EECS 16A HW12

Bryan Ngo

2019-11-20

1 Op Amp with Output Current

1.a

There are three equations that we concern ourselves with:

$$V_{out} = A(V_{\text{ref}} - V^-) \tag{1}$$

$$V^- = I_O R \tag{2}$$

$$V^- = V_{out} - \frac{I_O}{g_{ms}} \tag{3}$$

setting (2) and (3) equal and plugging in (1),

$$I_O R = AV_{\text{ref}} - AI_O R - \frac{I_O}{g_{ms}} \tag{4}$$

$$I_O \left(R + AR + \frac{1}{g_{ms}} \right) = AV_{\text{ref}} \tag{5}$$

$$I_O = \frac{AV_{\text{ref}}}{R + AR + \frac{1}{g_{ms}}} \tag{6}$$

Since there is no expression for R_L in the answer, the current is independent of R_L.

1.b

The circuit is considered a voltage-controlled current source.

2 Cauchy-Schwarz Inequality

2.a

$$||\mathbf{v}|| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle} = \sqrt{r^2 \cos^2(\theta) + r^2 \sin^2(\theta)} = r\cancelto{1}{\sqrt{\cos^2(\theta) + \sin^2(\theta)}} = r \tag{7}$$

$$||\mathbf{w}|| = \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle} = \sqrt{t^2 \cos^2(\phi) + t^2 \sin^2(\phi)} = t\cancelto{1}{\sqrt{\cos^2(\phi) + \sin^2(\phi)}} = t \tag{8}$$

2.b

$$\langle \mathbf{v}, \mathbf{w} \rangle = rt\cos(\theta)\cos(\phi) + rt\sin(\theta)\sin(\phi) = rt(\cos(\theta - \phi)) = ||\mathbf{v}||||\mathbf{w}||\cos(\theta - \phi) \tag{9}$$

2.c

Proof. By definition of the cosine function,

$$-1 \leq \cos(\theta - \phi) \leq 1 \tag{10}$$
$$(||\mathbf{v}||||\mathbf{w}||) \cdot -1 \leq (||\mathbf{v}||||\mathbf{w}||) \cdot \cos(\theta - \phi) \leq (||\mathbf{v}||||\mathbf{w}||) \cdot 1 \tag{11}$$
$$\text{abs}(||\mathbf{v}||||\mathbf{w}||\cos(\theta - \phi)) \leq ||\mathbf{v}||||\mathbf{w}|| \tag{12}$$
$$|\langle \mathbf{v}, \mathbf{w} \rangle| \leq ||\mathbf{v}||||\mathbf{w}|| \tag{13}$$

□

3 Mechanical Linear Correlation

3.b

$\mathbf{s}_1[n]$	0	0	0	2	-2	2	-2	0	0	0	
$\mathbf{s}_2[n+3]$	1	2	3	4	0	0	0	0	0	0	
$\langle \mathbf{s}_1[n], \mathbf{s}_2[n+3] \rangle$	0	0	0	8	0	0	0	0	0	0	$= 8$
$\mathbf{s}_1[n]$	0	0	0	2	-2	2	-2	0	0	0	
$\mathbf{s}_2[n+2]$	0	1	2	3	4	0	0	0	0	0	
$\langle \mathbf{s}_1[n], \mathbf{s}_2[n+2] \rangle$	0	0	0	6	-8	0	0	0	0	0	$= -2$

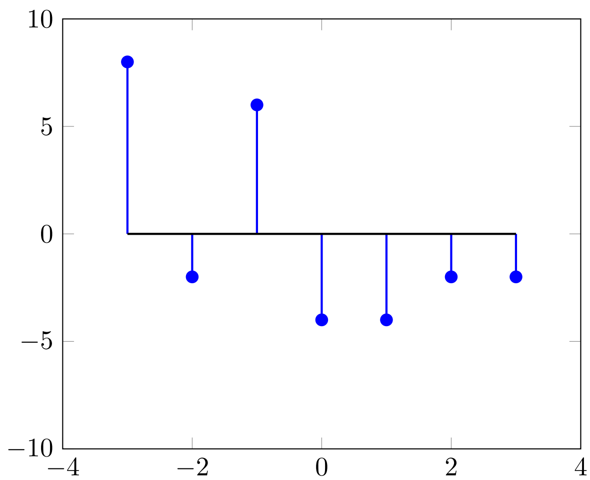
$\mathbf{s}_1[n]$	0	0	0	2	-2	2	-2	0	0	0	
$\mathbf{s}_2[n+1]$	0	0	1	2	3	4	0	0	0	0	
$\langle \mathbf{s}_1[n], \mathbf{s}_2[n+1] \rangle$	0	0	0	4	-6	8	0	0	0	0	$= 6$

$\mathbf{s}_1[n]$	0	0	0	2	-2	2	-2	0	0	0	
$\mathbf{s}_2[n]$	0	0	0	1	2	3	4	0	0	0	
$\langle \mathbf{s}_1[n], \mathbf{s}_2[n] \rangle$	0	0	0	2	-4	6	-8	0	0	0	$= -4$

