Data Management Plan

1. Importance of proper data practices

Modern science revolves around computer code: it is used to run simulations, analyze data, and create publishable figures and papers. If this code is not curated properly, subtle but crucial errors can creep into it and undermine the veracity of the scientific results (Soergel 2015). While it is nearly impossible to build completely bug-free code, by following proper data handling practices, such as the use of version control, errors in scientific codes can be reduced. Furthermore, open-source, published, and freely accessible code has been found to increase citation count and reproduceability (Piwowar & Vision 2013). Thus, the data management practices that I will follow, outlined below, will serve to not only increase the trustworthiness of published results but also increase the visability and impact of the work I carry out during my fellowship studies.

2. Description of expected data products

The research I am proposing here is purely computational, and there will be three types of data produced in these studes: Python code, raw data, and processed data. All products will be made open-source, openly-licensed, and free to use.

2.1. Python code

Each task will generate and run Python code which runs computer simulations, post-processes raw data, and generates publication-quality figures. Code used to run simulations and post-process data will be stored in version-controlled Git repository and made open to the public upon publication of any related work. These Git repositories will be linked to Zenodo repositories which automatically generate referencable DOIs upon the release of official code versions. By releasing official code versions in this way, I will ensure that future work will be able to reproduce my published results running precisely the same code. The code that I use to generate publication-quality figures will be packaged into separate Zenodo repositories, along with data (see below), and released upon submission of any work. These distribution plans will ensure that even novice users of Python or the Dedalus code will have direct access to my full research pipeline, and will be able to reproduce my simulations and produced figures with ease.

2.2. Raw & processed simulation data

The simulations that I will run in Dedalus will create rawdata in an HDF5 output format, which allows for extreme output versatility during simulation runs and simple location of data products afterwards. Unfortunately, it is not straightforward to publish complete datasets of the full time evolution of these simulations. Individual simulations can easily produce over a Terabyte

of data each, and such large quantities of data are cumbersome to make publically available, or to interact with as a consumer.

Rather than making the full set of simulation data available for all simulation times, I will instead publish select "checkpoint" data at simulation times. These checkpoints will contain the full simulation state and can be partnered with my released code in order to re-run key times in my published simulations. Upon the publication of time series data, or data which take long time averages, I will ensure that I publish a checkpoint that aligns with the beginning of these time intervals to improve reproduceability.

In addition to these raw checkpoint files, often times published figures feature a large collection of volume-averaged scalar data points, or partially averaged 1D profiles. All such data which appear in simulation figures, as well as some additional potentially useful measurements, will be published alongside figure generation code in Zenodo repositories.

REFERENCES

Piwowar, H., & Vision, T. 2013, *D*ata reuse and the open data citation advantage, PeerJ, 1, doi:10.7717/peerj.175

Soergel, D. A. W. 2015, *R*ampant software errors may undermine scientific results, F1000Research, 3, doi:10.12688/f1000research.5930.2