimensional turbulence is a mathematical model, which we have written an implementation for. We used the name ODT for the code implementation for convenience (and consistency with our current users), but we certainly want to avoid any confusion. We have read through the whole text and made edits to clarify this (including in the abstract, as requested). In most places, we change “ODT” to “ODT code”, though the latter was often used already; however, the extension should make the usage more clear through consistent terminology. Similarly, we replace “ODT” with “ODT model” in several places. A few places have deleted “ODT” where it is not necessary (such as “... ODT build process” replaced with “... build process”). Several other instances should be clear, for example “which runs one realization of ODT,” or “ODT simulations can also be run in parallel.” In our resubmission, we have included a document that highlights differences between the old and new versions to make it easy to track the changes made.

Also, you did not include the names of journals in almost all of the references. For example, the first one should be  
 Proceedings of the Combustion Institute 35 (2015) 1199-1206

This has been corrected. We apologize for the oversight, which was due to an error in our LaTeX processing.

# 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

One-dimensional turbulence (ODT) is an accurate and computationally efficient model for simulating turbulent flows. ODT has been applied to a wide range of flow problems including reaction, multiphase, differential diffusion, heat release, buoyancy, and wall flows. Applications include use as a standalone model and as a closure for large-eddy simulation (LES). Its strength lies in the ability to capture a full range of turbulent length and time scales. The ODT model is strongly coupled with its implementation, complicating its formulations. We present a modern, open-source, object-oriented C++ implementation of ODT. The code described here and made available online can be used as a starting point to understand, apply, and extend the ODT model, enabling its further application to turbulent flow research.

*Key words:* turbulence, reacting flows, one-dimensional turbulence

---

---

\*Corresponding author.

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

## 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	<a href="https://github.com/BYUignite/ODT">https://github.com/BYUignite/ODT</a>
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	ODT style guide
C9	Support email for questions	davidlignell@byu.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 researchers with valuable insights into complex systems, particularly reacting turbulent flows such as combustion processes. Turbulence is a complex phenomenon that affects the full range of a flow’s length and time scales. As a result, resolving the entire flow field by numerically solving the Navier-Stokes equations of fluid flow, as is done in direct numerical simulations (DNS), requires substantial computational resources. DNS is a powerful research tool, but its high computational cost makes it intractable for simulating most practical engineering flows. In order to achieve numerical solutions to practical flow problems, researchers use alternative frameworks that model turbulence rather than resolving it directly.

Large-eddy simulations (LES) address the problem of wide-ranging length and time scales by combining direct resolution of grid-scale quantities, as in DNS, with subgrid modeling of smaller turbulence structures. The more complex the flow, the more modeling is required; for example, a jet flame simulation might require subgrid modeling for the combustion chemistry, radiative heat transfer, and soot chemistry in addition to turbulence structures, all of which form a tightly coupled system in which each model interacts heavily with the others. While subgrid modeling makes LES more computationally

affordable than DNS, it can introduce empiricism into simulations, which can lead to inaccurate results. Additionally, unresolved quantities are often parameterized in state space with empirical relationships or assumed distributions that lack universal applicability. LES is a valuable simulation tool, but its approach to turbulence modeling can introduce unwanted empiricism and make errors difficult to isolate and quantify.

The one-dimensional turbulence model (ODT) functionally reverses the LES approach, modeling large-scale turbulent advection and directly resolving small-scale flow structures, simulating the full range of length and time scales in a single dimension. Depending on the configuration, large-scale structures can be easier to model than small-scale structures, especially in reacting flows. ODT's direct resolution of fine scales in one dimension bypasses subgrid modeling issues that complicate LES. Previous studies show that ODT can attain accuracy comparable to DNS at a fraction of the computational cost [1, 2], making it an attractive tool for simulating turbulent flows. Because the ODT model is one-dimensional, it is limited to homogeneous or boundary-layer flows, such as jets, wakes, and mixing layers, but these types of flows are common in nature and central to turbulence research. ODT's computational efficiency and resolution of a full range of scales make it a valuable tool that complements experimental studies and other simulation tools like DNS and LES.

Early applications of ODT focused on homogeneous turbulence, wakes, and mixing layers [3–5]. Later extension to variable-density flows and a spatial downstream coordinate system facilitated its growth and application to more complex flows, including combustion in jet flames [6–13], counterflow flames [14], wall fires [15], and sooting flames [1, 16–19], as well as other particle flows [20–23]. ODT has also served to complement LES through subgrid modeling studies [24–26] 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, ODT implementations have been extended to include cylindrical and spherical coordinate systems [30–32].

Many of the references cited above make use of a particular software implementation of the ODT model, referred to here as the *ODT code*. During the recent implementation of the cylindrical and spherical model formulations, this code was drastically overhauled and reorganized. 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 case in Section 3 is 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 formulation is described in detail in the literature [3, 5, 30, 33, 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 implementation of the ODT model 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  $\beta$ :

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

The first and second terms on the right-hand-side of this equation are the diffusive transport and source terms, respectively. Here, the term *diffusive* is not specific to mass transfer, but may refer to the transport of, e.g., heat, mass, and momentum, depending on the identity of the scalar  $\beta$ . In practice, we refer to the transport term as the “mixing term”. The symbols shown in Equation 1 are defined as follows:  $j_\beta$  is the diffusion flux of scalar  $\beta$  across the cell face area  $A_x$ , where the subscripts  $e$  and  $w$  refer to the “east” and “west” faces of a given grid cell, respectively;  $S_\beta$  is the Lagrangian source term derived from the conservation law for  $\beta$ ;  $\rho$  represents mass density; and  $V$  represents cell volume. The generic transport equation differs slightly in the spatial formulation of ODT, but its form is similar, 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 [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  $\beta_{LES}$  constrains eddies to those with timescales proportional to the elapsed simulation time. Sampled eddies that do not fit the defined parameters are rejected and not applied to the domain.

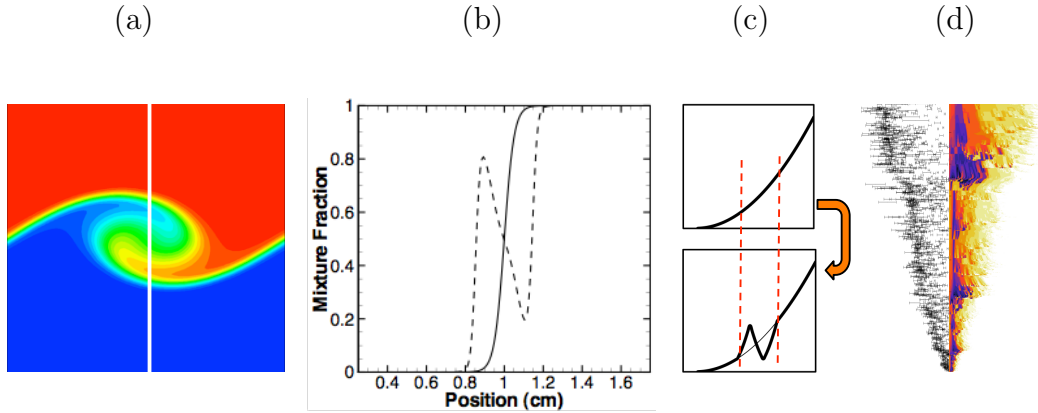


Figure 1: Schematic diagram of a triplet map. From left to right: (a) simulation of a single “eddy”, a two-dimensional Kelvin-Helmholtz instability with a vertical white line-of-sight; (b) mixture fraction profile along the indicated line-of-sight before and after the eddy showing increased gradients and surface area; (c) a triplet map operates on a profile to produce a mapped profile with three copies, compressed and lined up, with the middle copy spatially inverted; (d) illustrative eddy event sizes and locations with corresponding temperature profile for a single ODT realization of a wall fire, adapted from [15].

Eddy events modify domain variables using “triplet maps”, as illustrated in Figure 1. In an eddy region of size  $l$ , transported property profiles are modified (mapped) by making three copies, compressing each spatially by a factor of three, and realigning them with the middle copy spatially reversed. This map is continuous, conserves all quantities and statistical moments, increases scalar gradients, and decreases length scales consistent with turbulent eddies in real flows. Subsequent eddies in the same region will result in a cascade of scales. The eddy rates, used to sample and select eddy events, depend on eddy size and the local kinetic energy consistent with turbulent 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 the simulation 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 then gather statistics over these realizations. This is done via post-processing tools. Post-processing is specific to a given case and the needs of the user. Examples of post-processing files are included for different cases provided with the ODT code, as discussed further in Section 2.3.4.

