Group FARARA

11.1

In [1]:

```
import matplotlib.gridspec as gridspec
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage
from scipy.stats import cumfreq
from sklearn.model_selection import train_test_split

%matplotlib inline
```

In [2]:

```
image = ndimage.imread('testimg.jpg').flatten()
image = image / np.max(image)
```

In [3]:

```
image0 = image
image1 = image + np.random.normal(0, 0.05, image.shape)
image2 = image + np.random.normal(0, 0.1, image.shape)
```

In [ ]:

```
gs1 = gridspec.GridSpec(1, 3)
fig = plt.figure(figsize=(16, 6))

plt.subplot(gs1[0])
plt.hist(image0, bins=20, label="image 0")
plt.legend()


plt.subplot(gs1[1])
plt.hist(image1, bins=20, label="image 1")
plt.legend()



plt.subplot(gs1[2])
plt.hist(image2, bins=20, label="image 2")
plt.legend()
fig.tight_layout()
```

In [ ]:

```
plt.figure(figsize=(15, 8))
for key, i in enumerate([image0, image1, image2]):
    res = cumfreq(i, 100)
    x = res.lowerlimit + np.linspace(0, res.binsize*res.cumcount.si
ze, res.cumcount.size)
    cc = res.cumcount / np.max(res.cumcount)
    plt.plot(x, cc, label="image {}".format(key))
plt.legend()
plt.show()
```

In [4]:

```
def density_estimate(train, test, h):
    p = train.shape[0]

    train_matrix = np.repeat([train],len(test),axis=0).T
    #print(train_matrix)
    estimates = ((np.abs((train_matrix - test).T/h)) < 0.5).sum(axi
s=1)

    return (1/h) * (1/p) * estimates

def neg_log_like(train, test, h):
    # Adding very small value, to avoid calculating log of 0
    estimates = -np.log(density_estimate(train, test, h)+0.00000001
)
    return np.mean(estimates)
```

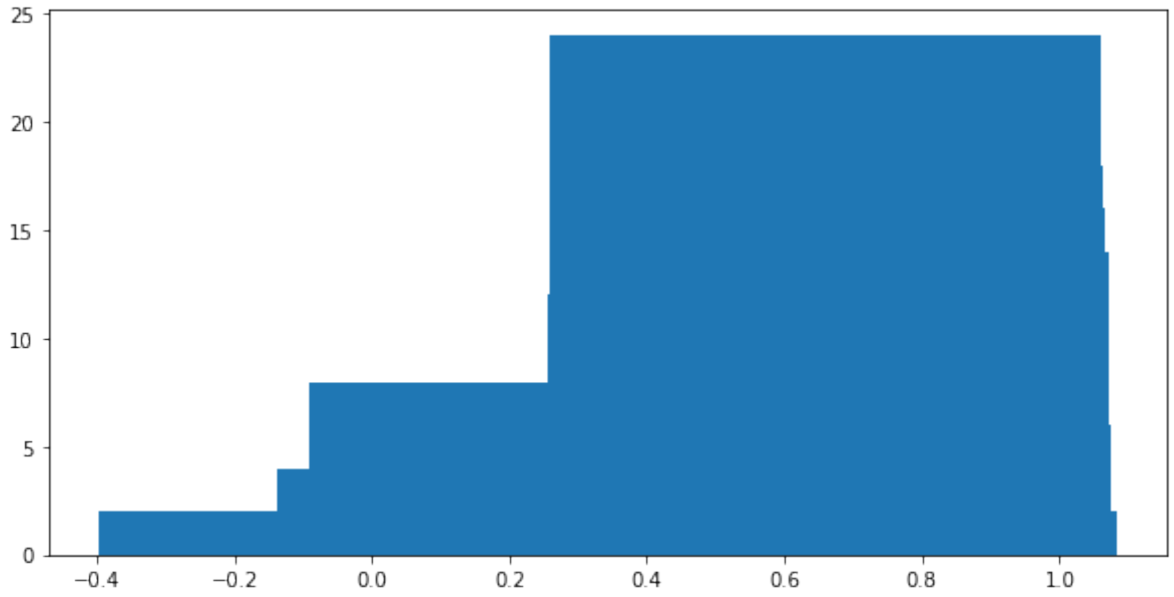
In [24]:

```
def mean_neg_log(data, h=0.005, N=10, P=5000, plot=False):
    nlls = 0
    for _ in range(N):
        train, test = train_test_split(image0, train_size=100)
        estimates = density_estimate(train, train, h)
        nll = neg_log_like(train, test[:P], h)
        nlls += nll

        if plot:
            plt.figure(figsize=(10,5))
            plt.bar(train, estimates)
            print(nll)
            plt.show()

    return nlls/N
```

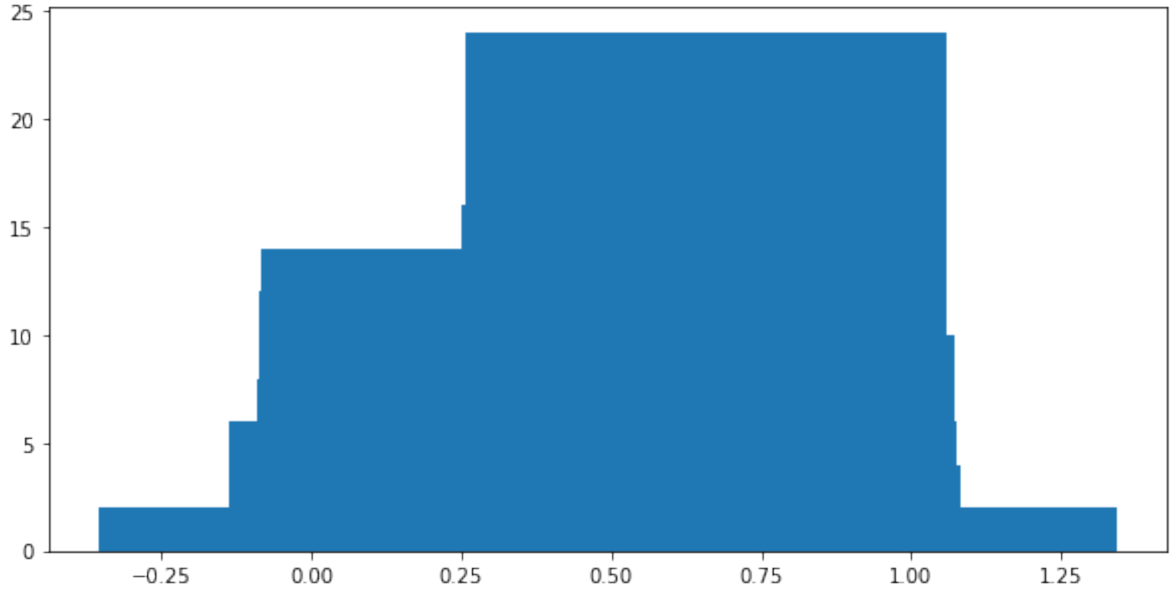
In [9]:

```
mean_neg_log(image0, plot=True)
```

