ASTR 5550: HW5

Jasmine Kobayashi

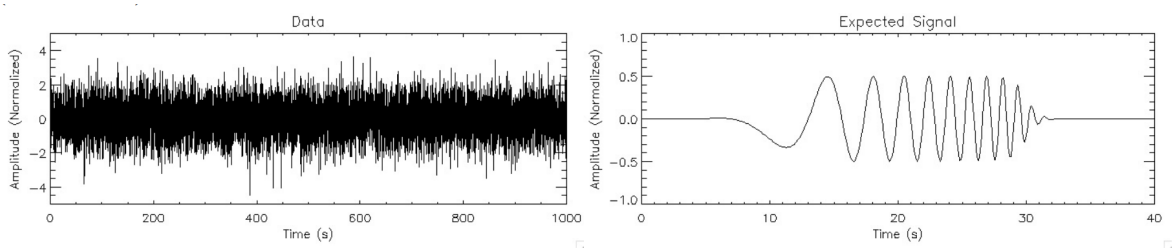
```
# Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import random
from scipy.stats import chi2
import os,sys
import seaborn as sns
sns.set_style('whitegrid')

# import helper script file
## change working directory
os.chdir("C:/Users/rokka/GH-repos/GitHubPages/Code-Reference-Notebook/CU-Boulder/AstroPhys/HW

## import my own code
import hw_helper_func2 as hf  # this is my own code I made (for probability/distribution fucn
```

3. Find the Signal

There have been many cases in which one must identify a signal buried in noise. It is easier than one thinks! LIGO, for example, offers an example in which distinct types of signals are expected. The data plotted below contains a signal embedded in random noise (fabricated). The data are ina file "`HW5_Data.txt`" (10000 values) and the signal is in the file "`HW5_Signal.txt`" (400 values).



Part (a)

Read in and verify signals. The cadence is 0.1 sec; generate time array for plotting. Let's start with the easiest method: cross-correlate the expected signal (400 points) with the data (1000 points). You will necessarily need to restrict your correlation to start at 20 sec and end at 980 sec to avoid edge effects. Plot the cross correlation as a function of time. Do you see a peak?

```
# read in data
data = np.loadtxt("hw5/HW5_Data.txt")
print("data shape:", data.shape)
data[:5]
```

```
data shape: (2000, 5)
```

```
array([[-1.4489671 , -0.05639172,  0.10236344,  0.03012188, -0.31627342],
       [-0.62731206,  1.5052398 ,  1.8104919 ,  0.41340446, -0.79464376],
       [ 0.88138539,  2.349612  , -0.44794255, -1.0353943 ,  0.92682785],
       [-0.07622421, -1.7109563 , -0.72154695,  1.3731886 ,  0.36365283],
       [-1.2074399 ,  0.32280344, -0.80500466,  1.4214896 ,  0.63947892]])
```

```
# Flatten 5d data to 1d
data_1d = data.flatten()
print(len(data_1d))
data_1d[:5]
```

```
10000
```

```
array([-1.4489671 , -0.05639172,  0.10236344,  0.03012188, -0.31627342])
```

```
# read in signal data
signal = np.loadtxt("hw5/HW5_Signal.txt")
print("signal data shape:", signal.shape)
signal[:5]
```

```
signal data shape: (80, 5)
```

```
array([[-6.2465821e-06, -1.3047378e-05, -2.0128356e-05, -2.6956645e-05,
        -3.2930560e-05],
       [-3.7414051e-05, -3.9777609e-05, -3.9443527e-05, -3.5932957e-05,
        -2.8911940e-05],
       [-1.8233456e-05, -3.9726125e-06,  1.3547713e-05,  3.3743037e-05,
         5.5767006e-05],
       [ 7.8525801e-05,  1.0070760e-04,  1.2082675e-04,  1.3728147e-04,
         1.4842331e-04],
       [ 1.5263583e-04,  1.4841952e-04,  1.3447950e-04,  1.0981237e-04,
         7.3788494e-05]])
```

```
# Flatten to 1d
signal_1d = signal.flatten()
print(len(signal_1d))
signal_1d[:5]
```

```
400
```

```
array([-6.2465821e-06, -1.3047378e-05, -2.0128356e-05, -2.6956645e-05,
       -3.2930560e-05])
```

```
# TODO: cross correlate (i.e. the rest of Q3 part a)
```

Part (b)

Isolate the maximum correlation value ($C_{\max}$) and then calculate the standard deviation (σ_C) of the 9600 correlation values. Waht is $C_{\max}/\sigma_C$? What is the probability of measuring such a correlation (or greater) given an Gaussian parent distribution?

