Assignment 2

Exercise 1.
Question.5 Code and Plots

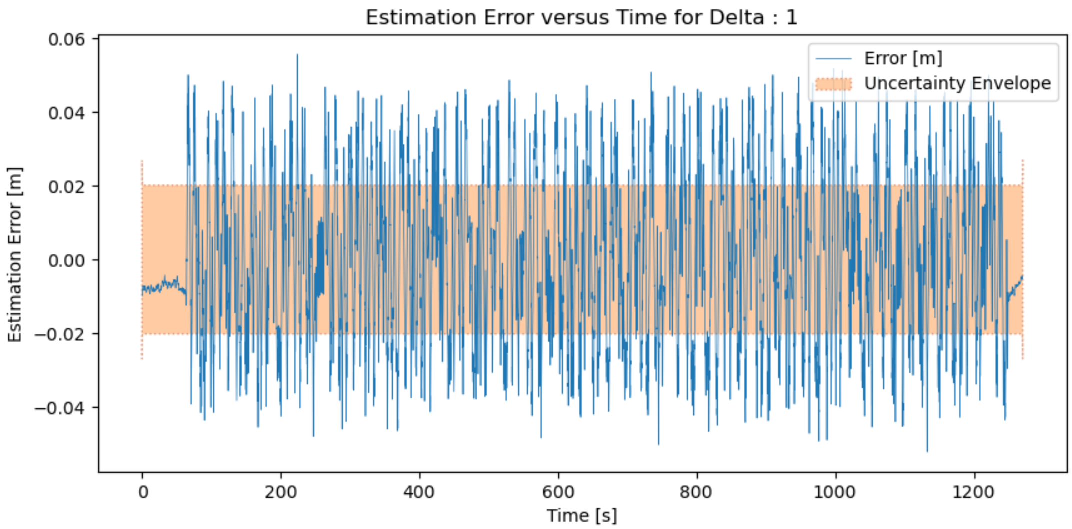
Code:

```
import numpy as np
import scipy.io
import scipy.linalg
from matplotlib import pyplot as plt
import matplotlib


def batch_est(d, y_d, measurement_var):
   K = 12709
   C_arr = np.diag(np.array([1] * K, dtype=np.float32))
   R_inv = np.diag(np.array([1] * K, dtype=np.float32)) * (1. /
measurement_var)
   if d == 1:
       return C_arr, R_inv, y_d
   else:
       C_delta = np.zeros(shape=(K // d, K))
       R_delta = np.zeros(shape=(K // d, K // d))
       y_delta = np.zeros(shape=(K // d, 1))

       for i, j in zip(range(d, K, d), range(len(C_delta))):
           C_delta[j] = C_arr[i]
           R_delta[j][j] = R_inv[i][i]
           y_delta[j] = y_d[i]
           C_delta = np.vstack(C_delta)
           R_delta = np.vstack(R_delta)
           y_delta = np.vstack(y_delta)
           print(y_delta)
       return C_delta, R_delta, y_delta


def q5(u_k, position_var, C_k, R_inv_k, y_kd):
   K = 12709
   A = np.tril(np.ones((K, K), dtype=np.float32), 0)  # not affected by
gamma
```