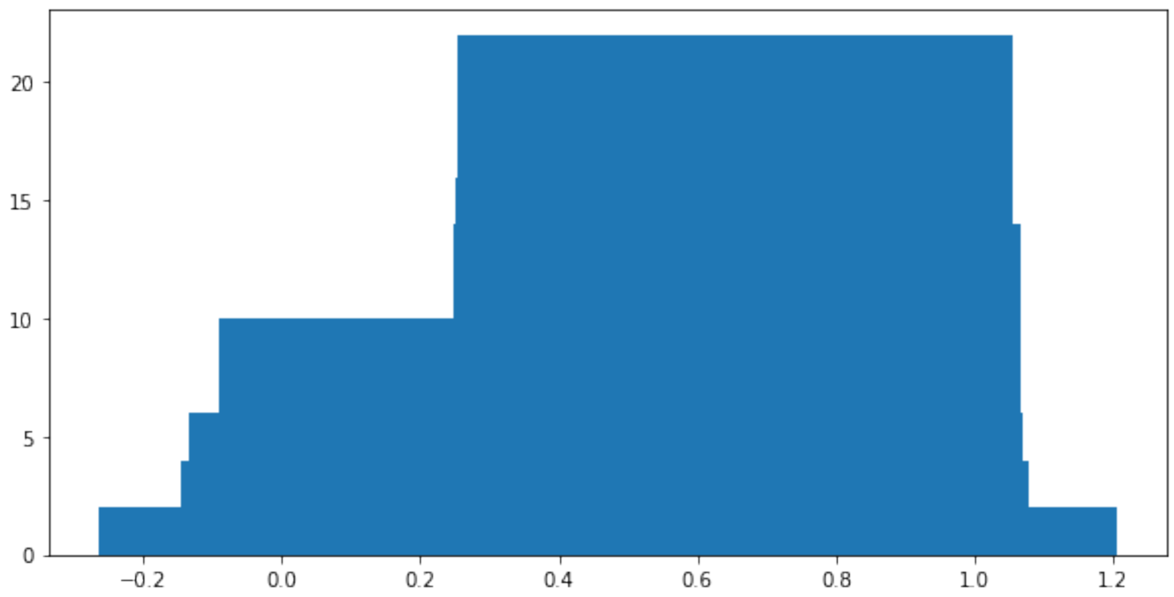
```
2.97878477984
```



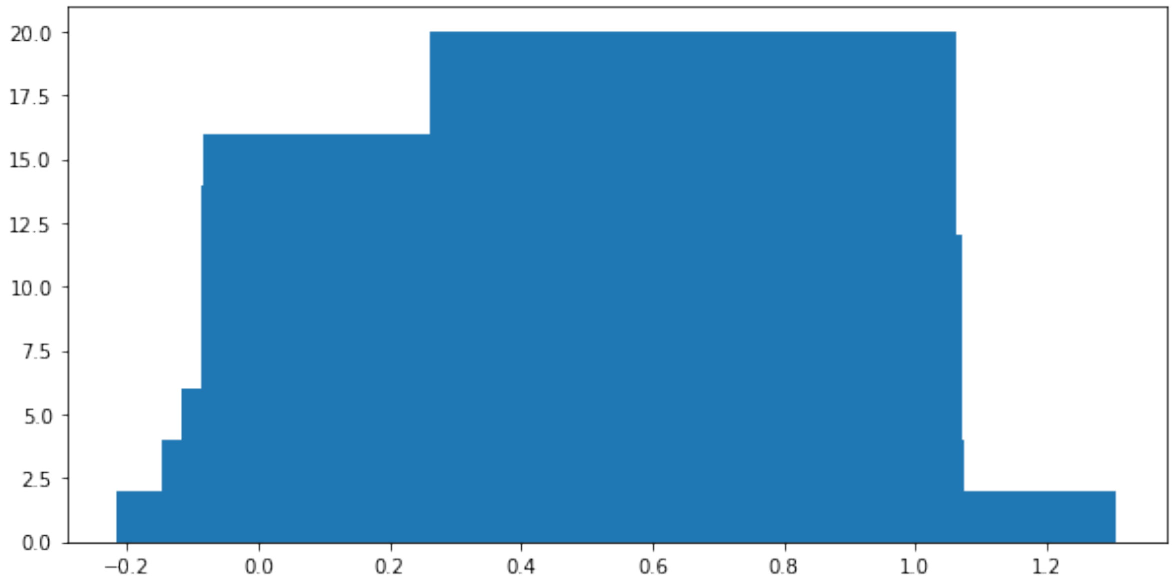
```
3.10376689561
```



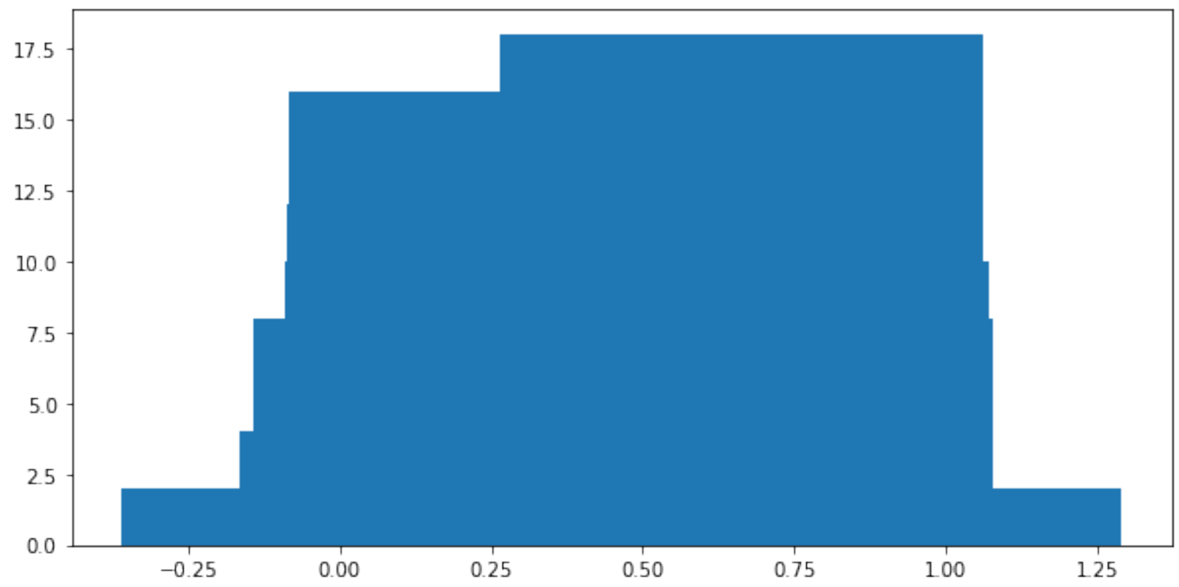
```
3.1856720421
```



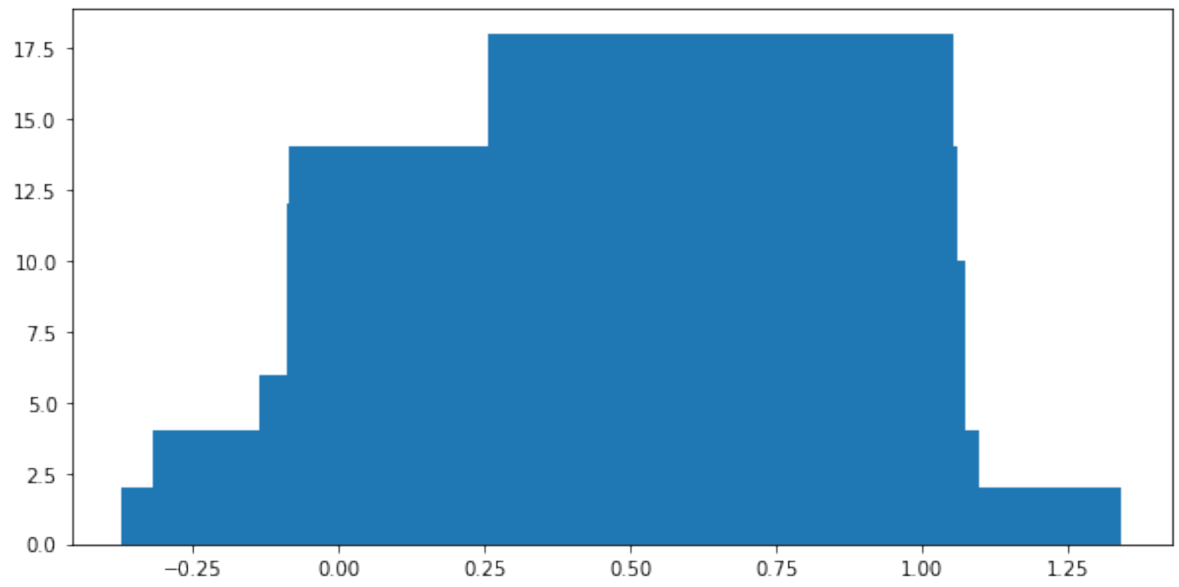
```
3.35407039479
