dss_python: Unofficial bindings for EPRI's OpenDSS

Python bindings and misc tools for using OpenDSS (EPRI Distribution System Simulator). Based on CFFI and `dss_capi`, aiming for full COM compatibility on Windows and Linux.

If you are looking for the C API library, see `dss_capi`.

Version 0.9.5, based on OpenDSS revision 2152. This is a work-in-progress but it's deemed stable enough to be made public. *Note that, while the interface with OpenDSS is stable (classic version), the OpenDSS-PM (actor-based parallel machine version) interface was integrated recently and is experimental.*

This module mimics the COM structure (as exposed via `win32com` or `comtypes`), effectively enabling multi-platform compatibility at Python level. Most of the COM documentation can be used as-is, but instead of returning tuples or lists, this modules returns/accepts NumPy arrays for numeric data exchange.

This module depends on CFFI, NumPy and, optionally, SciPy.Sparse for reading the sparse system admittance matrix.

Recent changes

- 2018-04-05 / version 0.9.5: Adds missing `ActiveCircuit.CktElements[index]` (or `...CktElements(index)`) and `ActiveCircuit.Buses[index]` (or `...Buses(index)`).
- 2018-03-07 / version 0.9.4: Allows using `len` on several classes, fixes DSSProperty, and includes COM helpstrings as docstrings. Contains changes up to OpenDSS revision 2152.
- 2018-02-16 / version 0.9.3: Integrates COM interface fixes from revision 2136 (`First Next` iteration on some elements)
- 2018-02-12 / version 0.9.2: Experimental support for OpenDSS-PM (at the moment, a custom patch is provided for FreePascal support) and port COM interface fixes (OpenDSS revision 2134)
- 2018-02-08 / version 0.9.1: First public release (OpenDSS revision 2123)

Missing features and limitations

Most limitations are inherited from `dss_capi`, i.e.:

- Only the 64-bit version of OpenDSS is built. A 32-bit version should be possible with a few changes.
- Currently not implemented:

  - `DSSEvents` from `DLL/ImplEvents.pas`: seems too dependent on COM.
  - `DSSProgress` from `DLL/ImplDSSProgress.pas`: would need a reimplementation depending on the target UI (GUI, text, headless, etc.)
- Although tests were successful on openSuse 42.3 (both CPython 3.6 and PyPy3.5 v5.10.1), Linux binaries are not yet available. For the time being, you need to build them yourself.

Extra features

Besides most of the COM methods, some of the unique DDLL methods are also exposed in adapted forms, namely the methods from `DYMatrix.pas`, especially `GetCompressedYMatrix` (check the source files for more information).

Since no GUI components are used in the FreePascal DLL, we are experimenting with different ways of handling OpenDSS errors. Currently, the `DSS.Text.Command` call checks for OpenDSS errors (through the `DSS.Error` interface) and converts those to Python exceptions. Ideally every error should be converted to Python exceptions, but that could negatively impact performance. You can manually trigger an error check by calling the function `CheckForError()` from the main module.

Installing

On Windows (64-bit Python 2.7 and 3.6), you can install directly from pip:

```
pip install dss_python
```

If successful, you can then import the `dss` module from your Python interpreter.

Building

Get this repository:

```
git clone https://github.com/PMeira/dss_python.git
```

Assuming you successfully built or downloaded the `dss_capi` (check its repository for instructions), keep the folder organization as follows:

```
dss_capi/
dss_python/
electricdss/
```

Open a command prompt in the `dss_python` subfolder and run the build process:

```
python setup.py build
python setup.py install
```

Example usage

If you were using `win32com` in code like:

```
import win32com.client
dss_engine = win32com.client.Dispatch("OpenDSSEngine.DSS")
```

or `comtypes`:

```
import comtypes.client
dss_engine = comtypes.client.CreateObject("OpenDSSEngine.DSS")
```

you can replace that fragment with:

```
import dss
dss.use_com_compat()
dss_engine = dss.DSS
```

Assuming you have a DSS script named `master.dss`, you should be able to run it as shown below:

```
import dss
dss.use_com_compat()
dss_engine = dss.DSS

dss_engine.Text.Command = "compile c:/dss_files/master.dss"
dss_engine.ActiveCircuit.Solution.Solve()
voltages = dss_engine.ActiveCircuit.AllBusVolts

for i in range(len(voltages) // 2):
    print('node %d: %f + j%f' % (i, voltages[2*i], voltages[2*i + 1]))
```

If you do not need the mixed-cased handling, you can omit the call to `use_com_compat()` and use the casing used in this project.

