Convert MATLAB buoy data to JSON format

October, 2020

This purpose of this notebook is to convert and reduce large MATLAB data files that store time series statistics derived from Western Australian Department of Transport Datawell Waverider Mk3 buoys. The data is trimmed for ease of use with modern Machine Learning and statistical analysis techniques, keeping only the one-dimensional representations of the wave measurements.

The files, at time of writing:

File	Size	Description
cott.mat	3.06 GB	Cottosloe Waverrider Mk3 buoy
rott.mat	6.10 GB	Rottnest Waverrider Mk3 buoy

Environment/package dependencies:

- `mat73` (for reading MATLAB `.mat` files)
- `numpy`
- Python 3.7.X (for use with the mat73 package)

Python 3.7 is available at Python.org, and the package requirements can be installed by running:

```
path/to/python3.7 -m pip install mat73
path/to/python3.7 -m pip install numpy
```

from the command line / terminal, or simply:

```
pip install <package-name>
```

if Python 3.7 is your only local Python install. If instead you use Conda/Anaconda Python distributions, you can set up a `conda` Python 3.7 environment by following the instructions on the Anaconda website.

Output files:

File	Size	Description
cott-waves.json	5.5 MB	Cottosloe Waverrider Mk3 buoy
rott-waves.json	11.1 MB	Rottnest Waverrider Mk3 buoy

Feel free to reach out at our GitHub repository, or by email to barrettjc1@gmail.com with any questions about how to utilise this notebook.

---

1 Imports and options

In [ ]:

```
import mat73
import numpy as np
import json
from json import JSONEncoder
import os
from os import path
```

Select buoy data:

In [ ]:

```
BUOY = 'cott'
# BUOY = 'rott'
```

In [ ]:

```
INPATH = '.'  # location of .mat files
OUTPATH = './Data/'  # location to save output json files
```

2 Load MatLab data

Given these files are named `cott.mat` and `rott.mat` for Cottosloe and Rottnest buoy data, respectively, and are sitting in the `INPATH` directory. Be advised that the following cell can take some time to run, given the large size of the input `.mat` files.

In [ ]:

```
mat_load = mat73.loadmat(path.join(INPATH, f'{BUOY}.mat'))
waves = mat_load[BUOY]['waves']
```

In [ ]:

```
# Data structure
for key in waves.keys():
    if key != 'metadata':
        print(key, waves[key].shape)
```

```
Dp (162035,)
Hs (162035,)
Tm (162035,)
Tp (162035,)
a1 (162035, 64)
a2 (162035, 64)
b1 (162035, 64)
b2 (162035, 64)
check_fac (162035, 64)
dirs (72,)
freq (64,)
position (755, 3)
spec1D (162035, 64)
spec2D (72, 64, 162035)
sst (762, 2)
time (162035,)
```

3 Keep only one-dimensional wave statistics

Here, only the variables representing:

- Date & Time
- Significant Wave Height
- Peak Wave Period
- Mean Wave Period

In [ ]:

```
keep_vars = ['Dp', 'Hs', 'Tm', 'Tp', 'time']
waves_slim = {k: v for k, v in waves.items() if k in keep_vars}
```

In [ ]:

```
class NumpyArrayEncoder(JSONEncoder):
    def default(self, obj):
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        return JSONEncoder.default(self, obj)
```

In [ ]:

```
if not path.exists(OUTPATH):
    os.mkdir(OUTPATH)

with open(path.join(OUTPATH, f'{BUOY}-waves.json'), 'w') as f:
    json.dump(waves_slim, f, cls=NumpyArrayEncoder, separators=(',', ':'))
```