```



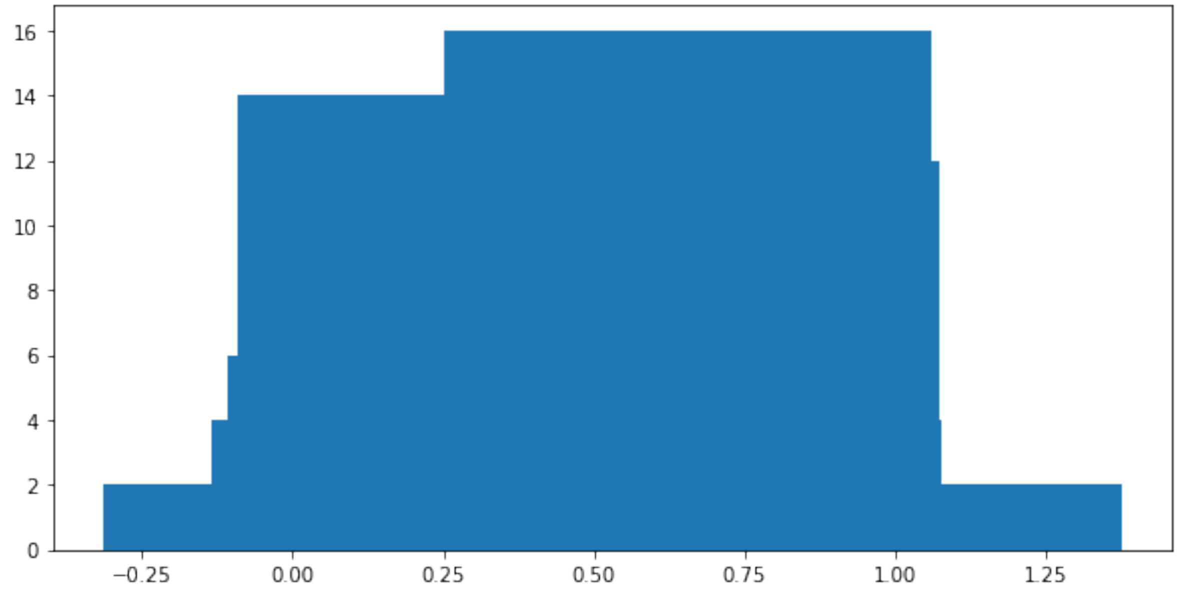
```
2.9789316155
```



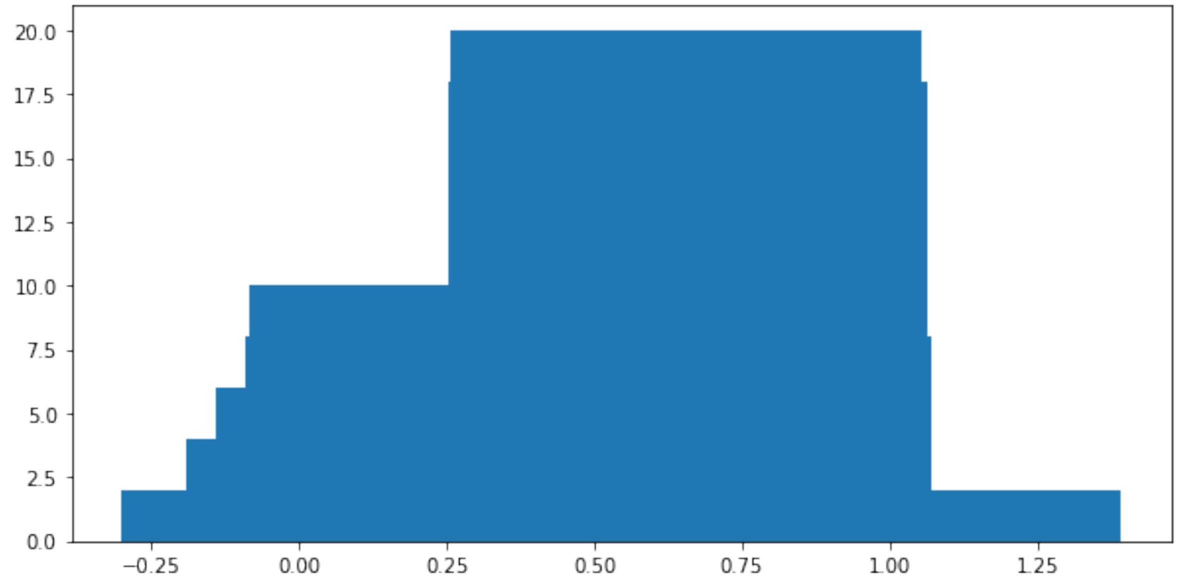
```
3.06244640592
```



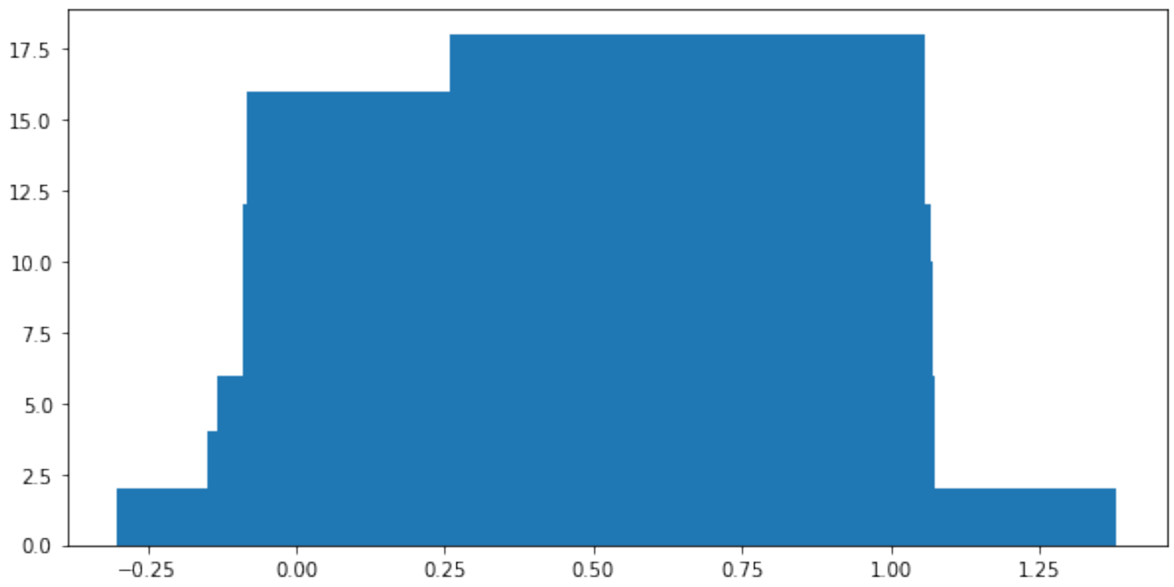
```
3.53406210097
```



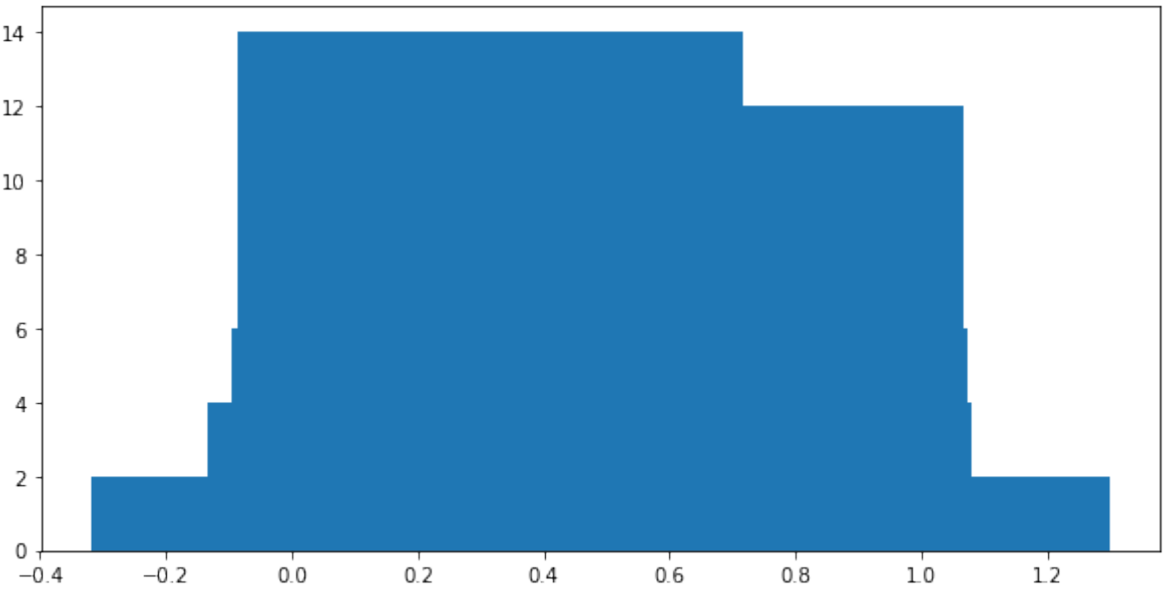
```