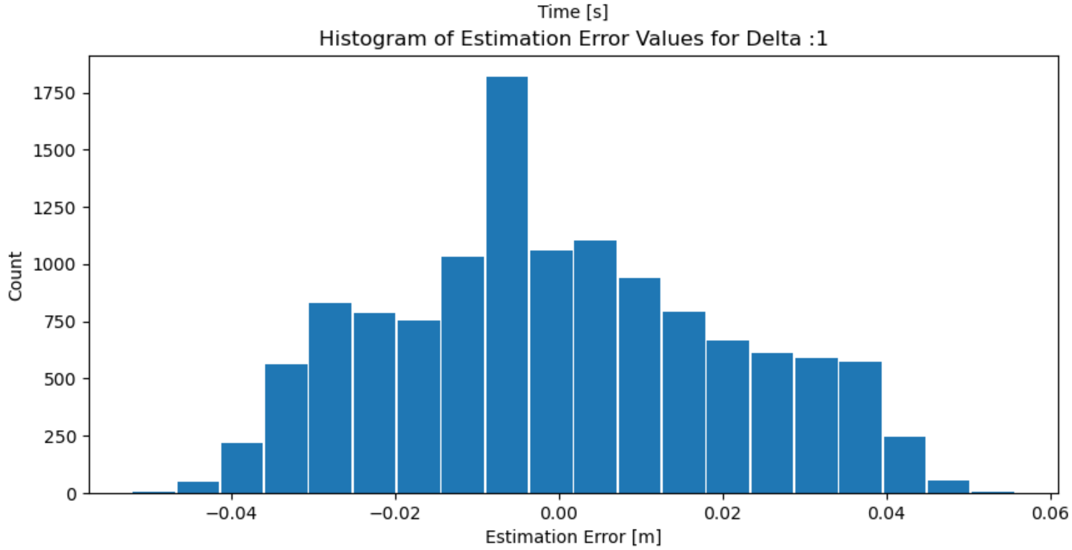
```python
    H = np.vstack([np.linalg.inv(A), C_k])
    H = np.delete(H, 0, 0)
    H = H.astype(np.float32)

    Q_inv = np.diag(np.array([1] * K, dtype=np.float32)) * (1. /
position_var)  # not affected
    print(Q_inv)
    W_inv = scipy.linalg.block_diag(Q_inv, R_inv_k)
    # print(W_inv)
    W_inv = np.delete(W_inv, 0, 0)
    W_inv = np.delete(W_inv, 0, 1)
    W_inv = W_inv.astype(np.float32)


    # Q = np.diag(np.array([1] * K, dtype=np.float32)) * (pos_var)
    # R = np.diag(np.array([1] * K, dtype=np.float32)) * (meas_var)
    # W = scipy.linalg.block_diag(Q, R)
    # W_inv_old = np.linalg.inv(W).astype(np.float32)
    # print(W_inv_old.dtype)

    z = np.vstack([u_k, y_kd])
    z = np.delete(z, 0, 0)
    z = z.astype(np.float32)

    lhs = H.T @ W_inv @ H
    np.save("q4_array", lhs)
    rhs = H.T @ W_inv @ z

    x_post = np.linalg.solve(lhs, rhs)
    print(x_post)
    np.save("x_post", x_post)




def plot(x_true, t, delta):
    x_post = np.load("x_post.npy")
    error = x_post - x_true
    inv_cov = np.load("q4_array.npy")
```

```
    x_var = np.linalg.inv(inv_cov)
    x_var_diag = x_var.diagonal()
    uncertainty = np.sqrt(x_var_diag) * 3
    abs_error = np.average(np.abs(x_post) - np.abs(x_true))
    # print("Avg Error :", abs_error)
    fig, ax = plt.subplots(2, 1, figsize=(10, 10), dpi=100)

    # ax[2].plot(x_true, linewidth=0.5, label="x_true [m]")
    # ax[2].plot(x_post, linewidth=0.5, label="x_est [m]")
    # ax[2].set_ylabel("X position [m]")
    # ax[2].set_xlabel("No. of Records")
    # ax[2].legend(loc="upper left")

    ax[0].fill_between(np.squeeze(t), -uncertainty, +uncertainty,
edgecolor='#CC4F1B', facecolor='#FF9848', alpha=0.5,
                       linestyle=':', label='Uncertainty Envelope')
    ax[0].plot(t, error, linewidth=0.5, label="Error [m]")
    ax[0].set_ylabel("Estimation Error [m]")
    ax[0].set_xlabel("Time [s]")
    ax[0].legend(loc="upper right")
    ax[0].set_title(f"Estimation Error versus Time for Delta : {delta}")

    ax[1].hist(error, rwidth=0.95, bins=20)
    # ax[1].axvline(x=np.max(uncert), color='r', linestyle='dashed',
linewidth=2)
    # ax[1].axvline(x=-np.max(uncert), color='r', linestyle='dashed',
linewidth=2)
    ax[1].set_ylabel("Count")
    ax[1].set_xlabel("Estimation Error [m]")
    ax[1].set_title(f"Histogram of Estimation Error Values for Delta
:{delta}")
    plt.savefig(f"delta{delta}.png")
    # plt.show()


if __name__ == '__main__':
    data =
scipy.io.loadmat('/home/shounak/Desktop/State_Estimation_Robotics_Course/A
ssignment_2/dataset1.mat')
```

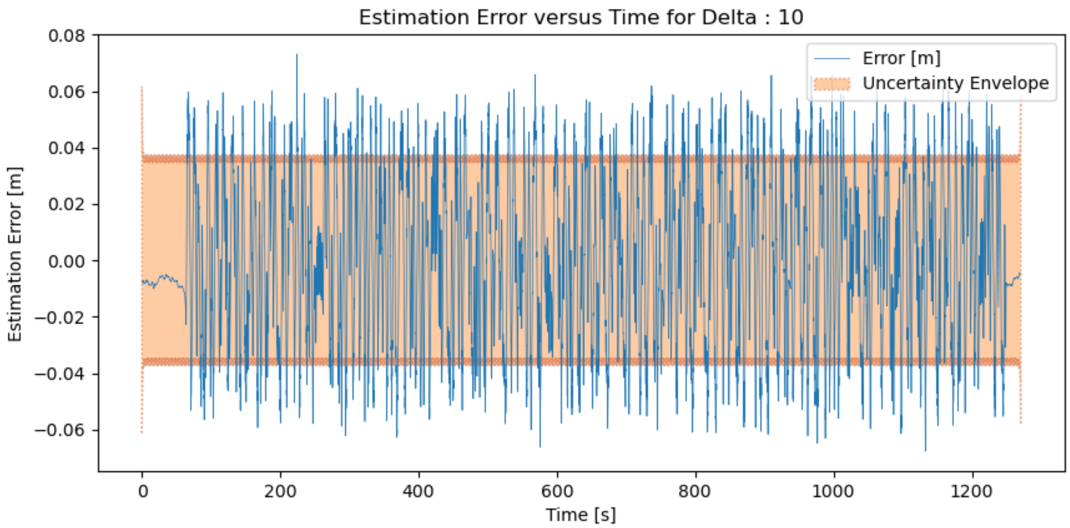
```python
# Timestep
t_step = 0.1
# x_c is the position of the cylinder's center
x_c = data["l"]
# range, r_k
r_k = data["r"]
# observation model
y_k = x_c - r_k
# true position, x_k, of the robot
x_true = data["x_true"]
# x_true = np.delete(x_true, 0, 0)
#print(x_true)
# data timestamps [s]
t = data["t"]
# speed, u_k
v = data["v"]
# control inputs
u = v * t_step
# variance of the range readings
r_var = data["r_var"]
# measurement noise variance
meas_var = r_var.item()
# variance of the speed readings
v_var = data["v_var"]
# process noise variance, position variance
pos_var = (t_step ** 2) * v_var.item()

# Total data points
p = x_true.shape[0]
for delta in [1, 10, 100, 1000]:
    C, R, y = batch_est(delta, y_k, meas_var)
    # print(C, R, y)
    q5(u, pos_var, C, R, y)
    plot(x_true, t,  delta)
```

