Simulation Guide

Meher Krishna Patel

Created on : Octorber, 2017

Last updated : May, 2020

Table of contents

Contents:

Chapter 1

Simulation Guidelines

Computer-simulations are essential to verify the theoretical models. Since creating individual simulator for each mathematical model can be a long and tedious process, therefore it is recommended to break-down the mathematical model into smaller pieces. Then, these pieces should be implemented using functions and classes, which can be reused by other mathematical models as well. Further, it is very important to document and test the simulators so that other can use it. We use Python3 along with some third-party libraries to design the simulators.

Complete process of simulation can be divided in following ways,

- Testing
- Simulation
- Documentation
- Version control
- Writing style

Requirements for above steps are explained next,

1.1 Testing

Please write the test cases before creating the simulator. The test cases should include all the possible values of inputs and corresponding outputs.

This is the most import process for designing the simulator as it gives us the chance to think thoroughly about the simulation process; therefore we get better insight of the mathematical model and simulation process. Also, tested codes encourage others to use it for simulating their mathematical models.

Preferred libraries for writing test cases are shown below,

- Pytest
- Numpy.testing
- Unittest
- Nose

Important: Testing is not optional process, it is mandatory for creating the simulators. Please read other tutorial i.e. PyTest Guide, where a simulator is designed with '**First Test, Then Implement**' philosophy.

1.2 Simulator

After creating the test cases, we can design the simulators. Remember, instead of creating one big simulator, write small reusable modules. Further, small modules are easy to test as well as understand.

Lastly, simulators must contains proper comments for easy understanding of the code. See, next section for requirements of comments.

Following libraries are quite useful to simulate the mathematical models,

- Numpy
- Scipy
- Cython
- Matplotlib
- Numba

Note: To make code reusable, write small methods for repetitive logics. Classes should be used for the cases, where various methods use same arguments.

Note that, first we need to write test cases for each module, therefore spend more time in designing and writing test cases.

1.3 Documentation

If simulators are documented properly, then these can be used by others as well. Note that, it is very difficult to understand our own code after sometime, therefore we also need proper documentation for later usage.

Following softwares can be used for creating documents,

- Python-Sphinx (required)
- Latex
- Jupyter-notebook

Note: Note that, Sphinx documents is mandatory as it generates HTML files with search features. Also, it can create various other formats as well e.g. Epub and Latex etc. Further, Sphinx document can be uploaded on ReadTheDoc websites easily.

Please use NumPy style docstrings (for comments) by enabling the napoleon extension in Sphinx.

Jupyter-notebook and Latex can be used for personal usage e.g. sharing codes on the blog or submitting reports etc. Jupyter-notebook is good for generating the HTML for short documents which can be easily uploaded on the blog, whereas Latex is useful for submitting work in PDF formats.

1.4 Version control

Git-Version-Control-System is required for maintaining the older versions of the codes. Also, we can share the codes on the web-repositories and work together in a group.

Preferred web repositories for version-control are following,

- BitBucket
- Github

Note: Version control is quite useful when we want to restore the previous code, in the cases when the code are modified incorrectly.

1.5 Writing style

Lastly, if common writing style is used to create the simulators, then it can be read easily by everyone. Therefore, use PEP-8 style guide to write codes, which makes codes more readable. Following library can be used to see the style-errors in the codes, which are not written according to PEP-8 guidelines.

- pep8

Note: All the above mentioned libraries are available in the Anaconda-Python-Package. But, the softwares i.e. Latex and Git need to be installed separately. Also, Pandoc can be installed to convert documents in various formats using Jupyter-notebook.