4.1291739196
```



```
2.90222828828
```



```
2.66737314794
```



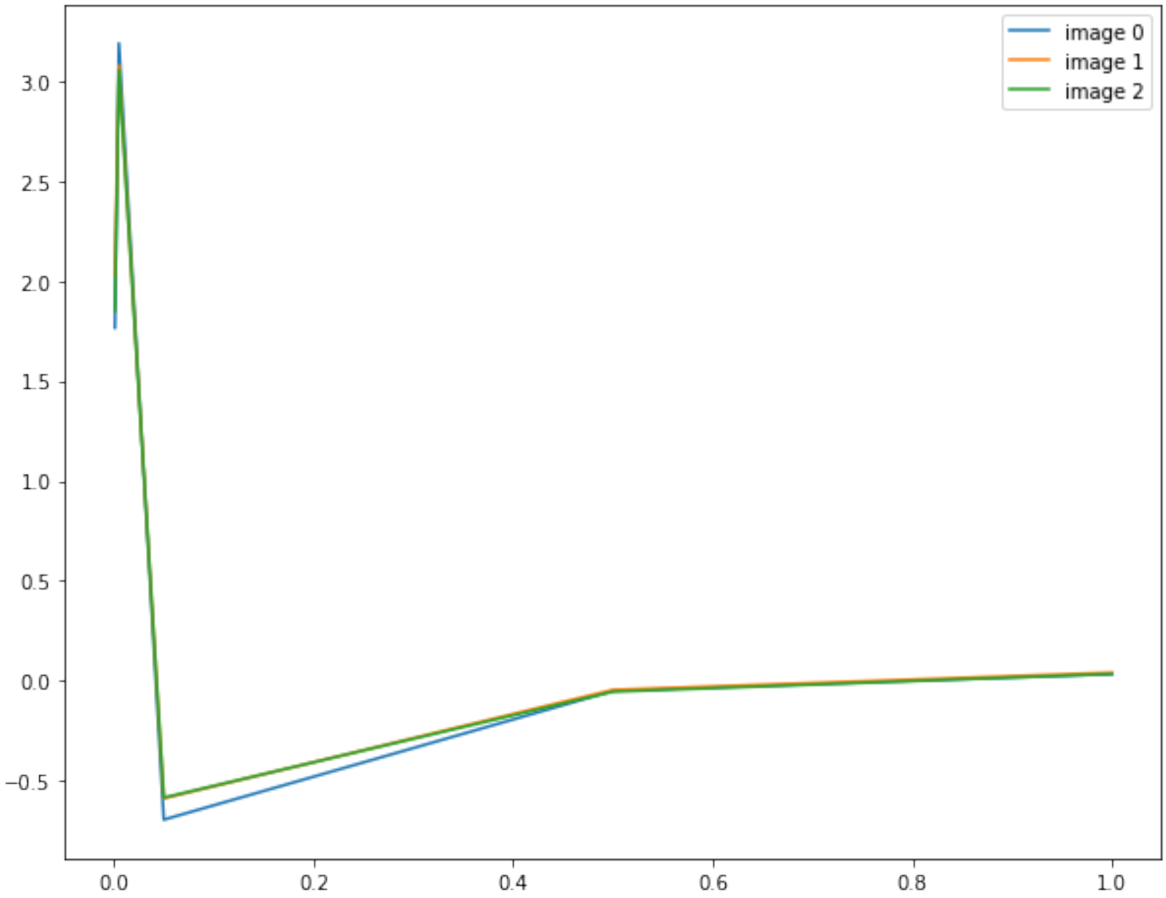
```
Out[9]: 3.1896509590543003
```

```
In [11]: for h in [0.001, 0.005, 0.05, 0.5]:
             print(mean_neg_log(image0, h=h, plot=False))
```

