01_QuickStart

May 8, 2020

1 `asammdf` Quick Start & Reference.

A one page introduction to `asammdf` for Measurement Data Format (MDF) data analysis.

For engineers wanting to drink from a fire hose or already experienced with Python data analysis.

Replaces some or all of the functionality of these tools with a fullly opensource, free, analysis stack:

- AVL CONCERTO 5™
- MathWorks® Vehicle Network Toolbox™
- ETAS Measure Data Analyzer (MDA V8)
- National Instruments DIAdem
- Vector CANape & vSignalyzer

```
[1]: %matplotlib inline
     import asammdf
     import seaborn as sns
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import matplotlib as mpl
```

2 Create MDF File

- Create a [0, 10)s time vector with 100Hz sampling rate (10 ms raster)
- Create a simple MDF file with two channels:
    - "EngineSpeedCmd" 0.5 Hz sinewave with amplitude=10, & dc offset=600.
    - "EngineSpeed" 0.5 Hz sinewave with amplitude=5, phi=pi/8, & dc offset=600.

2.0.1 Generate Engine Speed signals

```
[2]: # Time Vector
     t0=0   # [s]
     tf=10  # [s]
     f_sample = 100 # [Hz]
     dT = 1/f_sample # [s]
```

```
timestamps = np.arange(t0, tf, dT)
# Engine Speed
A = 10
f_signal = 0.5
dc_offset = 600
engine_speed_cmd = A*np.sin(2*np.pi*f_signal*timestamps)+dc_offset
engine_speed = A*0.5*np.sin(2*np.pi*f_signal*(timestamps-0.25))+dc_offset
```

2.0.2 Plot engine speed signal.

Direct plotting with matplotlib & seaborn.

```
[3]: sns.set(
        rc={
            "figure.figsize": (11.69, 8.27), # A4 paper size.
            "figure.facecolor": "w",
            "figure.edgecolor": "k",
            "axes.labelsize": 18,
            "axes.titlesize": 18,
            "legend.fontsize": 18,
        }
    )
    plt.plot(timestamps, engine_speed_cmd, timestamps, engine_speed)
    plt.xlabel("Time [s]")
    plt.ylabel("Engine Speed [rpm]")
    plt.title("Engine Speed vs Time")
    plt.legend(
        {"Engine Speed Cmd", "Engine Speed"},
        loc='upper right'
    );
    plt.savefig(
        "EngineSpeedPlot.png",
        transparent=False,
        bbox_inches='tight'
    );
```

Engine Speed vs Time
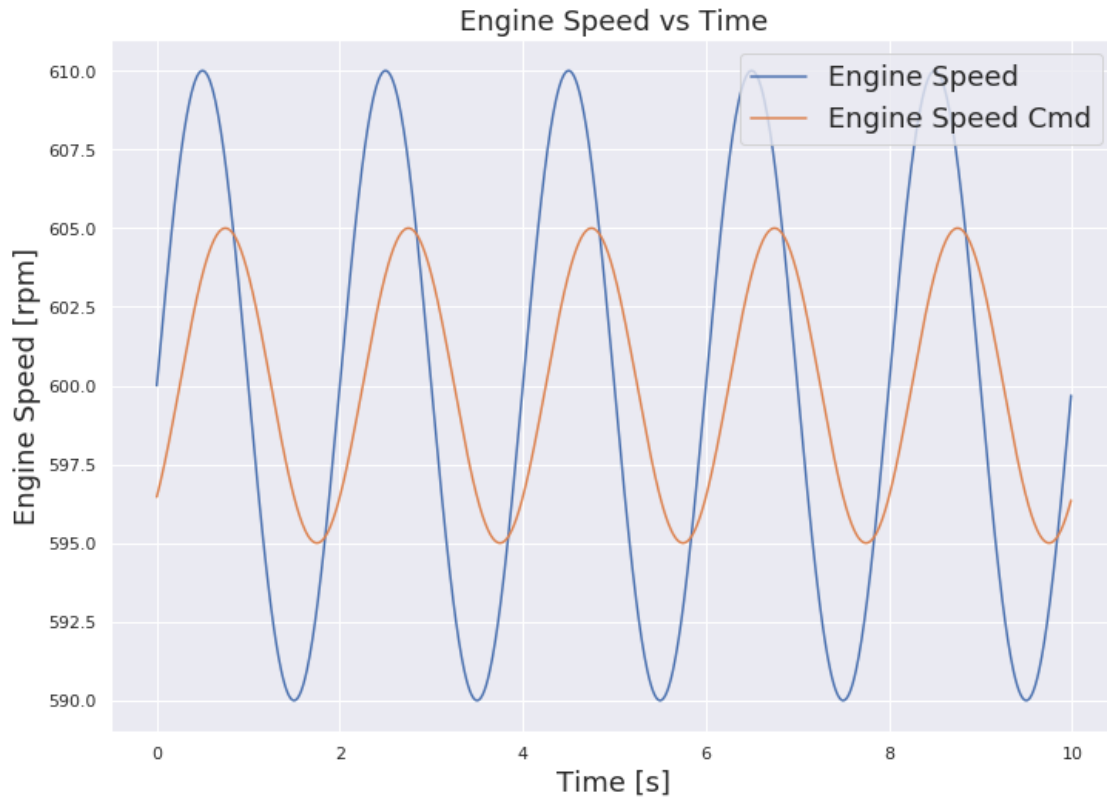
2.1 Generate `asammdf.Signal` & save MDF file.

