Astronomy 8824: Problem Set 6
Due Thursday, November 30

Fun with Fourier Transforms

1. Simple FFT Experiments

Start from the program `fftsimple.py` on the web page, which you may prettify any way you like.

This program generates a cosine wave on a 1-d grid of N points, computes its Fourier transform, and plots both the original function and the Fourier transform. Note that I have divided the FT by N; after your experimentation, see if you can figure out why.

For this problem, you can mostly just explain your results in words; provide plots if/when they are particularly relevant, but you don't need to show all of them.

The program arguments are n, the length of the array, freq, the frequency of the cosine wave, and x0, the phase of the wave. Run the program with

n=32, freq=4, x0=0

n=32, freq=8, x0=0

n=32, freq=4, x0=1

n=32, freq=8, x0=1

n=64, freq=8, x0=1

List the modulus and the real and imaginary parts for all modes that are significantly non-zero. What do you notice as you make these changes?

You are Fourier transforming a real function $f(x)$. How does this condition affect the Fourier transform $\tilde{f}(k)$?

By uncommenting the indicated line in the python script, switch from using a complex-to-complex fft to using a real-to-complex fft. Rerun the above cases and comment on what is different. The FT array returned by np.fft.rfft is only half as long as the one one returned by np.fft.fft. Why does it suffice for providing all of the information in the full FT?

To make this answer more precise, note that the length of the returned FT is actually $N/2 + 1$. Why doesn't $2 \times (N/2 + 1)$ contain 2 more independent pieces of information than the original N elements of the transformed function?

Using this real-to-complex version, run the following cases

n=32, freq=15, x0=1

n=64, freq=15, x0=1

n=32, freq=20, x0=1

n=32, freq=25, x0=1

n=64, freq=20, x0=1

n=64, freq=25, x0=1

Interpret the results.

Modify the program to add a constant to the function (a constant offset in $f(x)$). What happens to the FT?

Modify the program so that you can add a second cosine function with a different frequency, amplitude, and phase.

What happens when you FT a function that is a sum of two cosine waves of different frequencies? Illustrate your result with a plot.

2. Gaussian noise

Start from the program `fftnoise.py`, which generates Gaussian noise and measures its FT. Run a variety of cases and describe what you find, providing one or more illustrative plots.

What is the meaning of the quantities rms1 and rms2 output by this program? Explain why they are equal by appealing to Parseval's Theorem (see NR).

Uncomment the two lines that are needed to use `rfft` instead of `fft`. Check that you get equivalent results. Explain why rms2 is computed as it is in this case.

3. Mystery signals

Download the two files data1.out and data2.out from the web page. Each file represents noisy measurements of the radial velocity (in meters/second) of a star on 1024 consecutive nights.

Measure the Fourier transform of these two data sets and propose a physical interpretation of the results.

4. Gaussian convolution

Experiment with the program `gaussft.py` from the web page. This program generates a Gaussian function at a specified location and computes its Fourier transform. Note the use of a *periodic boundary condition*; if the Gaussian is centered at zero, then it appears equally on the beginning and end of the array. Verify that a Gaussian that is narrow in real space has a Fourier transform that is broad in k-space, and vice versa.

As you can see from the program and its output, the Fourier transform of a Gaussian $e^{-x^2/2\sigma^2}$ where the width σ is given in *pixels* in an array of length n is $e^{-2\pi^2 k^2 (\sigma/n)^2}$, i.e., a Gaussian of width $\sigma_k = n/(2\pi\sigma)$.

Write a program to convolve *two* Gaussians, each with an arbitrary location and width, using FFTs. Present illustrative plots, interpret the results, and demonstrate by test against an analytic result that your code works. Start with the case where both Gaussians are centered at zero, then try cases with one or both Gaussians centered elsewhere.

For plots, I recommend mapping values of $x > n/2$ to $x - n$ so that zero appears in the middle.