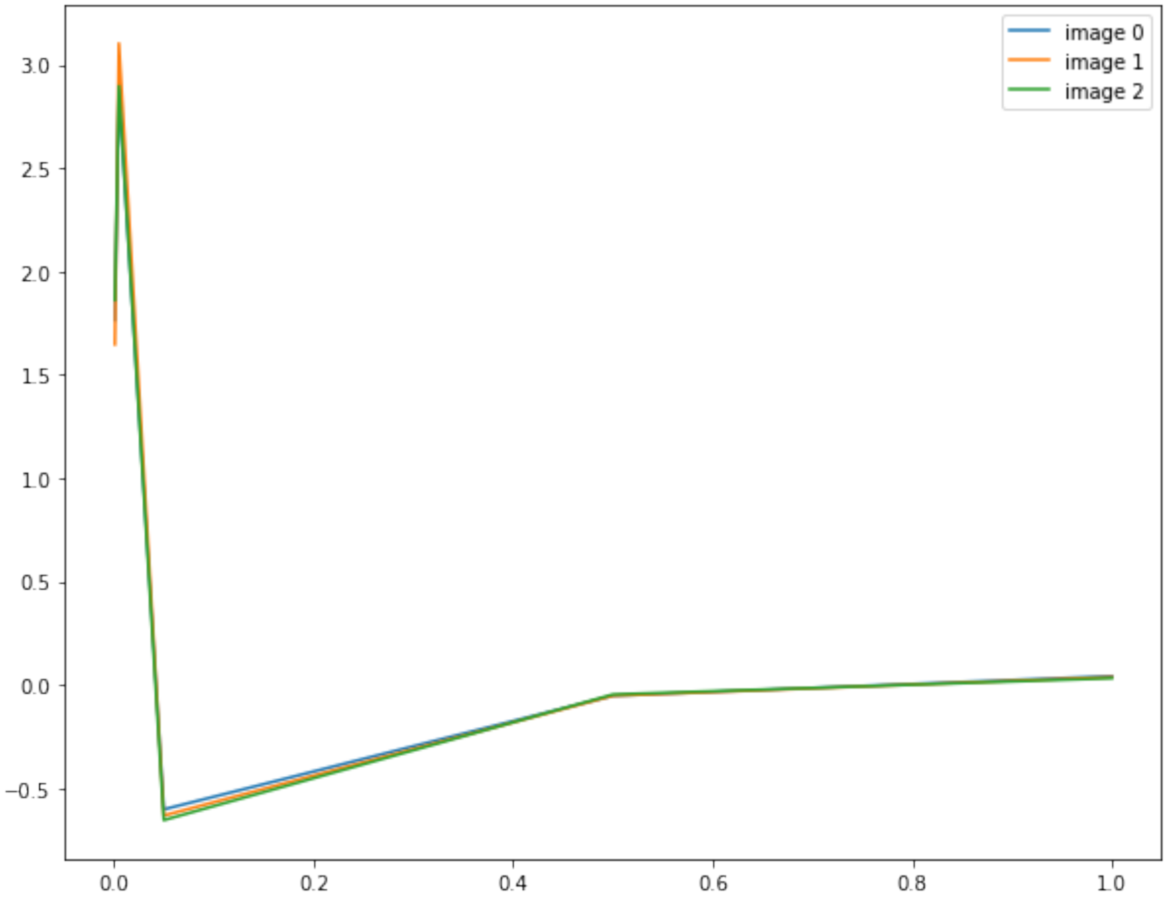
```
1.76817645765
2.82058157461
-0.621074053631
-0.0494772318339
```

In [27]:

```
hs = [0.001, 0.005, 0.05, 0.5]
results = np.zeros((3, len(hs)))
for data_i, data in enumerate([image0, image1, image2]):
    for h_i, h in enumerate(hs):
        results[data_i, h_i] = mean_neg_log(image0, h=h, plot=False
)

plt.figure(figsize=(10, 8))
for i in range(3):
    plt.plot(hs, results[i], label="image {}".format(i))
plt.legend()
plt.show()
```



In [28]:
```
results = np.zeros((3, len(hs)))
for data_i, data in enumerate([image0, image1, image2]):
    for h_i, h in enumerate(hs):
        results[data_i, h_i] = mean_neg_log(image0, h=h, P=500, plo
t=False)

plt.figure(figsize=(10, 8))
for i in range(3):
    plt.plot(hs, results[i], label="image {}".format(i))
plt.legend()
plt.show()
```

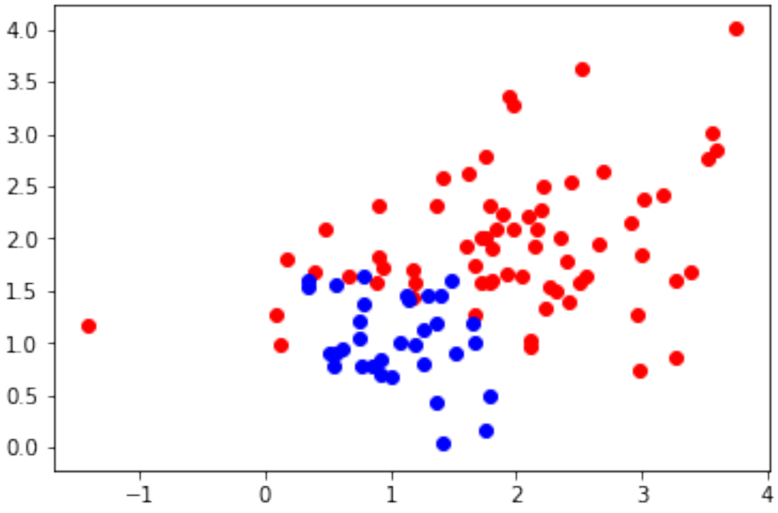


11.2

In [18]:
```
class1 = np.random.multivariate_normal([2,2], [[0.7, 0],[0, 0.7]],
67)
class2 = np.random.multivariate_normal([1,1], [[0.2, 0],[0, 0.2]],
33)
data = np.vstack([class1, class2])
```