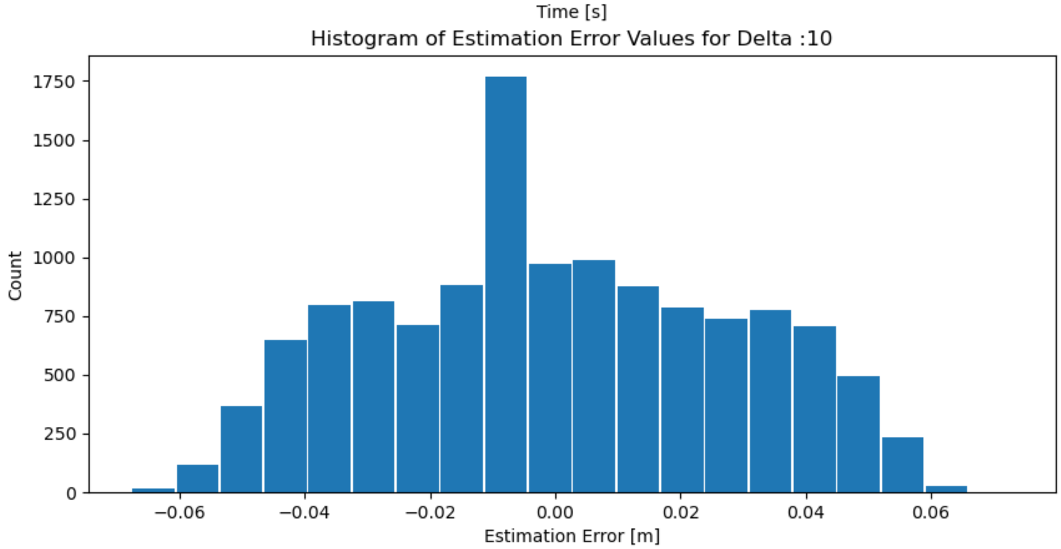


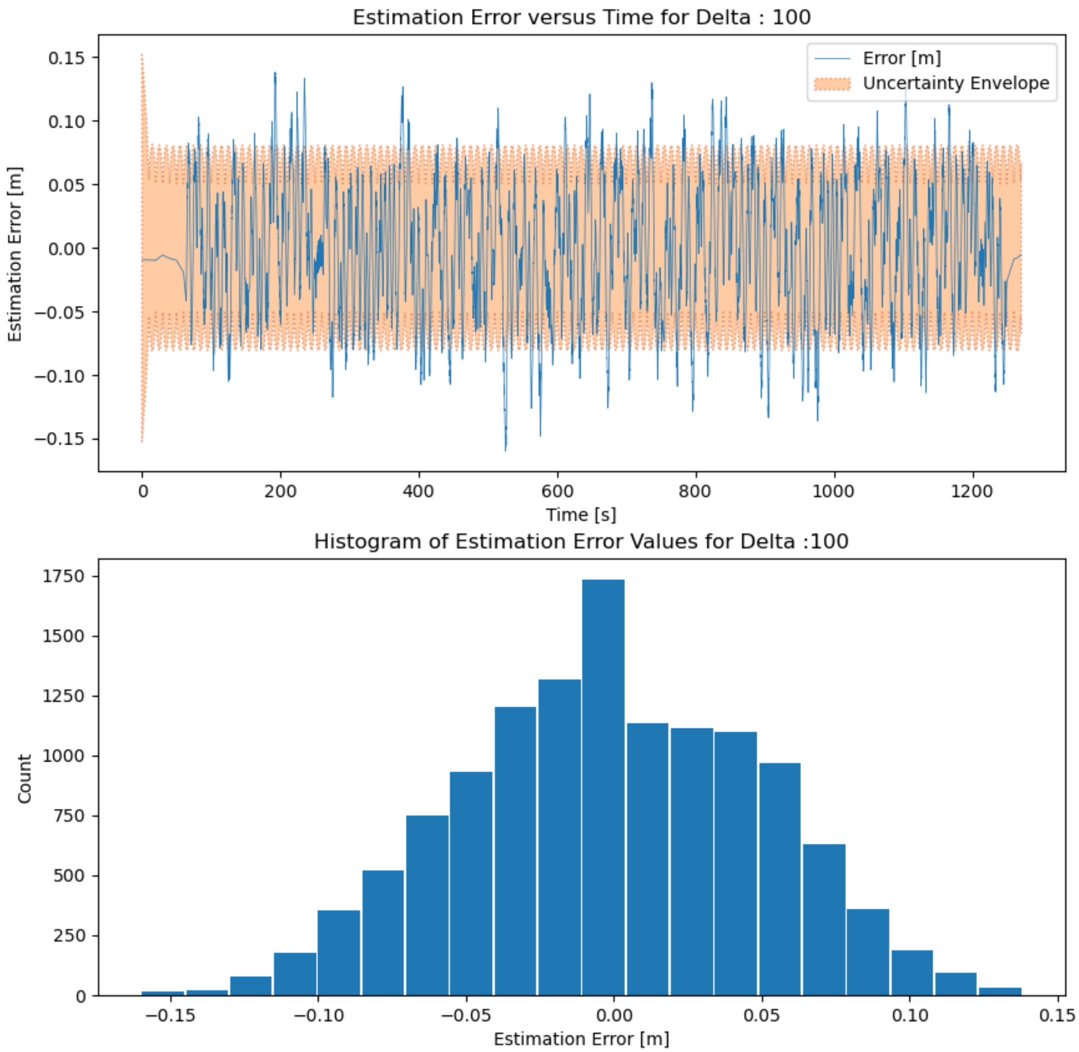
Estimation Error versus Time for Delta : 1
Error [m]
Uncertainty Envelope
Estimation Error [m]
Time [s]
0.06
0.04
0.02
0.00
−0.02
−0.04
0
200
400
600
800
1000
1200