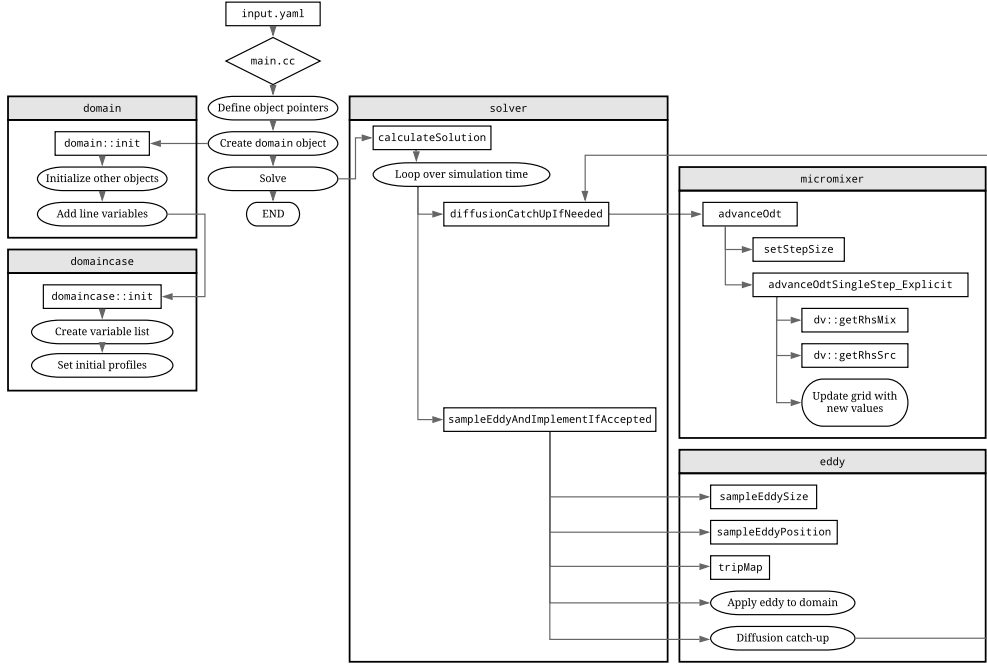


Figure 2: Structural outline of the ODT code, including its major class objects. For brevity, this diagram makes two simplifications. First, it assumes that diffusive advancement in the `micromixer` uses an explicit solution method; in practice, users specify either an explicit, semi-implicit, or Strang 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.

## 2.2. Software Architecture

The ODT code 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 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. 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**: a first-order explicit Euler method (pictured in Figure 2), a first-order semi-implicit method, and a second order Strang splitting method [37]. The latter two methods are useful for treating stiff chemistry, with source terms integrated using CVODE [38]. In reacting flow cases, enthalpy and chemical species mass fractions are transported, and physical properties and kinetic rates are computed with Cantera [39]. The **micromixer** is also the code’s primary point of interaction with the **mesher** object (not pictured in Figure 2), which manages the adaptive grid functions. Finally, the **eddy** object manages eddy event selection and implementation, which proceeds as described in Section 2.1.

### 2.3. Workflow

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

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

#### 2.3.1. Building the ODT code

The ODT code build process is automated with CMake. The user navigates to the **build** directory and edits the CMake configuration file. The CMake configuration file specifies Cantera’s location and must be changed to reflect the local installation location. For reacting flows, the chemical reaction rates can be computed by Cantera (default), or a from a user-specified

function, specified at compile time. Once the configuration settings are updated, the user runs CMake to apply the changes.

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

Once CMake and YAML have been prepared, the user can build the ODT code with the `make` command, which places the code executable in the `run` directory. The most common errors that occur during the build process involve incorrect file paths or incomplete installation of required packages. See the build documentation for troubleshooting help.

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

### 2.3.2. *Input files*

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

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

Input files contain simulation parameters in a human-readable format parsed by YAML. Within `input.yaml`, parameters are organized into sections, several of which are common to all input files. Parameter values present in the provided input files may be considered defaults that can be used to run a successful simulation of that case type. General code parameters are stored in the `params` class. Variables needed in a simulation but not provided in an input file (not recommended), are either given defaults specified in the `params` class or result in an error message. Details about individual parameters, including usage and typical values, are covered in the documentation.

### 2.3.3. *Running the ODT code*

To run a simulation, users must then navigate to the `run` directory, which contains the `odt.x` executable and several possible run scripts. The simplest

option is `runOneRlz.sh`, which runs one realization of ODT in the specified configuration. In this run script, the user must alter two variables near the top of the file: `inputDir`, which specifies which input directory and files to use; and `caseName`, which provides a name for the simulation and the data files it outputs. The `runManyRlz.sh` script runs many realizations in serial, one after the other; it differs from `runOneRlz.sh` only in that the user must also alter the `nRlz` variable, which specifies the number of realizations to run. Each realization has a different seed value, which can be randomized or directly specified in the input file. To run the simulation with either `runOneRlz.sh` or `runManyRlz.sh`, 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 (embarrassingly parallel), 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, including eddy event information; `data` contains the raw data files; and `post` contains post-processed data files once they are generated.

To use post-processing tools, users 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 Jupyter notebooks, 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.

As part of the post-processing tools, there is a `post/tools` directory that contains two files: `data.py.py` and `data.tools.py`. The first of these can be

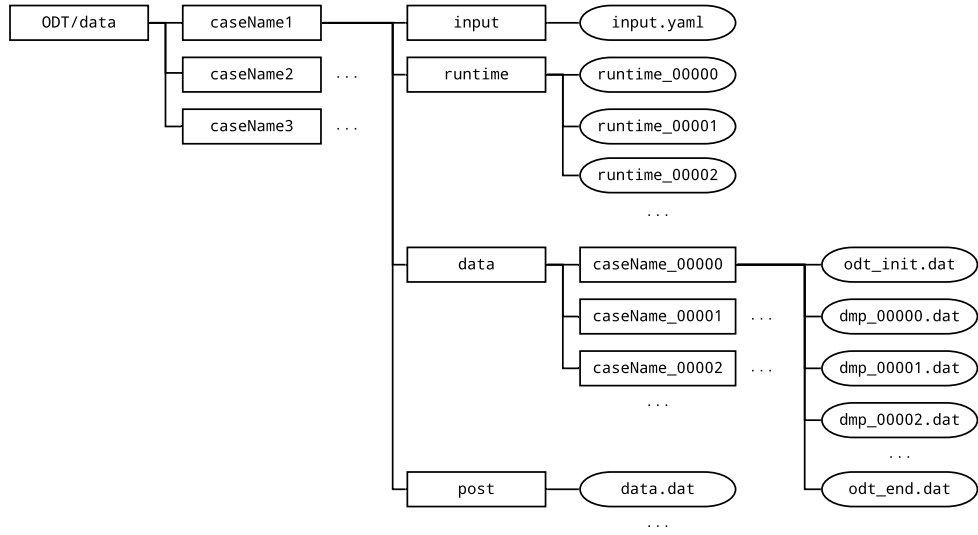


Figure 3: Data folder structure and file locations. Boxes indicate folders and ovals indicate files. For a given simulation, **caseName** will be replaced with the case name specified in the run script. The number attached to **caseName** folder names and to **runtime** data files indicates the realization number. The number attached to **dmp** files indicates the index of the list of “dump” times included in, or generated from the information in **input.yaml**.

used to convert the text data files generated during a run into binary numpy files. This can reduce both the number of data files and the file sizes. The number of files is reduced since all code realizations at a given output time are saved in the same binary file. The `data_tools.py` file is a module that contains a number of functions for processing the data files, such as functions to

- get the data from a given realization,
- return the number of realizations,
- get a list of the variable names in the data files,
- compute axial locations in the spatial ODT formulation,
- query a parameter from the input file,
- return the stoichiometric mixture fraction for combustion/mixing problems, and
- compute the domain bounds.

In addition, a few functions are provided for doing interpolation/extrapolation, and computing probability density functions (PDFs), and weighted PDFs. Many of these functions take an argument that is a dictionary of case-specific information, such as the case name and paths for key directories. This dictionary is set in the post-processing driver script based only on the user-specified case name.

### 3. Example Case: Canonical Jet Flame

The ODT model is uniquely suited to reacting flow simulations. Here, we present illustrative ODT simulation results of a round, turbulent jet flame based on and compared to the experimental DLR-A flame of Meier et al. [42]. This canonical flame configuration has been used extensively to study and validate turbulent combustion models [43–48].

The DLR-A fuel stream is mixture of 22.1%  $\text{CH}_4$ , 33.2%  $\text{H}_2$ , and 44.7%  $\text{N}_2$  (by volume) that issues into dry air via a nozzle with an inner diameter of 8 mm at a mean exit velocity of  $42.2 \text{ m}\cdot\text{s}^{-1}$ . The coflow air stream issues from a concentric nozzle 140 mm in diameter at a velocity of  $0.3 \text{ m}\cdot\text{s}^{-1}$ . The reported jet Reynolds number is 15,200.

Previous ODT simulations of turbulent jet flames have used the temporal planar formulation, but the spatial cylindrical formulation developed recently

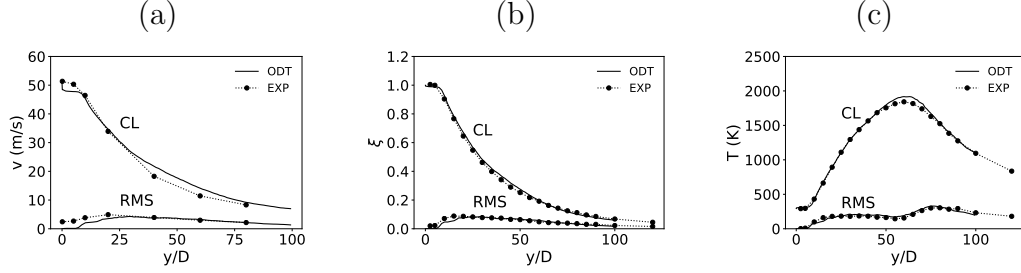
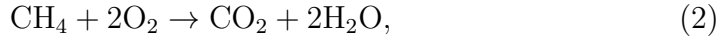


Figure 4: DLR jet flame example case results: (a) mean axial velocity and RMS velocity along the centerline versus downstream location; (b) mean axial mixture fraction and RMS mixture fraction along the centerline versus downstream location; (c) mean axial temperature (K) and RMS temperature (K) along the centerline versus downstream location.