In [19]:
```
m = np.hstack([np.zeros(67), np.ones(33)])
```

```
In [20]: #plt.scatter()
         plt.scatter(class1[:,0], class1[:,1], color='red')
         plt.scatter(class2[:,0], class2[:,1], color='blue')

Out[20]: <matplotlib.collections.PathCollection at 0x1151b83c8>
```



```
In [ ]:
```

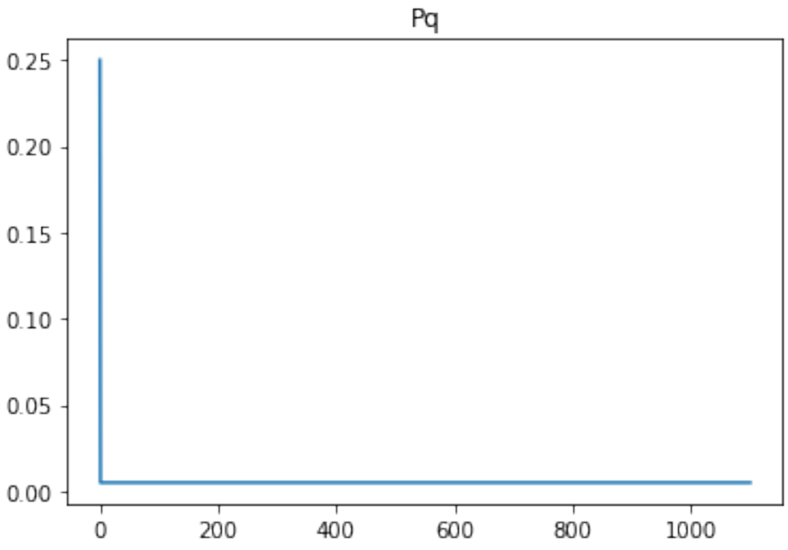
```
In [83]: def do_EM(data, M, tol = 0.001, use_sigma = False, sigma = 0.1):

             #initilization
             mu  = np.mean(data)
             nu  = np.random.rand()/100
             eta = np.random.rand()/100

             Pq  = []
             Pq.append(1/M)


             ws = []
             ws.append(mu+nu)

             sigmas = []
             if(use_sigma):
                 sigmas.append(sigma)
             else:
                 sigmas.append(np.mean(np.square(data-mu))+eta)

             P_qx = []

             counter = 0
             while(True):

                 # E STEP
                 P_xq = np.random.normal((np.abs(data-ws[counter])),sigmas[c
         ounter])
                 P_qx.append((P_xq*Pq[counter])/((P_xq*Pq[counter]).sum()))
```

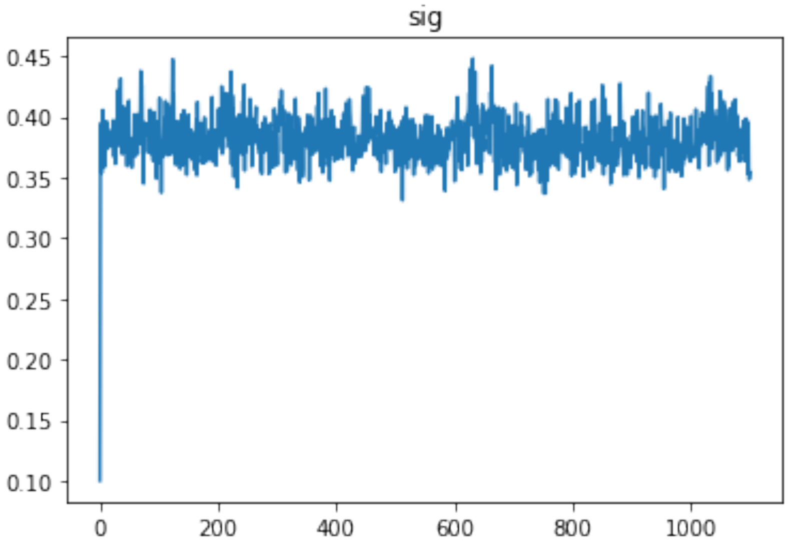
```
        # M STEP
        # calculate w_new
        ws.append(((P_qx[counter]*data).sum())/(P_qx[counter].sum()
))

        #calculate sigma_new
        sigmas.append( (1/M) * (np.square(data-ws[counter])*P_qx[co
unter]).sum() )

        Pq.append(np.mean(P_qx[counter]))

        counter += 1
        # check if things changed a lot
        if (counter > 2):
            if (np.linalg.norm(ws[counter-1]-ws[counter-2])<tol):
                if (np.abs(sigmas[counter-1]-sigmas[counter-2])<tol
):
                    if (np.abs(Pq[counter-1]-Pq[counter-2])<tol):
                        print (counter)
                        plt.plot(np.arange(len(Pq)),Pq)
                        plt.title('Pq')
                        plt.show()

                        plt.plot(np.arange(len(sigmas)),sigmas)
                        plt.title('sig')
                        plt.show()

                        plt.plot(np.arange(len(ws)),ws)
                        plt.title('ws')
                        plt.show()
                        break
```

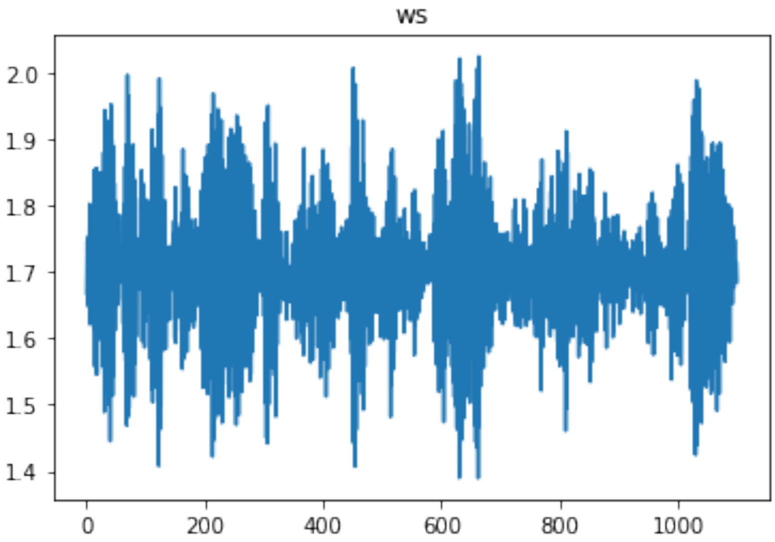
In [84]:

```
for s in [0.1,0.5,1,1.5]:
    do_EM(data, 4, use_sigma = True, sigma = s)
```

