AmpScan Internship Diary

Oliver Stocks

July 2018

1 Background

1.1 Development Stack

- OS - Windows 10
- Version Control - GIT
- Python Version - 3.6.5 (Anaconda)
- Sphinx Version - 1.7.4

1.2 Conda Environment

A conda environment was used for development as opposed to pip and virtualenv. This was a conscious decision as it fit in with the anaconda workflow far more conveniently - for example, instead of adding Python to path or making a specific version available to the virtual environment I could simply run anacondas built-in prompt and call `conda create −−name [env name]`. Conda's installation tools and the jupyter development suite are also extremely convenient.

1.3 GIT

The AmpScan module is hosted on the University of Southamptons GitLab account. As such a University account is required for access to developers. An introduction to Git is not provided here as many such resources exist online (I found `http://dont-be-afraid-to-commit.readthedocs.io/en/latest/git/index.html` to be particularly useful).

2 Documentation: 2/07 -

2.1 Setting Up Sphinx

Documentation for the project was generated using Python's Sphinx library. Install from the command line using;

```
conda install −c anaconda sphinx.
```

or if you are not using anaconda;

```
pip install -U Sphinx
```

The documentation files were created using the built-in command;

```
sphinx-quickstart
```

from inside a new "docs" folder - the choice of name here is not arbitrary as it will later be used by the hosting service. Quickstart asks numerous setup questions. Most of the defaults can be accepted except for autodocs which defaults to no - we will use this tool later to automatically generate documentation from docstrings - and create .nojekyll which also defaults to no - only necessary to change because we will have files beginning with an underscore published on github. Note that quickstart was called from within the docs folder. This is worthy of note as it alters the autodocs configuration. The docs directory took on the following structure:

```
docs
 ├── conf.py
 ├── index.rst
 ├── make.bat
 ├── Makefile
 ├── _build
 ├── _static
 └── _templates
```

The initial documentation files (e.g. Introduction.rst) were created within the docs folder. This can be done quickly from the command line by using echo [some text] > [file name]. Before generating html files, index.rst was edited to add the new files to the contents. Here the additional lines "Intro" and "Fordevs" haves been added according to the names of my new files (Intro.rst and Fordevs.rst). Note that a blank line must be included **before** the list of contents:

```
.. toctree::
   :maxdepth: 2
   :caption: Contents:

   Intro
   Fordevs
```

Now using the built-in command;

```
make html
```

a set of html files are generated within the docs_build folder. These can be viewed within a browser session.

2.2 Autodocs

Sphinx provides a built-in tool called autodocs that reads through all the code in a given module and generates documentation from the docstrings. We enabled this during setup although it can be reconfigured by altering `conf.py`. Note that in order to create readable documentation, the docstrings must be written in a standard format (discussed in the next section).

First autodocs needs to be configured using the built-in `apidocs` tool. The output path should be left within the docs directory (where we call this command) and must be given the name `source` (again this is necessary for hosting). The last option attached to apidocs is the directory where the python functions are stored - after cloning this can be found at `AmpScan\AmpScan`:

```
sphinx-apidoc -o source ..\AmpScan
```

The docs directory now contains a folder called source that contains the .rst files needed to generate autodocs. Calling `make html` at this point results in a series of failed import errors. The `conf.py` file must be altered to add the directory the python package (Ampscan) to Path. The relevant code can be uncommented from the file and the absolute path specified using double dot notation:

```
import os
import sys
sys.path.insert(0, os.path.abspath('..'))
```

Calling make html at this stage resulted in numerous errors. Inspection of the traceback however indicates that the majority of these errors are a result of incorrectly formatted docstrings - this is to be expected at this stage. A couple of modules (e.g. surrogateModelGui2) were not found, however. It is not entirely clear why this is the case. The remaining modules (including those with incorrect formatting) were rendered successfully as html files in `docs\\_build \\html`.

I am continueing under the assumption that correcting the format of the module docstrings will either remove these errors or at least make the issue clearer.

2.3 Docs Template