$\mathbf{s}_1[n]$	0	0	0	2	-2	2	-2	0	0	0	
$\mathbf{s}_2[n-1]$	0	0	0	0	1	2	3	4	0	0	
$\langle \mathbf{s}_1[n], \mathbf{s}_2[n-1] \rangle$	0	0	0	0	-2	4	-6	0	0	0	$= -4$

$\mathbf{s}_1[n]$	0	0	0	2	-2	2	-2	0	0	0	
$\mathbf{s}_2[n-2]$	0	0	0	0	0	1	2	3	4	0	
$\langle \mathbf{s}_1[n], \mathbf{s}_2[n-2] \rangle$	0	0	0	0	0	2	-4	0	0	0	$= -2$

$\mathbf{s}_1[n]$	0	0	0	2	-2	2	-2	0	0	0	
$\mathbf{s}_2[n-3]$	0	0	0	0	0	0	1	2	3	4	
$\langle \mathbf{s}_1[n], \mathbf{s}_2[n-3] \rangle$	0	0	0	0	0	0	-2	0	0	0	$= -2$



3.c

The iPython notebook tells us that when the vectors correlated switch spots, the entire graph flips. This tells us that

$$\mathrm{corr}_{\mathbf{s}_1}(\mathbf{s}_2)[k] = \mathrm{corr}_{\mathbf{s}_2}(\mathbf{s}_1)[-k] \tag{14}$$

4 Audio File Matching

4.a

If $\mathbf{x}_1 = \mathbf{x}_2$,

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \langle \mathbf{x}_1, \mathbf{x}_1 \rangle = ||\mathbf{x}_1||^2 = \underbrace{1^2 + 1^2 + \cdots}_{N \text{ times}} = N \tag{15}$$

In the second case,

$$\langle \mathbf{x}_1, \mathbf{x}_2 \rangle = 1(1) + 1(-1) + \cdots = 0 \tag{16}$$

This only works when N is even because then there are as many negatives as there are positives so it cancels out. We can use the inner product to determine the similarity of two vectors, since geometrically it gives us a measure of how ”parallel” two vectors are.

4.b

Finding $\text{corr}_{\mathbf{x}}(\mathbf{y}_1[k])$,

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n+2]$	1	1	1	0	0	0	0	0	0	0	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[-2])$	0	0	1	0	0	0	0	0	0	0	0	0	$= 1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n+1]$	0	1	1	1	0	0	0	0	0	0	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[-1])$	0	0	-1	-1	0	0	0	0	0	0	0	0	$= -2$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n]$	0	0	1	1	1	0	0	0	0	0	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[0])$	0	0	-1	1	-1	0	0	0	0	0	0	0	$= -1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-1]$	0	0	0	1	1	1	0	0	0	0	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[1])$	0	0	0	1	1	1	0	0	0	0	0	0	$= 3$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-2]$	0	0	0	0	1	1	1	0	0	0	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[2])$	0	0	0	0	1	-1	-1	0	0	0	0	0	$= -2$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-3]$	0	0	0	0	0	1	1	1	0	0	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[3])$	0	0	0	0	0	-1	1	-1	0	0	0	0	$= -2$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-4]$	0	0	0	0	0	0	1	1	1	0	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[4])$	0	0	0	0	0	0	1	1	1	0	0	0	$= 3$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-5]$	0	0	0	0	0	0	0	1	1	1	0	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[5])$	0	0	0	0	0	0	0	1	-1	-1	0	0	$= -2$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-6]$	0	0	0	0	0	0	0	0	1	1	1	0	
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[6])$	0	0	0	0	0	0	0	0	-1	1	0	0	$= 0$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-7]$	0	0	0	0	0	0	0	0	0	1	1	1	0
$\text{corr}_{\mathbf{x}}(\mathbf{y}_1[7])$	0	0	0	0	0	0	0	0	0	1	0	0	$= 1$

Here we find $\max\{\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[k])\} = 3$ at $k = 1, 4$. For $\mathbf{y}_2$,

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}[n+2]$	1	1	1	0	0	0	0	0	0	0	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[-2])$	0	0	-1	0	0	0	0	0	0	0	0	0	$= -1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n+1]$	0	1	1	1	0	0	0	0	0	0	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[-1])$	0	0	-1	1	0	0	0	0	0	0	0	0	$= 0$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n]$	0	0	1	1	1	0	0	0	0	0	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[0])$	0	0	-1	1	1	0	0	0	0	0	0	0	$= 1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-1]$	0	0	0	1	1	1	0	0	0	0	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[1])$	0	0	0	1	1	-1	0	0	0	0	0	0	$= 1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-2]$	0	0	0	0	1	1	1	0	0	0	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[2])$	0	0	0	0	1	-1	1	0	0	0	0	0	$= 1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-3]$	0	0	0	0	0	1	1	1	0	0	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[3])$	0	0	0	0	0	-1	1	1	0	0	0	0	$= 1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-4]$	0	0	0	0	0	0	1	1	1	0	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[4])$	0	0	0	0	0	0	1	1	-1	0	0	0	$= 1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-5]$	0	0	0	0	0	0	0	1	1	1	0	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[5])$	0	0	0	0	0	0	0	1	-1	1	0	0	$= 1$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-6]$	0	0	0	0	0	0	0	0	1	1	1	0	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[6])$	0	0	0	0	0	0	0	0	-1	1	0	0	$= 0$

$\mathbf{x}[n]$	0	0	-1	1	1	-1	1	1	-1	1	0	0	
$\mathbf{y}_1[n-7]$	0	0	0	0	0	0	0	0	0	1	1	1	
$\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[7])$	0	0	0	0	0	0	0	0	0	1	0	0	$= 1$

Here we find $\max\{\mathrm{corr}_{\mathbf{x}}(\mathbf{y}_1[k])\} = 1$ at $k \in [0, 5]$.

