EE16A: Homework 1

(PRACTICE) Problem 2: Finding Charges from Potential Measurements

In [1]:
```
import numpy as np
twoRoot = np.sqrt(2)
fiveRoot = np.sqrt(5)
tenRoot = np.sqrt(10)
oneOverTwoRoot = 1/twoRoot
oneOverFiveRoot = 1/fiveRoot
oneHalf = 1/2
one = 1
u1 = (4 + 3 * fiveRoot + tenRoot) / (2 * fiveRoot)
u2 = (2 + 4 * twoRoot) / twoRoot
u3 = (4 + fiveRoot + 3 * tenRoot) / (2 * fiveRoot)

a = np.array([
[oneOverTwoRoot, oneOverFiveRoot, oneHalf],
[one, oneOverTwoRoot, one],
[oneHalf, oneOverFiveRoot, oneOverTwoRoot]
])
b = np.array([u1, u2, u3])
x = np.linalg.solve(a, b)
x
```

Out[1]:
```
array([ 1.,  2.,  3.])
```

Problem 4: The Framingham Risk Score

In [25]:
```
import numpy as np

p1 = 0.1550
p2 = 0.1108
p3 = 0.0940
p4 = 0.0105

R1 = np.log(np.log(1 - p1) / np.log(0.95)) + 25.66
R2 = np.log(np.log(1 - p2) / np.log(0.95)) + 25.66
R3 = np.log(np.log(1 - p3) / np.log(0.95)) + 25.66
R4 = np.log(np.log(1 - p4) / np.log(0.95)) + 25.66

a = np.array([
[np.log(66), np.log(198), np.log(55), np.log(132)],
[np.log(61), np.log(180), np.log(47), np.log(124)],
[np.log(60), np.log(180), np.log(50), np.log(120)],
[np.log(23), np.log(132), np.log(45), np.log(132)]
])
b = np.array([R1, R2, R3, R4])
x = np.linalg.solve(a, b)
x

# Tip: np.log works element-wise on an np.array
```

Out[25]:
```
array([ 2.30985691,  1.16955491, -0.69451695,  2.82002675])
```

Problem 5: Filtering Out The Troll

In [26]:
```
import numpy as np
import matplotlib.pyplot as plt
import wave as wv
import scipy
from scipy import io
import scipy.io.wavfile
from scipy.io.wavfile import read
from IPython.display import Audio
import warnings
warnings.filterwarnings('ignore')
sound_file_1 = 'm1.wav'
sound_file_2 = 'm2.wav'
```

Let's listen to the recording of the first microphone (it can take some time to load the sound file).

In [27]:
```
Audio(url='m1.wav', autoplay=False)
```

Out[27]:

And this is the recording of the second microphone (it can take some time to load the sound file).

In [28]:
```
Audio(url='m2.wav', autoplay=False)
```

Out[28]:

We read the first recording to the variable `corrupt1` and the second recording to `corrupt2`.

In [29]:
```
rate1,corrupt1 = scipy.io.wavfile.read('m1.wav')
rate2,corrupt2 = scipy.io.wavfile.read('m2.wav')
```

Enter the gains of the two recordings to get the clean speech.

Note: The square root of a number a can be written as `np.sqrt(a)` in IPython.

In [30]:
```
# enter the gains u (recording 1) and v (recording 2)
u = 2 / (np.sqrt(2) + np.sqrt(6))
v = (2 * np.sqrt(3)) / (np.sqrt(2) + np.sqrt(6))
```

Weighted combination of the two recordings:

In [31]:
```
s1 = u*corrupt1 + v*corrupt2
```

Let's listen to the resulting sound file (make sure your speaker's volume is not very high, the sound may be loud if things go wrong).

In [32]:
```
Audio(data=s1, rate=rate1)
```

Out[32]: