mwavepy Manual

Alex Arsenovic

10/03/2010

Contents

1 About **3**

2 Installation **4**
- 2.1 Dependencies . . . 4
- 2.2 Platform independent . . . 4
  - 2.2.1 setuptools . . . 5
  - 2.2.2 distutils . . . 5

3 Basic Usage **6**
- 3.1 Basic Plotting . . . 6

4 Advanced Usage **7**

5 Architecture **8**
- 5.1 Classes, and Their Inheritance . . . 8
- 5.2 Individual Class Architectures . . . 8
  - 5.2.1 Frequency . . . 8
  - 5.2.2 Network . . . 10
  - 5.2.3 touchstone . . . 11
  - 5.2.4 WorkingBand . . . 11
  - 5.2.5 Calibration . . . 12

6 Future Work **13**

1 About

mwavepy is a compilation of functions and class's for microwave/RF engineering written in python. It is useful for things such as touchstone file manipulation, calibration, data analysis, data acquisition, and plotting. The main interface is command line, using the python interpreter. **mwavepy** is also very useful for scripting tasks. Being that it has been built around my specific needs as a graduate student, it is far from complete, but is more of a framework for future development. I've tried to make mwavepy as simple yet flexible as possible.

2 Installation

2.1 Dependencies

The requirements are basically a python environment setup to do numerical/scientific computing. If you are new to python, you should consider using pythonxy[1], which provides most everything you need to get started.

Here is a list of the requirements,

Necessary

- python (>=2.6) [2]
- matplotlib (aka pylab) [3]
- numpy [4]
- scipy [5]

Recomended

- ipython [6] (for interactive shell)

Optional

- pyvisa [7] - for instrument control
- pythics [8] - for VI gui interface design

2.2 Platform independent

Python has many choices for module installation, listed here are installation instructions using setuptools, distutils.

[1] http://www.pythonxy.com/
[2] http://www.python.org/
[3] http://matplotlib.sourceforge.net/
[4] http://numpy.scipy.org/
[5] http://www.scipy.org/
[6] http://ipython.scipy.org/moin/
[7] http://pyvisa.sourceforge.net/pyvisa/
[8] http://code.google.com/p/pythics

2.2.1 setuptools

Install the dependencies, if you have setuptools[9] installed just open a terminal and type:

```
easy_install mwavepy
```

This should download and install all mwavepy.

2.2.2 distutils

If you dont have setuptools installed, either check out the up-to-date svn version, or download and extract the source package. Open a terminal and cd into the mwavepy directory, and run

```
python setup.py install
```

Linux¶

For those who want to use their distribution package manager. To install the requirements in a debian-based linux system just type,

sudo apt-get install python-pyvisa python-numpy python-scipy python-matplotlib ipython python

(you will probably have to go fetch pyvisa yourself, or use easy_install.) Then you can checkout the current Source or Download a release. Install of same as described in previous section (and is given in README.txt)

Windows¶

I recommend using the platform independent instructions, as it will install all dependencies for you. Otherwise, you have to fetch and install all python modules listed under Requirements.

Then you can download mwavepy. There are two choices:

* windows binary (python module only) * platform independent source, has examples and docs

Both are under the Download tab.

[9] http://pypi.python.org/pypi/setuptools

3 Basic Usage

3.1 Basic Plotting

```
import mwavepy as mvy
import pylab


# create a Network type from a touchstone file of a horn antenna
horn = mvy.Network('horn.s2p')

# plot magnitude of S11
pylab.figure(1)
pylab.title('Return Loss (Mag)')
horn.plot_s_db(m=0,n=0) # m,n are S-Matrix indecies

# plot phase of S11
pylab.figure(2)
pylab.title('Return Loss (Phase)')
# all keyword arguments are passed to matplotlib.plot command
horn.plot_s_deg(0,0, label='Broadband Horn Antenna', color='r', linewidth=2)

# plot unwrapped phase of S11
pylab.figure(3)
pylab.title('Return Loss (Unwrapped Phase)')
horn.plot_s_deg_unwrapped(0,0)

# plot complex S11 on polar grid
pylab.figure(4)
horn.plot_s_polar(0,0, show_legend=False)
pylab.title('Return Loss')

# uncomment to save all figures,
#mvy.save_all_figs('.', format = ['png'])

# show the plots
pylab.show()
```