4.c

We will expect a correlation peak at $k = 4$, as you will get

$$\langle \mathbf{x}, \mathbf{y}[4] \rangle = 2 \cdot 1 + 2 \cdot 2 + 3 \cdot 3 = 15 \tag{17}$$

which is clearly what is not intended, since although the magnitudes are highest here, that does not necessarily mean the pitches, or directions, line up.

5 GPS Receivers

5.a

We observe that there is a relatively low correlation until the gold code matches with itself at offset zero, at which point a large spike of 1023 is observed, confirming a high auto-correlation.

5.b

We observe a very low correlation between satellite 13 and satellite 10, with a peak of about 80.

5.c

We notice that even noise yields a very low correlation between it and satellite 10, with a max correlation of only about 100.

5.d

We notice that even *Gaussian* noise yields a very low correlation between it and satellite 10, with a max correlation of only about 100.

5.e

We find peaks for `signal1` at Satellites $\{4, 7, 13, 19\}$.

5.f

Satellite 3 emits a signal $\{1, -1, -1, -1, 1\}$.

5.g

Satellite 5 has a delay of 0, and Satellite 20 has a delay of +506.

6 *Gold*en Positioning System

6.a

The satellite is transmitting the message $\{1, -1, -1, 1\}$.

6.b

$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n+1]$	1	1	-1	1	-1	0	0	0	0	0	0	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[-1]$	0	1	-1	-1	-1	0	0	0	0	0	0	0	$= -2$

$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n]$	0	1	1	-1	1	-1	0	0	0	0	0	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[0]$	0	1	1	1	1	1	0	0	0	0	0	0	$= 5$

$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n-1]$	0	0	1	1	-1	1	-1	0	0	0	0	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[1]$	0	0	1	-1	-1	-1	1	0	0	0	0	0	$= -1$

$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n-2]$	0	0	1	1	-1	1	-1	0	0	0	0	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[2]$	0	0	1	-1	-1	-1	1	0	0	0	0	0	$= -1$

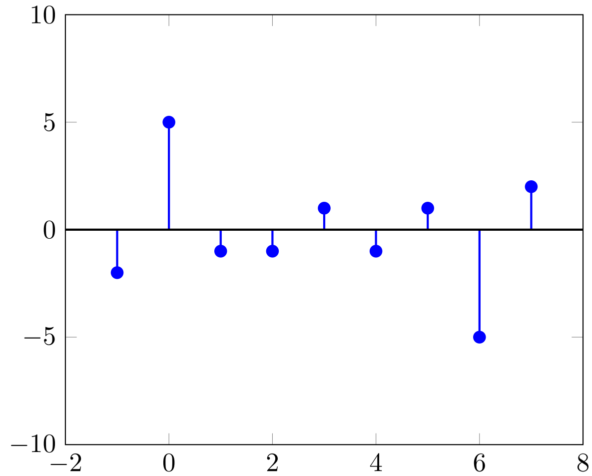
$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n-3]$	0	0	0	1	1	-1	1	-1	0	0	0	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[3]$	0	0	0	-1	1	1	-1	1	0	0	0	0	$= 1$

$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n-4]$	0	0	0	0	1	1	-1	1	-1	0	0	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[4]$	0	0	0	0	1	-1	1	-1	-1	0	0	0	$= -1$

$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n-5]$	0	0	0	0	0	1	1	-1	1	-1	0	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[5]$	0	0	0	0	0	-1	-1	1	1	1	0	0	$= 1$

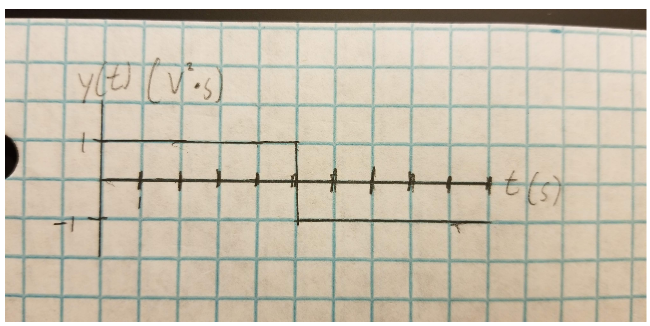
$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n-6]$	0	0	0	0	0	0	1	1	-1	1	-1	0	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[6]$	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	$= -5$