[30] more closely matches the experimental configuration. This simulation uses the experimentally reported velocity profiles and jet dimensions. The fuel was diluted with  $N_2$  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  $O_2$ ,  $N_2$ ,  $CH_4$ ,  $H_2$ ,  $H_2O$ , and  $CO_2$ . We assume that reactions proceed to the products of complete combustion and apply simple, fast reaction rates according to the following chemical equations:



These assumptions are reasonable for illustrating ODT in the DLR-A reacting jet configuration with variable properties and heat release.

This simulation uses ODT parameters  $C = 20$ ,  $\beta_{LES} = 17$ , and  $Z = 400$ . The values of  $C$  and  $\beta_{LES}$  were adjusted to give good agreement with the experimental data, and the value of  $Z$  is the same as the spatial simulations in [15]. 1024 independent flow realizations were performed in parallel and the results ensemble averaged. Downstream distance  $y$  and radial position  $r$  are normalized by the jet diameter  $D$ .

Figure 4 displays the simulation results: axial mean and centerline values for (a) axial velocity  $v$ , (b) mixture fraction  $\xi$ , and (c) temperature  $T$ . The ODT simulation results track the experimental data well for all three variables. The centerline temperature peaks about 100 K above the experimental data, but this small difference can be attributed to thermal radiation (which was neglected in this simulation) and the assumption that reactions proceed

to the products of complete combustion rather than their equilibrium state. The centerline velocity shows a fast initial decrease that can be attributed to diffusion. Similarly, the increase in centerline velocity RMS values is delayed; this occurs due to the elapsed time model for large eddy suppression, which limits disturbances in the early stages of the flow to small eddies that occur on the jet edges away from the centerline.

To replicate this example case, build the code with the `CHEMISTRY = SIMPLEDLR` flag in the `user_config` file and edit the desired run script with `inputDir = "../input/jetFlame/DLR_A"` and a new case name such as `caseName = "jetFlame_example"`. The input file for this case, located at `ODT/input/jetFlame/DLR_A/input.yaml`, contains the appropriate parameters and does not need to be modified to match this example case. See the code documentation for data post-processing instructions.

## 4. Impact

The ODT model is complex, challenging to implement, and difficult to cleanly separate from its implementation, all of which presents barriers to practitioners who wish to use the model. For instance, triplet maps can be represented mathematically but are usually implemented as previously described, assuming an adaptive grid with variable cell sizes. Alternative formulations are used for fixed grids. The ODT code implementation presented here reduces these barriers by providing an open-source, well-documented, modularly styled code that can be used directly or modified to suit particular research goals. Its structure facilitates addressing new research questions, such as soot-flame-radiation-chemistry interactions, multi-phase reacting flows using Lagrangian particle models (such as diesel sprays), and coupled near-wall heat and momentum transfer. The code presented here has been extended to and demonstrated in such systems already, facilitated by the object-oriented structure. For example, this code has been extended to solve steady laminar flamelets [49], requiring inheritance from the `solver`, `micromixer`, and `dv` (domain variable) classes.

The structure of the ODT code, using an adaptive mesh and a Lagrangian formulation for diffusive advancement of transport equations, has several advantages, including simplicity of formulation and computational efficiency. The adaptive mesh minimizes the number of computational cells required and can be adapted to any variable of interest. This is especially important for flows with complex and expensive chemistry and flows with widely varying resolution demands such as near-wall flows. The Lagrangian formulation implies that cell faces move with fluid velocity (such as is present during flame expansion, for example). This treatment is inherently stable, removing

the need for complex and diffusive hyperbolic conservation law treatments and their associated CFL constraints. The conservation equations, as in Equation 1, are similarly simplified.

The code has been run on Linux, Mac, and Windows machines, including parallel clusters of up to 512 processor cores. Several research groups have used the code and contributed to its development. These include groups at Sandia National Laboratories, Brandenburg University of Technology Cottbus-Senftenberg, Chalmers University, and Combustion Science and Engineering, Inc. The latter company is the only organization known to use the code in a commercial setting, though on a combustion research project.

## 5. Conclusion

A clean and extensible implementation of the one-dimensional turbulence (ODT) model has been presented in an open-source, object-oriented C++ framework, motivated by advanced turbulent reacting flow research. The build process, directory structure, input file format, run process, and post-processing procedure were described in detail, including a representative reacting jet example case. Access to a proven, documented code and a modern build procedure will aid researchers in applying and extending the model to new research questions.

## 6. Conflict of Interest

There are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

## Acknowledgements

Special thanks to Alan R. Kerstein for his significant contributions to model development. Additional thanks to Heiko Schmidt, Juan Medina, and Marten Klein of Brandenburg University of Technology Cottbus-Senftenberg, Michael Oevermann and Marco Fistler of Chalmers University of Technology, and John Hewson of Sandia National Laboratories for code contributions, and insightful conversations.



## References

- [1] D. O. Lignell, G. C. Fredline, A. D. Lewis, Comparison of one-dimensional turbulence and direct numerical simulations of soot formation and transport in a nonpremixed ethylene jet flame, *Proceedings of the Combustion Institute* 35 (2) (2015) 1199–1206. doi:10.1016/j.proci.2014.05.046.
- [2] A. W. Abboud, C. Schulz, T. Saad, S. T. Smith, D. D. Harris, D. O. Lignell, A numerical comparison of precipitating turbulent flows between large-eddy simulation and one-dimensional turbulence, *AIChE Journal* 61 (10) (2015) 3185–3197. doi:10.1002/aic.14870.
- [3] A. R. Kerstein, One-dimensional turbulence: model formulation and application to homogeneous turbulence, shear flows, and buoyant stratified flows, *Journal of Fluid Mechanics* 392 (1999) 277–334. doi:10.1017/S0022112099005376.
- [4] A. R. Kerstein, T. D. Dreeben, Prediction of turbulent free shear flow statistics using a simple stochastic model, *Physics of Fluids* 12 (2) (2000) 418–424. doi:10.1063/1.870319.
- [5] A. R. Kerstein, W. T. Ashurst, S. Wunsch, V. Nilsen, One-dimensional turbulence: vector formulation and application to free shear flows, *Journal of Fluid Mechanics* 447 (2001) 85–109. doi:10.1017/S0022112001005778.
- [6] T. Echehki, A. R. Kerstein, T. D. Dreeben, J.-Y. Chen, ‘one-dimensional turbulence’ simulation of turbulent jet diffusion flames: model formulation and illustrative applications, *Combustion and Flame* 125 (3) (2001) 1083–1105. doi:10.1016/S0010-2180(01)00228-0.
- [7] J. C. Hewson, A. R. Kerstein, Stochastic simulation of transport and chemical kinetics in turbulent  $\text{CH}_4/\text{H}_2/\text{N}_2$  flames, *Combustion Theory and Modelling* 5 (4) (2001) 669–697. doi:10.1088/1364-7830/5/4/309.
- [8] J. C. Hewson, A. R. Kerstein, Local extinction and reignition in non-premixed turbulent  $\text{CH}_4/\text{H}_2/\text{N}_2$  jet flames, *Combustion Science and Technology* 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, *Combustion*

- and Flame 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<sub>2</sub> jets with extinction and reignition, *Proceedings of the Combustion Institute* 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, *Zeitschrift für Physikalische Chemie* 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, *Proceedings of the Combustion Institute* 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, *Combustion and Flame* 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, *Fire Technology* 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 for soot evolution in fire, in: H. Bockhorn, A. D’Anna, A. F. Sarofim, H. Wang (Eds.), *Combustion Generated Fine Carbonaceous Particles*, KIT Scientific Publishing, 2009, pp. 571–588.

- [18] D. O. Lignell, J. C. Hewson, One-dimensional turbulence simulation: overview and application to soot formation in nonpremixed flames, in: SIAM Conference on Computational Science and Engineering, 2015.
- [19] A. J. Ricks, J. C. Hewson, A. R. Kerstein, J. P. Gore, S. R. Tieszen, W. T. Ashurst, A spatially developing one-dimensional turbulence (ODT) study of soot and enthalpy evolution in meter-scale buoyant turbulent flames, *Combustion Science and Technology* 182 (1) (2010) 60–101. doi:10.1080/00102200903297003.
- [20] G. Sun, J. C. Hewson, D. O. Lignell, Evaluation of stochastic particle dispersion modeling in turbulent round jets, *International Journal of Multiphase Flow* 89 (2017) 108–122. doi:10.1016/j.ijmultiphaseflow.2016.10.005.
- [21] J. R. Schmidt, J. O. L. Wendt, A. R. Kerstein, Non-equilibrium wall deposition of inertial particles in turbulent flow, *Journal of Statistical Physics* 137 (2) (2009) 233–257. doi:10.1007/s10955-009-9844-8.
- [22] G. Sun, D. O. Lignell, J. C. Hewson, C. R. Gin, Particle dispersion in homogeneous turbulence using the one-dimensional turbulence model, *Physics of Fluids* 26 (10) (2014) 103301. doi:10.1063/1.4896555.
- [23] M. Fistler, D. O. Lignell, A. R. Kerstein, M. Oevermann, Numerical studies of turbulent particle-laden jets using spatial approach of one-dimensional turbulence, in: ILASS-Europe 28th Conference on Liquid Atomization and Spray Systems, 2017.
- [24] S. Cao, T. Echehki, A low-dimensional stochastic closure model for combustion large-eddy simulation, *Journal of Turbulence* 9. doi:10.1080/14685240701790714.
- [25] R. C. Schmidt, A. R. Kerstein, S. Wunsch, V. Nilsen, Near-wall LES closure based on one-dimensional turbulence modeling, *Journal of Computational Physics* 186 (1) (2003) 317–355. doi:10.1016/S0021-9991(03)00071-8.
- [26] R. C. Schmidt, A. R. Kerstein, R. McDermott, ODTLES: A multi-scale model for 3D turbulent flow based on one-dimensional turbulence modeling, *Computer Methods in Applied Mechanics and Engineering* 199 (13-16) (2010) 865–880. doi:10.1016/j.cma.2008.05.028.

