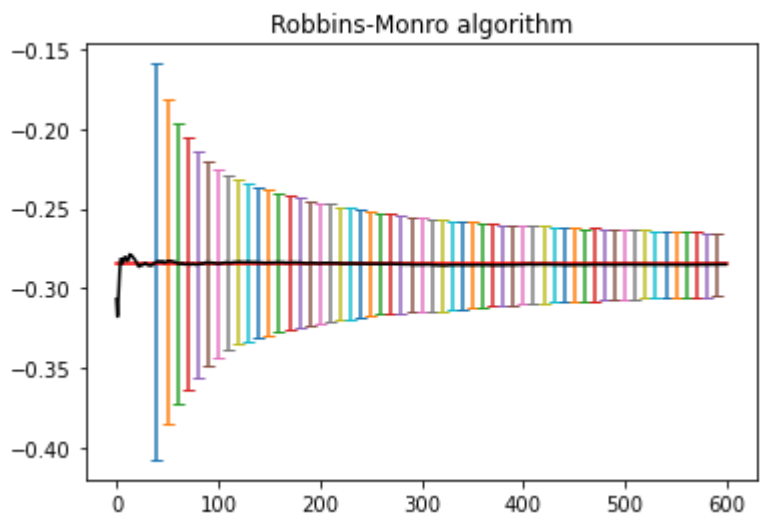
Question 4.1

In [1]:
```
import numpy as np
from matplotlib import pyplot as plt
```

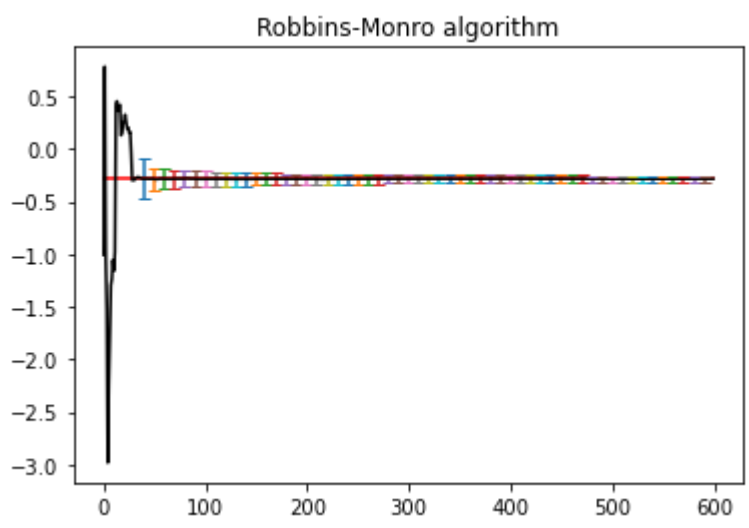
.1 & .2

In [2]:
```
N = 600  # Number of Data
L = 2  # Dimension of the unknown vector
theta = np.random.randn(L, 1)  # Unknown parameter
w = np.zeros((L, 1))  # Initialization


Iter_n = 1000 #number of iterations
noisev = 0.1
wtot = np.zeros((N, Iter_n))
inputvec = lambda n: X[:, n].copy()


for t in range(0, Iter_n):
    X = np.random.randn(L, N)
    noise = np.random.randn(N, 1) * np.sqrt(noisev)
    y = np.zeros((N, 1))
    y[0:N] = np.dot(X[:, 0:N].conj().T, theta)
    y = y + noise
    w = np.zeros((L, 1))
    for i in range(0, N):
        mu = 1 / (i+1)  # Step size
        e = y[i] - np.dot(w.conj().T, inputvec(i))  # Error computation
        w = w + mu * e * inputvec(i)
        wtot[i][t] = w[0][0]

theta1 = theta[0] * np.ones((N, 1))
plt.title("Robbins-Monro algorithm")
plt. plot(theta1, color='red')
mean_w = np.mean(wtot.conj().T, axis=0)
plt.plot(mean_w, color='k', linestyle='solid')

for i in range(0, N):
    if i % 10 == 0 and i > 30:
        plt.errorbar(i, mean_w[i], yerr=np.std(wtot[i, :], axis=0), capsize=3)

plt.show()
```

