sign up log in tour help

x Dismiss

Join the Stack Overflow Community

Stack Overflow is a community of 6.4 million programmers, just like you, helping each other. Join them; it only takes a minute:

Sign up

how to extract frequency associated with fft values in python

I used `fft` function in numpy which resulted in a complex array. How to get the exact frequency values?

python numpy fft

edited Oct 10 '12 at 23:35

Ashish Gupta
7,582 11 48 102

asked Sep 12 '10 at 12:59

ria
84 1 1 5

3 Answers

`np.fft.fftfreq` tells you the frequencies associated with the coefficients:

```
import numpy as np

x = np.array([1,2,1,0,1,2,1,0])
w = np.fft.fft(x)
freqs = np.fft.fftfreq(len(x))

for coef,freq in zip(w,freqs):
    if coef:
        print('{c:>6} * exp(2 pi i t * {f})'.format(c=coef,f=freq))

# (8+0j) * exp(2 pi i t * 0.0)
#    -4j * exp(2 pi i t * 0.25)
#     4j * exp(2 pi i t * -0.25)
```

The OP asks how to find the frequency in Hertz. I believe the formula is `frequency (Hz) = abs(fft_freq * frame_rate)`.

Here is some code that demonstrates that.

First, we make a wave file at 440 Hz:

```
import math
import wave
import struct

if __name__ == '__main__':
    # http://stackoverflow.com/questions/3637350/how-to-write-stereo-wav-files-in-python
    # http://www.sonicspot.com/guide/wavefiles.html
    freq = 440.0
    data_size = 40000
    fname = "test.wav"
    frate = 11025.0
    amp = 64000.0
    nchannels = 1
    sampwidth = 2
    framerate = int(frate)
    nframes = data_size
    comptype = "NONE"
    compname = "not compressed"
    data = [math.sin(2 * math.pi * freq * (x / frate))
            for x in range(data_size)]
    wav_file = wave.open(fname, 'w')
    wav_file.setparams(
        (nchannels, sampwidth, framerate, nframes, comptype, compname))
    for v in data:
        wav_file.writeframes(struct.pack('h', int(v * amp / 2)))
    wav_file.close()
```

This creates the file `test.wav`. Now we read in the data, FFT it, find the coefficient with maximum power, and find the corresponding fft frequency, and then convert to Hertz:

```
import wave
import struct
```

```
import numpy as np

if __name__ == '__main__':
    data_size = 40000
    fname = "test.wav"
    frate = 11025.0
    wav_file = wave.open(fname, 'r')
    data = wav_file.readframes(data_size)
    wav_file.close()
    data = struct.unpack('{n}h'.format(n=data_size), data)
    data = np.array(data)

    w = np.fft.fft(data)
    freqs = np.fft.fftfreq(len(w))
    print(freqs.min(), freqs.max())
    # (-0.5, 0.499975)

    # Find the peak in the coefficients
    idx = np.argmax(np.abs(w))
    freq = freqs[idx]
    freq_in_hertz = abs(freq * frate)
    print(freq_in_hertz)
    # 439.8975
```

edited Nov 28 '14 at 13:29

answered Sep 12 '10 at 15:45

unutbu

391k 49 728 841

@unutbu: thanks a lot. Let me try it:) – ria Sep 12 '10 at 15:50

@~unutbu:But can I get the frequency values in Hertz?I want to make wav files. – ria Sep 12 '10 at 18:29

1 @PavelShvechikov: Oops, yes. You are absolutely right. Thanks for the correction. – unutbu Nov 28 '14 at 13:27

1 I found it. Basically my data is 2 channel data but your code may not working for me. – AQU Jun 14 at 11:49

1 I made the wav generation script channels to 2 and then with the script I am getting the freq specified in the wav generation script. But when I record the same. I am getting exactly half of the peak frequency value. What may I go wrong. Thanks in advance – AQU Jun 14 at 12:26

Frequencies associated with DFT values (in python)

By *fft*, Fast Fourier Transform, we understand a member of a large family of algorithms that enable the *fast* computation of the DFT, Discrete Fourier Transform, of an equisampled signal.