- [27] E. Gonzalez-Juez, A. R. Kerstein, D. O. Lignell, Fluxes across double-diffusive interfaces: a one-dimensional-turbulence study, *Journal of Fluid Mechanics* 677 (2011) 218–254. 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, *Geophysical & Astrophysical Fluid Dynamics* 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, *Physics of Fluids* 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, *Theoretical and Computational Fluid Dynamics* 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, *Physics of Fluids* 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, *Theoretical and Computational Fluid Dynamics* 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] G. Strang, On the construction and comparison of difference schemes, SIAM Journal on Numerical Analysis 5 (3) (1968) 506–517. doi:10.1137/0705041.
- [38] A. C. Hindmarsh, R. Serban, D. R. Reynolds, CVODE, [https://computing.llnl.gov/sites/default/files/public/cv\\_guide.pdf](https://computing.llnl.gov/sites/default/files/public/cv_guide.pdf) (2020).  
URL <https://computing.llnl.gov/projects/sundials/cvode>
- [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] W. Meier, R. S. Barlow, Y.-L. Chen, J.-Y. Chen, Raman/Rayleigh/LIF measurements in a turbulent CH<sub>4</sub>/H<sub>2</sub>/N<sub>2</sub> jet diffusion flame: experimental techniques and turbulence–chemistry interaction, Combustion and Flame 123 (3) (2000) 326–343. doi:10.1016/S0010-2180(00)00171-1.  
URL <https://tnfworkshop.org/data-archives/simplejet/dlrflames/>
- [43] H. Pitsch, Unsteady flamelet modeling of differential diffusion in turbulent jet diffusion flames, Combustion and Flame 123 (3) (2000) 358–374. doi:10.1016/S0010-2180(00)00135-8.
- [44] R. P. Lindstedt, H. Ozarovskiy, Joint scalar transported PDF modeling of nonpiloted turbulent diffusion flames, Combustion and Flame 143 (4) (2005) 471–490. doi:10.1016/j.combustflame.2005.08.030.
- [45] H. Wang, S. B. Pope, Large eddy simulation/probability density function modeling of a turbulent CH<sub>4</sub>/H<sub>2</sub>/N<sub>2</sub> jet flame, Proceedings of the Combustion Institute 33 (1) (2011) 1319–1330. doi:10.1016/j.proci.2010.08.004.
- [46] M. Fairweather, R. M. Woolley, First-order conditional moment closure modeling of turbulent, nonpremixed methane flames, Combustion and

539 Flame 138 (1-2) (2004) 3–19. doi:10.1016/j.combustflame.2004.03.  
540 001.

541 [47] K. W. Lee, D. H. Choi, Prediction of NO in turbulent diffusion flames  
542 using eulerian particle flamelet model, Combustion Theory and Mod-  
543 elling 12 (5) (2008) 905–927. doi:10.1080/13647830802094351.

544 [48] K. W. Lee, D. H. Choi, Analysis of NO formation in high temperature  
545 diluted air combustion in a coaxial jet flame using an unsteady flamelet  
546 model, International Journal of Heat and Mass Transfer 52 (5-6) (2009)  
547 1412–1420. doi:10.1016/j.ijheatmasstransfer.2008.08.015.

548 [49] N. Peters, Laminar diffusion flamelet models in non-premixed turbulent  
549 combustion, Progress in Energy and Combustion Science 10 (1984) 319–  
550 339.

## 551 Current executable software version

552 Software information is provided in Table 2.

Nr.	(Executable) software meta-data description	Please fill in this column
S1	Current software version	1.0
S2	Permanent link to executables of this version	For example: <a href="https://github.com/BYUignite/ODT">https://github.com/BYUignite/ODT</a>
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	<a href="https://github.com/BYUignite/ODT">https://github.com/BYUignite/ODT</a>
S7	Support email for questions	davidlignell@byu.edu

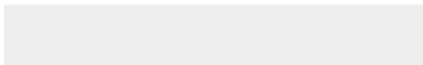
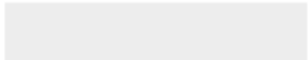
Table 2: Software metadata (optional)

[Click here to view linked References](#)



[Click here to access/download](#)

**LaTeX Source File**  
elsarticle-num.bst



[Click here to view linked References](#)



[Click here to access/download](#)

**LaTeX Source File**  
elsarticle.cls





## Conflict of Interest

There are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

# 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

One-dimensional turbulence (ODT) is an accurate and computationally efficient ~~technique~~ [model](#) for ~~modeling~~ [simulating](#) turbulent flows. ODT has been applied to a wide range of flow problems including reaction, multiphase, differential diffusion, heat release, buoyancy, and wall flows. Applications include use as a standalone model and as a closure for large-eddy simulation (LES). Its strength lies in the ability to capture a full range of turbulent length and time scales. The ODT model is strongly coupled with its implementation, complicating its formulations. We present a modern, open-source, object-oriented C++ implementation of ODT. ~~This code can be used as a starting point to understand, apply, and extend the ODT model, enabling its further application to turbulent flow research.~~ The code described here and made available online can be used as a starting point to understand, apply, and extend the ODT model, enabling its further application to turbulent flow research.

*Keywords:* turbulence, reacting flows, one-dimensional turbulence

---

---

\*Corresponding author.

*Email address:* [davidlignell@byu.edu](mailto:davidlignell@byu.edu) (David O. Lignell)

## 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	<a href="https://github.com/BYUignite/ODT">https://github.com/BYUignite/ODT</a>
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	ODT style guide
C9	Support email for questions	davidlignell@byu.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 researchers with valuable insights into complex systems, particularly reacting turbulent flows such as combustion processes. Turbulence is a complex phenomenon that affects the full range of a flow's length and time scales. As a result, resolving the entire flow field by numerically solving the Navier-Stokes equations of fluid flow, as is done in direct numerical simulations (DNS), requires substantial computational resources. DNS is a powerful research tool, but its high computational cost makes it intractable for simulating most practical engineering flows. In order to achieve numerical solutions to practical flow problems, researchers use alternative frameworks that model turbulence rather than resolving it directly.

Large-eddy simulations (LES) address the problem of wide-ranging length and time scales by combining direct resolution of grid-scale quantities, as in DNS, with subgrid modeling of smaller turbulence structures. The more complex the flow, the more modeling is required; for example, a jet flame simulation might require subgrid modeling for the combustion chemistry, radiative heat transfer, and soot chemistry in addition to turbulence structures, all of which form a tightly coupled system in which each model interacts heavily with the others. While subgrid modeling makes LES more computationally

22 affordable than DNS, it can introduce empiricism into simulations, which  
 23 can lead to inaccurate results. Additionally, unresolved quantities are often  
 24 parameterized in state space with empirical relationships or assumed distri-  
 25 butions that lack universal applicability. LES is a valuable simulation tool,  
 26 but its approach to turbulence modeling can introduce unwanted empiricism  
 27 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. Depending on the configuration, large-scale  
 32 structures can be easier to model than small-scale structures, especially in  
 33 reacting flows. ODT’s direct resolution of fine scales in one dimension by-  
 34 passes subgrid modeling issues that complicate LES. Previous studies show  
 35 that ODT can attain accuracy comparable to DNS at a fraction of the com-  
 36 putational cost [1, 2], making it an attractive tool for simulating turbulent  
 37 flows. Because the ODT model is one-dimensional, it is limited to homoge-  
 38 neous or boundary-layer flows, such as jets, wakes, and mixing layers, but  
 39 these types of flows are common in nature and central to turbulence research.  
 40 ODT’s computational efficiency and resolution of a full range of scales make it  
 41 a valuable tool that complements experimental studies and other simulation  
 42 tools like DNS and LES.

43 Early applications of ODT focused on homogeneous turbulence, wakes,  
 44 and mixing layers [3, 4, 5]. Later extension to variable-density flows and a  
 45 spatial downstream coordinate system facilitated its growth and application  
 46 to more complex flows, including combustion in jet flames [6, 7, 8, 9, 10,  
 47 11, 12, 13], counterflow flames [14], wall fires [15], and sooting flames [1, 16,  
 48 17, 18, 19], as well as other particle flows [20, 21, 22, 23]. ODT has also  
 49 served to complement LES through subgrid modeling studies [24, 25, 26] and  
 50 has been applied to various other flow configurations such as double-diffusive  
 51 interfaces [27], Rayleigh-Taylor mixing [28], and stratified turbulence [29].  
 52 Most recently, ~~the ODT code was~~ ODT implementations have been extended  
 53 to include cylindrical and spherical coordinate systems [30, 31, 32].

