1-2

assert human_binary.shape[0] == num_sequences, 'Error: the number of sequences was entered incorrectly'

Check whether the number of rows in x_list is cot=rrect or not.

assert human_binary.shape[0] == num_sequences, 'Error: the number of sequences was entered incorrectly'

Check whether the number of columns in x_list is cot=rrect or not.

1-3

```
import h5py

import numpy as np

import matplotlib.pyplot as plt


DEBUG = False

DATA_FNAME = 'ZhengWen_hw1_1.hd5'


if DEBUG:

    num_sequences = 3

    sequence_length = 4

else:

    num_sequences = 25

    sequence_length = 20


### Enter your data here...

### Be sure to generate the data by hand:

###     copy-n-paste

###     use a random number generator

###

x_list = [

    [ 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1],

    [ 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0],

    [ 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0],
```

```
    [ 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0],
    [ 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0],
    [ 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0],
    [ 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0],
    [ 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1],
    [ 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
    [ 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1],
    [ 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1],
    [ 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0],
    [ 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0],
    [ 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0],
    [ 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
    [ 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0],
    [ 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1],
    [ 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0],
    [ 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1],
    [ 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0],
    [ 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0],
    [ 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1],
    [ 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1],
    [ 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0],
    [ 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1],
]

# convert list to a numpy array...
human_binary = np.asarray(x_list)


### do some error trapping:
```

```
assert human_binary.shape[0] == num_sequences, 'Error: the number of sequences was entered incorrectly'

assert human_binary.shape[1] == sequence_length, 'Error: the length of the seqeunces is incorrect'


# the with statement opens the file, does the business, and close it up for us...

with h5py.File(DATA_FNAME, 'w') as hf:

    hf.create_dataset('human_binary', data = human_binary)

    ## note you can write several data arrays into one hd5 file, just give each a different name.


###################

# Let's read it back from the file and then check to make sure it is as we wrote...

with h5py.File(DATA_FNAME, 'r') as hf:

    hb_copy = hf['human_binary'][:]


### this will throw and error if they are not the same...

np.testing.assert_array_equal(human_binary, hb_copy)
```

2

```
import numpy as np


def ReLu(array):

    [rows, cols] = array.shape


    for i in range(rows):

        for j in range(cols):

            if array[i, j] <= 0:

                array[i, j] = 0

    return array


x = [[1], [-1]]
```

```
W1 = [[1, -2], [3, 4]]
W2 = [[2, 2], [3, -3]]
b1 = [[1], [0]]
b2 = [[0], [-4]]

x = np.array(x)
W1 = np.array(W1)
W2 = np.array(W2)
b1 = np.array(b1)
b2 = np.array(b2)

l1 = ReLu(np.dot(W1, x) + b1)
y = ReLu(np.dot(W2, l1) + b2)

print(y)
[[8]
 [8]]
```

3-1

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k]$$

$$y[n] = x[n] \star h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[k+n]$$

So,

$$x[n] \star h[-n] = \sum_{k=-\infty}^{+\infty} x[k]h[-k+n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = x[n] * h[n]$$

In two-dimensional space,

$$y[i][j] = x[i][j] * h[i][j] = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[m][n]h[i-m][j-n]$$

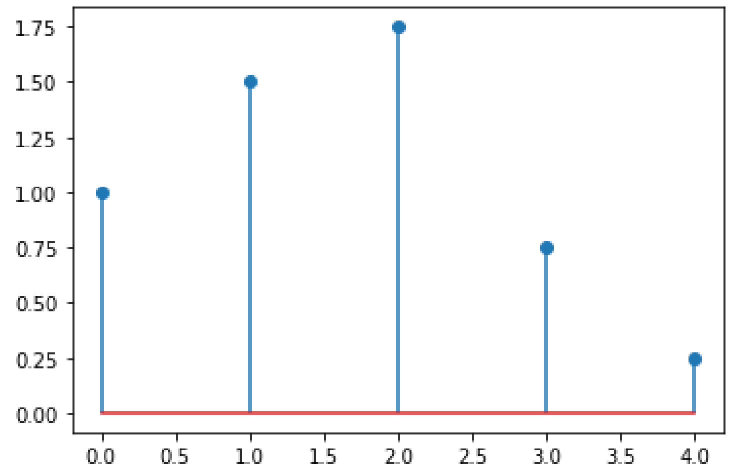
$$y[i][j] = x[i][j] \star h[i][j] = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[m][n]h[m+i][n+j]$$