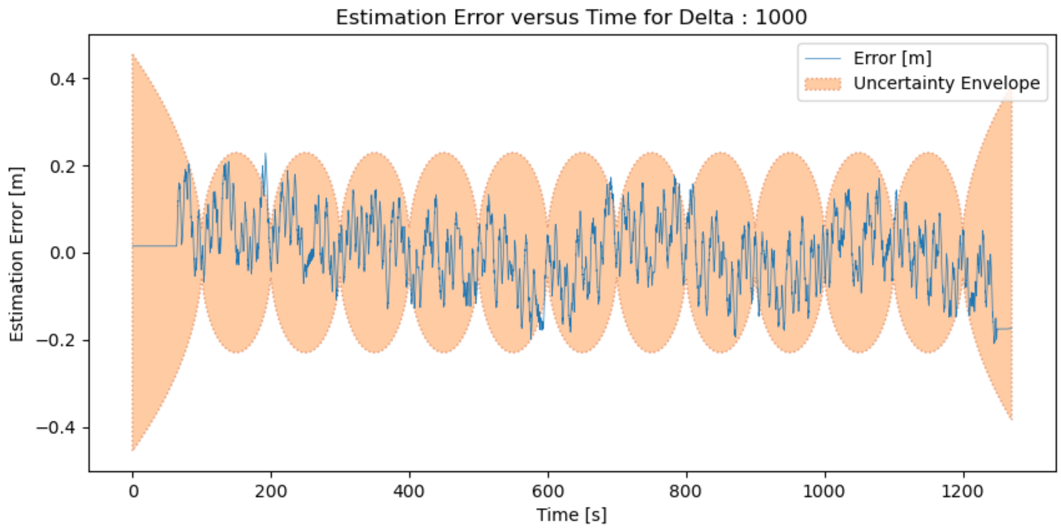
Histogram of Estimation Error Values for Delta :1
Count
Estimation Error [m]
1750
1500
1250
1000
750
500
250
0
−0.04
−0.02
0.00
0.02
0.04
0.06


Estimation Error versus Time for Delta : 10
Error [m]
Uncertainty Envelope
Estimation Error [m]
Time [s]
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0
200
400
600
800
1000
1200

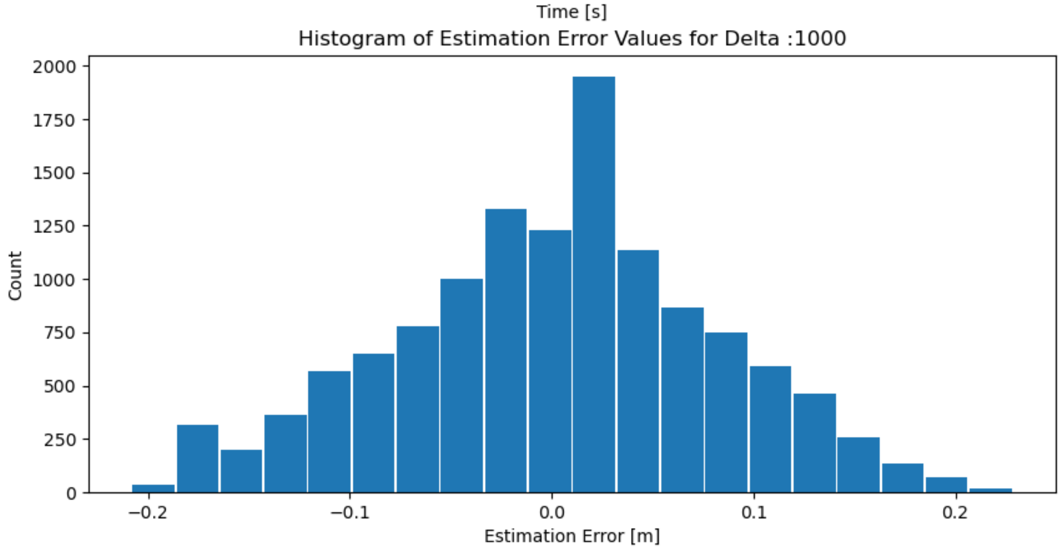

Histogram of Estimation Error Values for Delta :10
Count
Estimation Error [m]
1750
1500
1250
1000
750
500
250
0
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06



Estimation Error versus Time for Delta : 100
Error [m]
Uncertainty Envelope
Estimation Error [m]
Time [s]
Histogram of Estimation Error Values for Delta :100
Count
Estimation Error [m]



Estimation Error versus Time for Delta : 1000
Error [m]
Uncertainty Envelope
Estimation Error [m]
Time [s]
0.4
0.2
0.0
−0.2
−0.4
0
200
400
600
800
1000
1200



Histogram of Estimation Error Values for Delta :1000
Count
Estimation Error [m]
2000
1750
1500
1250
1000
750
500
250
0
−0.2
−0.1
0.0
0.1
0.2

