FOURIER TRANSFORM IN PYTHON

OCT 26, 2016

FFT FUNCTIONS

Python's default FFT function, np.fft.fft(), requires evenly spaced data.

- import numpy as np
- np.fft.fft(ArrayName)
- np.amax(ArrayName)
- pow(ArrayName, exponent)
- abs(ArrayName)

SUNSPOT PYTHON CODE AND DATA

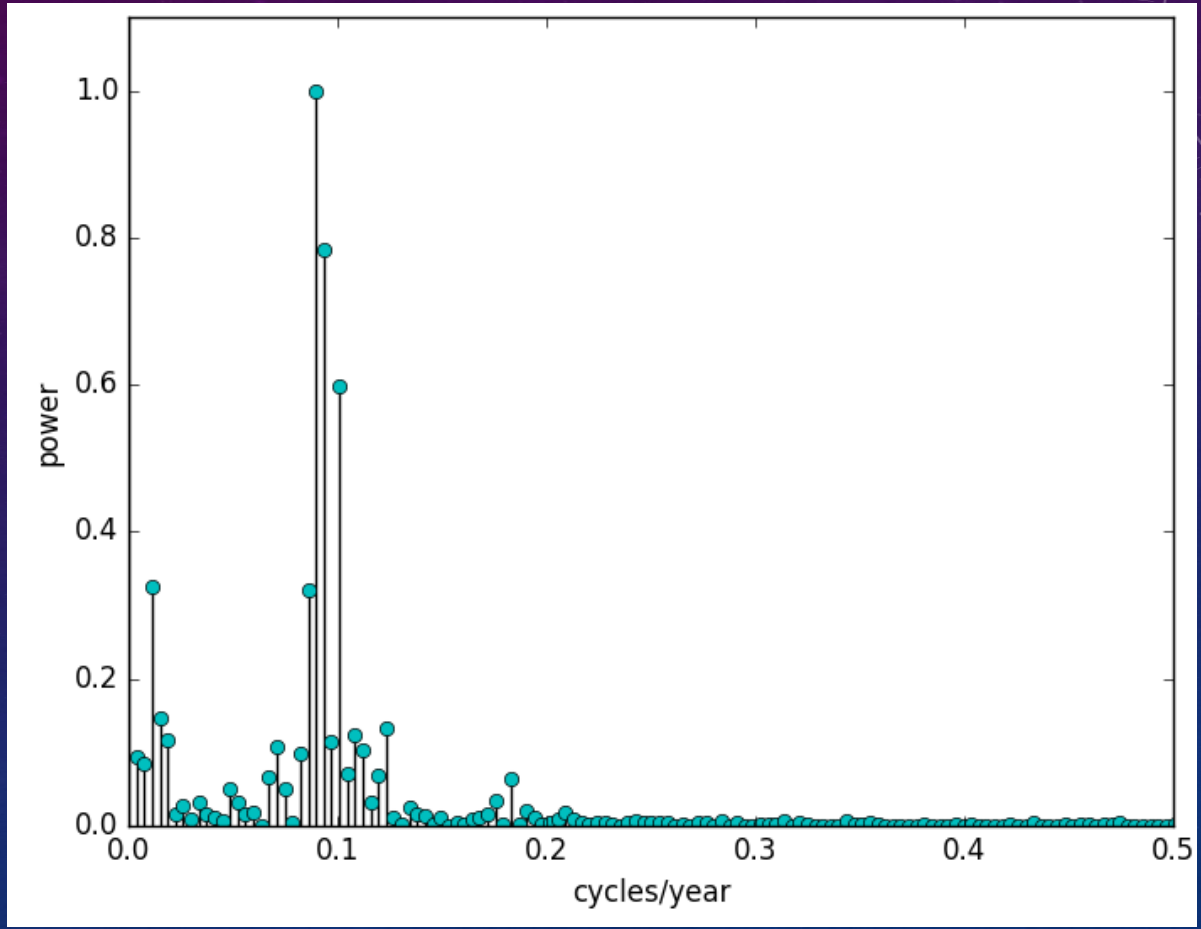
~rjs/aosc652/week_09/sunspot.py

sunspot_number_monthly.dat

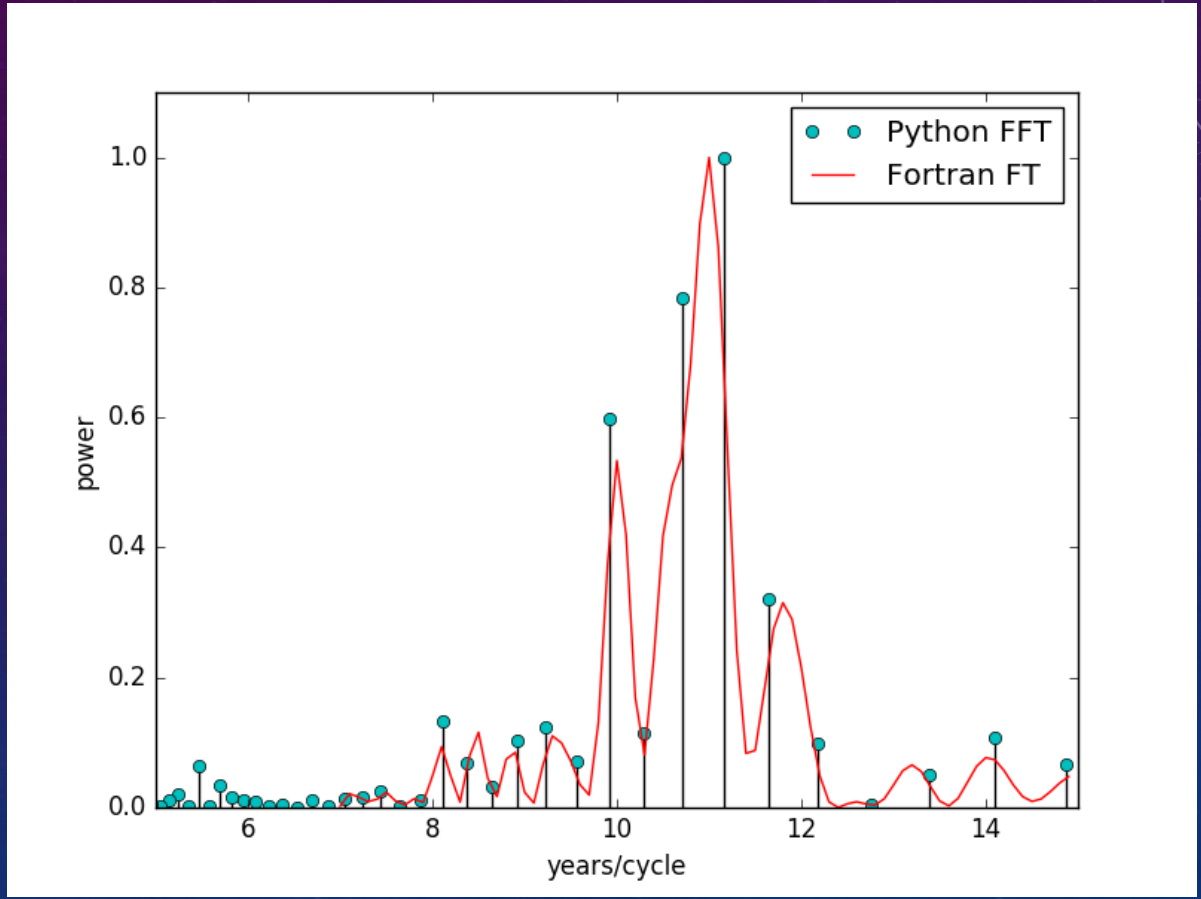
sunspot_number_monthly.pwr

~rjs/aosc652/week_09/vostok_climate_record.dat

POWER VS FREQUENCY



POWER VS FREQUENCY



1-D PLOTTING

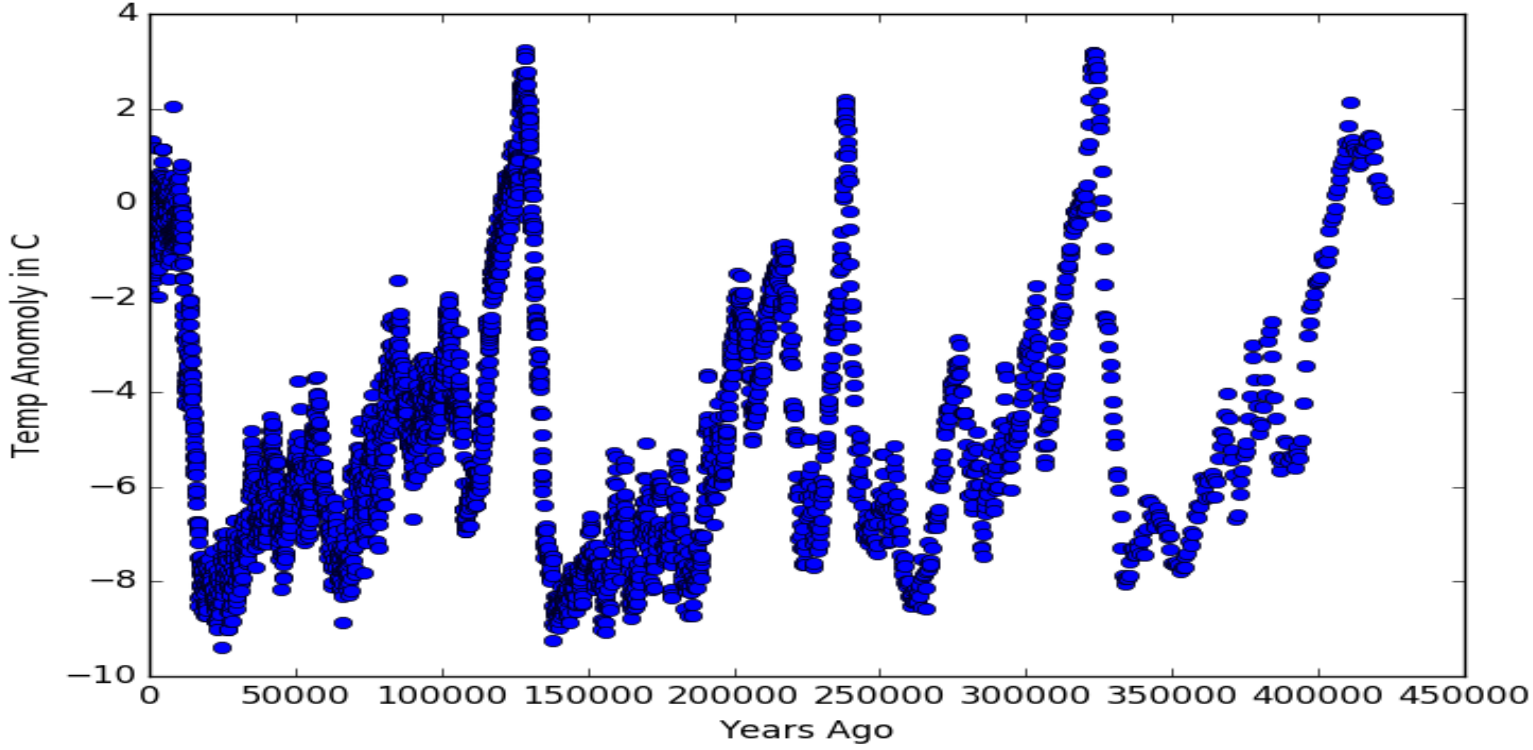
import matplotlib.pyplot as plt

```
t1=1./f
pmax = 1.1
plt.plot(t1,power_norm, marker='o', color='c', ls='none', label="Python FFT")

plt.vlines(t1,0,power_norm)

plt.plot(time2,power2_norm, label="Fortran FT", c='r')