A DFT converts a list of *N* complex numbers to a list of *N* complex numbers, with the understanding that both lists are periodic with period *N*.

Here we deal with the `numpy` implementation of the *fft*.

In many cases, you think of

- a signal *x* defined in the time domain of length *N*, sampled at a constant interval *dt*,
- its DFT *X* (here specifically `X = np.fft.fft(x)`), whose elements are sampled on the frequency axis with a sample rate *dw*.

Some definition

- the period (aka duration) of the signal `x` , sampled at `dt` with `N` samples is is

  ```
  T = dt*N
  ```

- the fundamental frequencies (in Hz and in rad/s) of `X` , your DFT are

  ```
  df = 1/T
  dw = 2*pi/T # =df*2*pi
  ```

- the top frequency is the Nyquist frequency

  ```
  ny = dw*N/2
  ```

  (and it's not `dw*N`)

The frequencies associated with a particular element in the DFT

The frequencies corresponding to the elements in `X = np.fft.fft(x)` for a given index `0<=n<N` can be computed as follows:

```
def rad_on_s(n, N, dw):
    return dw*n if n<N/2 else dw*(n-N)
```

or in a single sweep

```
w = np.array([dw*nif n<N/2 else dw*(n-N) for n in range(N)])
```

if you prefer to consider frequencies in Hz, `s/w/f/`

```
f = np.array([df*n if n<N/2 else df*(n-N) for n in range(N)])
```

Using those frequencies

If you want to modify the original signal `x -> y` applying an operator in the frequency domain in the form of a function of frequency only, the way to go is computing the `w`'s and

```
Y = X*f(w)
y = ifft(Y)
```

Introducing `np.fft.fftfreq`

Of course `numpy` has a convenience function `np.fft.fftfreq` that returns *dimensionless frequencies* rather than *dimensional ones* but it's as easy as

```
f = np.fft.fftfreq(N)*N*df
w = np.fft.fftfreq(N)*N*dw
```

edited Apr 8 at 0:07

yuqian
167 1 8

answered Nov 28 '14 at 14:52

gboffi
5,096 1 11 34

The frequency is just the index of the array. At index *n*, the frequency is 2π*n* / the array's length (radians per unit). Consider:

```
>>> numpy.fft.fft([1,2,1,0,1,2,1,0])
array([ 8.+0.j,  0.+0.j,  0.-4.j,  0.+0.j,  0.+0.j,  0.+0.j,  0.+4.j,
        0.+0.j])
```

the result has nonzero values at indices 0, 2 and 6. There are 8 elements. This means

$$y \sim \frac{8\,e^{2\pi i/8 \times 0} - 4i\,e^{2\pi i/8 \times 2} + 4i\,e^{2\pi i/8 \times 6}}{8}$$

answered Sep 12 '10 at 13:16

kennytm
322k 63 758 818

1 thanks KennyTM.but how do u do it using python code ? – ria Sep 12 '10 at 14:14

@ria: Just associate each element with 0,1,2,3,4,.... – kennytm Sep 12 '10 at 14:57

I'm sorry. But I couldn't get it clearly.Can you tell me what are 't' and 'e' above? why did you introduce 'i*t' in the equation 2ππn/8, Is there a function in SciPy doing this calculation? – ria Sep 12 '10 at 15:48

1 @ria: e is 2.71828.... See en.wikipedia.org/wiki/Euler%27s_formula. t is the index of the original array, e.g. t=0 -> 1, t=1 -> 2, t=2 -> 1, etc. Basically, if you want to get the frequency, they are just 0/8, 1/8, 2/8, ..., 7/8. – kennytm Sep 12 '10 at 16:05

@ KennyTM:I see,the exponent 'e'.I understood. – ria Sep 12 '10 at 18:30