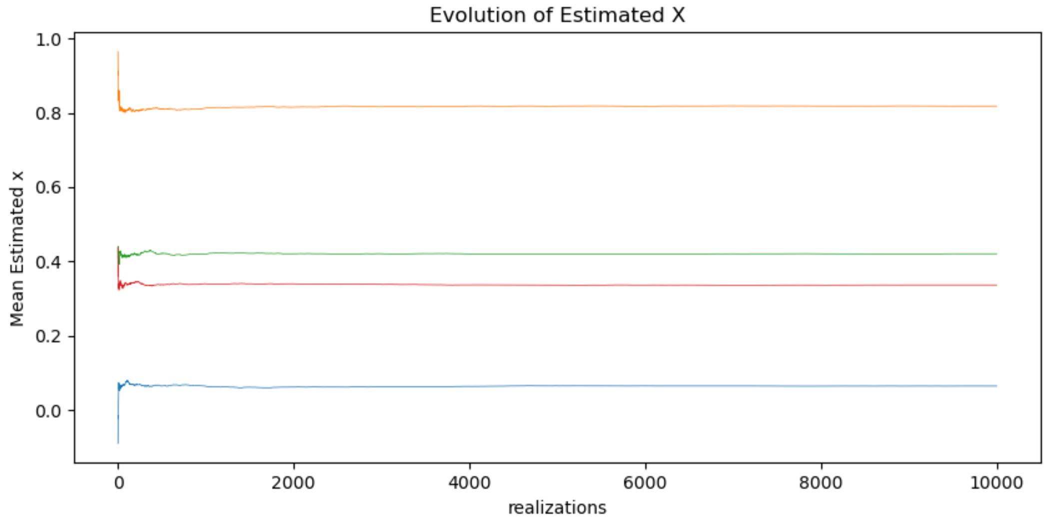
Question 2. Code and Plots

```
import numpy as np
import scipy.linalg
from matplotlib import pyplot as plt
import matplotlib


def plot_means(x_hat, error_hat):

   fig, ax = plt.subplots(2, 1, figsize=(10, 10), dpi=100)

   for i, j in zip(range(x_hat.shape[0]), range(error_hat.shape[0])):

       ax[0].plot(x_hat[i,:], linewidth=0.5)
       ax[0].set_ylabel("Mean Estimated x")
       ax[0].set_xlabel("realizations")
       ax[0].set_title("Evolution of Estimated X")

       ax[1].plot(error_hat[j,:], linewidth=0.5)
       ax[1].set_ylabel("Mean Error")
       ax[1].set_xlabel("realizations")
       ax[1].set_title("Evolution of error")

   plt.grid()

def plot_covariance(cov_diff_frob, cov_diff_trace):

   fig, ax = plt.subplots(2, 1, figsize=(10, 10), dpi=100)

   ax[0].plot(cov_diff_frob[1:-1], linewidth=0.5)
   ax[0].set_ylabel("Frobenius norm of covariance difference")
   ax[0].set_xlabel("realizations")
   ax[0].set_title("Evolution of the Frobenius norm of the difference of
covariances")

   ax[1].plot(cov_diff_trace[1:-1], linewidth=0.5)
   ax[1].set_ylabel("Absolute trace of covariance difference")
   ax[1].set_xlabel("realizations")
```

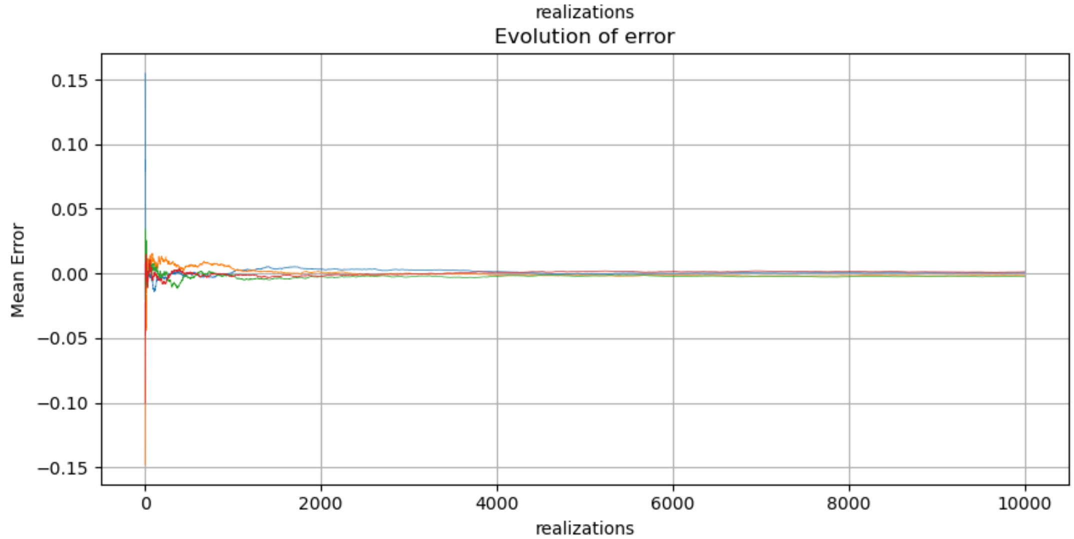
```
    ax[1].set_title("Evolution of the trace of the difference of
covariances:")

    plt.grid()

def plot_variance(var_hat, var_diff):

    fig, ax = plt.subplots(2, 1, figsize=(10, 10), dpi=100)

    ax[0].plot(var_hat[1:-1], linewidth=0.5)
    ax[0].set_ylabel("Mean of estimated variance")
    ax[0].set_xlabel("realizations")
    ax[0].set_title(" Evolution of the variance estimator")

    ax[1].plot(var_diff[1:-1], linewidth=0.5)
    ax[1].set_ylabel("variance differences")
    ax[1].set_xlabel("realizations")
    ax[1].set_title(" Evolution of the error of the noise variance")

    plt.grid()

def means(M, K, X):
    x_hat_mean = np.zeros(shape=(M,K))
    for i in range(K):
        temp_x = np.take(X, range(i+1), axis=1)
        temp_mean = np.mean(temp_x, axis =1)
        x_hat_mean[:,i] = temp_mean
    #print(temp_x)
    return x_hat_mean

def cov_err(M, K, err, cov_given):
    err_hat_cov = []
    for i in range(K):
        temp_err = np.take(err, range(i+1), axis=1)
        temp_cov = np.cov(temp_err, bias =False)
        err_hat_cov.append(temp_cov)

    cov_diff = []
    cov_diff_frob = []
    cov_diff_trace = []
```

```
    for j in range(K):
        #covariance differences
        cov_d = cov_given - err_hat_cov[j]
        cov_diff.append(cov_d)
        # Frobenius norm difference
        cov_diff_frob.append(np.linalg.norm(cov_d))
        # trace of the dierence of covariances
        cov_diff_trace.append(abs(cov_d.trace()))

    return cov_diff_frob, cov_diff_trace

def var_hat_n(M, K, X, var_given):
    var_hat = []
    for i in range(K):
        temp_x_var = np.take(X, range(i+1), axis=1)
        temp_var = np.mean(np.var(temp_x_var, axis =1))
        var_hat.append(temp_var)

    var_diff = []

    for j in range(K):
        #variance differences
        var_d = var_given - var_hat[j]
        var_diff.append(var_d)

    #print(temp_x)
    return var_hat, var_diff

if __name__ == '__main__':

    K = 10000
    M = 4
    N = 6
    H = np.random.random((N, M))
    H = H.astype(np.float32)
    H_T = H.T
    H_T_H = H_T @ H
    H_T_H_inv = np.linalg.inv(H_T_H)
    H_const = H_T_H_inv @ H_T
```