4 Advanced Usage

5 Architecture

5.1 Classes, and Their Inheritance

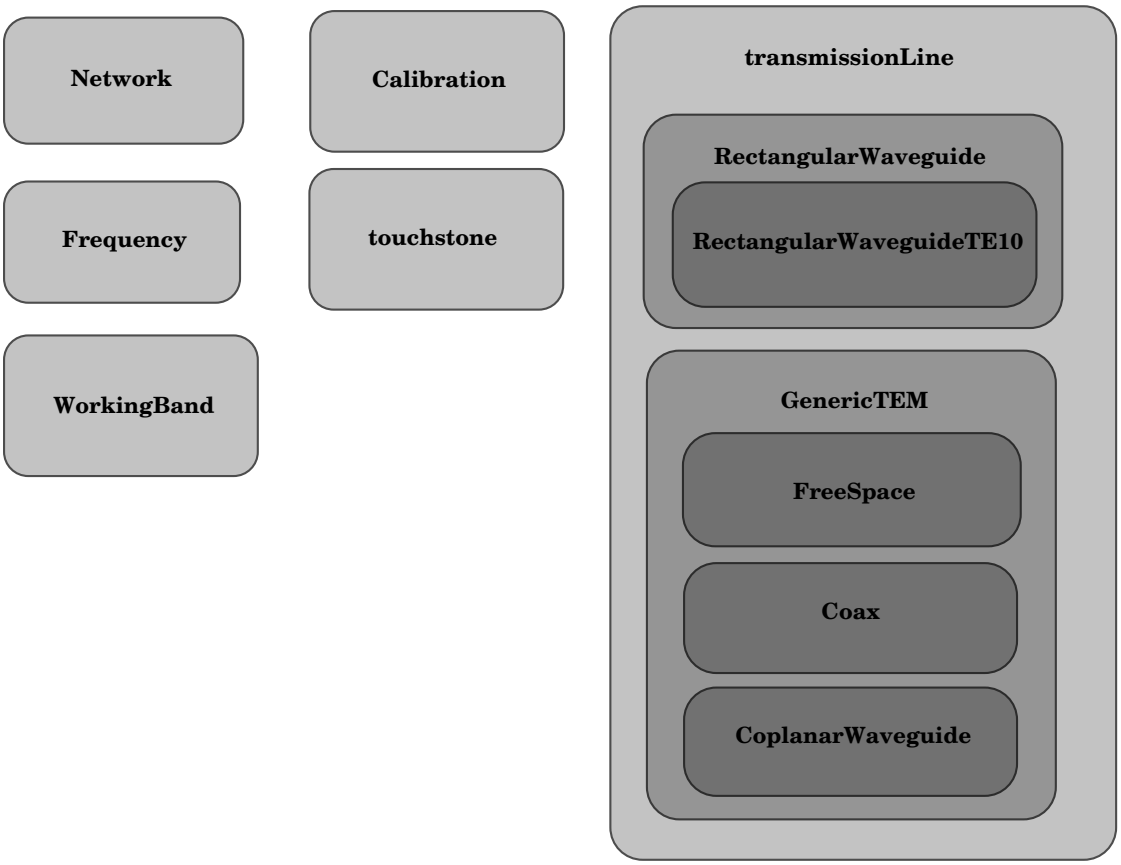

Figure 5.1: Main Classes and their inheritance

5.2 Individual Class Architectures

```
import mwavepy as mv
mv.Network('Interface 1/caled/short.s1p').plot_s_deg
```

5.2.1 Frequency

```
freq = mv.Frequency(start = 80,stop=120, npoints = 201, unit='ghz')
```

The frequency object was created to make storing and manipulating frequency information easier and more rigid. A major convenience this class provides is the acounting of the frequency vector's unit. Other objects, such as Network, and Calibration require a frequency vector to be meaningful. This vector is commonly referenced when a plot is generated, which one generally doesnt was in units of Hz. If the Frequency object did not exist other objects which require frequency information would have to implement the unit and multiplier bagage.