$\mathbf{r}$	0	1	1	-1	1	-1	-1	-1	1	-1	1	0	
$\mathbf{g}[n-7]$	0	0	0	0	0	0	0	1	1	-1	1	-1	
$\text{corr}_{\mathbf{r}}(\mathbf{g})[7]$	0	0	0	0	0	0	0	-1	1	1	1	0	$= 2$

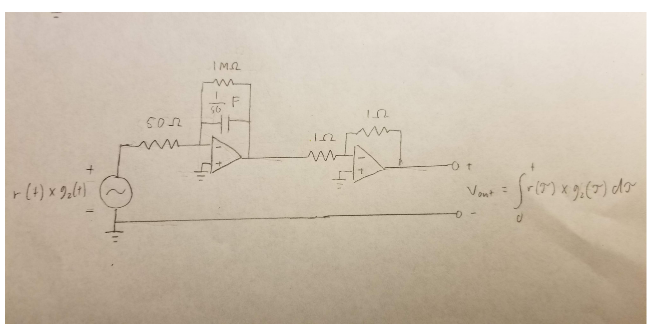


The peaks show where the signals match up with the Gold code, and their sign represents the modulation, in this case $\{1, -1\}$.

6.c



6.d



6.e

The equation for the distance is

$$d_i = c(T_i - S_i) \tag{18}$$

6.f

Four satellites are needed in order to uniquely determine the position of the satellite in 3D space.

7 Homework Process and Study Group

I did this homework by myself.

prob12

November 22, 2019

1 EECS16A Homework 12

1.1 Question 3: Mechanical Correlation

1.1.1 Part (c)

[1]:
```
import numpy as np
s1 = [2, -2, 2, -2]
s2 = [1, 2, 3, 4]

# Use the function np.correlate with mode='full' for linear cross correlation.
## your code here
print(np.correlate(s1, s2, 'full'))
print(np.correlate(s2, s1, 'full'))
```

```
[ 8 -2  6 -4 -4 -2 -2]
[-2 -2 -4 -4  6 -2  8]
```

1.2 Question 4: Audio File Matching

This notebook continues the audio file matching problem. Be sure to have song.wav and clip.wav in the same directory as the notebook.

In this notebook, we will look at the problem of searching for a small audio clip inside a song.

The song "Mandelbrot Set" by Jonathan Coulton is licensed under CC BY-NC 3.0

If you have trouble playing the audio file in IPython, try opening it in a different browser. I encountered problem with Safari but Chrome works for me.

[ ]:
```
import numpy as np
import wave
import matplotlib.pyplot as plt
import scipy.io.wavfile
import operator
from IPython.display import Audio
%matplotlib inline

given_file = 'song.wav'
target_file = 'clip.wav'
rate_given,  given_signal  = scipy.io.wavfile.read(given_file)
```

```
rate_target, target_signal = scipy.io.wavfile.read(target_file)
given_signal  = given_signal[:2000000].astype(float)
target_signal = target_signal.astype(float)
def play_clip(start, end, signal=given_signal):
    scipy.io.wavfile.write('temp.wav', rate_given, signal[start:end].astype(np.
 ↪int16))
    return Audio(url='temp.wav', autoplay=True)

def run_comparison(target_signal, given_signal, idxs=None):
    # Run everything if not called with idxs set to something
    if idxs is None:
        idxs = [i for i in range(len(given_signal)-len(target_signal))]
    return idxs, [vector_compare(target_signal, given_signal[i:
 ↪i+len(target_signal)])
                 for i in idxs]

play_clip(0, len(given_signal))
```

We will load the song into the variable `given_signal` and load the short clip into the variable `target_signal`. Your job is to finish code that will identify the short clip's location in the song. The clip we are trying to find will play after executing the following block.

[ ]:
```
Audio(url=target_file, autoplay=True)
```

Your task is to define the function 'vector_compare' and run the following code. Because the song has a lot of data, you should use the provided examples from the previous parts of the problem before running the later code. Do you results here make sense given your answers to previous parts of the problem?

[ ]:
```
def vector_compare(short_clip, segment_of_song):
    """This function compares two vectors, returning a number.
    The test vector with the highest return value is regarded as being closest␣
 ↪to the desired vector."""
    return ## your code here

print("PART A:")
print(vector_compare(np.array([1,1,1]), np.array([1,1,1])))
print(vector_compare(np.array([1,1,1]), np.array([-1,-1,-1])))
print("PART C:")
print(vector_compare(np.array([1,2,3]), np.array([1,2,3])))
print(vector_compare(np.array([1,2,3]), np.array([2,3,4])))
print(vector_compare(np.array([1,2,3]), np.array([3,4,5])))
print(vector_compare(np.array([1,2,3]), np.array([4,5,6])))
print(vector_compare(np.array([1,2,3]), np.array([5,6,7])))
print(vector_compare(np.array([1,2,3]), np.array([6,7,8])))
```

1.2.1 Part (e)

Run the following code that runs vector_compare on every subsequence in the song- it will probably take at least 5 minutes. How do you interpret this plot to find where the clip is in the song?