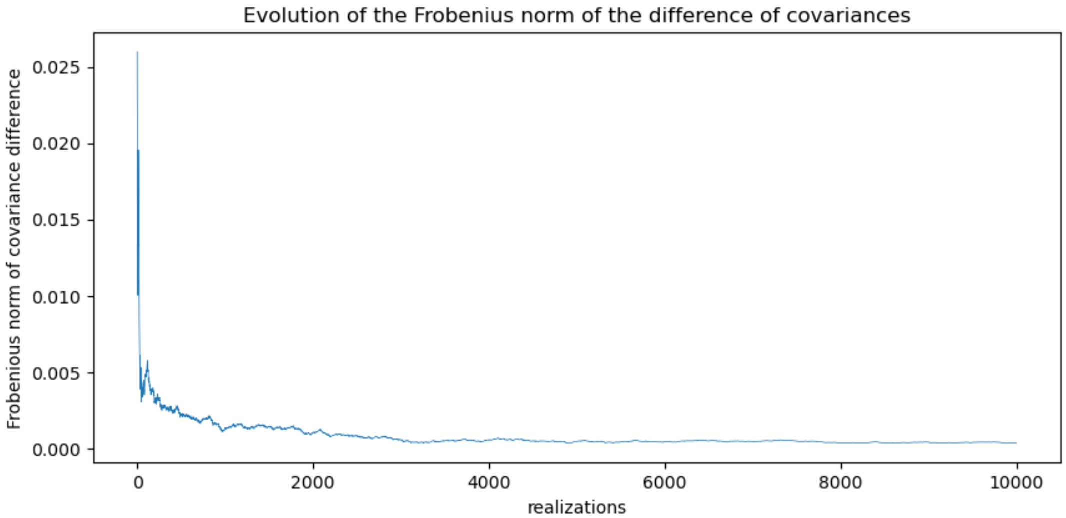
```python
x_true = np.random.random((M,1))
var = 0.01

#realization
y = np.full((N,K), H @ x_true)

#adding noise n, multivariate Gaussian
for i in range(K):
    mean = np.zeros(N)
    covariance = np.diag(np.array([1] * N, dtype=np.float32)) * var
    noise = np.array([np.random.multivariate_normal(mean, covariance)])
    y[:, i] = y[:,i] + noise

# a
#estimate x and error
x_est = H_const @ y
error_x_est = x_true - x_est

#mean across the realization
x_hat = means(M, K, x_est)
error_hat = means(M, K, error_x_est)
plot_means(x_hat, error_hat)

# b
#given covariance
cov_given = var * np.linalg.inv(H.T @ H)
#covariance across the realization
cov_diff_frob, cov_diff_trace = cov_err(M, K, error_x_est,cov_given)
plot_covariance(cov_diff_frob, cov_diff_trace )

# c
# given sigma_sq = 0.01
var_hat, var_diff = var_hat_n(M, K, x_est, var)
plot_variance(var_hat, var_diff )

plt.show()
```

