Assignment 2

Kalman Filter

Irsh Vijay
21EC39055

1 Python Scripts

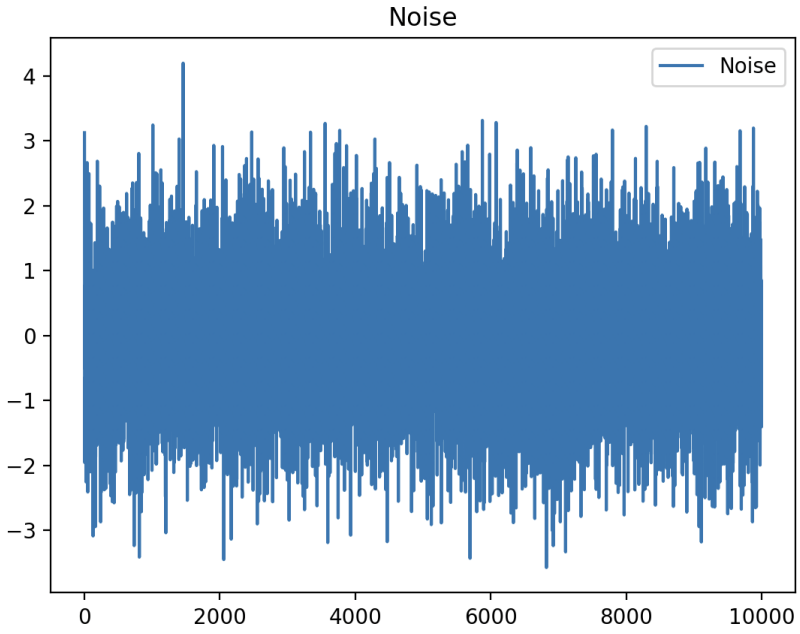
```
class KalmanFilter:
    def __init__(self, A, B, C, P, Q, R, x0):
        self.A = A
        self.B = B
        self.C = C
        self.P = P
        self.Q = Q
        self.R = R
        self.x = x0

    def predict(self, u = None):
        self.x = self.A @ self.x
        if u:
            self.x += self.B @ u
        self.P = self.A @ self.P @ self.A.T + self.Q

    def update(self, y):
        self.gain = self.P @ self.C.T / (self.C @ self.P @ self.C.T + self.R)

        self.x = self.x + self.gain @ (y - self.C @ self.x)
        self.P = (np.eye(self.gain.shape[0]) - self.gain @ self.C) @ self.P

    def filter(self, measurement, control_input = None):
        self.predict(u = control_input)
        self.update(y = measurement)
        return self.x
```

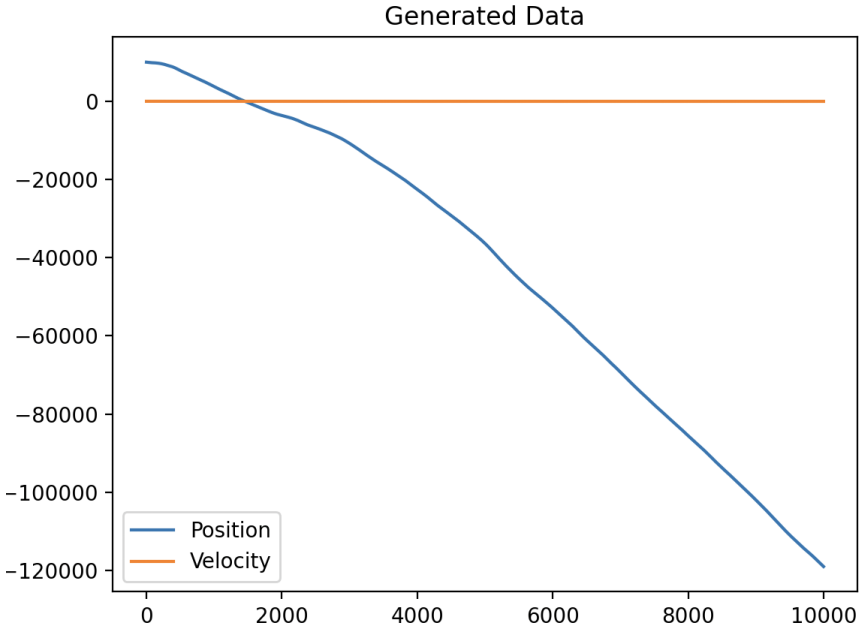
```
class KalmanExperiment:
    def __init__(self, A, B, C, P0, Q, R, x0, num_points, sigma=1):
        self.A = A
        self.check_eigenvalues()
        self.B = B
        self.C = C
        self.P0 = P0
        self.Q = Q
        self.R = R
        self.x0 = x0

        self.num_points = num_points
        self.n = self.A.shape[0]
        self.sigma = sigma

        self.x = np.zeros((num_points, self.n))
        self.y = np.zeros((num_points, C.shape[0]))

        self.kalman_filter = KalmanFilter(A, B, C, P0, Q, R, x0)

    def check_eigenvalues(self):
        for eigenval in np.abs(np.linalg.eigvals(self.A)):
            assert eigenval <= 1, f'Unstable A: {eigenval}'

    def generate_data(self):
        self.w = np.random.normal(0, self.sigma, (num_points, 1))

        self.x[0] = self.x0

        for k in range(1, self.num_points):
            self.x[k] = self.A @ self.x[k-1] + self.B @ self.w[k]
            measurement_noise = np.random.multivariate_normal(np.zeros(self.y.shape[1]),
            self.y[k] = self.C @ self.x[k] + measurement_noise

    def run(self):
        self.generate_data()

        estimated_states = np.zeros((self.num_points, self.n))
        for k in range(self.num_points):
            measurement = self.y[k]
            estimated_state = self.kalman_filter.filter(measurement)
            estimated_states[k] = estimated_state

        return estimated_states
```

