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

Victoria B. Stephens, David O. Lignell*

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

Abstract

Write this last. About 100 words.

Keywords: turbulence, reacting flows, one-dimensional turbulence

Code Metadata

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

Table 1: Code metadata (mandatory)

1. Motivation and significance

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

*Corresponding author.

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

Preprint submitted to SoftwareX *April 24, 2020*

searchers 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 can 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, or 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. Because large-scale structures are much easier to study and model than small-scale structures, ODT mitigates or sidesteps many of the 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 model is one-dimensional, it is limited to homogeneous or boundary layer flows such as jets, wakes, and mixing layers; such flows, however, are extremely common in both nature and 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.

2. Software description

2.1. Model description

Describe how ODT works here. Turbulent advection via eddy events implemented by triplet maps. Planar vs. cylindrical configurations. Eddy events happen concurrently with solution of transport equations. Evolution of transport equations in only one dimension. Limited to one-dimensional flows: jets, wakes, mixing layers. This is fine though because those are common in nature and important to research.

2.2. Software Architecture

Things to put in this section (from SoftwareX template)

- overview of overall software architecture
- optional: pictorial overview
- implementation details
  - software dependencies?
  - how to build and run (or leave this in the code README files? maybe brief description here with full details in README.)
  - how to use post processing tools? (TODO: prune post processing tools as much as possible so there's no excess)

Summary of directory structure:

- source: source code lives here; yaml files generated by building yaml in build folder get put here as well
  - more details on directory structure necessary here? not sure if relevant
- build: go here to build the code
  - Run: make yaml (only need to do this once, the first time you download/build the code)
  - set options/paths for cmake in user_config file
  - Run: cmake -C user_config ../source
  - Run: make -j8
- input: input files live here; change input files to specify case details

- each case type has its own folder; make your own if necessary
- within each case folder, you should see file input.yaml. this is your all-important input file that makes each case unique
  - talk about most important ODT parameters/options here? or leave to documentation?

- run: go here to actually run the code; don't forget to make your changes to the input file first
  - after building, you'll see odt.x in this folder. you can run it directly (not recommended) or from within one of the bash scripts (better)
  - don't forget to give your case a name; this name will be used to label data files dumped to the data directory
- data: data and runtime files dumped here, organized by case name and individual realizations
  - TO DO: raw/processed file types and what they contain, how to use them effectively
- post: post processing tools located here, organized by case type
- doc: doxygen documentation (if generated) lives here

3. Example Cases

Present the major functionalities of the software.

- simulating turbulent flow cases, reacting or nonreacting flows
- can simulate laminar flow cases too
- does these things a whole lot faster than other methods do (LES, DNS especially)
- testing LES subgrid modeling assumptions
- simulating cases that DNS can't get to because needed simulation length is too long (i.e. late-flame phenomena)

Example cases:

- From cylindrical ODT paper:
  1. pipe flow

2. nonreacting round jet
3. round jet flame

- Other possible example cases:
  - something to compare with DNS/LES results to illustrate efficiency
  - no sooting cases since no soot in the code
  - planar vs. cylindrical comparison

Things to talk about with example cases:

- compare to DNS/LES cases; computational efficiency, accuracy
- how data looks before post processing, how post processing works

4. Impact

Questions to answer in this section (from SoftwareX template)

1. How can new research questions be pursued with this software?
   - possibility of parametric studies (much harder with DNS/LES/RANS)
   - study of late-flame soot and radiation interactions, soot emissions as smoke
   - comparative radiation model studies?
2. How does the software improve pursuit of existing research questions?
   - late-flame behavior becomes easier to study
   - validation of LES subgrid models
   - soot stuff, especially late in the flame (because soot moves slowly compared to gas species and therefore short simulation times like in DNS aren't enough to study it effectively)
3. How does the software change the daily practice of its users?
   - cases take hours or days rather than weeks using supercomputer resources
   - test cases can be run on local computers (unlike something like DNS) and as background tasks without disrupting other tasks
   - ODT as a tool complements other approaches, can cover blind spots and be used in validation

4. How widespread is the software? Who uses it? (Within and outside of intended research area and/or group.)
    - BYU group
    - JCH at Sandia
    - Chalmers group in Sweden (Marco Fistler, etc.)
    - German university group (Heiko Schmidt, Juan Media, Marten Klein, etc.)
    - TO DO: find other groups who have used or currently use ODT
5. How is the software used in commercial settings (if any)? Has it led to creation of spin-off companies?
    - No commercial use (I think).

5. Conclusion

Write this part next to last

6. Conflict of Interest

We wish to confirm that 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

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

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 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 61 (10) (2015) 3185–3197. doi:10.1002/aic.14870.

Current executable software version

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

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

Table 2: Software metadata (optional)