[ ]:
```
import time

t0 = time.time()
idxs, song_compare = run_comparison(target_signal, given_signal)
t1 = time.time()
plt.plot(idxs, song_compare)
print ("That took %(time).2f minutes to run" % {'time':(t1-t0)/60.0} )
```

1.3 Question 5: GPS Receivers

[2]:
```
%pylab inline
import numpy as np
import matplotlib.pyplot as plt
import scipy.io
import sys
```

```
Populating the interactive namespace from numpy and matplotlib
```

[3]:
```
## RUN THIS FUNCTION BEFORE YOU START THIS PROBLEM
## This function will generate the gold code associated with the satellite ID␣
↪using linear shift registers
## The satellite_ID can be any integer between 1 and 24
def Gold_code_satellite(satellite_ID):
    codelength = 1023
    registerlength = 10

    # Defining the MLS for G1 generator
    register1 = -1*np.ones(registerlength)
    MLS1 = np.zeros(codelength)
    for i in range(codelength):
        MLS1[i] = register1[9]
        modulo = register1[2]*register1[9]
        register1 = np.roll(register1,1)
        register1[0] = modulo

    # Defining the MLS for G2 generator
    register2 = -1*np.ones(registerlength)
    MLS2 = np.zeros(codelength)
    for j in range(codelength):
        MLS2[j] = register2[9]
        modulo =␣
↪register2[1]*register2[2]*register2[5]*register2[7]*register2[8]*register2[9]
        register2 = np.roll(register2,1)
```

```
        register2[0] = modulo

    delay = np.
↪array([5,6,7,8,17,18,139,140,141,251,252,254,255,256,257,258,469,470,471,472,473,474,509,51
    G1_out = MLS1;
    shamt = delay[satellite_ID - 1]
    G2_out = np.roll(MLS2,shamt)

    CA_code = G1_out * G2_out

    return CA_code
```

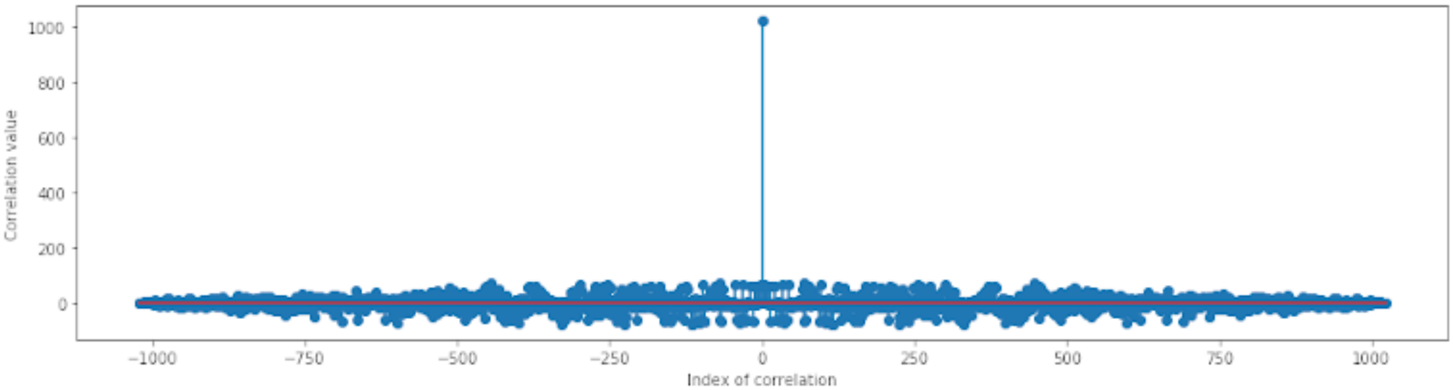
1.3.1 Part (a)

[4]:
```
def cross_correlation(array1, array2):
    """ This function should return two arrays or a matrix with one row␣
↪corresponding to
    the offset and other to the correlation value. array1 and array2 do not␣
↪have to be
    arrays of equal length.
    Think of array1 as the received signal and array2 as the signature.
    The function should return correlation values as well as the indices of the␣
↪nonzero values of the correlation
    Hint: look up np.correlate
    """

    #correlated_array = #Your code here (it is just one line) np.
↪correlate(array1, array2, 'full')
    correlated_array = np.correlate(array1, array2, 'full')

    #Since both the arrays start at 0, the last "shift" where the signals␣
↪overlap is the length of the first array
    end_index = len(array1)

    #Similarly, the first "shift" where the signals overlap is the negative of␣
↪the length of the second array, offset by one because of the zero index.
    st_index = -len(array2) + 1

    indices = np.arange(st_index, end_index)
    return (indices, correlated_array)
```