54 ~~During the recent implementation of the cylindrical and spherical model~~  
 55 ~~formulations, the ODT code was drastically overhauled and reorganized,~~  
 56 ~~resulting in its current configuration.~~ Many of the references cited above  
 57 make use of a particular software implementation of the ODT model, referred  
 58 to here as the *ODT code*. During the recent implementation of the cylindri-  
 59 cal and spherical model formulations, this code was drastically overhauled  
 60 and reorganized. The ODT code presented here is a pared down version  
 61 of the development code, representing the fundamental aspects of the ODT  
 62 model and its most reliable functions. The example case in Section 3 is 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 [formulation](#) 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 [implementation of the ODT model](#) 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  $\beta$ :

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

The first and second terms on the right-hand-side of this equation are the diffusive transport and source terms, respectively. Here, the term *diffusive* is not specific to mass transfer, but may refer to the transport of, e.g., heat, mass, and momentum, depending on the identity of the scalar  $\beta$ . In practice, we refer to the transport term as the “mixing term”. The symbols shown in Equation 1 are defined as follows:  $j_{\beta}$  is the diffusion flux of scalar  $\beta$  across the cell face area  $A_x$ , where the subscripts  $e$  and  $w$  refer to the “east” and “west” faces of a given grid cell, respectively;  $S_{\beta}$  is the Lagrangian source term derived from the conservation law for  $\beta$ ;  $\rho$  represents mass density; and  $V$  represents cell volume. The generic transport equation differs slightly in the spatial formulation of ODT, but its form is similar, 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 [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

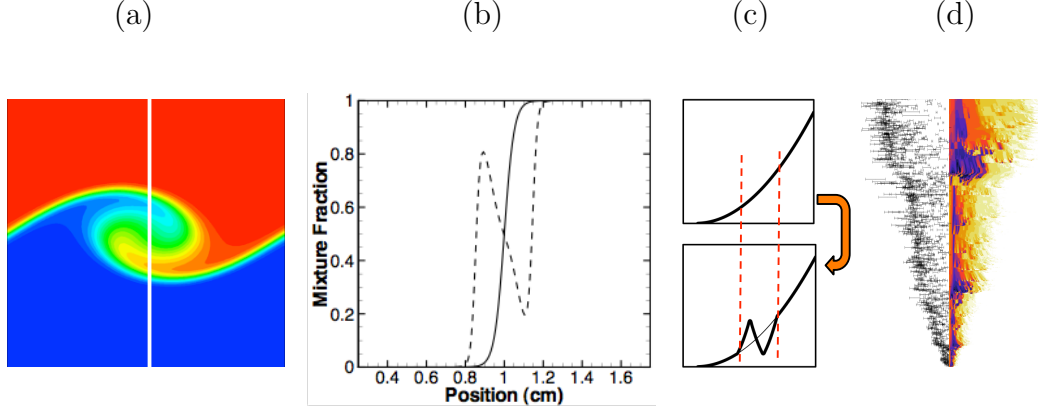


Figure 1: Schematic diagram of a triplet map. From left to right: (a) simulation of a single “eddy”, a two-dimensional Kelvin-Helmholtz instability with a vertical white line-of-sight; (b) mixture fraction profile along the indicated line-of-sight before and after the eddy showing increased gradients and surface area; (c) a triplet map operates on a profile to produce a mapped profile with three copies, compressed and lined up, with the middle copy spatially inverted; (d) illustrative eddy event sizes and locations with corresponding temperature profile for a single ODT realization of a wall fire, adapted from [15].

eddy events; and the large eddy suppression parameter  $\beta_{LES}$  constrains eddies to those with timescales proportional to 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 in Figure 1. In an eddy region of size  $l$ , transported property profiles are modified (mapped) by making three copies, compressing each spatially by a factor of three, and realigning them with the middle copy spatially reversed. This map is continuous, conserves all quantities and statistical moments, increases scalar gradients, and decreases length scales consistent with turbulent eddies in real flows. Subsequent eddies in the same region will result in a cascade of scales. The eddy rates, used to sample and select eddy events, depend on eddy size and the local kinetic energy consistent with turbulent 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 the simulation 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

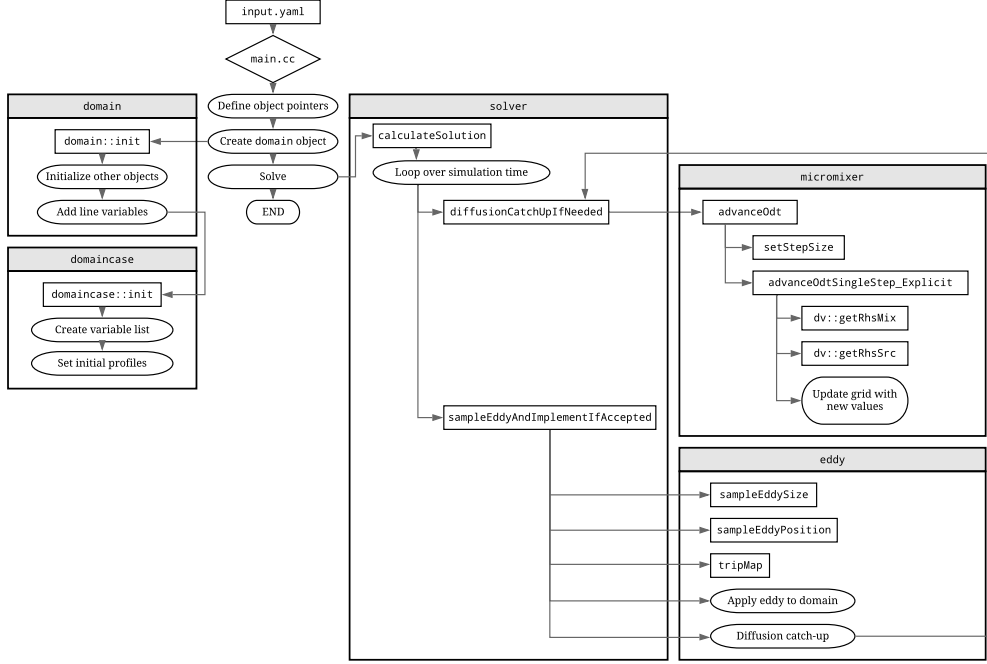


Figure 2: Structural outline of the ODT code, including its major class objects. For brevity, this diagram makes two simplifications. First, it assumes that diffusive advancement in the `micromixer` uses an explicit solution method; in practice, users specify either an explicit, semi-implicit, or Strang 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.

121 then gather statistics over these realizations. This is done via post-processing  
 122 tools. Post-processing is specific to a given case and the needs of the user.  
 123 Examples of post-processing files are included for different cases provided  
 124 with the ODT code, as discussed further in Section 2.3.4.

## 125 2.2. Software Architecture

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

131 Figure 2 illustrates the ODT code’s most important objects and structural  
 132 features. User inputs are provided to the executable in YAML [36] format via

133 `input.yaml`; the location of the specific `input.yaml` file to be used is deter-  
134 mined by the case name and case type specified in the run script. The `main`  
135 function defines storage for the main objects, but, once created, the `domain`  
136 object is responsible for object initialization as well as variable initialization  
137 and storage via a case-specific `domaincase` object.

138 Three primary objects handle the code’s main functions: the `solver`,  
139 `micromixer`, and `eddy` objects. The `solver` coordinates the ODT solution  
140 process, marching along the simulation time and invoking diffusive advance-  
141 ment and eddy events. The `micromixer` handles diffusive advancement by  
142 setting step sizes, interacting with the transported domain variables, and  
143 solving the system of ODEs defined by Equation 1 (or its equivalent in the  
144 spatial formulation). The `micromixer` includes three solution methods that  
145 can be specified in `input.yaml`: a first-order explicit Euler method (pic-  
146 tured in Figure 2), a first-order semi-implicit method, and a second order  
147 Strang splitting method [37]. The latter two methods are useful for treating  
148 stiff chemistry, with source terms integrated using CVODE [38]. In reacting  
149 flow cases, enthalpy and chemical species mass fractions are transported, and  
150 physical properties and kinetic rates are computed with Cantera [39]. The  
151 `micromixer` is also the code’s primary point of interaction with the `mesher`  
152 object (not pictured in Figure 2), which manages the adaptive grid functions.  
153 Finally, the `eddy` object manages eddy event selection and implementation,  
154 which proceeds as described in Section 2.1.

### 155 2.3. Workflow

156 This section outlines the process a user goes through in order to suc-  
157 cessfully build the ODT code and run a simulation. The ODT `code` is a  
158 standalone, self-contained package, and users interact with its files primarily  
159 via the command line rather than a graphical interface. For more details,  
160 please see the package documentation.