Convert the `numpy` arrays into `asammdf.Signals` & save.

2.1.1 Signal Object

```
[4]: asammdf.Signal?
```

```
[5]: engine_speed_cmd_sig = asammdf.Signal(
         samples=engine_speed_cmd,
         timestamps=timestamps,
         unit='rpm',
         name='EngineSpeedCmd',
         conversion=None,
         comment='Swept sine plant identification, X(t)',
         raw=True,
         master_metadata=None,
         display_name='Engine Speed Command',
         attachment=(),
         source=None,
```

```
        bit_count=None,
        stream_sync=False,
        invalidation_bits=None,
        encoding=None,
    )
engine_speed_sig = asammdf.Signal(
        samples=engine_speed,
        timestamps=timestamps,
        unit='rpm',
        name='EngineSpeed',
        conversion=None,
        comment='Swept sine plant identification, Y(t)',
        raw=True,
        master_metadata=None,
        display_name='Engine Speed',
        attachment=(),
        source=None,
        bit_count=None,
        stream_sync=False,
        invalidation_bits=None,
        encoding=None,
    )
signals = [
        engine_speed_cmd_sig,
        engine_speed_sig,
    ]
```

2.1.2 MDF Object

```
[6]: asammdf.MDF?
```

Write the file directly in and out of a context manager.

```
[7]: with asammdf.MDF(version="4.10") as mdf:
        mdf.append(signals, "Created by Python")
        # save new file
        mdf.save("quickstart_example.mf4", overwrite=True)
```

```
[8]: mdf = asammdf.MDF(version="4.10")
    mdf.append(signals, "Created by Python")
        # save new file
    mdf.save("quickstart_example.mf4", overwrite=True)
    mdf.close()
```

fsspec example.

Filesystem Spec is a project to unify various projects and classes to work with remote filesystems and file-system-like abstractions using a standard pythonic interface. - https://filesystem-spec.readthedocs.io/en/latest/

This allows reading & writing directly from S3, Azure Datalakes, Google Storage, and others.

```python
[9]: import fsspec
with fsspec.open("quickstart_example_fs.mf4", "wb") as fid:
    with asammdf.MDF(version="4.10") as mdf:
        mdf.append(signals, "Created by Python")
        # save new file
        mdf.save(fid, overwrite=True)
```

3   Read & Analyze MDF

```python
[10]: %matplotlib inline
import asammdf
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
```

Open a file with and without `fsspec`.

```python
[11]: with fsspec.open("quickstart_example_fs.mf4", "rb") as fid:
    mdf = asammdf.MDF(fid)
```

```python
[12]: mdf = asammdf.MDF("quickstart_example.mf4")
```