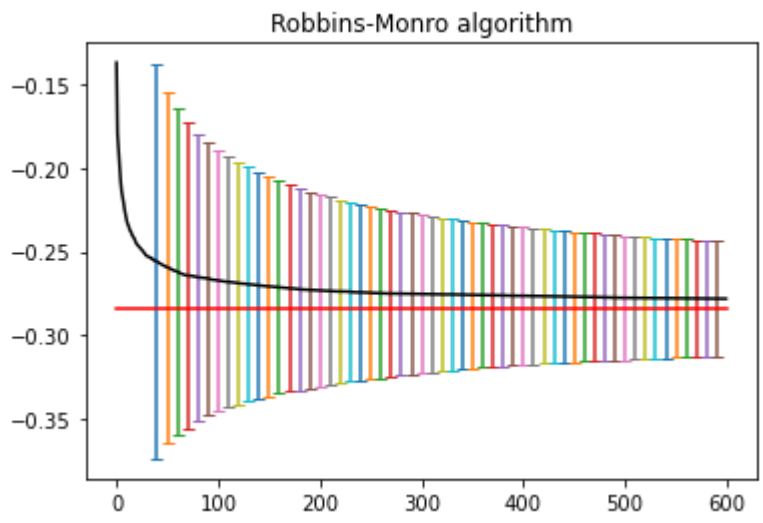


.3

In [3]:
```
for t in range(0, Iter_n):
    X = np.random.randn(L, N)
    noise = np.random.randn(N, 1) * np.sqrt(noisev)
    y = np.zeros((N, 1))
    y[0:N] = np.dot(X[:, 0:N].conj().T, theta)
    y = y + noise
    w = np.zeros((L, 1))
    for i in range(0, N):
        mu = 1 / (i+1)*4  # Step size
        e = y[i] - np.dot(w.conj().T, inputvec(i))  # Error computation
        w = w + mu * e * inputvec(i)
        wtot[i][t] = w[0][0]

theta1 = theta[0] * np.ones((N, 1))
plt.title("Robbins-Monro algorithm")
plt. plot(theta1, color='red')
meanw = np.mean(wtot.conj().T, axis=0)

plt.plot(meanw, color='k', linestyle='solid')


for i in range(0, N):
    if i % 10 == 0 and i > 30:
        plt.errorbar(i, meanw[i], yerr=np.std(wtot[i, :], axis=0), capsize=3)

plt.show()
```



In [4]:
```
for t in range(0, Iter_n ):
    X = np.random.randn(L, N)
    noise = np.random.randn(N, 1) * np.sqrt(noisev)
    y = np.zeros((N, 1))
    y[0:N] = np.dot(X[:, 0:N].conj().T, theta)
    y = y + noise
    w = np.zeros((L, 1))
    for i in range(0, N):
        mu = 1 / ((i+1)*2) # Step size
        e = y[i] - np.dot(w.conj().T, inputvec(i))  # Error computation
        w = w + mu * e * inputvec(i)
        wtot[i][t] = w[0][0]

theta1 = theta[0] * np.ones((N, 1))
plt.title("Robbins-Monro algorithm")
plt. plot(theta1, color='red')
meanw = np.mean(wtot.conj().T, axis=0)
plt.plot(meanw, color='k', linestyle='solid')


for i in range(0, N):
    if i % 10 == 0 and i > 30:
        plt.errorbar(i, meanw[i], yerr=np.std(wtot[i, :], axis=0), capsize=3)

plt.show()
```



In this case it can be seen how the line converges with the unknown parameter quickly. Also, it can be concluded that the algorithm is quite susceptible to the step size sequence,the error is affected when the step size.