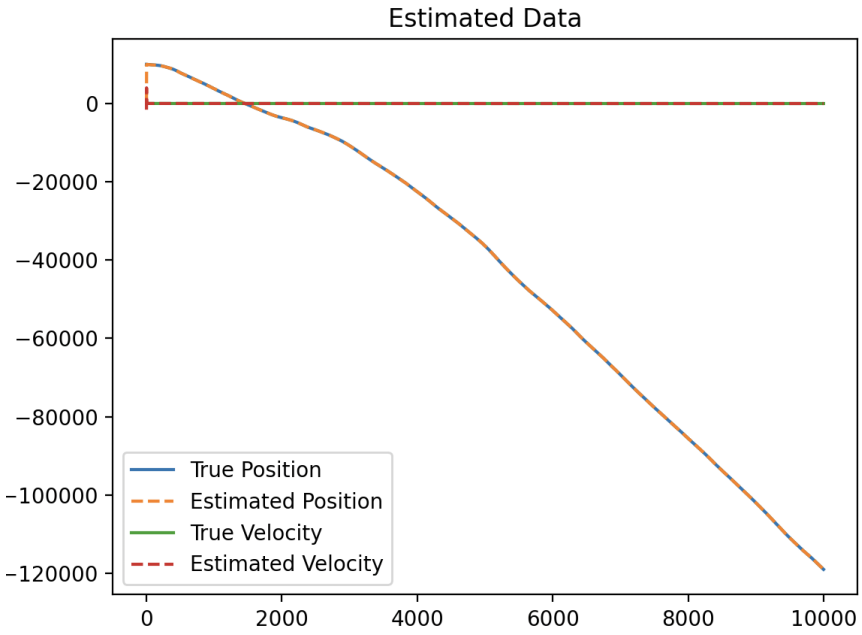
2 Plots

2.1 Example 1

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0.05 \\ 0.1 \end{bmatrix}, C = \begin{bmatrix} 1 \\ 0.1 \end{bmatrix}, P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}, R = 1$$