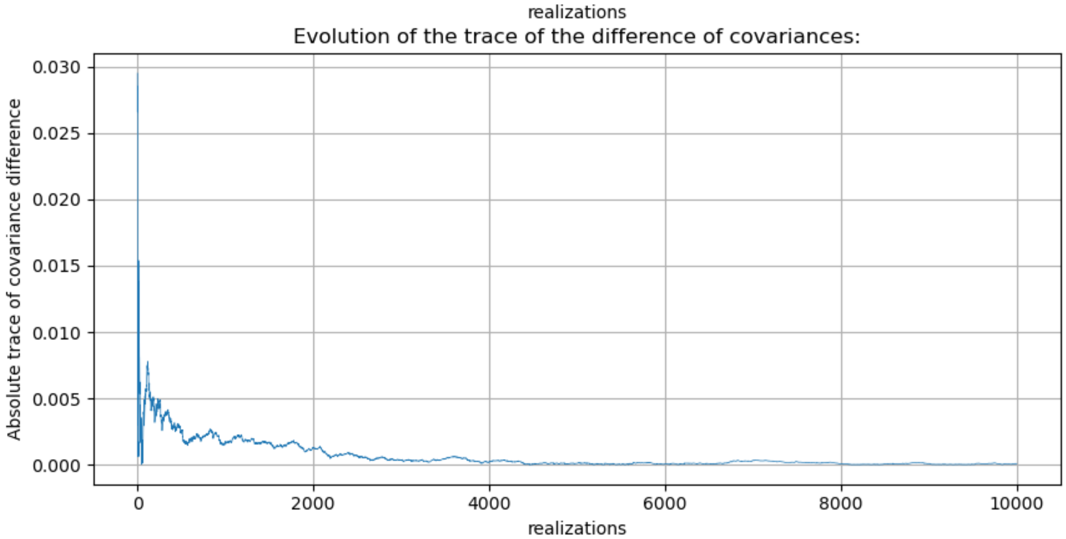


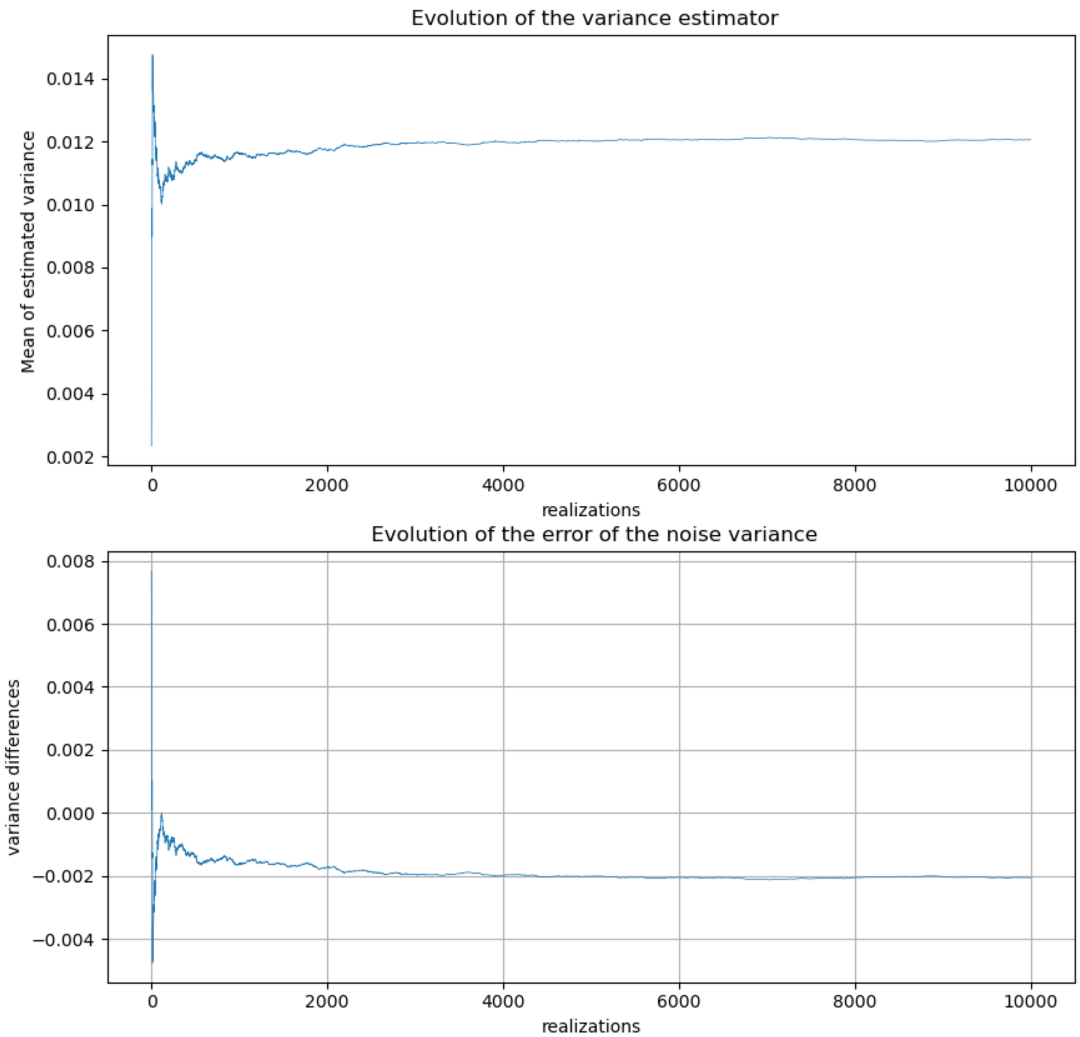
Evolution of Estimated X
Mean Estimated x
realizations



Evolution of error
Mean Error
realizations


Evolution of the Frobenius norm of the difference of covariances
Frobenious norm of covariance difference
0.025
0.020
0.015
0.010
0.005
0.000
0
2000
4000
6000
8000
10000
realizations


Evolution of the trace of the difference of covariances:
Absolute trace of covariance difference
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
2000
4000
6000
8000
10000
realizations


Evolution of the variance estimator
Mean of estimated variance
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0
2000
4000
6000
8000
10000
realizations
Evolution of the error of the noise variance
variance differences
0.008
0.006
0.004
0.002
0.000
−0.002
−0.004
0
2000
4000
6000
8000
10000
realizations

Date: __/__/____

Assignment - 2

Name: Shounak

1. Calculating σ_q^2 → associated with Process noise. $\sim N(0, \sigma_q^2)$

Therefore, integrating the velocity noise.

$$\sigma^2_{vel} = (\text{std. of velocity})^2$$

As per the histogram, the system is discrete and the velocity per time step follows a gaussian.