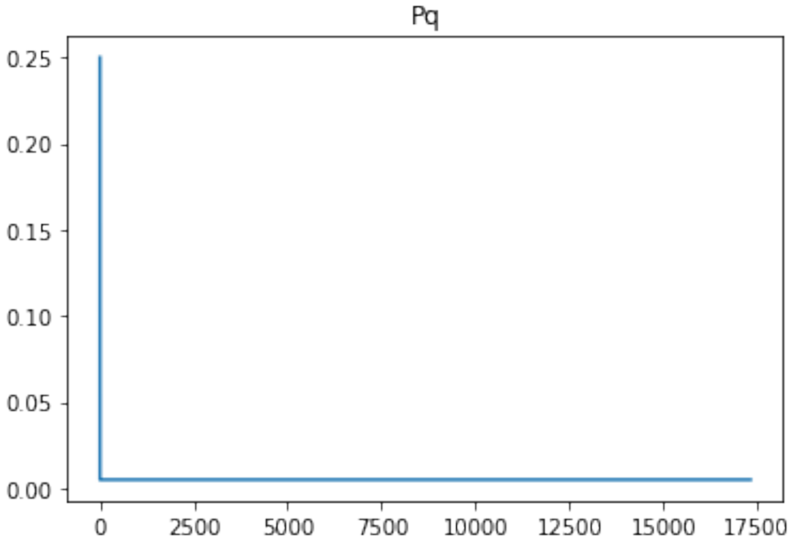
```
1100
```

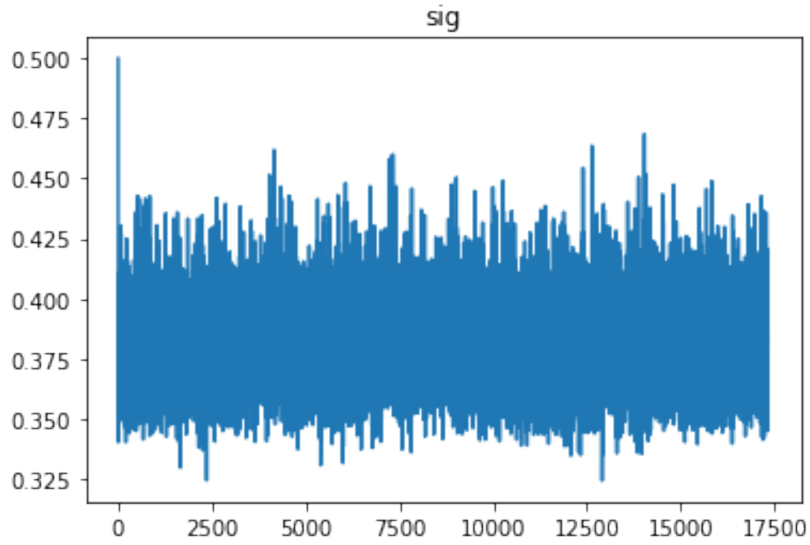


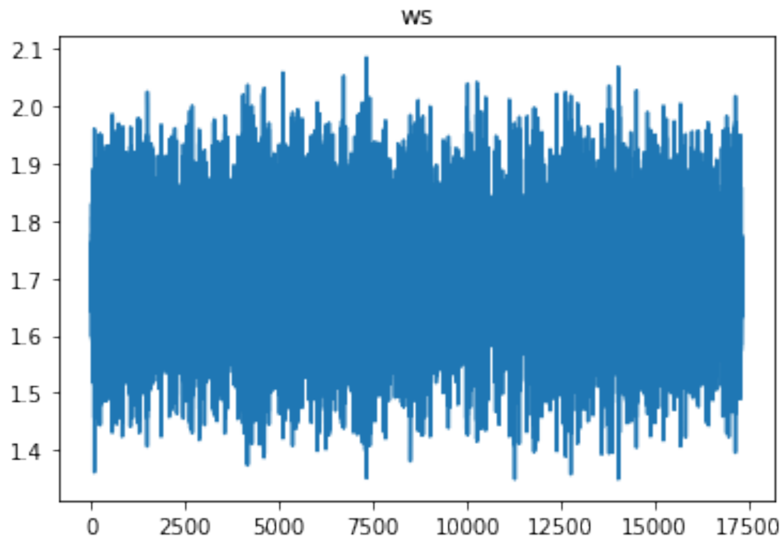

sig
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
200
400
600
800
1000


ws
2.0
1.9
1.8
1.7
1.6
1.5
1.4
0
200
400
600
800
1000

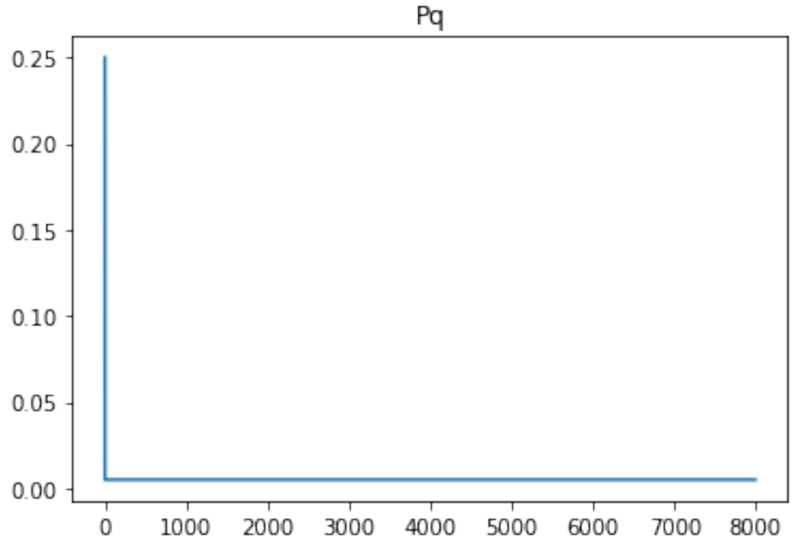
```
17328
```

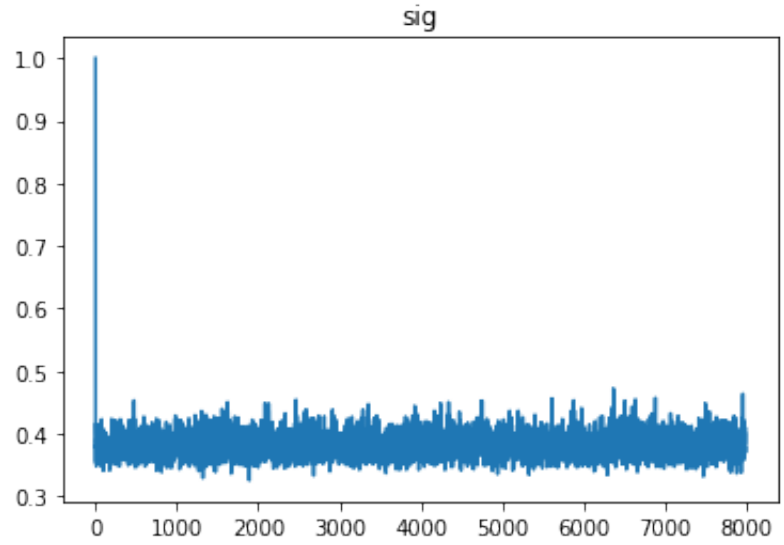

Pq
0.25
0.20
0.15
0.10
0.05
0.00
0
2500
5000
7500
10000
12500
15000
17500

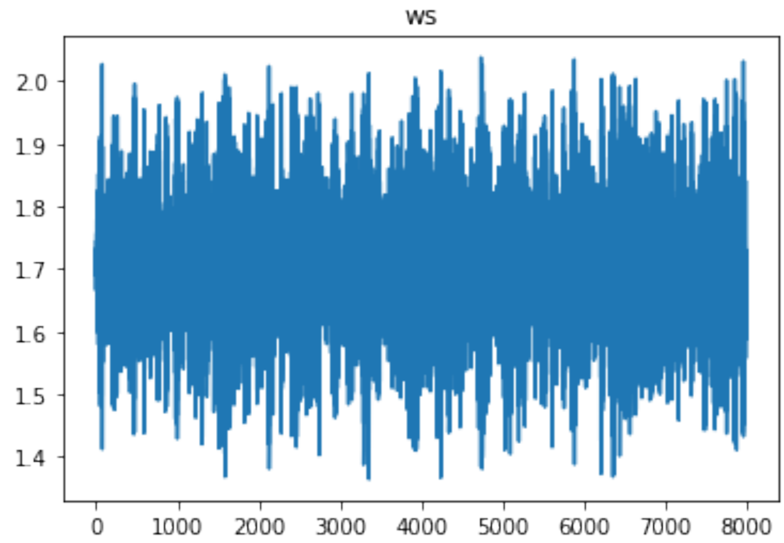

sig
0.500
0.475
0.450
0.425
0.400
0.375
0.350
0.325
0
2500
5000
7500
10000
12500
15000
17500


ws
2.1
2.0
1.9
1.8
1.7
1.6
1.5
1.4
0
2500
5000