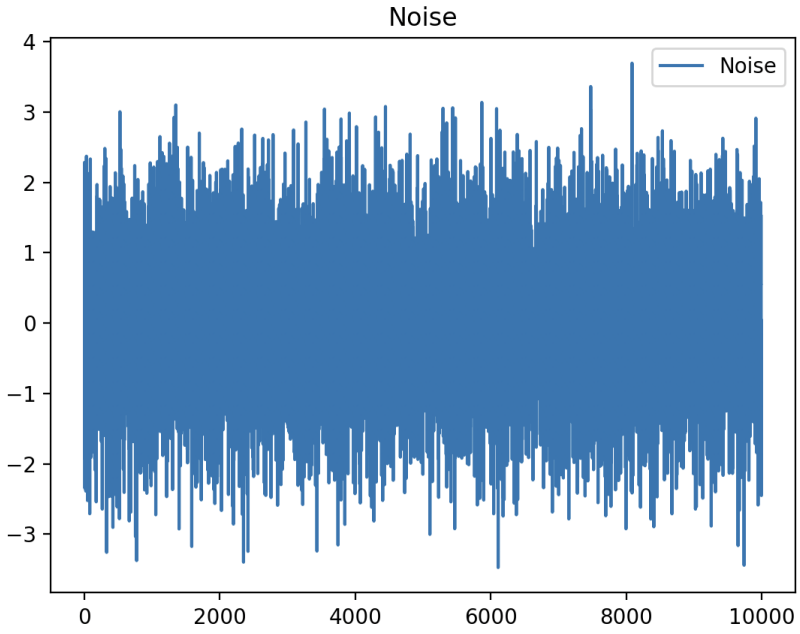


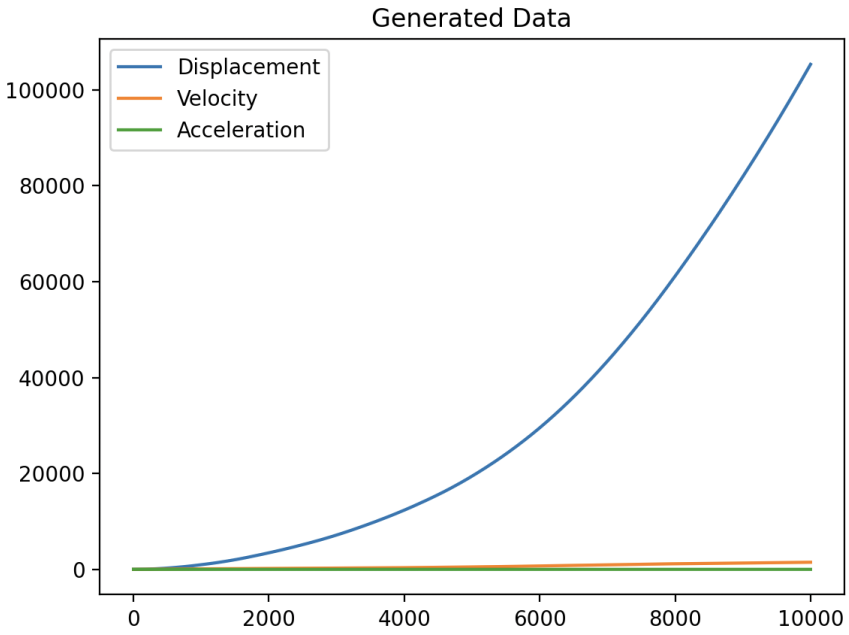


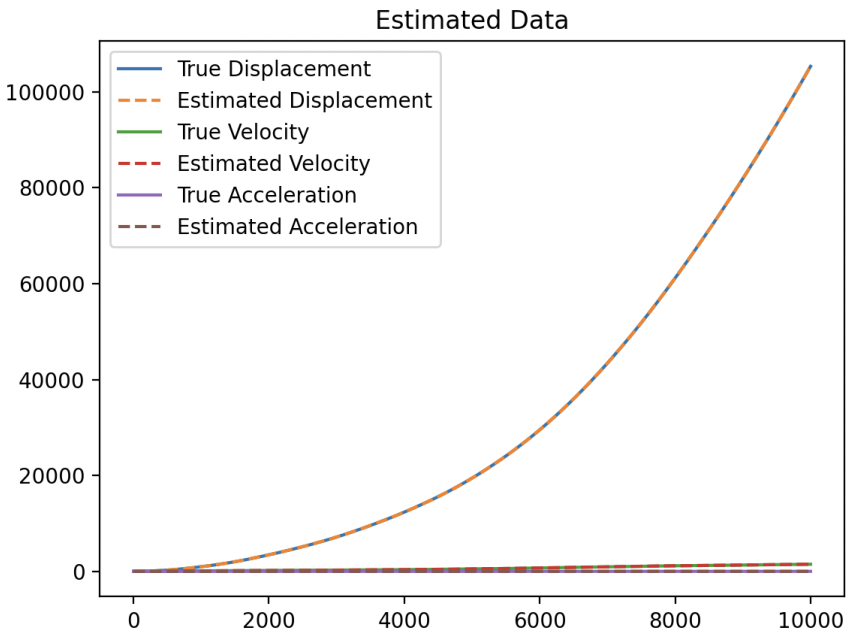


2.2 Example 2

$$A = \begin{bmatrix} 1 & 1/60 & 0 \\ 0 & 1 & 1/60 \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0.05 \\ 0.05 \\ 0.1 \end{bmatrix}, C = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, Q = \begin{bmatrix} 0.05 & 0.05 & 0 \\ 0.05 & 0.05 & 0 \\ 0 & 0 & 1 \end{bmatrix}, R = 0.5$$




Generated Data
Displacement
Velocity
Acceleration
100000
80000
60000
40000
20000
0
0
2000
4000
6000
8000
10000


Estimated Data
True Displacement
Estimated Displacement
True Velocity
Estimated Velocity
True Acceleration
Estimated Acceleration
100000
80000
60000
40000
20000
0
0
2000
4000
6000
8000
10000