So,

$$x[i][j] \star h[-i][-j] = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[m][n]h[-m+i][-n+j] = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} x[m][n]h[i-m][j-n]$$
$$= x[i][j] * h[i][j]$$

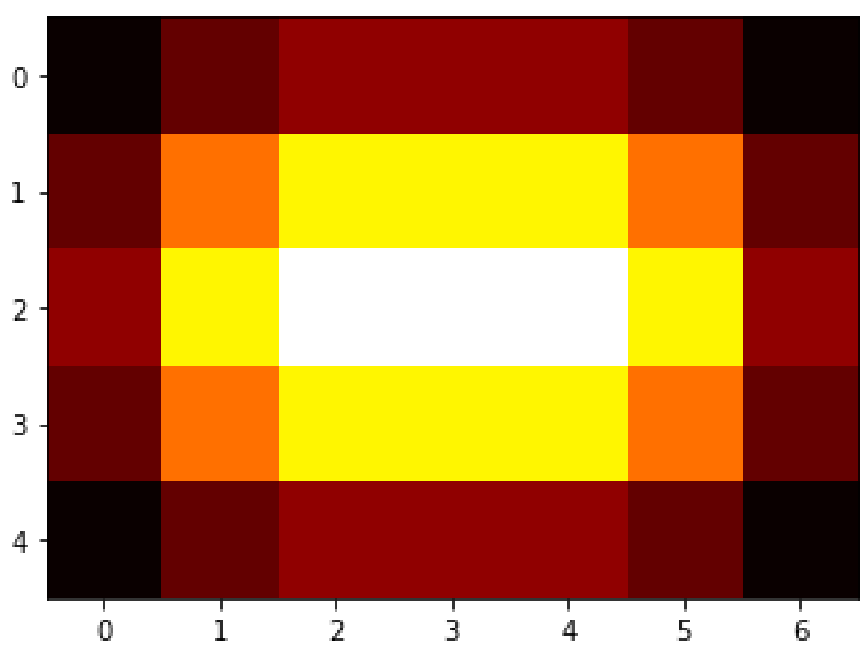
3-2

```
import numpy as np

from scipy import signal

import matplotlib.pyplot as plt

%matplotlib inline


x = [1, 1, 1]

h = [1, 0.5, 0.25]


X_1 = np.array(x)

H_1 = np.array(h)

y_conv = np.convolve(X_1, H_1)


print('x*y = ', y_conv)


plt.stem(y_conv, use_line_collection=True)


x*y =  [1.   1.5  1.75 0.75 0.25]
```



3-3

```
x = np.array([[1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1]])
y = np.array([[.25, .5, .25],
        [0.5, 1, 0.5],
        [.25, .5, .25]])
res = signal.convolve2d(x, y)
plt.imshow(res, cmap='hot')
```



4-1-(a)

