NC STATE UNIVERSITY

Software project

Tanisha Khurana

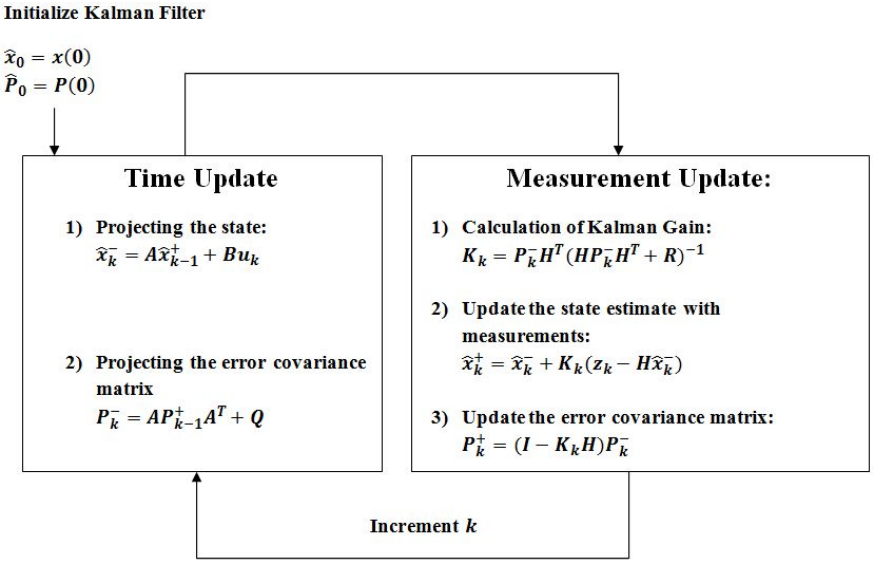
Date: 3/29/2023

ECE 751 Detection and Estimation theory

Background

- Kalman filter estimates the state of a system with incomplete or noisy information which is a gaussian noise.
- It uses the system model and the measurements to estimate the true state of the system and its uncertainty.
- In this project, we are implementing a discrete-time, time-invariant Kalman filter as a MATLAB function.
- For the benchmark case we are using the Kalman filter to estimate the position and velocity of a moving object based on noisy measurements of its position.
- Kalman filter will be used to make future predictions using past values to correct the estimated state.

Kalman filter prototype:



- The filter takes in a set of parameters and a sequence of K observations, and produces a sequence of state estimates, P matrices, and Kalman gain matrices.
- We first modeled the system as a set of equations that describe the relationship between the state of the system, the observations and the noise.
- We update the estimate of the state based on each new observation.
- The filter also calculates the uncertainty in our estimate, represented by the P matrix, and uses this information to adjust the gain of the filter for future observations.

Interface

Inputs:

N: scalar value representing the number of iterations.

F: NxN matrix representing the state transition model.

C: 1xN matrix representing the observation model.

G: Nx1 matrix representing the process noise model.

Q: scalar value representing the variance of the process noise.

R: scalar value representing the variance of the measurement noise.

X_hat_0: 1xN matrix representing the initial state estimate.

P_0: NxN matrix representing the initial error covariance matrix.

Y: MxN matrix representing the observed data.

Outputs:

X_hat: NxN matrix representing the estimated state.
P: NxN matrix representing the error covariance matrix.
K_hat: NxN matrix representing the Kalman gain.
P_mse: NxN matrix representing the mean squared error of the state estimate.

Interface

The function can be called as follows:

[X_hat, P, K_hat, P_mse] = kalman_filter_estimate(N, F, C, G, Q, R, X_hat_0, P_0, Y);

Here, N, F, C, G, Q, R, X_hat_0, P_0, and Y are the input arguments. The output arguments are X_hat, P, K_hat, and P_mse. Once the function is called, the values of the output arguments can be used for further analysis or visualization.

Benchmark case:

- The benchmark case is a simple tracking problem where we have a constant velocity model. The state vector contains two components - position and velocity.

- The dynamic model equations describe how the position and velocity evolve over time.

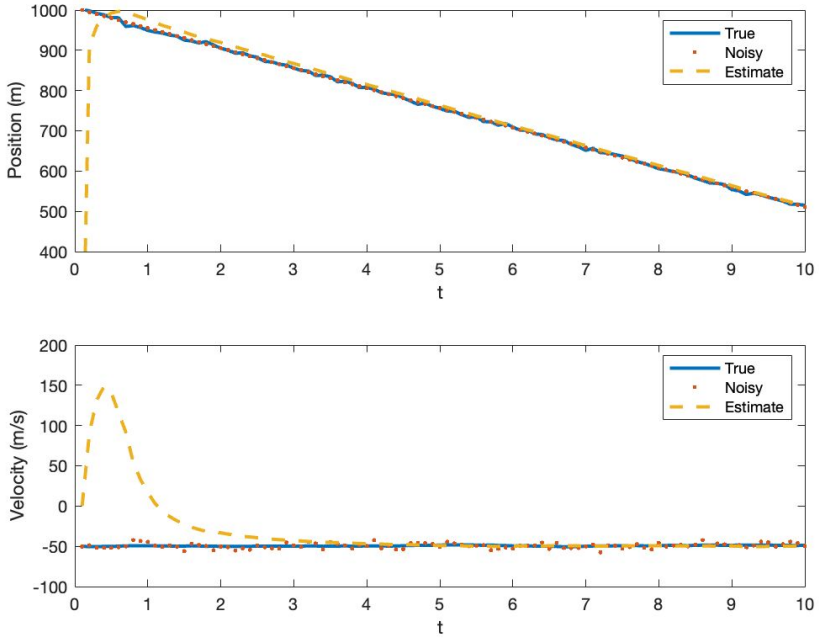
- The problem is to estimate the true trajectory of the object being tracked,given noisy measurements of its position. The benchmark case is often used to compare the performance of different tracking algorithms.