161 Within the main download package, several directories organize the ODT  
162 code. The `source`, `build`, and `run` directories contain the ODT source code,  
163 compilation tools, and run scripts, respectively. The `input` directory contains  
164 subdirectories corresponding to several possible case types, each populated  
165 with the appropriate input files. The `data` directory, initially empty, holds  
166 the raw data files and runtime information generated ~~by ODT~~, as well as  
167 post-processed data files generated from within the `post` directory. Finally,  
168 the `docs` directory contains documentation optionally generated by Doxygen  
169 [40] during the build process.



### 170 2.3.1. Building *ODT* the *ODT* code

171 The *ODT* *code* build process is automated with CMake. The user nav-  
172 igates to the `build` directory and edits the CMake configuration file. The  
173 CMake configuration file specifies Cantera’s location and must be changed to  
174 reflect the local installation location. For reacting flows, the chemical reac-  
175 tion rates can be computed by Cantera (default), or a from a user-specified  
176 function, specified at compile time. Once the configuration settings are up-  
177 dated, the user runs CMake to apply the changes.

178 The `yaml-cpp` package is required in order to process input files. If it  
179 has not yet been installed, it must be built and installed at this point. For  
180 convenience, the *ODT* *code* package automates this process. YAML need  
181 only be installed once for a given instance of the *ODT* package. Rebuilding  
182 the *ODT* code with CMake does not affect the YAML installation.

183 Once CMake and YAML have been prepared, the user can build the  
184 *ODT* code with the `make` command, which places the code executable in the  
185 `run` directory. The most common errors that occur during the build process  
186 involve incorrect file paths or incomplete installation of required packages.  
187 See the build documentation for troubleshooting help.

188 Once the code is built, the user may optionally build a local copy of the  
189 documentation via Doxygen, which must be previously installed. This step is  
190 not required in order to run the *ODT* code. As an alternative, documentation  
191 can be accessed at the code repository wiki.

### 192 2.3.2. Input files

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

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

205 Input files contain simulation parameters in a human-readable format  
206 parsed by YAML. Within `input.yaml`, parameters are organized into sec-  
207 tions, several of which are common to all input files. Parameter values present  
208 in the provided input files may be considered defaults that can be used to  
209 run a successful simulation of that case type. General code parameters are

210 stored in the `params` class. Variables needed in a simulation but not provided  
 211 in an input file (not recommended), are either given defaults specified in the  
 212 `params` class or result in an error message. Details about individual param-  
 213 eters, including usage and typical values, are covered in the documentation.

### 214 2.3.3. Running ~~ODT~~ the ODT code

215 To run a simulation, users must then navigate to the `run` directory, which  
 216 contains the `odt.x` executable and several possible run scripts. The simplest  
 217 option is `runOneRlz.sh`, which runs one realization of ODT in the specified  
 218 configuration. In this run script, the user must alter two variables near the  
 219 top of the file: `inputDir`, which specifies which input directory and files to  
 220 use; and `caseName`, which provides a name for the simulation and the data  
 221 files it outputs. The `runManyRlz.sh` script runs many realizations in serial,  
 222 one after the other; it differs from `runOneRlz.sh` only in that the user must  
 223 also alter the `nRlz` variable, which specifies the number of realizations to  
 224 run. Each realization has a different seed value, which can be randomized  
 225 or directly specified in the input file. To run the simulation with either  
 226 `runOneRlz.sh` or `runManyRlz.sh`, save the run script and execute it at the  
 227 command line. Users will see some output on the command line, but no data,  
 228 which is instead output to the `data` directory.

229 ODT simulations can also be run in parallel using MPI. Two run scripts,  
 230 `slrmJob.sh` and `slrmJob_array.sh`, are configured using Slurm [41], a com-  
 231 mon workload manager used for massively parallel computing resources. This  
 232 allows many ODT realizations to run in parallel rather than in serial, reduc-  
 233 ing overall simulation time. Individual realizations are independent (embar-  
 234 rassingly parallel), but users must take care with case names and input file  
 235 changes to ensure that individual realizations or entire cases are not overrid-  
 236 den accidentally.

### 237 2.3.4. Data files and post-processing

238 For a given simulation, data is output to the `data` directory, which con-  
 239 tains subdirectories for each simulation, specified by the `caseName` variable  
 240 in the run script. Figure 3 illustrates the structure of the `data` directory and  
 241 the locations of files within it. Each case folder is subdivided into `input`,  
 242 `runtime`, `data`, and `post`. The `input` folder contains a copy of the input  
 243 files used for the simulation; `runtime` contains runtime output, including  
 244 eddy event information; `data` contains the raw data files; and `post` contains  
 245 post-processed data files once they are generated.

246 To use post-processing tools, users navigate to the `post` directory. Within  
 247 the `post` directory, data processing tools are organized by case type, which

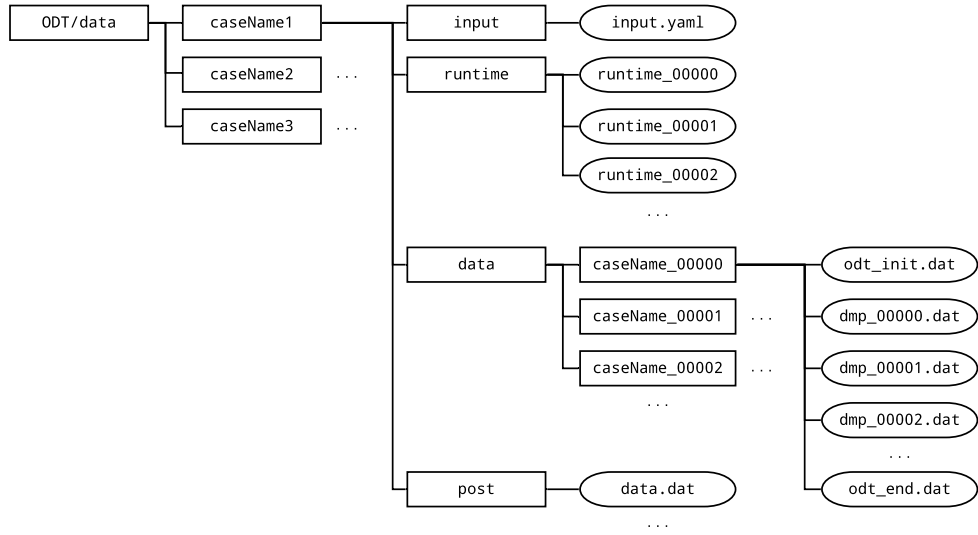


Figure 3: Data folder structure and file locations. Boxes indicate folders and ovals indicate files. For a given simulation, **caseName** will be replaced with the case name specified in the run script. The number attached to **caseName** folder names and to **runtime** data files indicates the realization number. The number attached to **dmp** files indicates the index of the list of “dump” times included in, or generated from the information in **input.yaml**.

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 Jupyter notebooks, 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.

As part of the post-processing tools, there is a `post/tools` directory that contains two files: `data_py.py` and `data_tools.py`. The first of these can be used to convert the text data files generated during a run into binary numpy files. This can reduce both the number of data files and the file sizes. The number of files is reduced since all code realizations at a given output time are saved in the same binary file. The `data_tools.py` file is a module that contains a number of functions for processing the data files, such as functions to

- get the data from a given realization,
- return the number of realizations,
- get a list of the variable names in the data files,
- compute axial locations in the spatial ODT formulation,
- query a parameter from the input file,
- return the stoichiometric mixture fraction for combustion/mixing problems, and
- compute the domain bounds.

In addition, a few functions are provided for doing interpolation/extrapolation, and computing probability density functions (PDFs), and weighted PDFs. Many of these functions take an argument that is a dictionary of case-specific information, such as the case name and paths for key directories. This dictionary is set in the post-processing driver script based only on the user-specified case name.

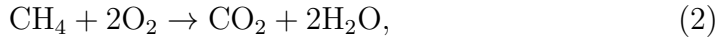
### 3. Example Case: Canonical Jet Flame

**ODT** The ODT model is uniquely suited to reacting flow simulations. Here, we present illustrative ODT simulation results of a round, turbulent jet flame based on and compared to the experimental DLR-A flame of Meier

et al. [42]. This canonical flame configuration has been used extensively to study and validate turbulent combustion models [43, 44, 45, 46, 47, 48].

The DLR-A fuel stream is mixture of 22.1% CH<sub>4</sub>, 33.2% H<sub>2</sub>, and 44.7% N<sub>2</sub> (by volume) that issues into dry air via a nozzle with an inner diameter of 8 mm at a mean exit velocity of 42.2 m·s<sup>-1</sup>. The coflow air stream issues from a concentric nozzle 140 mm in diameter at a velocity of 0.3 m·s<sup>-1</sup>. The reported jet Reynolds number is 15,200.

Previous ODT [studies simulations](#) 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. The fuel was diluted with N<sub>2</sub> 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 O<sub>2</sub>, N<sub>2</sub>, CH<sub>4</sub>, H<sub>2</sub>, H<sub>2</sub>O, and CO<sub>2</sub>. We assume that reactions proceed to the products of complete combustion and apply simple, fast reaction rates according to the following chemical equations:



These assumptions are reasonable for illustrating ODT in the DLR-A reacting jet configuration with variable properties and heat release.