Hint:

$$P = \frac{1}{2}\left[1 - \text{erf}\left(\frac{C_{\max}}{\sqrt{2}\sigma_C}\right)\right]; \text{ use double precision}$$

Part (c)

At this point one may think that this event cannot possibly be a random fluctuation. However, how many 0.1 s periods are there in a year? There also are ~10^4 different variations in the expected signal. Given that information, how many random events would one expect in a year?

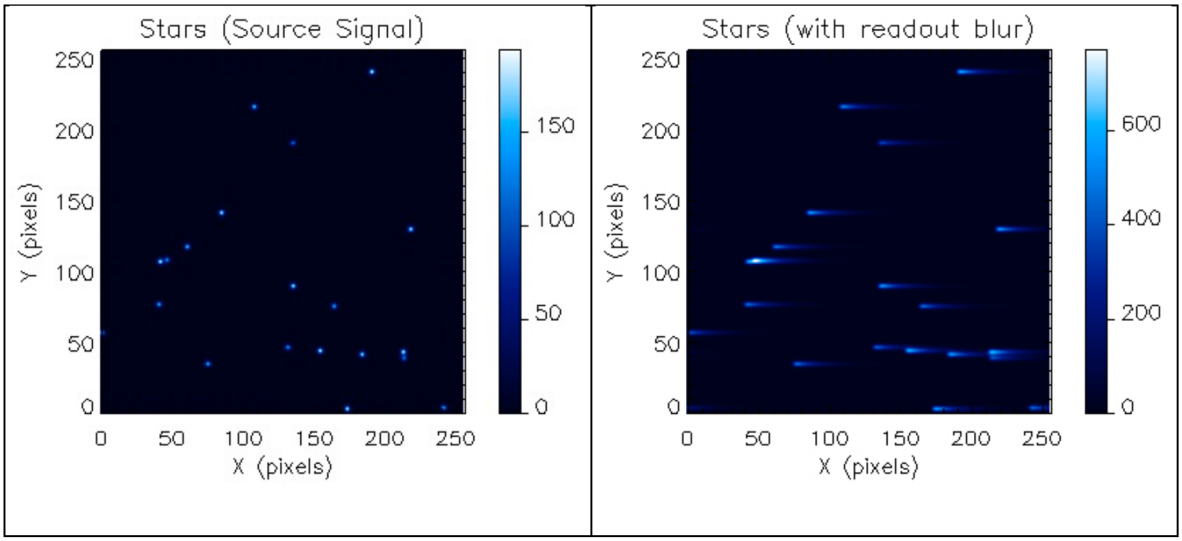
Part (d)

However, suppose that one made two such measurements (as did LIGO) and that the second data set had a correlation of 4.6 σ_C within the expected speed-of-light delay. Given the 2$^{\text{nd}}$ detection, what is the probability that these two measurements are from a random fluctuation?

4. Convolution Theorem

An older telescope has a horizontal CCD readout blur (see below). Unfortunately, the CCD readout has a memory of the previous pixel that exponentially fades causing horizontal streaks.

Examine the two images below.



Part (a)

The blurred image is in a text file "`HW5_BlurredImage.txt`", which contains 256×256-point array (I_{Blur}). Read in this image and plot to verify

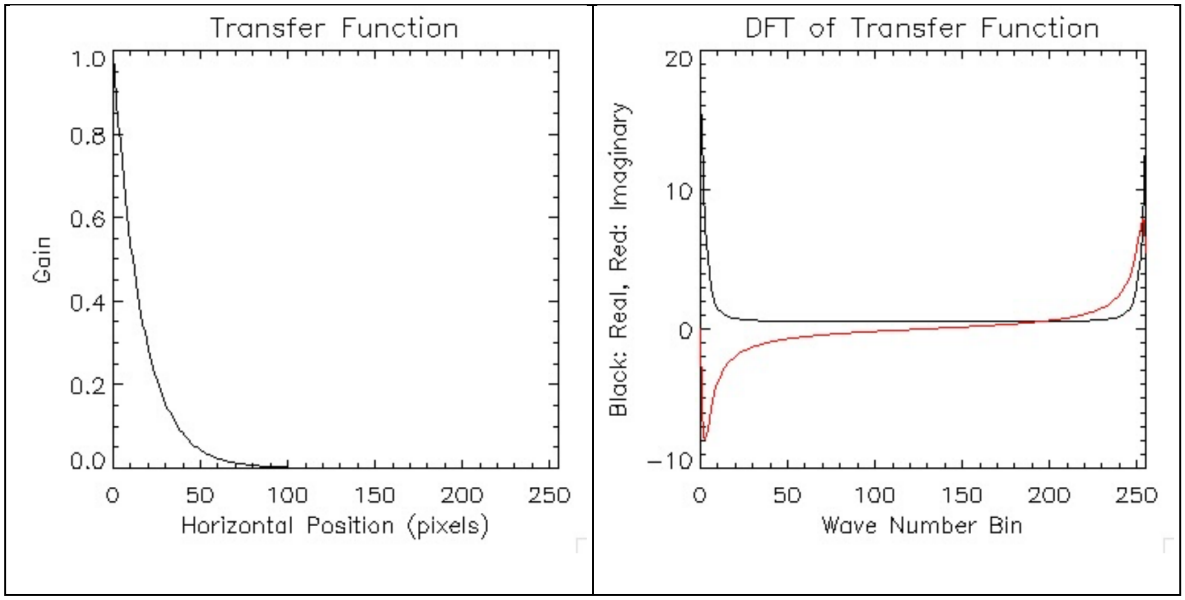
```
# Read BlurredImage text file
# TODO: Figure out how to format to workable data to plot
I_blur = pd.read_fwf("hw5/HW5_BlurredImage.txt",sep=" ",header=None)
I_blur

# TODO: Plot Blurred Image data
```