plt.xlim(5,15)plt.ylim(0,pmax)plt.xlabel('years/cycle')
plt.ylabel('power')
plt.legend(loc=0)
plt.show()
```

THE VOSTOK CLIMATE RECORD



POWER SPECTRUM OF THE VOSTOK CLIMATE RECORD

4,105
Depth (m),Age(yrs before present),dD(per mil wrt SMOW),T diff wrt recent mean value (C)
....

8	149	-442.9	-0.81
9	170	-437.9	0.02
10	190	-435.8	0.36
11	211	-443.7	-0.95
12	234	-449.1	-1.84
13	258	-444.6	-1.09
14	281	-442.5	-0.75
15	304	-439.3	-0.22
16	327	-440.9	-0.48
17	351	-442.5	-0.75
18	375	-436.6	0.23
19	397	-430	1.33
20	420	-435.9	0.35
21	444	-436.9	0.18
22	469	-438.5	-0.08
23	495	-444.5	-1.08
24	523	-446.4	-1.39

1749	12	1749.958	142.0	-1.0	-1
1750	01	1750.042	122.2	-1.0	-1
1750	02	1750.123	126.5	-1.0	-1
1750	03	1750.204	148.7	-1.0	-1
1750	04	1750.288	147.2	-1.0	-1
1750	05	1750.371	150.0	-1.0	-1
1750	06	1750.455	166.7	-1.0	-1
1750	07	1750.538	142.3	-1.0	-1
1750	08	1750.623	171.7	-1.0	-1
1750	09	1750.707	152.0	-1.0	-1
1750	10	1750.790	109.5	-1.0	-1
1750	11	1750.874	105.5	-1.0	-1
1750	12	1750.958	125.7	-1.0	-1
1751	01	1751.042	116.7	-1.0	-1

ADDITIONAL FUNCTION NEEDED

np.interp(*x*, *xp*, *fp*, *left=None*, *right=None*, *period=None***)**

Returns the one-dimensional piecewise linear interpolant to a function with given values at discrete data-points.

https://docs.scipy.org/doc/numpy/reference/generated/numpy.interp.html

ADDITIONAL FUNCTION NEEDED

Part B requires conditioning the data with the Hanning Cosine Taper Function.

I would like you to code this in Python by hand. Though an interesting comparison would be to plot your results against the results using the SciPy function…

- **scipy.signal.hanning()**

Remember to import scipy