$$X = \sum_{i=1}^{N} x_i \cdot T_s$$

$$\sigma_q^2 = var\left(\sum_{i=1}^{N} x_i T_s\right).$$

(variance of position)

$$= E\left[(X - \mu)(X - \mu)^T\right] \quad (1)$$

Now, $\mu = E[X]$

$$= E\left[\sum_{i=1}^{N} x_i T_s\right]$$

$$= \sum_{i=1}^{N} T_s \, E[x_i]$$

$$= \sum_{i=1}^{N} T_s \, \mu_v$$ → mean velocity

#learnthesmarterway

Continuing (4)

$$\sigma_q^2 = E\left[(x-\mu_v)(x-\mu_v)^T\right]$$

$$= E\left[\left(\sum_{i=1}^{N} T_s \cdot x_i - \sum_{i=1}^{N} T_s \mu_v\right)\left(\sum_{j=1}^{N} T_s \cdot x_j - \sum_{j=1}^{N} T_s \mu_v\right)^T\right]$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} T_s T_s \, E\left[(x_i-\mu_v)(x_j-\mu_v)^T\right]$$

$$= \sum_{i=1}^{N} T_s^2 \sigma_{vel}^2$$

Given, $T_s = 0.1$

$\sigma_{vel} = 0.047554 \, m/s.$

$$= (0.1)^2 (0.047554)^2$$

$$= 2.256 \times 10^{-5} \, m^2$$

Calculating $\sigma_r^2 \rightarrow$ associated with measurement noise. $\sim N(0, \sigma_r^2)$

Therefore, integrating the range error,

$$\sigma_{range}^2 = (\text{std of range})^2 = (0.019155)^2$$

$$= 3.67 \times 10^{-4} \, m^2$$

Date: __/__/____

Q.

$x = x_{1:k} = (x_1, \dots, x_k)$

$u = u_{1:n} = (u_1 \dots u_n)$, $v_k = u_k T$.

$y = y_{1:n} = (y_1 \dots y_n)$

MAP, $\hat{x} = \arg\max_x p(x \mid u, y)$

Rewriting the MAP estimate using Bayes' Rule:-

$$\hat{x} = \arg\max_x p(x \mid u, y) = \arg\max_x \frac{p(y \mid x, u)\, p(x \mid u)}{p(y \mid u)}$$

$$= \arg\max_x p(y \mid x, u)\, p(x \mid u)$$

⇒ we drop the denominator because it does not depend on x and u in $p(y \mid x, u)$, since it does not affect y in our system if x is ~~are~~ known.

Optimizing it by taking the negative log f^n.

$$\hat{x} = \arg\min_x \left(-\ln p(y \mid x)\, p(x \mid u)\right).$$

from an optimization point of view we want to solve following

$$\hat{x} = \arg\min_x J(x)$$

#learnthesmarterway

⇒ We then define an overall objective f^n $J(x)$, that we will seek to minimum w.r.t x.

$$J(x) = \sum_{k=1}^{K} \left(J_{w,k}(x) + J_{y,k}(x) \right)$$

$$J_{w,k}(x) = \frac{1}{2} \left(x_k - A_{k-1} x_{k-1} - u_k T \right)^T \times Q_k^{-1} \left(x_k - A_{k-1} x_{k-1} - u_k T \right),$$

$$k = 1 \dots K$$

$$J_{y,k}(x) = \frac{1}{2} \left(y_k - C_k x_k \right)^T R_k^{-1} \left(y_k - C_k x_k \right),$$

$k =$

Date: __/__/___

3. In order to find the optimal estimates,

we can first express our cost fn from last question more compactly as,

$$J(x) = \frac{1}{2}(Z-Hx)^T W^{-1}(Z-HX)$$

where,

$$Z = \begin{bmatrix} v \\ y \end{bmatrix} \quad H = \begin{bmatrix} A^T \\ C \end{bmatrix} \quad W = \begin{bmatrix} Q & \\ & R \end{bmatrix}$$

~~Since J(x) is exactly a paraboloid, we~~

Estimating x^* using batch least squares soln, we set the partial derivative of $J(x)$ w.r.t to x, to zero:

$$\left.\frac{\partial J(x)}{\partial x^T}\right|_{x^*} = -H^T W^{-1}(Z - Hx^*) = 0$$

$$\Rightarrow (H^T W^{-1} H)x^* = H^T W^{-1} Z$$

$$x^* = (H^T W^{-1} H)^{-1} H^T W^{-1} Z$$

Date: __/__/____

4. Given $k = 12709$

Solving for a subset of poses, $x_\delta, x_{2\delta}, x_{3\delta} \cdots x_k$

and $r_\delta, r_{2\delta}, r_{3\delta} \cdots r_k$.

$\therefore$

$$z = \begin{bmatrix} v \\ y \end{bmatrix} \quad H = \begin{bmatrix} A^{-1} \\ C \end{bmatrix} \quad W = \begin{bmatrix} Q & \\ & R \end{bmatrix}$$

if $\delta = 1000$

$$C = \begin{bmatrix} C_\delta & & & \\ & C_{2\delta} & & \\ & & \ddots & C_k \end{bmatrix}_{12 \times 12709} \quad R = \begin{bmatrix} R_\delta & & & \\ & R_{2\delta} & & \\ & & \ddots & R_k \end{bmatrix}_{12 \times 12}$$

$$y = \begin{bmatrix} y_\delta \\ y_{2\delta} \\ \vdots \\ \vdots \end{bmatrix}_{12 \times 1}$$

$$\therefore \; z = \begin{bmatrix} v \\ y \end{bmatrix}_{12720 \times 1} \quad H = \begin{bmatrix} A^{-1} \\ C \end{bmatrix}_{12720 \times 12709}$$

$$W^{-1} = \begin{bmatrix} Q & \\ & R \end{bmatrix}_{12720 \times 12720}$$

To get the optimal position estimates;

$$\{\hat{X}\}_{12709\times 1} = \left(H^T W^{-1} H\right)^{-1} H^T W^{-1} Z$$