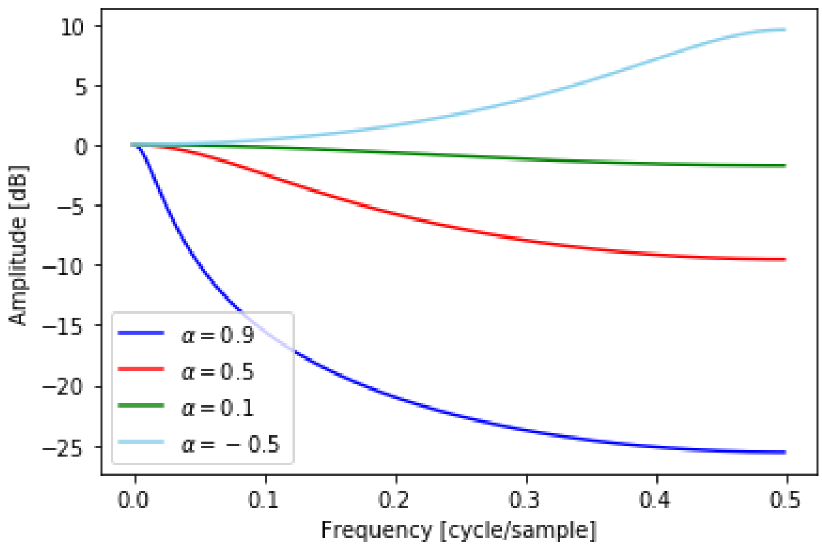
Because pole of H(z) is α, so a[1] = α

When ω=0, the gain of H(z) is 1, so b[0] = 1- α

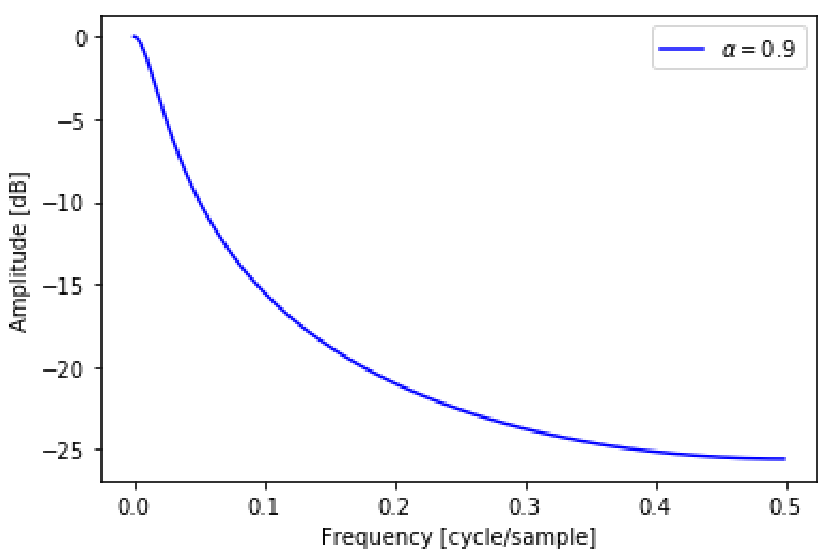
So $H(z) = \frac{1-\alpha}{1-\alpha z^{-1}}$

4-1-(b)

```
import matplotlib.pyplot as plt

import numpy as np

from scipy import signal

%matplotlib inline



w1, h1 = signal.freqz([0.1, 0], [1, -0.9])

fig = plt.figure()

plt.plot(w1/(2*np.pi), 20*np.log10(abs(h1)), color='b', label=r"$\alpha=0.9$")


w2, h2 = signal.freqz([0.5, 0], [1, -0.5])

plt.plot(w2/(2*np.pi), 20*np.log10(abs(h2)), color='r',label=r"$\alpha=0.5$")


w3, h3 = signal.freqz([0.9, 0], [1, -0.1])

plt.plot(w3/(2*np.pi), 20*np.log10(abs(h3)), color='g',label=r"$\alpha=0.1$")


w4, h4 = signal.freqz([1.5, 0], [1, 0.5])

plt.plot(w4/(2*np.pi), 20*np.log10(abs(h4)), color='skyblue',label=r"$\alpha=-0.5$")


plt.ylabel('Amplitude [dB]')

plt.xlabel('Frequency [cycle/sample]')

plt.legend()
```