List the channels in the database.

```python
[13]: mdf.channels_db
```

```python
[13]: {'time': ((0, 0),), 'EngineSpeedCmd': ((0, 1),), 'EngineSpeed': ((0, 2),)}
```

```python
[14]: mdf.get_channel_unit("EngineSpeedCmd")
```

```python
[14]: 'rpm'
```

```python
[15]: mdf.get_channel_comment("EngineSpeedCmd")
```

```python
[15]: 'Swept sine plant identification, X(t)'
```
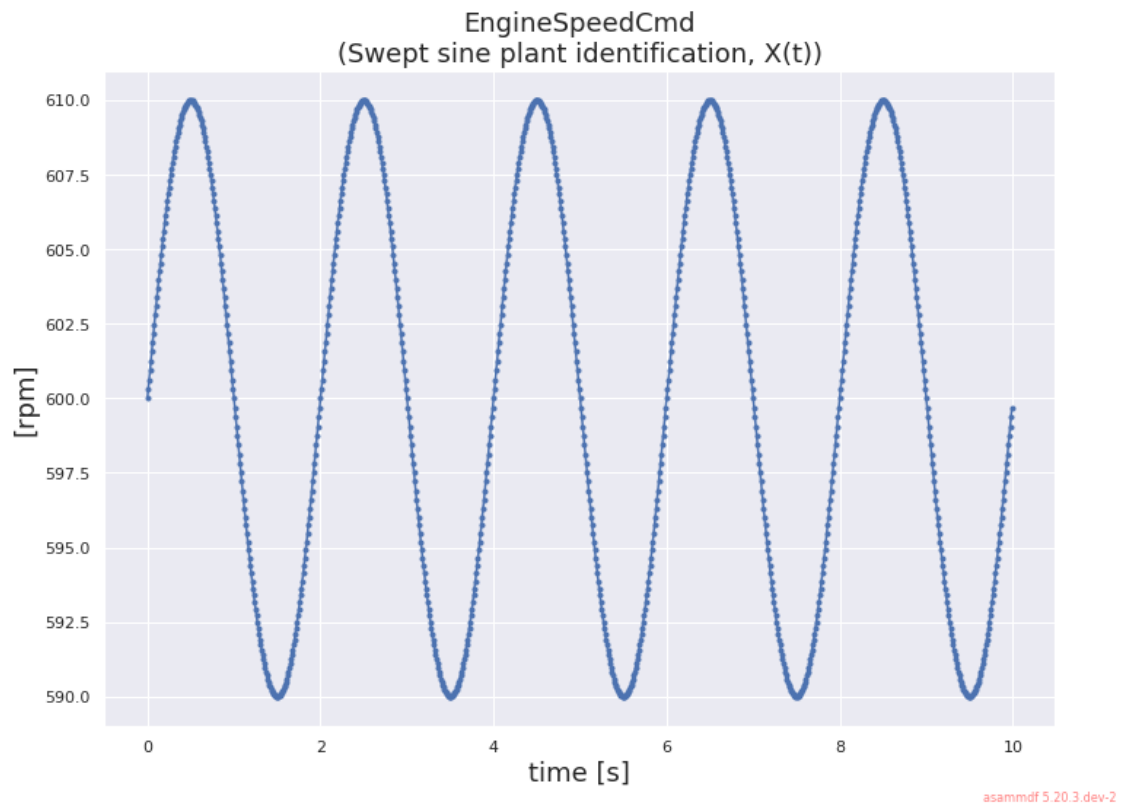
```python
[16]: sns.set(
    rc={
        "figure.figsize": (11.69, 8.27), # A4 paper size.
```
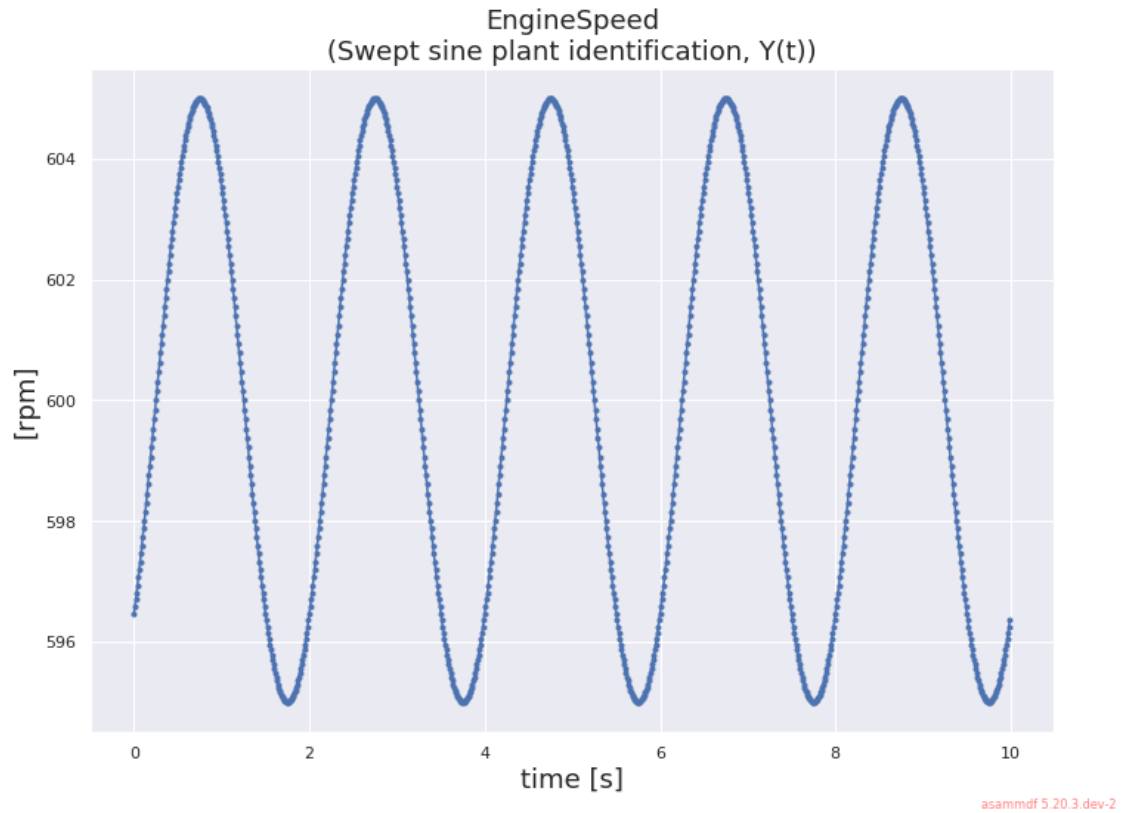
```
        "figure.facecolor": "w",
        "figure.edgecolor": "k",
        "axes.labelsize": 18,
        "axes.titlesize": 18,
        "legend.fontsize": 18,
    }
)
for channel in mdf.iter_channels():
    channel.plot()
```