[5]:
```
# Plot the auto-correlation of satellite 10 with itself. Fill in the function␣
↪call.
array_10 = Gold_code_satellite(10)

(ind_10, self_10) = cross_correlation(array_10, array_10)
```

```
plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

[5]: Text(0, 0.5, 'Correlation value')



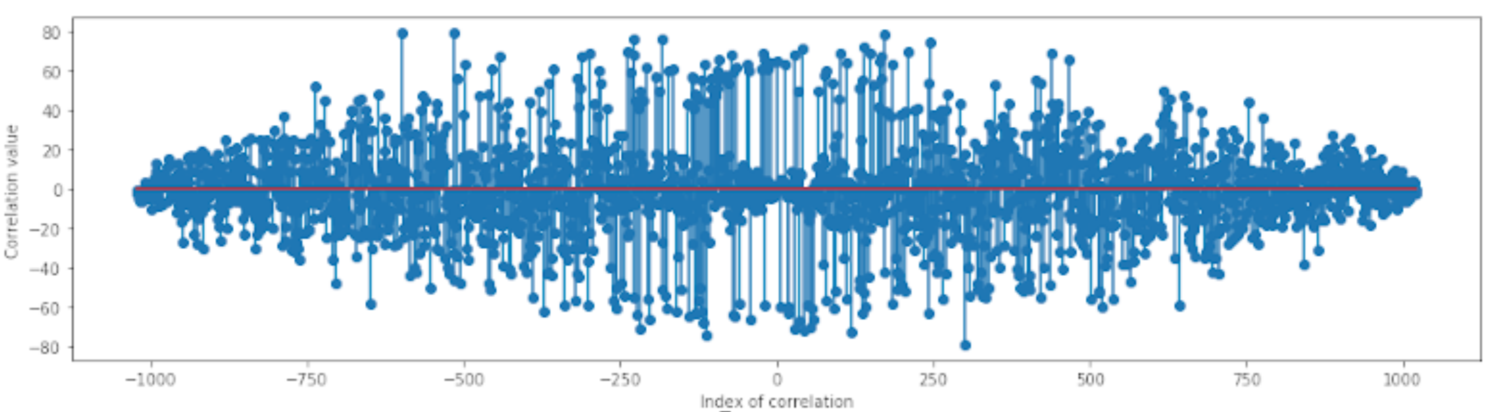
The autocorrelation peaks at 1023 when the signals are perfectly aligned (offset 0). The correlation of a Gold code with a shifted version of itself is not significant.

1.3.2 Part (b)

Plot the cross correlation when array1 = satellite 13 and array2 = satellite10

[6]:
```
#Your code here
array_10 = Gold_code_satellite(10)
array_13 = Gold_code_satellite(13)

(ind_10, self_10) = cross_correlation(array_13, array_10)

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

[6]: Text(0, 0.5, 'Correlation value')

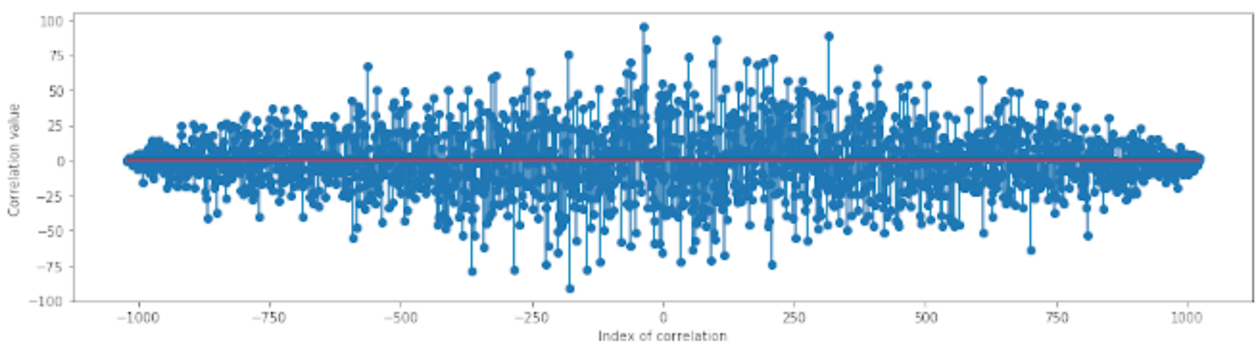


We see that the cross-correlation of a Gold code of any satellite with any other satellite is very low. This indicates that when given some unknown data, we can differentiate between different satellites.