	0	1	2	3	4
0	4.330307e+00	4.067947e+00	3.821483e+00	3.589951e+00	3.372447e+00
1	3.168120e+00	2.976174e+00	2.795856e+00	2.626464e+00	2.467335e+00
2	2.317846e+00	2.177415e+00	2.045492e+00	1.921562e+00	1.805141e+00
3	1.695773e+00	1.593031e+00	1.496514e+00	1.405845e+00	1.320669e+00
4	1.240654e+00	1.165486e+00	1.094873e+00	1.028538e+00	9.662221e-01
...	...	...	...	...	...
13307	1.335766e-21	1.254836e-21	1.178809e-21	1.107389e-21	1.040296e-21
13308	9.772671e-22	9.180575e-22	8.624352e-22	8.101829e-22	7.610964e-22
13309	7.149839e-22	6.716652e-22	6.309711e-22	5.927425e-22	5.568300e-22
13310	5.230934e-22	4.914008e-22	4.616283e-22	4.336597e-22	4.073856e-22
13311	3.827033e-22	NaN	NaN	NaN	NaN

Part (b)

Fortunately, the transfer function is fairly well known. It is in the text file "`HW5_TransferFunc.txt`", which contains a 256-point array. The transfer function and it's DFT are shown below.



Part (c)

Read in and perform a DFT on the transfer function, $h(j)$. Plot your results and compare to above (and/or perform a reverse DFT to check). Pixel numbers are positive integers sot that one can calculate:

$$\tilde{h}(n) = \sum_{j=0}^{N-1} h(j) e^{-i2\pi nj/N}$$

```
jarr = [0,1,...255], npts=256
for n = 0, npts-1 do h_rl(n) = total(h*cos(jarr*n*2*!dpi/npts)) ; h is 256-point transfer fu
for n = 0, npts-1 do h_im(n) = -total(h*sin(jarr*n*2*!dpi/npts))
```

Note: One can use an FFT (with no window) if you prefer, but take care to understand how it works. I recommend doing a reverse FFT to check.

```
# read in transfer function
# TODO: Somehow read as 1d array
transfer = pd.read_fwf("hw5/HW5_TransferFunc.txt",sep=" ",header=None)
transfer
```