7500
10000
12500
15000
17500

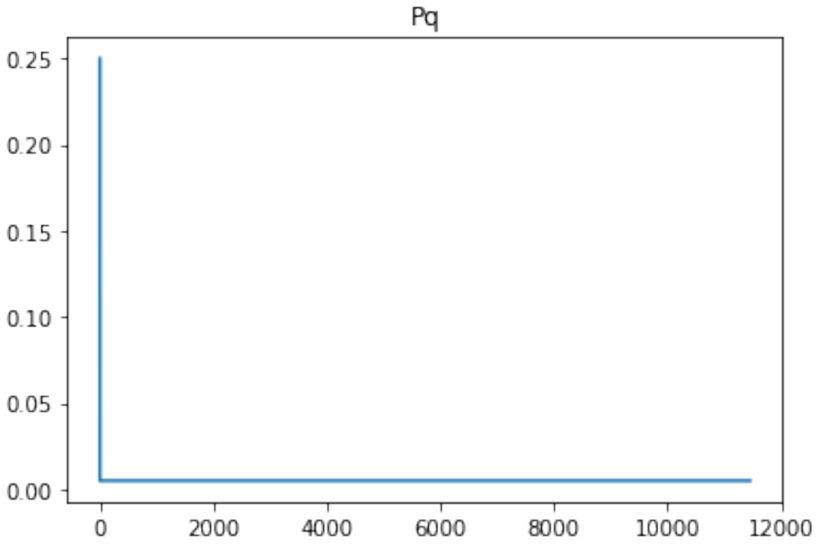
```
7992
```

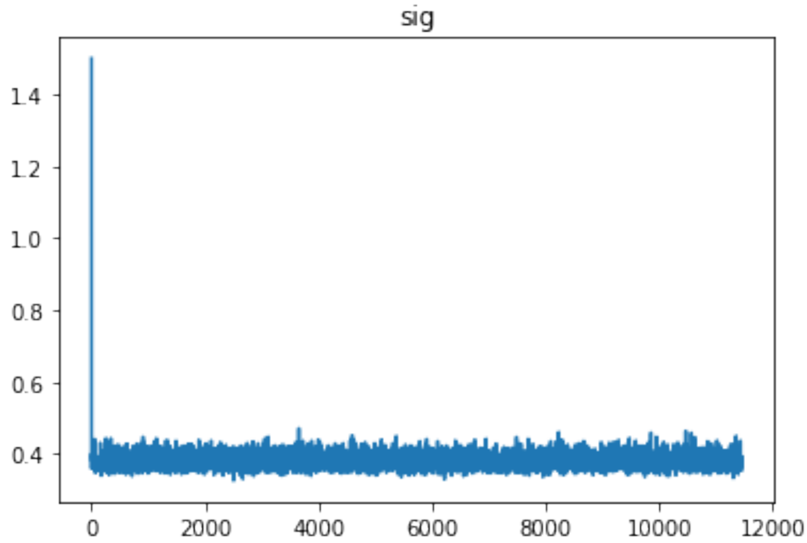

Pq
0.25
0.20
0.15
0.10
0.05
0.00
0
1000
2000
3000
4000
5000
6000
7000
8000

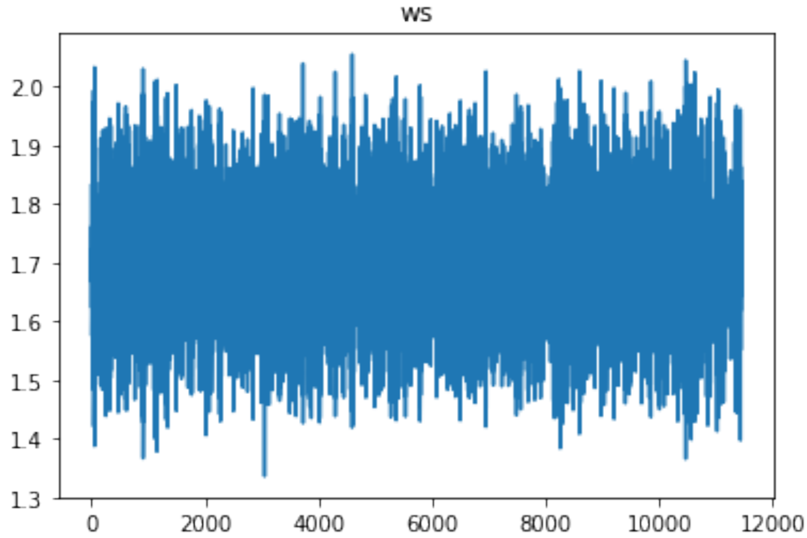

sig
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
1000
2000
3000
4000
5000
6000
7000
8000


ws
2.0
1.9
1.8
1.7
1.6
1.5
1.4
0
1000
2000
3000
4000
5000
6000
7000
8000

```
11455
```


Pq
0.25
0.20
0.15
0.10
0.05
0.00
0
2000
4000
6000
8000
10000
12000


sig
1.4
1.2
1.0
0.8
0.6
0.4
0
2000
4000
6000
8000
10000
12000


ws
2.0
1.9
1.8
1.7
1.6
1.5
1.4
1.3
0
2000
4000
6000
8000
10000
12000