1.3.3 Part (c)

[7]:
```
## THIS IS A HELPER FUNCTION FOR PART C THAT GENERATES +-1 RANDOM NOISE
def integernoise_generator(length_of_noise):
    noise_array = np.random.randint(2, size = length_of_noise)
    noise_array = 2 * noise_array - np.ones(size(noise_array))
    return noise_array

# YOUR CODE HERE

noise = integernoise_generator(1023)

(ind_10, self_10) = cross_correlation(noise, array_10)

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

[7]:
```
Text(0, 0.5, 'Correlation value')
```



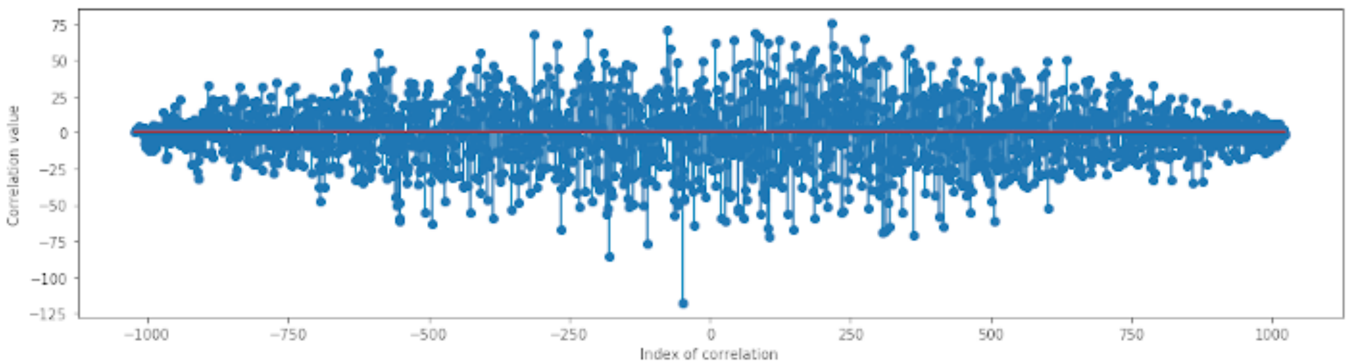
We see that the cross-correlation of the Gold code of any satellite with integer noise is very low. This indicates that we can still figure out the presence of a satellite even if it is buried in noise.

1.3.4 Part (d)

[8]:
```
## THIS IS A HELPER FUNCTION FOR PART D THAT GENERATES REAL VALUED RANDOM NOISE
def gaussiannoise_generator(length_of_noise):
    noise_array = np.random.normal(0, 1, length_of_noise)
    return noise_array
```

```
# YOUR CODE HERE

gauss_noise = gaussiannoise_generator(1023)

(ind_10, self_10) = cross_correlation(gauss_noise, array_10)

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

[8]:
```
Text(0, 0.5, 'Correlation value')
```



We see that the Gold code of any satellite with Gaussian noise is very low. This indicates that we can still figure out the presence of a satellite even if it is buried in Gaussian noise.

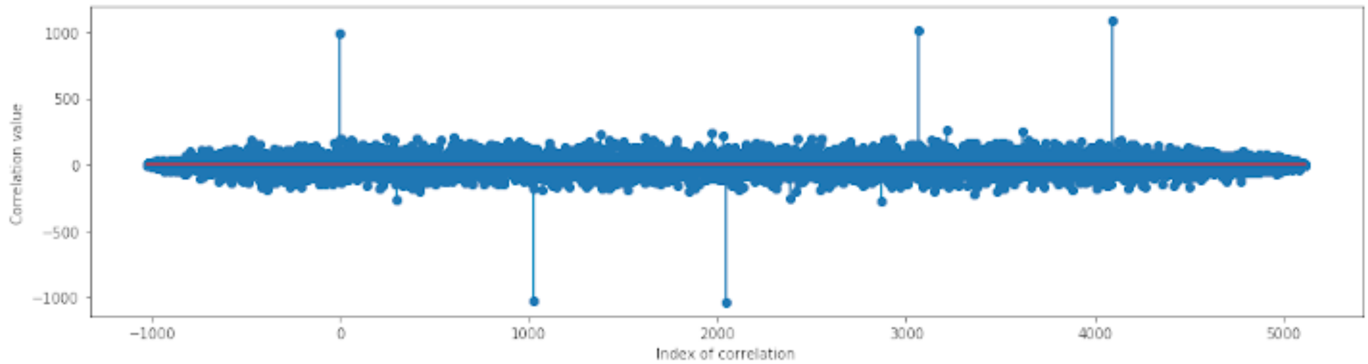
1.3.5 Part (e)

Hint: you can use a absolute value threshold of 800 for the cross-correlation to detect if a given satellite is present. np.argwhere may be useful for detecting peak locations.

[9]:
```
#Now let us see which signals are present in the data signal that is in data1.
↪npy
signal1 = np.load('data1.npy')
```

[10]:
```
#Here try plotting the cross-correlations of data1.npy with a few of the␣
↪satellite gold codes.
#How can you detect if the satellite is present?

(ind_10, self_10) = cross_correlation(signal1, array_13)

plt.figure(figsize=(16, 4))
plt.stem(ind_10, self_10)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

[10]:
```
Text(0, 0.5, 'Correlation value')
```



[11]:
```
## This helper function returns 1 if peak (greater than threshold or less than␣
↪-threshold) is found, else it returns 0.
## You do not have to use this function as there are other solutions to this␣
↪part as well

def find_peak(signal, threshold):
    max_value = np.amax(signal)
    min_value = np.amin(signal)
    if max_value > threshold:
        ret_value = 1
    elif min_value < -1 * threshold:
        ret_value = 1
    else:
        ret_value = 0
    return ret_value
```

[12]:
```
## USE 'np.load' FUNCTION TO LOAD THE DATA
## USE DATA1.NPY AS THE SIGNAL ARRAY
# YOUR CODE HERE
signal1 = np.load('data1.npy')

def find_sat(signal):
    sat_list = [cross_correlation(signal, Gold_code_satellite(i)) for i in␣
↪range(1, 25)]
    peaks = [j + 1 for j in range(len(sat_list)) if find_peak(sat_list[j][1],␣
↪800)]
    return peaks

find_sat(signal1)
```

[12]:
```
[4, 7, 13, 19]
```

1.3.6 Part (f)

[13]:
```
## USE DATA2.NPY AS THE SIGNAL ARRAY