	0	1	2	3	4
0	1.000000e+00	9.394131e-01	8.824969e-01	8.290291e-01	7.788008e-01

	0	1	2	3	4
1	7.316156e-01	6.872893e-01	6.456485e-01	6.065307e-01	5.697828e-01
2	5.352614e-01	5.028316e-01	4.723665e-01	4.437473e-01	4.168620e-01
3	3.916056e-01	3.678794e-01	3.455908e-01	3.246525e-01	3.049828e-01
4	2.865048e-01	2.691463e-01	2.528396e-01	2.375208e-01	2.231302e-01
5	2.096114e-01	1.969117e-01	1.849814e-01	1.737739e-01	1.632455e-01
6	1.533550e-01	1.440637e-01	1.353353e-01	1.271357e-01	1.194330e-01
7	1.121969e-01	1.053992e-01	9.901341e-02	9.301449e-02	8.737903e-02
8	8.208500e-02	7.711172e-02	7.243976e-02	6.805085e-02	6.392786e-02
9	6.005467e-02	5.641614e-02	5.299806e-02	4.978707e-02	4.677062e-02
10	4.393693e-02	4.127493e-02	3.877421e-02	3.642500e-02	3.421812e-02
11	3.214495e-02	3.019738e-02	2.836782e-02	2.664910e-02	2.503451e-02
12	2.351775e-02	2.209288e-02	2.075434e-02	1.949690e-02	1.831564e-02
13	1.720595e-02	1.616349e-02	1.518420e-02	1.426423e-02	1.340001e-02
14	1.258814e-02	1.182547e-02	1.110900e-02	1.043594e-02	9.803655e-03
15	9.209682e-03	8.651695e-03	8.127515e-03	7.635094e-03	7.172507e-03
16	6.737947e-03	6.329715e-03	5.946217e-03	5.585954e-03	5.247518e-03
17	4.929587e-03	4.630919e-03	4.350346e-03	4.086771e-03	3.839166e-03
18	3.606563e-03	3.388053e-03	3.182781e-03	2.989946e-03	2.808794e-03
19	2.638618e-03	2.478752e-03	2.328572e-03	2.187491e-03	2.054958e-03
20	1.930454e-03	1.813494e-03	1.703620e-03	1.600403e-03	1.503439e-03
21	1.412350e-03	1.326780e-03	1.246395e-03	1.170880e-03	1.099940e-03
22	1.033298e-03	9.706933e-04	9.118820e-04	8.566338e-04	8.047330e-04
23	7.559767e-04	7.101744e-04	6.671471e-04	6.267267e-04	5.887553e-04
24	5.530844e-04	5.195747e-04	4.880952e-04	4.585230e-04	4.307425e-04
25	4.046452e-04	3.801290e-04	3.570981e-04	3.354626e-04	3.151380e-04
26	2.960447e-04	2.781083e-04	2.612586e-04	2.454297e-04	2.305599e-04
27	2.165910e-04	2.034684e-04	1.911408e-04	1.795602e-04	1.686812e-04
28	1.584613e-04	1.488606e-04	1.398416e-04	1.313691e-04	1.234098e-04
29	1.159328e-04	1.089088e-04	1.023103e-04	9.611165e-05	9.028854e-05
30	8.481823e-05	7.967936e-05	7.485183e-05	7.031679e-05	6.605651e-05
31	6.205435e-05	5.829466e-05	5.476277e-05	5.144486e-05	4.832797e-05
32	4.539993e-05	4.264929e-05	4.006530e-05	3.763786e-05	3.535750e-05
33	3.321530e-05	3.120288e-05	2.931240e-05	2.753645e-05	2.586810e-05
34	2.430083e-05	2.282852e-05	2.144541e-05	2.014610e-05	1.892551e-05
35	1.777887e-05	1.670170e-05	1.568980e-05	1.473920e-05	1.384620e-05
36	1.300730e-05	1.221922e-05	1.147890e-05	1.078343e-05	1.013009e-05
37	9.516342e-06	8.939776e-06	8.398143e-06	7.889325e-06	7.411335e-06
38	6.962305e-06	6.540480e-06	6.144212e-06	5.771953e-06	5.422248e-06
39	5.093731e-06	4.785117e-06	4.495202e-06	4.222851e-06	3.967002e-06
40	3.726653e-06	3.500867e-06	3.288760e-06	3.089504e-06	2.902320e-06
41	2.726478e-06	2.561289e-06	2.406108e-06	2.260329e-06	2.123383e-06

	0	1	2	3	4
42	1.994734e-06	1.873879e-06	1.760346e-06	1.653692e-06	1.553500e-06
43	1.459378e-06	1.370959e-06	1.287897e-06	1.209867e-06	1.136565e-06
44	1.067704e-06	1.003015e-06	9.422455e-07	8.851577e-07	8.315287e-07
45	7.811489e-07	7.338215e-07	6.893615e-07	6.475952e-07	6.083594e-07
46	5.715008e-07	5.368753e-07	5.043477e-07	4.737908e-07	4.450852e-07
47	4.181189e-07	3.927864e-07	3.689886e-07	3.466327e-07	3.256313e-07
48	3.059023e-07	2.873686e-07	2.699578e-07	2.536019e-07	2.382370e-07
49	2.238029e-07	2.102434e-07	1.975054e-07	1.855391e-07	1.742979e-07
50	1.637377e-07	1.538174e-07	1.444980e-07	1.357433e-07	1.275191e-07
51	1.197931e-07	NaN	NaN	NaN	NaN

Part (d)

Perform a DFT on the blurred image line by line in y. Be careful here. One must perform 256 DFTs; one for each position in y. I recommend finding a y value a star and testing your code.

$$\tilde{I}_{\text{Blur}}(n, y) = \sum_{j=0}^{N-1} I_{\text{Blur}}(j, y) e^{-i2\pi nj/N}$$

```
#TODO: Perform DFT on Blurred Image
```

Part (e)

Carefully calculate (watch out for signs!) line by line in y:

$$\tilde{I}_{\text{Source}}(n, y) = \frac{\tilde{I}_{\text{Blur}}(n, y)\tilde{h}^*(n)}{|\tilde{h}(n)|^2}$$

Part (f)

Perform a reverse DFT on $\tilde{I}_{Source}$. Plot your results. The image should be close but not exactly equal to the source image plotted above.

Hint: The most common error is in reconstructing $\tilde{I}_{\text{Source}}$. Check your reverse DFT by deriving h from $\tilde{h}$.

```
#TODO: Reverse DFT

#TODO: Plot results
```