FFT-tools Documentation

Release 0.5

W. Trevor King

January 28, 2015

Contents

Module documentation:

FFT_tools	Wrap Numpy's fft module to reduce clutter.

CHAPTER 1

FFT_tools

Wrap Numpy's `fft` module to reduce clutter.

Provides a unitary discrete FFT and a windowed version based on `numpy.fft.rfft()`.

Main entry functions:

- `unitary_rfft()`
- `power_spectrum()`
- `unitary_power_spectrum()`
- `avg_power_spectrum()`
- `unitary_avg_power_spectrum()`

Functions

`avg_power_spectrum`(data[, freq, chunk_size, ...])	Compute the avgerage power spectrum of *data*.
`ceil_pow_of_two`(num)	Round *num* up to the closest exact a power of two.
`floor_pow_of_two`(num)	Round *num* down to the closest exact a power of two.
`power_spectrum`(data[, freq])	Compute the power spectrum of the time series *data*.
`round_pow_of_two`(num)	Round *num* to the closest exact a power of two on a log scale.
`unitary_avg_power_spectrum`(data[, freq, ...])	Compute the unitary average power spectrum of *data*.
`unitary_power_spectrum`(data[, freq])	Compute the unitary power spectrum of the time series *data*.
`unitary_rfft`(data[, freq])	Compute the unitary Fourier transform of real data.
`window_hann`(length)	Returns a Hann window array with length entries

Classes

`TestRFFT`([methodName])	Ensure Numpy's FFT algorithm acts as expected.
`TestUnitaryAvgPowerSpectrum`([methodName])	
`TestUnitaryPowerSpectrum`([methodName])	
`TestUnitaryRFFT`([methodName])	Verify `unitary_rfft()`.

class `FFT_tools.`**`TestRFFT`**(*methodName='runTest'*)

Ensure Numpy's FFT algorithm acts as expected.

The expected return values are [1]:

$$X_k = \sum_{m=0}^{n-1} x_m \exp^{-2\pi imk/n}$$

test_rfft()
Test NumPy's builtin numpy.fft.rfft()

class FFT_tools.**TestUnitaryRFFT**(*methodName='runTest'*)
Verify unitary_rfft().

gaussian(*a*, *t*)
Gaussian function.

$$\text{gaussian}(a, t) = \exp^{-at^2}$$

rect(*t*)
Rectangle function.

$$\text{rect}(t) = \begin{cases} 1 & \text{if } |t| < 0.5, \\ 0 & \text{if } |t| \geq 0.5. \end{cases}$$

run_gaussian(*a=1.0*, *time_shift=5.0*, *samp_freq=25.6*, *samples=256*)
Test unitary_rttf() on known function gaussian().

Analytic result:

$$\text{rfft}(\text{gaussian}(a, t)) = \sqrt{\pi/a} \cdot \text{gaussian}(1/a, \pi f)$$

run_parsevals(*xs*, *freq*, *freqs*, *Xs*)
Check the discretized integral form of Parseval's theorem

Which is:

$$\sum_{m=0}^{n-1} |x_m|^2 dt = \sum_{k=0}^{n-1} |X_k|^2 df$$

run_rect(*a=1.0*, *time_shift=5.0*, *samp_freq=25.6*, *samples=256*)
Test unitary_rfft() on known function rect().

Analytic result:

$$\text{rfft}(\text{rect}(at)) = 1/|a| \cdot \text{sinc}(f/a)$$

test_gaussian()
Test unitary FFTs on variously shaped gaussian functions.

test_parsevals()
Test unitary rfft on Parseval's theorem

test_rect()
Test unitary FFTs on variously shaped rectangular functions.

FFT_tools.**avg_power_spectrum**(*data*, *freq=1.0*, *chunk_size=2048*, *overlap=True*, *window=<function window_hann at 0x7f9cf37df410>*)
Compute the avgerage power spectrum of *data*.

[1] See the *Background information* section of numpy.fft.

data [iterable] Real (not complex) data taken with a sampling frequency *freq*.

freq [real] Sampling frequency.

chunk_size [int] Number of samples per chunk. Use a power of two.

overlap: {True,False} If *True*, each chunk overlaps the previous chunk by half its length. Otherwise, the chunks are end-to-end, and not overlapping.

window: iterable Weights used to "smooth" the chunks, there is a whole science behind windowing, but if you're not trying to squeeze every drop of information out of your data, you'll be OK with the default Hann window.

freq_axis,power [numpy.ndarray] Arrays ready for plotting.

The average power spectrum is computed by breaking *data* into chunks of length *chunk_size*. These chunks are transformed individually into frequency space and then averaged together.

See Numerical Recipes 2 section 13.4 for a good introduction to the theory.

If the number of samples in *data* is not a multiple of *chunk_size*, we ignore the extra points.

FFT_tools.**ceil_pow_of_two**(*num*)

Round *num* up to the closest exact a power of two.

```
>>> ceil_pow_of_two(3)
4
>>> ceil_pow_of_two(11)
16
>>> ceil_pow_of_two(15)
16
```

FFT_tools.**floor_pow_of_two**(*num*)

Round *num* down to the closest exact a power of two.

```
>>> floor_pow_of_two(3)
2
>>> floor_pow_of_two(11)
8
>>> floor_pow_of_two(15)
8
```

FFT_tools.**power_spectrum**(*data*, *freq=1.0*)

Compute the power spectrum of the time series *data*.

data [iterable] Real (not complex) data taken with a sampling frequency *freq*.

freq [real] Sampling frequency.

freq_axis,power [numpy.ndarray] Arrays ready for plotting.

If the number of samples in *data* is not an integer power of two, the FFT ignores some of the later points.

unitary_power_spectrum,avg_power_spectrum

FFT_tools.**round_pow_of_two**(*num*)

Round *num* to the closest exact a power of two on a log scale.

```
>>> round_pow_of_two(2.9) # Note rounding on *log scale*
4
>>> round_pow_of_two(11)
8
```

```
>>> round_pow_of_two(15)
16
```

FFT_tools.**unitary_avg_power_spectrum**(*data*, *freq=1.0*, *chunk_size=2048*, *overlap=True*, *window=<function window_hann at 0x7f9cf37df410>*)

Compute the unitary average power spectrum of *data*.

avg_power_spectrum,unitary_power_spectrum

FFT_tools.**unitary_power_spectrum**(*data*, *freq=1.0*)

Compute the unitary power spectrum of the time series *data*.

power_spectrum,unitary_avg_power_spectrum

FFT_tools.**unitary_rfft**(*data*, *freq=1.0*)

Compute the unitary Fourier transform of real data.

Unitary = preserves power [Parseval's theorem].

data [iterable] Real (not complex) data taken with a sampling frequency *freq*.

freq [real] Sampling frequency.

freq_axis,trans [numpy.ndarray] Arrays ready for plotting.

If the units on your data are Volts, and your sampling frequency is in Hz, then *freq_axis* will be in Hz, and *trans* will be in Volts.

FFT_tools.**window_hann**(*length*)

Returns a Hann window array with length entries

The Hann window with length L is defined as

$$w_i = \frac{1}{2}(1 - \cos(2\pi i/L))$$

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*

f

A

C

F

G

P

R

T

U

W