This simulation uses ODT parameters  $C = 20$ ,  $\beta_{LES} = 17$ , and  $Z = 400$ . The values of  $C$  and  $\beta_{LES}$  were adjusted to give good agreement with the experimental data, and the value of  $Z$  is the same as the spatial simulations in [15]. 1024 independent flow realizations were performed in parallel and the results ensemble averaged. Downstream distance  $y$  and radial position  $r$  are normalized by the jet diameter  $D$ .

Figure 4 displays the simulation results: axial mean and centerline values for (a) axial velocity  $v$ , (b) mixture fraction  $\xi$ , and (c) temperature  $T$ . The ODT [simulation](#) results track the experimental data well for all three variables. The centerline temperature peaks about 100 K above the experimental data, but this small difference can be attributed to thermal radiation (which was neglected in this simulation) and the assumption that reactions proceed to the products of complete combustion rather than their equilibrium state. The centerline velocity shows a fast initial decrease that can be attributed to diffusion. Similarly, the increase in centerline velocity RMS values is delayed; this occurs due to the elapsed time model for large eddy suppression, which

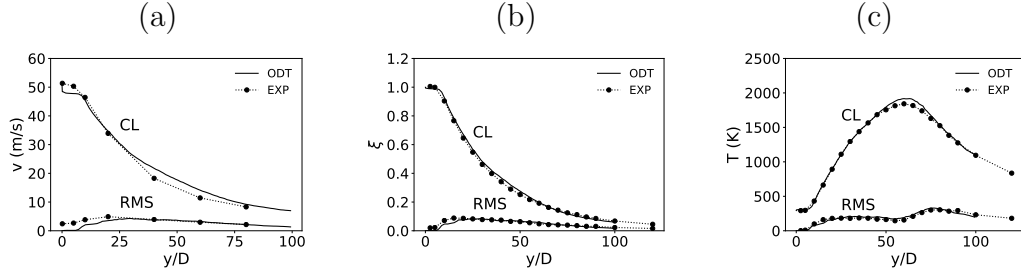


Figure 4: DLR jet flame example case results: (a) mean axial velocity and RMS velocity along the centerline versus downstream location; (b) mean axial mixture fraction and RMS mixture fraction along the centerline versus downstream location; (c) mean axial temperature (K) and RMS temperature (K) along the centerline versus downstream location.

limits disturbances in the early stages of the flow to small eddies that occur on the jet edges away from the centerline.

To replicate this example case, build the code with the `CHEMISTRY = SIMPLEDLR` flag in the `user_config` file and edit the desired run script with `inputDir = "../input/jetFlame/DLR_A"` and a new case name such as `caseName = "jetFlame_example"`. The input file for this case, located at `ODT/input/jetFlame/DLR_A/input.yaml`, contains the appropriate parameters and does not need to be modified to match this example case. See the code documentation for data post-processing instructions.

#### 4. Impact

The ODT model is complex, challenging to implement, and difficult to cleanly separate from its implementation, all of which presents barriers to practitioners who wish to use the model. For instance, triplet maps can be represented mathematically but are usually implemented as previously described, assuming an adaptive grid with variable cell sizes. Alternative formulations are used for fixed grids. The [code ODT code implementation](#) presented here reduces these barriers by providing an open-source, well-documented, modularly styled code that can be used directly or modified to suit particular research goals. Its structure facilitates addressing new research questions, such as soot-flame-radiation-chemistry interactions, multi-phase reacting flows using Lagrangian particle models (such as diesel sprays), and coupled near-wall heat and momentum transfer. The code presented here has been extended to and demonstrated in such systems already, facilitated by the object-oriented structure. For example, this code has been extended to solve steady laminar flamelets [49], requiring inheritance from the `solver`,

331 `micromixer`, and `dv` (domain variable) classes.

332 The structure of the ODT code, using an adaptive mesh and a Lagrangian  
333 formulation for diffusive advancement of transport equations, has several  
334 advantages, including simplicity of formulation and computational efficiency.  
335 The adaptive mesh minimizes the number of computational cells required and  
336 can be adapted to any variable of interest. This is especially important for  
337 flows with complex and expensive chemistry and flows with widely varying  
338 resolution demands such as near-wall flows. The Lagrangian formulation  
339 implies that cell faces move with fluid velocity (such as is present during  
340 flame expansion, for example). This treatment is inherently stable, removing  
341 the need for complex and diffusive hyperbolic conservation law treatments  
342 and their associated CFL constraints. The conservation equations, as in  
343 Equation 1, are similarly simplified.

344 The code has been run on Linux, Mac, and Windows machines, includ-  
345 ing parallel clusters of up to 512 processor cores. Several research groups  
346 have used the code and contributed to its development. These include  
347 groups at Sandia National Laboratories, Brandenburg University of Tech-  
348 nology Cottbus-Senftenberg, Chalmers University, and Combustion Science  
349 and Engineering, Inc. The latter company is the only organization known  
350 to use the code in a commercial setting, though on a combustion research  
351 project.

## 352 5. Conclusion

353 A clean and extensible implementation of the one-dimensional turbulence  
354 (ODT) model has been presented in an open-source, object-oriented C++  
355 framework, motivated by advanced turbulent reacting flow research. The  
356 build process, directory structure, input file format, run process, and post-  
357 processing procedure were described in detail, including a representative re-  
358 acting jet example case. Access to a proven, documented code and a modern  
359 build procedure will aid researchers in applying and extending the model to  
360 new research questions.

## 361 6. Conflict of Interest

362 There are no known conflicts of interest associated with this publication  
363 and there has been no significant financial support for this work that could  
364 have influenced its outcome.

## 365 Acknowledgements

366 Special thanks to Alan R. Kerstein for his significant contributions to  
367 model development. Additional thanks to Heiko Schmidt, Juan Medina, and  
368 Marten Klein of Brandenburg University of Technology Cottbus-Senftenberg,  
369 Michael Oevermann and Marco Fistler of Chalmers University of Technology,  
370 and John Hewson of Sandia National Laboratories for code contributions, and  
371 insightful conversations.

## 372 References

- 373 [1] D. O. Lignell, G. C. Fredline, A. D. Lewis, Comparison of one-  
374 dimensional turbulence and direct numerical simulations of soot for-  
375 mation and transport in a nonpremixed ethylene jet flame, Proceedings  
376 of the Combustion Institute 35 (2) (2015) 1199–1206. doi:10.1016/j.  
377 proci.2014.05.046.
- 378 [2] A. W. Abboud, C. Schulz, T. Saad, S. T. Smith, D. D. Harris, D. O.  
379 Lignell, A numerical comparison of precipitating turbulent flows between  
380 large-eddy simulation and one-dimensional turbulence, AIChE Journal  
381 61 (10) (2015) 3185–3197. doi:10.1002/aic.14870.
- 382 [3] A. R. Kerstein, One-dimensional turbulence: model formulation and  
383 application to homogeneous turbulence, shear flows, and buoyant strat-  
384 ified flows, Journal of Fluid Mechanics 392 (1999) 277–334. doi:  
385 10.1017/S0022112099005376.
- 386 [4] A. R. Kerstein, T. D. Dreeben, Prediction of turbulent free shear flow  
387 statistics using a simple stochastic model, Physics of Fluids 12 (2) (2000)  
388 418–424. doi:10.1063/1.870319.
- 389 [5] A. R. Kerstein, W. T. Ashurst, S. Wunsch, V. Nilsen, One-dimensional  
390 turbulence: vector formulation and application to free shear flows,  
391 Journal of Fluid Mechanics 447 (2001) 85–109. doi:10.1017/  
392 S0022112001005778.
- 393 [6] T. Echekki, A. R. Kerstein, T. D. Dreeben, J.-Y. Chen, ‘one-dimensional  
394 turbulence’ simulation of turbulent jet diffusion flames: model formula-  
395 tion and illustrative applications, Combustion and Flame 125 (3) (2001)  
396 1083–1105. doi:10.1016/S0010-2180(01)00228-0.
- 397 [7] J. C. Hewson, A. R. Kerstein, Stochastic simulation of transport and  
398 chemical kinetics in turbulent CH<sub>4</sub>/H<sub>2</sub>/N<sub>2</sub> flames, Combustion Theory