WARNING:root:Signal plotting requires pyqtgraph or matplotlib

WARNING:root:Signal plotting requires pyqtgraph or matplotlib

EngineSpeed
(Swept sine plant identification, Y(t))

3.0.1 Export to `pandas.DataFrame`:

```
[17]: df = mdf.to_dataframe()
```

```
[18]: df.describe()
```

[18]:
	EngineSpeedCmd	EngineSpeed
count	1000.000000	1000.000000
mean	600.000000	600.000000
std	7.074606	3.537303
min	590.000000	595.000000
25%	592.928932	596.464466
50%	600.000000	600.000000
75%	607.071068	603.535534
max	610.000000	605.000000

```
[19]: df
```

[19]:
	EngineSpeedCmd	EngineSpeed
timestamps		

7

```
0.00              600.000000    596.464466
0.01              600.314108    596.577264
0.02              600.627905    596.693441
0.03              600.941083    596.812880
0.04              601.253332    596.935465
...                    ...            ...
9.95              598.435655    595.954915
9.96              598.746668    596.049225
9.97              599.058917    596.147434
9.98              599.372095    596.249445
9.99              599.685892    596.355157

[1000 rows x 2 columns]
```
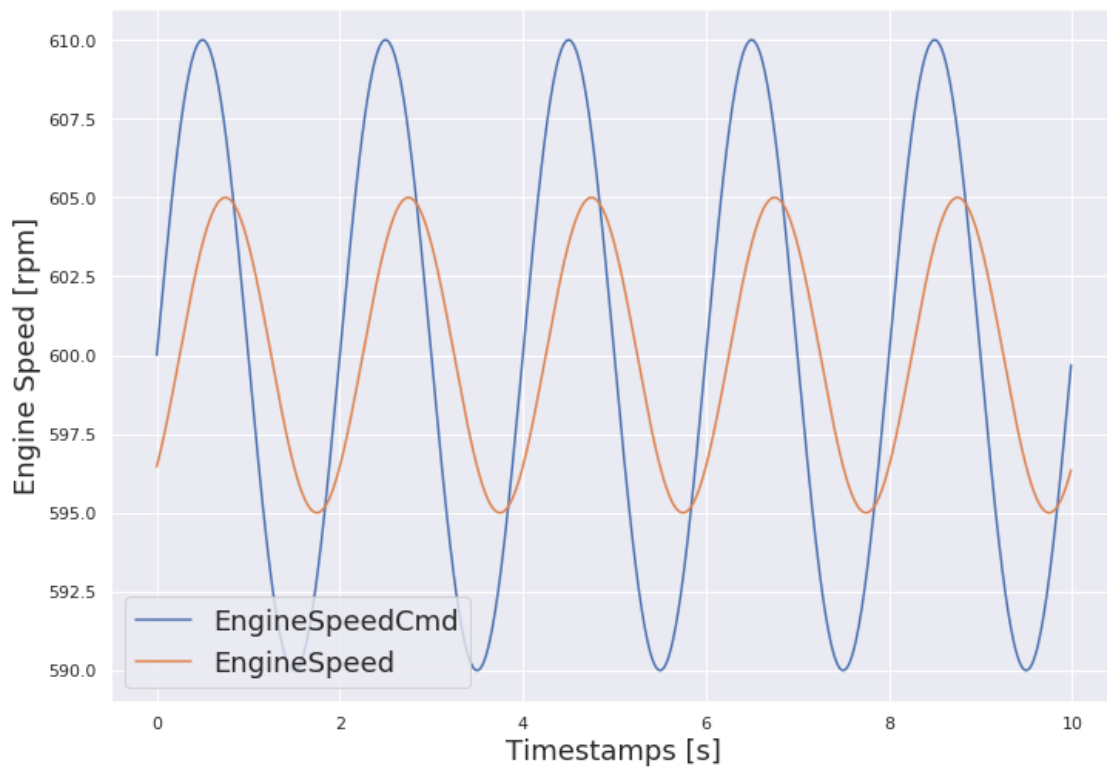
Reset Index

Convert the 'timestamps' index back into a column for direct plotting from `pandas`.

```python
[20]:  # Convert the
       df.reset_index(inplace=True)
```

```python
[21]:  df.plot(x="timestamps", y={"EngineSpeed", "EngineSpeedCmd"})
       plt.ylabel("Engine Speed [rpm]")
       plt.xlabel("Timestamps [s]");
```

4 Export

`asammdf` natively supports:

- csv
- hdf5
- mat
- parquet

```
[22]: mdf.export?
```

```
[23]: mdf.export(fmt="csv")
      !ls *.csv
```

quickstart_example.ChannelGroup_0.csv

```
[24]: pd.read_csv("quickstart_example.ChannelGroup_0.csv")
```

```
[24]:      timestamps  EngineSpeedCmd  EngineSpeed
      0          0.00      600.000000   596.464466
      1          0.01      600.314108   596.577264
      2          0.02      600.627905   596.693441
      3          0.03      600.941083   596.812880
      4          0.04      601.253332   596.935465
      ..          ...             ...          ...
      995        9.95      598.435655   595.954915
      996        9.96      598.746668   596.049225
      997        9.97      599.058917   596.147434
      998        9.98      599.372095   596.249445
      999        9.99      599.685892   596.355157

      [1000 rows x 3 columns]
```

```
[25]: mdf.export(fmt="hdf5")
      !ls *.hdf
```

quickstart_example.hdf

```
[26]: mdf.export(fmt="mat")
      !ls *.mat
```

quickstart_example.mat