## Authors Response to Reviewers

* Original reviewer comments are listed. Author responses are shown in blue text.

**Review 1**

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

This paper presents an easy to use software for the One-dimensional Turbulence (ODT) model that helps to remove the barrier for researchers wanting to learn about ODT. The code is easy to understand and change to suit different requirements. The paper is well-written, the software is carefully explained, and the example provided demonstrates the use of the software. I recommend publication subject to some minor edits and clarifications as noted below.

*We appreciate the reviewer’s and their careful review of the paper.*

* Building OPT:
  + YAML: specify the name of the needed yaml package is: yaml-cpp
* *We made this clarification in section 2.3.1 as requested.*
* Input files: Consider adding a feature that helps to specify the dumpTimes. For example, add the capability to parse the following values (Start, End, Step ) from the input file rather than specifying all the times you want to save the data manually.
* *We have now included this feature as requested. The user can specify times either through the original dumpTimes key, or through a new dumpTimesGen key that includes start, stop, and step variables. If both are present in the input file dumpTimesGen takes precedence, unless a negative start time is specified. This is demonstrated in the newly modified input file for the channel flow case, with comments to note the behavior and usage. We have also edited the paper and code documentation appropriately.*
* Running ODT:
  + The running interface is not trivial and limits the user to a predefined data storage location.This may cause problems for some users.
  + My recommendation is to put all these run scripts under a single python interface that will allow the user to select the number of realization, the number of processes (when running in parallel), the input file, ... .
    - consider using python classes for the interface, it will make you code much cleaner
    - check argparse module in python!
* *We have carefully considered the reviewer’s suggestion. The running of the code falls into two broad categories: running smaller simulations on a desktop computer, and running larger parallel simulations on Linux clusters. The scripts that we have provided are well-documented, and we have also provided a video going through the process. In addition, many of our current users are used to the existing interface, and changing this might cause confusion. For single desktop jobs, the runOneRlz.sh script is easy to use, with two variables for the user to edit at the top. For parallel jobs, users will be familiar with scripts using job submission software (e.g., slurm), and examples are provided that can be edited. We will keep this option open.*
* Data files and Post-processing:
  + In the paper, you mentioned in line 118 that the package contains auxiliary data processing and visualization tools. However, later in section 2.3.4 Data files and post-processing you mentioned in line 237 that those tools are organized by case type meaning that these post-processing tools are specific for the input files provided as examples. This may cause confusion to the user.
  + *The reference on line 118 is a short note that post-processing tools are provided. The description in section 2.3.4 is more detailed and provides a more complete description. We certainly want to avoid confusion. We have edited the comment on line 118 to note that the post-processing tools are specific to the case, and include a pointer to the later discussion.*
  + Also, you didn’t mention the tools you provided that help the user load the data. (ODT/Post/tools)
  + *This is a good comment. We have added a paragraph to the end of section 2.3.4 that describes these tools.*
    - I recommend that you put these functions under a python class that takes as a value the path to the data you saved. This will give flexibility to the user to load any data stored on his local or remote machines.
    - *This is a valid suggestion for implementation, and is very similar to the approach that we have chosen. Namely, the functions are in a python module and can be easily accessed by importing the module. Many of these functions take a single dictionary argument that provides the case name as well as relevant paths (e.g., to the data or post directories of the given case). This dictionary is build in the driver script based on the user-supplied case name. We have explaned this in the new paragraph in section 2.3.4 noted above.*
    - Describe those functions in the paper.

*Done, as requested.*

* + Also, I would recommend that the post-processing data examples you provided be written in Jupyter notebooks for clarity.
  + *This is a good recommendation. We have included a Jupyter notebook for post-processing the channelFlow, pipeFlow, and laminar\_mixing\_layer cases, with the latter already present. These will process the data and show comparison plots with the results. The previous stats.py functions are retained for consistency with the demonstration video and the codeocean module. The README files are updated. We have not made a jupyter notebook for the coldJet or jetFlame cases since these are larger simulations that are meant to be run and post-processed on a parallel processor with a job submission script, and that is how we and previous users have interacted with the code.*
* Impact:
  + In the impact section line 298 the reference is missing.
* *This has been corrected.*
* Lines 78 and 79: very cumbersome sentence. Suggest you use: the first term on the RHS of eq. 1 is... while the second term is...
* *We have reworded this paragraph: rather than jumping into symbol definitions, we note the physical meanings of the two additive terms on the right-hand-side of the equation. We also note the generality of the diffusive mixing, and that transport is refered to as mixing, which has relevance to understanding the naming in the source code. We then define the variables used.*

**Review 2**

Reviewer #2: The paper describes an open-source code for simulating one-dimensional turbulence (ODT).

As highlighted from the Authors in the impact Section, implementing ODT is not an easy task hindering the spreading and use of this useful model. The present open-source tool will be a first step to increase the popularity of the technique among the scientific community in multiphase and reactive flows.

The code is easy to read, to eventually modify and to use. The documentation and the provided examples are clear and exhaustive.

*We thank the reviewer for their time in performing this review.*

The Reviewer has a personal curiosity on the technique (outside the scope of the Journal). ODT has been used in combination with Lagrangian particle models. We know those ( heavy ) particles tend to stay away from the vortical regions, but if we are in 1-D, what is vorticity? Should we look to other "equivalent" observables?

*It is true that ODT cannot capture vortex structures, but many relevant aspects of turbulent flows, including turbulent particle dispersion are represented by ODT through the capture of a wide range of length and time scales. There are aspects of correlation of turbulence intensity with vortex structures that are relevant, for example. For more information, see the discussion on page 111 (section 6.4.2) of Guangyuan Sun’s dissertation: https://search.proquest.com/pqdtglobal/docview/1762246838/D46E7CC1A3EA44FAPQ/7?accountid=4488*

Minor, a reference appears as [? ] on page 12.

*This has been corrected.*