- 399 and Modelling 5 (4) (2001) 669–697. doi:10.1088/1364-7830/5/4/  
400 309.
- 401 [8] J. C. Hewson, A. R. Kerstein, Local extinction and reignition in non-  
402 premixed turbulent  $\text{CH}_4/\text{H}_2/\text{N}_2$  jet flames, Combustion Science and  
403 Technology 174 (5-6) (2002) 35–66. doi:10.1080/713713031.
- 404 [9] D. O. Lignell, D. S. Rappleye, One-dimensional-turbulence simulation of  
405 flame extinction and reignition in planar ethylene jet flames, Combustion  
406 and Flame 159 (9) (2012) 2930–2943. doi:10.1016/j.combustflame.  
407 2012.03.018.
- 408 [10] N. Punati, J. C. Sutherland, A. R. Kerstein, E. R. Hawkes, J. H. Chen,  
409 An evaluation of the one-dimensional turbulence model: comparison  
410 with direct numerical simulations of  $\text{CO}/\text{H}_2$  jets with extinction and  
411 reignition, Proceedings of the Combustion Institute 33 (1) (2011) 1515–  
412 1522. doi:10.1016/j.proci.2010.06.127.
- 413 [11] A. Abdelsamie, D. O. Lignell, D. Thévenin, Comparison between ODT  
414 and DNS for ignition occurrence in turbulent premixed jet combustion: safety-relevant applications, Zeitschrift für Physikalische Chemie  
415 231 (10) (2017) 1709–1735. doi:10.1515/zpch-2016-0902.
- 417 [12] D. O. Lignell, V. B. Lansinger, A. R. Kerstein, A cylindrical formula-  
418 tion of the one-dimensional turbulence (ODT) model for turbulent jet  
419 flames, in: AIChE Annual Meeting 2017, American Institute of Chemical Engineers, 2017.
- 421 [13] B. Goshayeshi, J. C. Sutherland, Prediction of oxy-coal flame stand-  
422 off using high-fidelity thermochemical models and the one-dimensional  
423 turbulence model, Proceedings of the Combustion Institute 35 (3) (2015)  
424 2829–2837. doi:10.1016/j.proci.2014.07.003.
- 425 [14] Z. Jozefik, A. R. Kerstein, H. Schmidt, S. Lyra, H. Kolla, J. H. Chen,  
426 One-dimensional turbulence modeling of a turbulent counterflow flame  
427 with comparison to DNS, Combustion and Flame 162 (8) (2015) 2999–  
428 3015. doi:10.1016/j.combustflame.2015.05.010.
- 429 [15] E. I. Monson, D. O. Lignell, M. A. Finney, C. Werner, Z. Jozefik,  
430 A. R. Kerstein, R. S. Hintze, Simulation of ethylene wall fires using  
431 the spatially-evolving one-dimensional turbulence model, Fire Technol-  
432 ogy 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 for soot evolution in fire, in: H. Bockhorn, A. D’Anna, A. F. Sarofim, H. Wang (Eds.), Combustion Generated Fine Carbonaceous Particles, KIT Scientific Publishing, 2009, pp. 571–588.
- [18] D. O. Lignell, J. C. Hewson, One-dimensional turbulence simulation: overview and application to soot formation in nonpremixed flames, in: SIAM Conference on Computational Science and Engineering, 2015.
- [19] A. J. Ricks, J. C. Hewson, A. R. Kerstein, J. P. Gore, S. R. Tieszen, W. T. Ashurst, A spatially developing one-dimensional turbulence (ODT) study of soot and enthalpy evolution in meter-scale buoyant turbulent flames, Combustion Science and Technology 182 (1) (2010) 60–101. doi:10.1080/00102200903297003.
- [20] G. Sun, J. C. Hewson, D. O. Lignell, Evaluation of stochastic particle dispersion modeling in turbulent round jets, International Journal of Multiphase Flow 89 (2017) 108–122. doi:10.1016/j.ijmultiphaseflow.2016.10.005.
- [21] J. R. Schmidt, J. O. L. Wendt, A. R. Kerstein, Non-equilibrium wall deposition of inertial particles in turbulent flow, Journal of Statistical Physics 137 (2) (2009) 233–257. doi:10.1007/s10955-009-9844-8.
- [22] G. Sun, D. O. Lignell, J. C. Hewson, C. R. Gin, Particle dispersion in homogeneous turbulence using the one-dimensional turbulence model, Physics of Fluids 26 (10) (2014) 103301. doi:10.1063/1.4896555.
- [23] M. Fistler, D. O. Lignell, A. R. Kerstein, M. Oevermann, Numerical studies of turbulent particle-laden jets using spatial approach of one-dimensional turbulence, in: ILASS-Europe 28th Conference on Liquid Atomization and Spray Systems, 2017.
- [24] S. Cao, T. Echekki, A low-dimensional stochastic closure model for combustion large-eddy simulation, Journal of Turbulence 9. doi:10.1080/14685240701790714.

- [25] R. C. Schmidt, A. R. Kerstein, S. Wunsch, V. Nilsen, Near-wall LES closure based on one-dimensional turbulence modeling, *Journal of Computational Physics* 186 (1) (2003) 317–355. doi:10.1016/S0021-9991(03)00071-8.
- [26] R. C. Schmidt, A. R. Kerstein, R. McDermott, ODTLES: A multi-scale model for 3D turbulent flow based on one-dimensional turbulence modeling, *Computer Methods in Applied Mechanics and Engineering* 199 (13-16) (2010) 865–880. doi:10.1016/j.cma.2008.05.028.
- [27] E. Gonzalez-Juez, A. R. Kerstein, D. O. Lignell, Fluxes across double-diffusive interfaces: a one-dimensional-turbulence study, *Journal of Fluid Mechanics* 677 (2011) 218–254. 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, *Geophysical & Astrophysical Fluid Dynamics* 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, *Physics of Fluids* 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, *Theoretical and Computational Fluid Dynamics* 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, *Physics of Fluids* 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,

- 502 Theoretical and Computational Fluid Dynamics 27 (3-4) (2013) 273–  
503 295. doi:10.1007/s00162-012-0267-9.
- 504 [35] Y. A. Çengel, J. M. Cimbala, Fluid Mechanics, 2nd Edition, Çengel  
505 series in engineering thermal-fluid sciences, McGraw-Hill Higher Educa-  
506 tion, 2010.
- 507 [36] J. Beder, yaml-cpp v0.6.3 (2008).  
508 URL <https://github.com/jbeder/yaml-cpp/>
- 509 [37] G. Strang, On the construction and comparison of difference schemes,  
510 SIAM Journal on Numerical Analysis 5 (3) (1968) 506–517. doi:10.  
511 1137/0705041.
- 512 [38] A. C. Hindmarsh, R. Serban, D. R. Reynolds, CVODE,  
513 [https://computing.llnl.gov/sites/default/files/public/cv\\_guide.pdf](https://computing.llnl.gov/sites/default/files/public/cv_guide.pdf)  
514 (2020).  
515 URL <https://computing.llnl.gov/projects/sundials/cvode>
- 516 [39] D. G. Goodwin, R. L. Speth, H. K. Moffat, B. W. Weber, Cantera  
517 (2018). doi:10.5281/zenodo.1174508.  
518 URL <https://cantera.org/>
- 519 [40] D. van Heesch, Doxygen (2018).  
520 URL <https://www.doxygen.nl/>
- 521 [41] A. B. Yoo, M. A. Jette, M. Grondona, Slurm: simple linux utility for re-  
522 source management, in: D. G. Feitelson, L. Rudolph, U. Schwiegelshohn  
523 (Eds.), Job Scheduling Strategies For Parallel Processing, Vol. 2862 of  
524 Lecture notes in computer science, 0302-9743, Springer, 2003, pp. 44–60.  
525 doi:10.1007/10968987-3.
- 526 [42] W. Meier, R. S. Barlow, Y.-L. Chen, J.-Y. Chen, Raman/Rayleigh/LIF  
527 measurements in a turbulent CH<sub>4</sub>/H<sub>2</sub>/N<sub>2</sub> jet diffusion flame: experimen-  
528 tal techniques and turbulence–chemistry interaction, Combustion and  
529 Flame 123 (3) (2000) 326–343. doi:10.1016/S0010-2180(00)00171-1.  
530 URL [https://tnfworkshop.org/data-archives/simplejet/  
531 dlrflames/](https://tnfworkshop.org/data-archives/simplejet/dlrflames/)
- 532 [43] H. Pitsch, Unsteady flamelet modeling of differential diffusion in turbu-  
533 lent jet diffusion flames, Combustion and Flame 123 (3) (2000) 358–374.  
534 doi:10.1016/S0010-2180(00)00135-8.

- [44] R. P. Lindstedt, H. Ozarovsky, Joint scalar transported PDF modeling of nonpiloted turbulent diffusion flames, *Combustion and Flame* 143 (4) (2005) 471–490. doi:10.1016/j.combustflame.2005.08.030.
- [45] H. Wang, S. B. Pope, Large eddy simulation/probability density function modeling of a turbulent CH<sub>4</sub>/H<sub>2</sub>/N<sub>2</sub> jet flame, *Proceedings of the Combustion Institute* 33 (1) (2011) 1319–1330. doi:10.1016/j.proci.2010.08.004.
- [46] M. Fairweather, R. M. Woolley, First-order conditional moment closure modeling of turbulent, nonpremixed methane flames, *Combustion and Flame* 138 (1-2) (2004) 3–19. doi:10.1016/j.combustflame.2004.03.001.
- [47] K. W. Lee, D. H. Choi, Prediction of NO in turbulent diffusion flames using eulerian particle flamelet model, *Combustion Theory and Modelling* 12 (5) (2008) 905–927. doi:10.1080/13647830802094351.
- [48] 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, *International Journal of Heat and Mass Transfer* 52 (5-6) (2009) 1412–1420. doi:10.1016/j.ijheatmasstransfer.2008.08.015.
- [49] N. Peters, Laminar diffusion flamelet models in non-premixed turbulent combustion, *Progress in Energy and Combustion Science* 10 (1984) 319–339.

## Current executable software version

Software information is provided in Table 2.

Nr.	(Executable) software meta-data description	Please fill in this column
S1	Current software version	1.0
S2	Permanent link to executables of this version	For example: <a href="https://github.com/BYUignite/ODT">https://github.com/BYUignite/ODT</a>
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	<a href="https://github.com/BYUignite/ODT">https://github.com/BYUignite/ODT</a>
S7	Support email for questions	davidlignell@byu.edu

Table 2: Software metadata (optional)