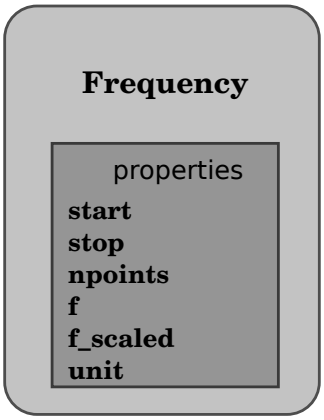

Figure 5.2: Frequency class architecture

5.2.2 Network

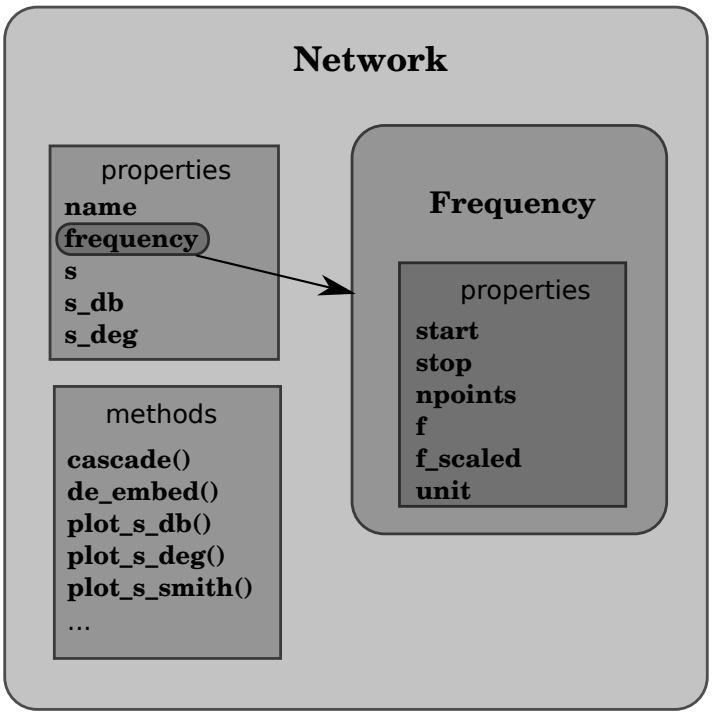

Figure 5.3: Network class architecture

5.2.3 touchstone

5.2.4 WorkingBand