If you want to play with the experimental OpenDSS-PM interface, it is installed side-by-side and you can import it as:

```
import dss.pm
dss.pm.use_com_compat()
dss_engine = dss.pm.DSS
```

Testing

Since the DLL is built using FreePascal, which is not officially supported by EPRI, the results are validated running sample networks provided in the official OpenDSS distribution. The only modifications are done directly by the script, removing interactive features and some minor other minor issues.

The validation scripts is `tests/validation.py` and requires the same folder structure as the building process. You need `win32com` to run it.

Currently, the following sample files from the official OpenDSS repository are used:

```
Distrib/EPRITestCircuits/ckt5/Master_ckt5.dss
Distrib/EPRITestCircuits/ckt7/Master_ckt7.dss
Distrib/EPRITestCircuits/ckt24/Master_ckt24.dss
Distrib/IEEETestCases/8500-Node/Master-unbal.dss
Distrib/IEEETestCases/IEEE 30 Bus/Master.dss
Distrib/IEEETestCases/NEVTestCase/NEVMASTER.DSS
Distrib/IEEETestCases/37Bus/ieee37.dss
Distrib/IEEETestCases/4Bus-DY-Bal/4Bus-DY-Bal.DSS
Distrib/IEEETestCases/4Bus-GrdYD-Bal/4Bus-GrdYD-Bal.DSS
Distrib/IEEETestCases/4Bus-OYOD-Bal/4Bus-OYOD-Bal.DSS
Distrib/IEEETestCases/4Bus-OYOD-UnBal/4Bus-OYOD-UnBal.DSS
Distrib/IEEETestCases/4Bus-YD-Bal/4Bus-YD-Bal.DSS
Distrib/IEEETestCases/4Bus-YY-Bal/4Bus-YY-Bal.DSS
Distrib/IEEETestCases/123Bus/IEEE123Master.dss
Distrib/IEEETestCases/123Bus/SolarRamp.DSS
Distrib/IEEETestCases/13Bus/IEEE13Nodeckt.dss
Test/IEEE13_LineSpacing.dss
Test/IEEE13_LineGeometry.dss
Test/IEEE13_LineAndCableSpacing.dss
Test/IEEE13_Assets.dss
Test/CableParameters.dss
Test/Cable_constants.DSS
Test/BundleDemo.DSS
Test/IEEE13_SpacingGeometry.dss
Test/TextTsCable750MCM.dss
Test/TestDDRegulator.dss
Test/XYCurvetest.dss
Test/PVSystemTestHarm.dss
Test/TestAuto.dss
Test/Stevenson.dss
Test/YgD-Test.dss
Test/Master_TestCapInterface.DSS
Test/LoadTest.DSS
Test/IEEELineGeometry.dss
```

```
Test/ODRegTest.dss
Test/MultiCircuitTest.DSS
Test/TriplexLineCodeCalc.DSS
Test/PVSystemTest-Duty.dss
Test/PVSystemTest.dss
Test/REACTORTest.DSS
```

On Windows 10, remember to set the compatibility layer to Windows 7 (set the environment variable `__COMPAT_LAYER=WIN7RTM`), otherwise you may encounter issues with COM due to ASLR on Python 3.6.

There is no validation on Linux yet, since we cannnot run the COM module there. The most likely solution will be to pickle the data on Windows and load them on Linux.

Roadmap

Besides bug fixes, the main funcionality of this library is mostly done. Notable desirable features that may be implemented are:

- More and better documentation
- Create wheels for Linux distributions, maybe using the Anaconda stack.
- Create a more "Pythonic" API. This would break compatibility with COM, but may result in a more pleasant environment for using OpenDSS in Python.

Questions?

If you have any question, feel free to open a ticket on Github or contact me through Twitter. Please allow me a few days to respond.

Credits / Acknowlegement

`dss_python` is based on EPRI's OpenDSS via the `dss_capi` project, check its licensing information.

This project is licensed under the (new) BSD, available in the `LICENSE` file. It's the same license OpenDSS uses (`OPENDSS_LICENSE`).

I thank my colleagues at the University of Campinas, Brazil, for providing feedback and helping me test this module.