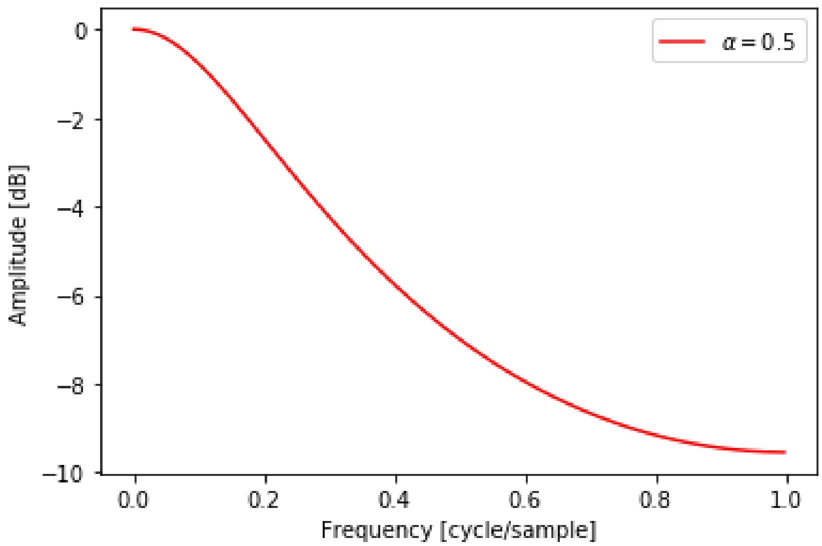

```
w1, h1 = signal.freqz([0.1, 0], [1, -0.9])

fig = plt.figure()

plt.plot(w1/(2*np.pi), 20*np.log10(abs(h1)), color='b', label=r"$\alpha=0.9$")

plt.ylabel('Amplitude [dB]')

plt.xlabel('Frequency [cycle/sample]')

plt.legend()
```




```
w2, h2 = signal.freqz([0.5, 0], [1, -0.5])

fig = plt.figure()

plt.plot(w2/(np.pi), 20*np.log10(abs(h2)), color='r',label=r"$\alpha=0.5$")

plt.ylabel('Amplitude [dB]')
```

plt.xlabel('Frequency [cycle/sample]')

plt.legend()



w3, h3 = signal.freqz([0.9, 0], [1, -0.1])

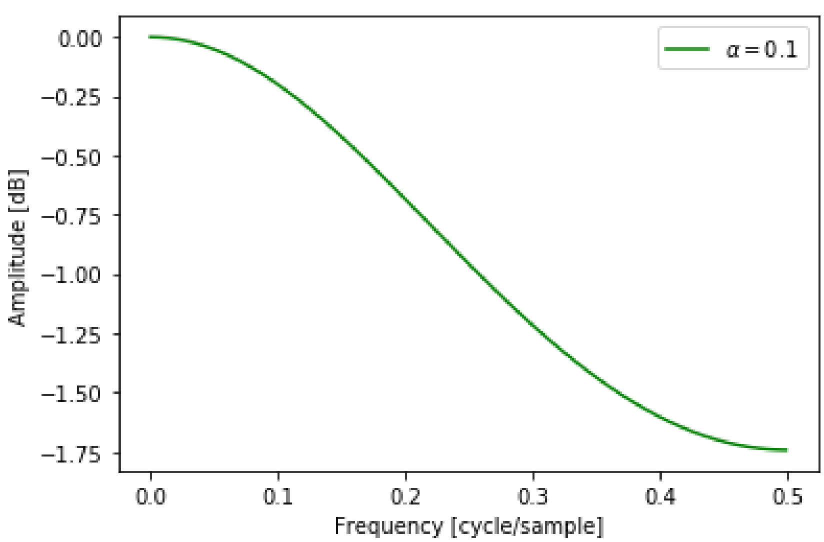
fig = plt.figure()

plt.plot(w3/(2*np.pi), 20*np.log10(abs(h3)), color='g',label=r"$\alpha=0.1$")

plt.ylabel('Amplitude [dB]')

plt.xlabel('Frequency [cycle/sample]')

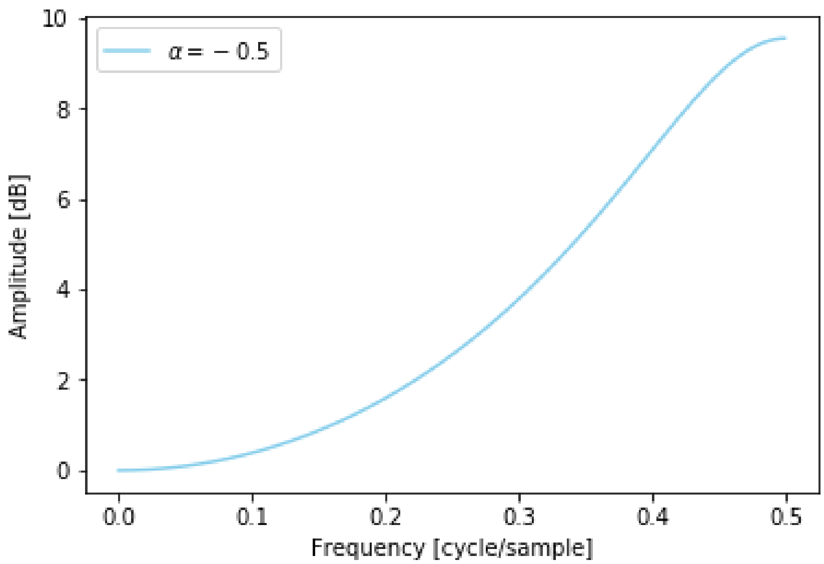
plt.legend()