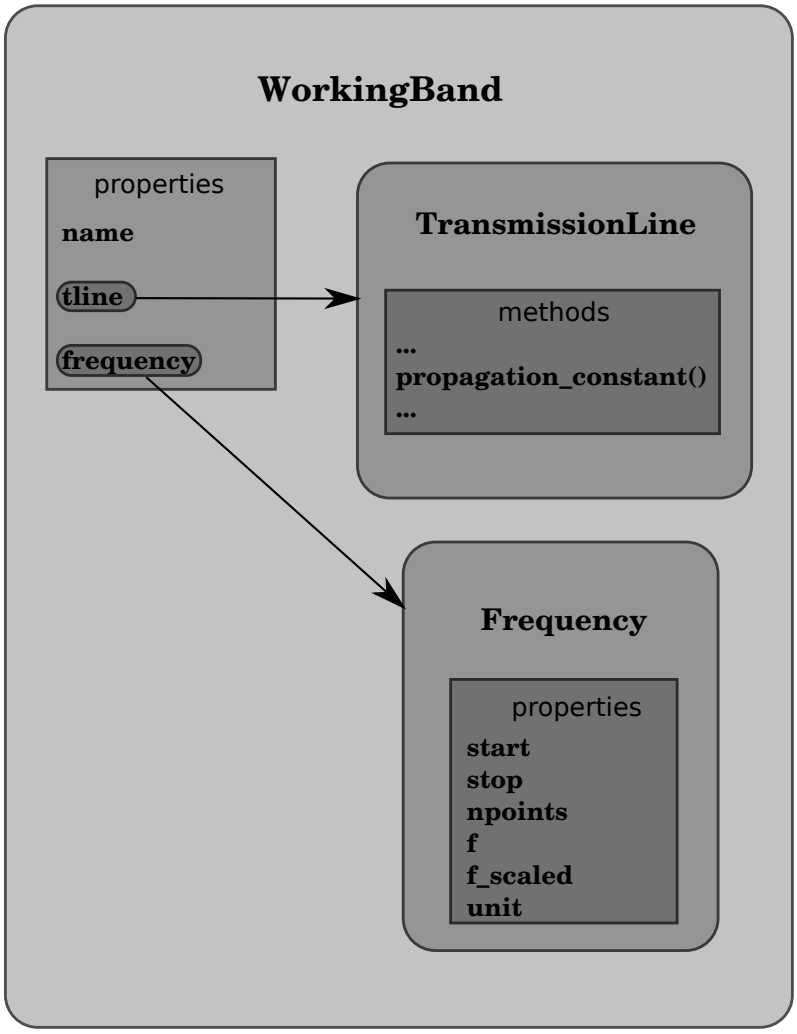

Figure 5.4: WorkingBand class architecture

5.2.5 Calibration

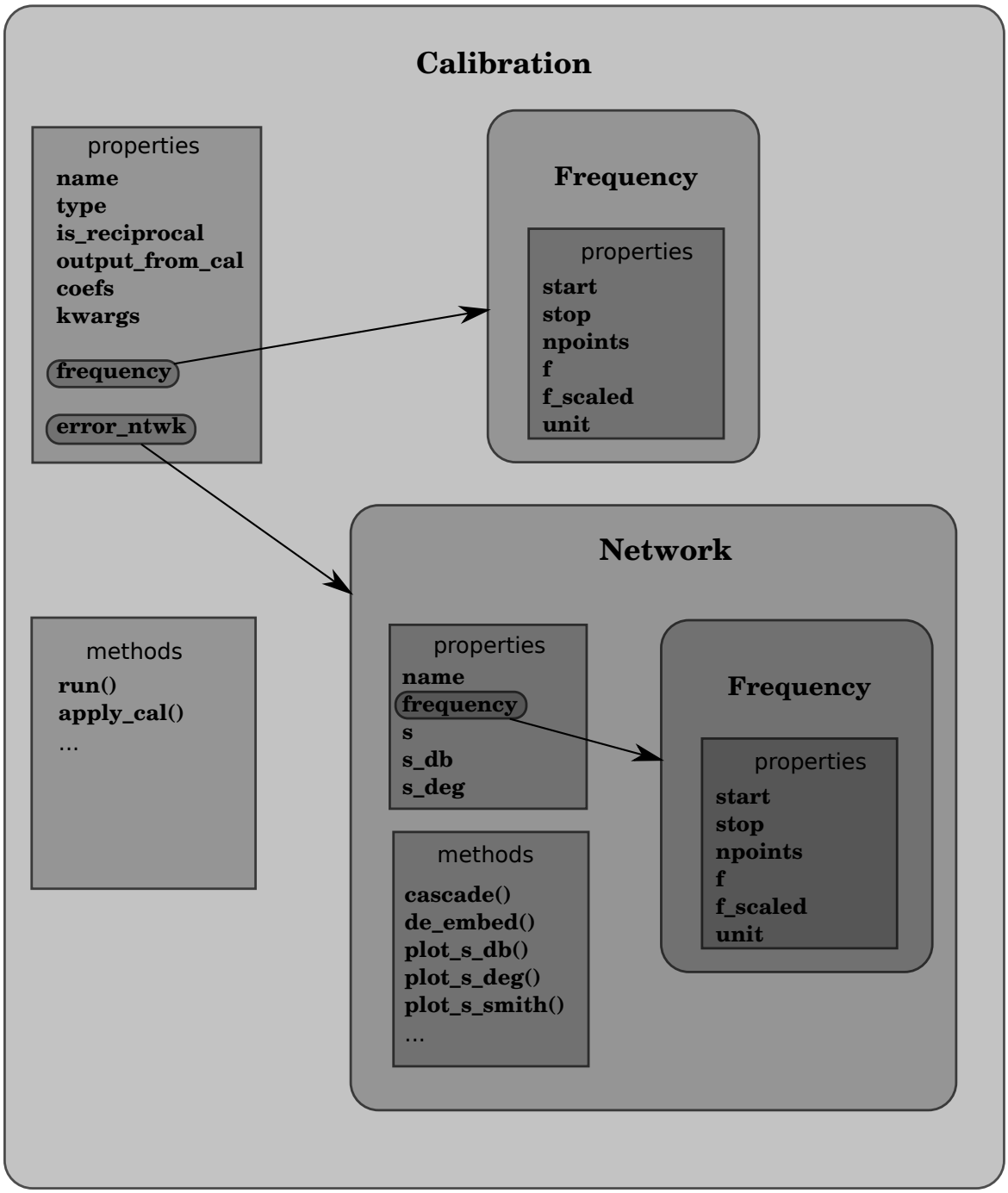

Figure 5.5: Calibration class architecture

6 Future Work