usage_FST

January 9, 2024

1 Hawk data full structure test data API

The aim of this notebook is to guide the user through the use of the API package `hawk` for interacting with the dataseries collected as part of the TRIC-DT project at the LVV in Sheffield in 202. In order to use this notebook, the hawk package is required. The package is freely available and can be installed with pip (requres python 3.9+):

```
pip install git+https://github.com/MDCHAMP/hawk-data
```

For basic usage of the `hawk` package for interacting with data collected on the aircraft see the starboard wing test and related documentation.

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_theme("notebook")
```

```
sns.set_style("ticks")
sns.set_palette("Set2")

from hawk import FST
```

1.1 Basic usage

The `hawk` package contains the `hawk.FST` function for interacting with the data from the starboard wing test.

[ ]:
```
data_dir = "./hawk_data"
data = FST(data_dir)
data.describe()
```

During the test campaigns a great deal of data were collected. Within the `data` object we created above, the various test series and signals are organised like a file-tree structure. The `describe` method above returns information pertaining to where in the tree we are currently. The `explore` method provides a look at what is contained within the tree beneath us. Lets see the result of calling the `explore` method on `data` (the top of the tree).

[ ]:
```
data.explore(2)
```

As expected, this producdes a structured description of which test series are available. The only argument controls the depth through the tree that the `explore` function will search.

Lets now try to access a test series that may or may not be downloaded in `data_dir`.

[ ]:
```
test_series = data['HS_WN_01'] # whie noise healthy state 01
test_series.explore(1)
```

Just by accessing the data in our code, the relevant files have been downloded and saved to disk in `data_dir`. If we were to access the data again, the downloaded data wold be used automatically.

Looking at the output of the `explore` function, we can see that there are a number of sensor addresses, lets now take a look at the output of the `describe` function:

[ ]:
```
test_series.describe()
```

[ ]:
```
sensor = "LLC-07"  # Lower leading edge central position 07 (wing tip)
sensor_data = test_series[sensor]
sensor_data.explore(1)
```

1.1.1 Dataset structure

In order to avoid downloading all the data every time, the dataset has been divided into a number of independent files, each one correspondiong to one of the test series repeats. Overall there are 71 test series.

The `hawk` package relies on a single 'header' `.hdf5` file for accessing all of the data simultaneously without loading it all in to memory (or even having all of the data on disk). This works thanks to the `ExternalLink` feature of the `.hdf5` spec, more details of which can be found here.