w4, h4 = signal.freqz([1.5, 0], [1, 0.5])

fig = plt.figure()

```
plt.plot(w4/(2*np.pi), 20*np.log10(abs(h4)), color='skyblue',label=r"$\alpha=-0.5$")
plt.ylabel('Amplitude [dB]')
plt.xlabel('Frequency [cycle/sample]')
plt.legend()
```

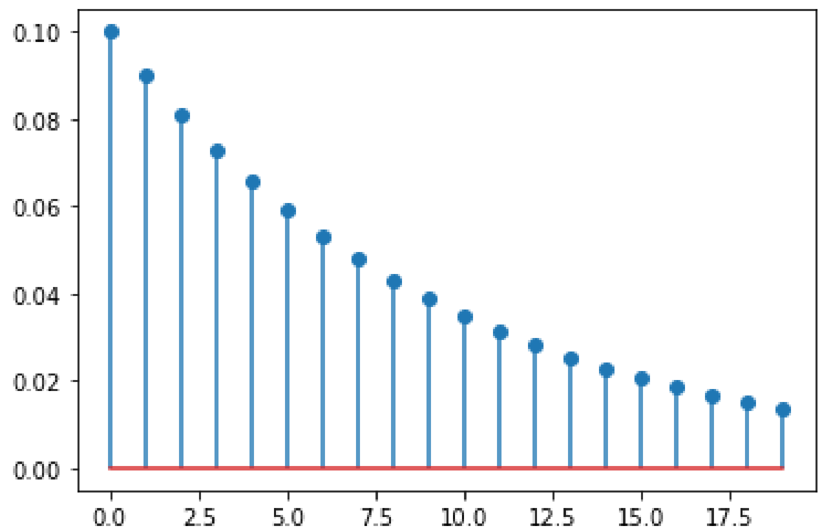


(c)

The reverse transform of H(z) is $x[n] = (1-\alpha) * \alpha^n u[n]$

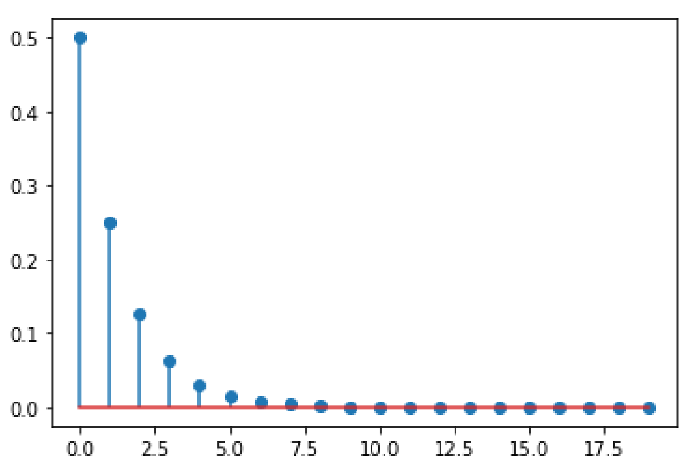
```
x1 = np.array([n for n in range(20)])
y1 = []
for each in x1:
    y1.append((1-0.9)*0.9**each)
y1 = np.array(y1)
plt.stem(y1,use_line_collection=True)
for n in range(len(y1)):
    if y1[n] <= y1[0] * 0.2:
        print("Time constant is", n, "when alpha = 0.9")
        break
```

```
Time constant is 16 when alpha = 0.9
```