# YOUR CODE HERE --- first write code to figure out which satellite is present
signal2 = np.load('data2.npy')

sats = find_sat(signal2)
print(sats)
```

```
[3]
```

[14]:
```
## Once you have figured out which satellite is present, proceed to find the␣
↪data transmitted

def find_message(signal, sat_id):
    message = []
    mod_signal = cross_correlation(signal, Gold_code_satellite(sat_id))
    for i in mod_signal[1]:
        if i > 800:
            message += [1]
        elif i < -800:
            message += [-1]
    return message

print([find_message(signal2, i) for i in sats])
```

```
[[1, -1, -1, -1, 1]]
```

1.3.7 Part (g)

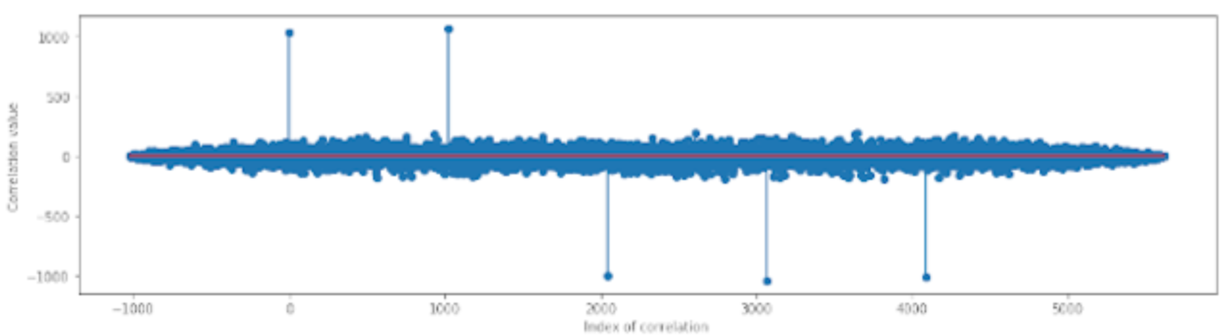
[15]:
```
## USE DATA3.NPY AS THE SIGNAL ARRAY

# YOUR CODE HERE
signal3 = np.load('data3.npy')

s3_sats = find_sat(signal3)
print(s3_sats, [find_message(signal3, i) for i in s3_sats])
```

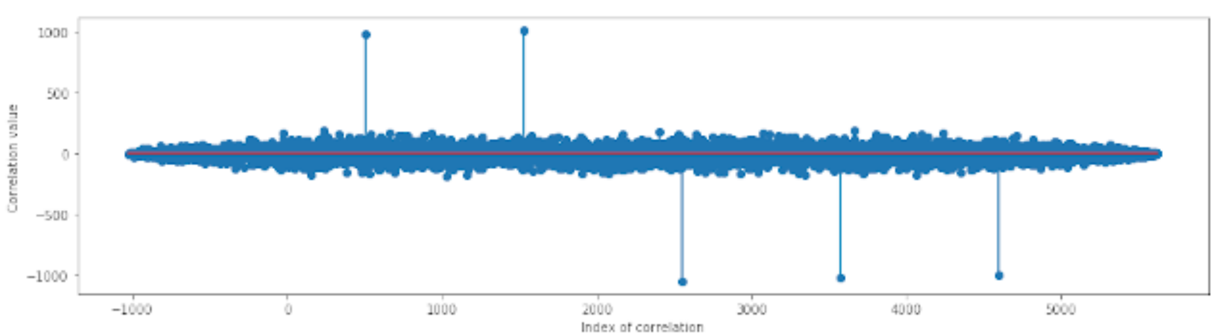
```
[5, 20] [[1, 1, -1, -1, -1], [1, 1, -1, -1, -1]]
```

[16]:
```
## We know that the data is 1, 1, -1, -1, -1, so we just find the positions of␣
↪the first 1 in both the satellite correlations.
## plot the appropriate cross_correlation and find the location of the first 1
## Do this for as many satellites as there are present
## Your code here
```

```
(ind_5, self_5) = cross_correlation(signal3, Gold_code_satellite(s3_sats[0]))

plt.figure(figsize=(16, 4))
plt.stem(ind_5, self_5)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")

(ind_20, self_20) = cross_correlation(signal3, Gold_code_satellite(s3_sats[1]))

plt.figure(figsize=(16, 4))
plt.stem(ind_20, self_20)
plt.xlabel("Index of correlation")
plt.ylabel("Correlation value")
```

[16]: Text(0, 0.5, 'Correlation value')





[17]:
```
def find_offset(signal, sat_id):
    corr = cross_correlation(signal, Gold_code_satellite(sat_id))
    for i in range(len(corr[1])):
        if find_peak(corr[1][i], 800):
            return corr[0][i]

print(find_offset(signal3, 5), find_offset(signal3, 20))
```

0 506