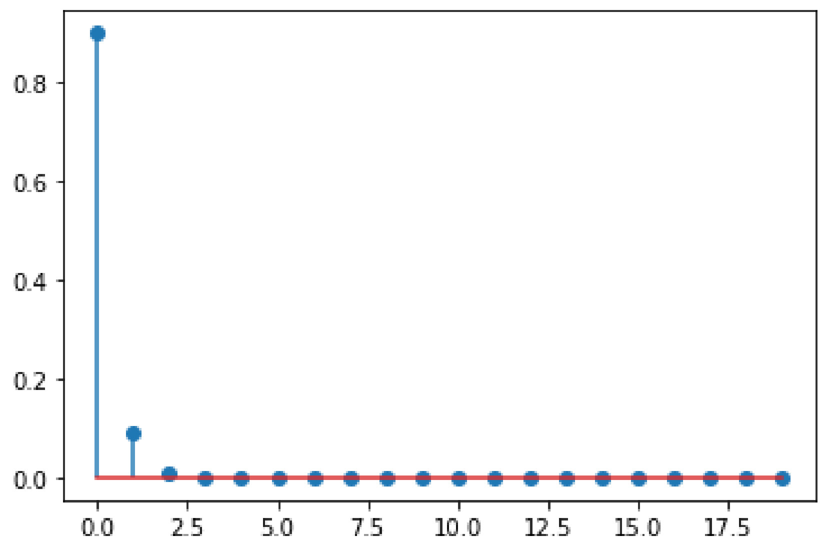



```
x2 = np.array([n for n in range(20)])
y2 = []
for each in x2:
  y2.append((1-0.5)*0.5**each)
y2 = np.array(y2)
plt.stem(y2,use_line_collection=True)
for n in range(len(y2)):
  if y2[n] <= y2[0] * 0.2:
    print("Time constant is", n, "when alpha = 0.5")
    break
```

`Time constant is 3 when alpha = 0.5`



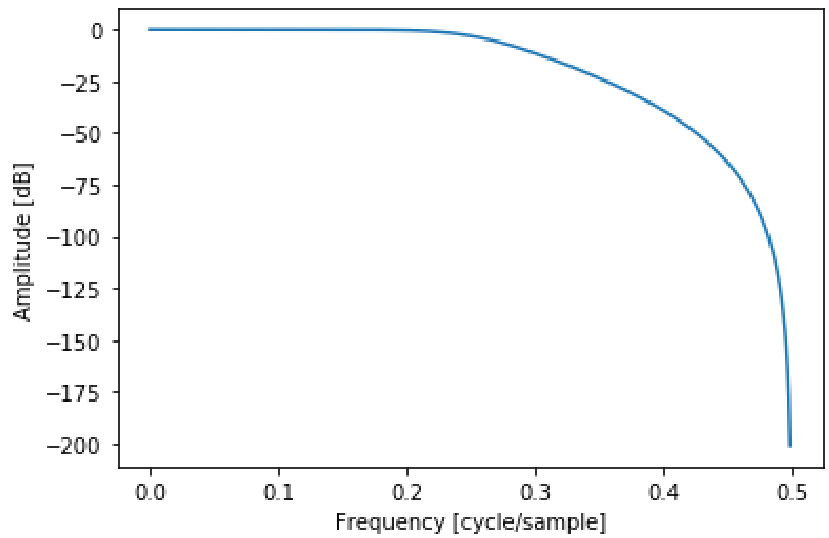
```
x3 = np.array([n for n in range(20)])
y3 = []
for each in x3:
    y3.append((1-0.1)*0.1**each)
y3 = np.array(y3)
plt.stem(y3,use_line_collection=True)
for n in range(len(y3)):
    if y3[n] <= y3[0] * 0.2:
        print("Time constant is", n, "when alpha = 0.1")
        break
```




```
Time constant is 1 when alpha = 0.1
```

4-2-(a)

```
b, a = signal.butter(4, 0.5, btype='lowpass', analog=False, output='ba')
w5, h5 = signal.freqz(b,a)
fig = plt.figure()
plt.plot(w5/(2*np.pi), 20*np.log10(abs(h5)))
plt.ylabel('Amplitude [dB]')
plt.xlabel('Frequency [cycle/sample]')
```




```
x = np.random.randn(300)

y = signal.lfilter(b, a, x)

fig = plt.figure()

plt.plot(np.array([n for n in range(300)]), x, label="Random Signal",color='b')

plt.plot(np.array([n for n in range(300)]), y, label="Filtered Signal",color='r')

plt.legend()
```