- For the benchmark example of the velocity motion model, I have a set of 100 measurements ranging from 1 to 10 with an interval of 0.1.

  Thus my T= 0.1, N =100 and my time ranges from t = (1:N)*T;

Benchmark results:

Below are the results on the right from the textbook (Detection, estimation, and modulation theory by Harry L. Van Trees) Example 9.10. And on the left are the results from the MATLAB code. We can see the results obtained from the code closely matches the one in the textbook.



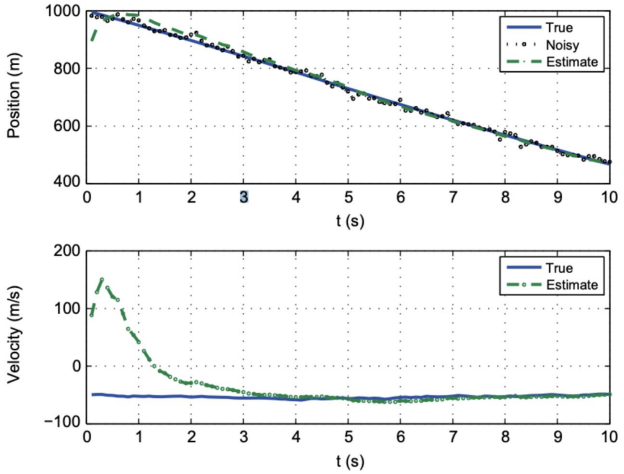
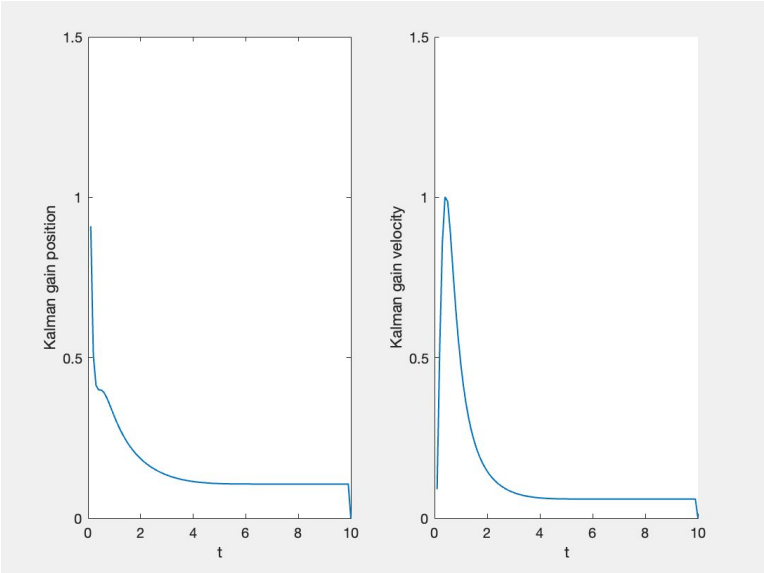


Figure 9.38: A realization of Kalman tracker with range-only observation.

Benchmark results:

Below are the results on the right from the textbook (Detection, estimation, and modulation theory by Harry L. Van Trees) Example 9.10. And on the left are the results from the MATLAB code. We can see the results obtained from the code closely matches the one in the textbook.



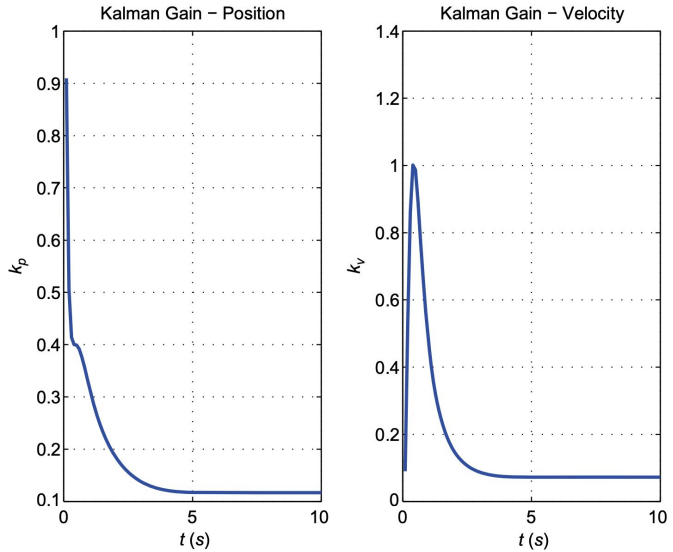
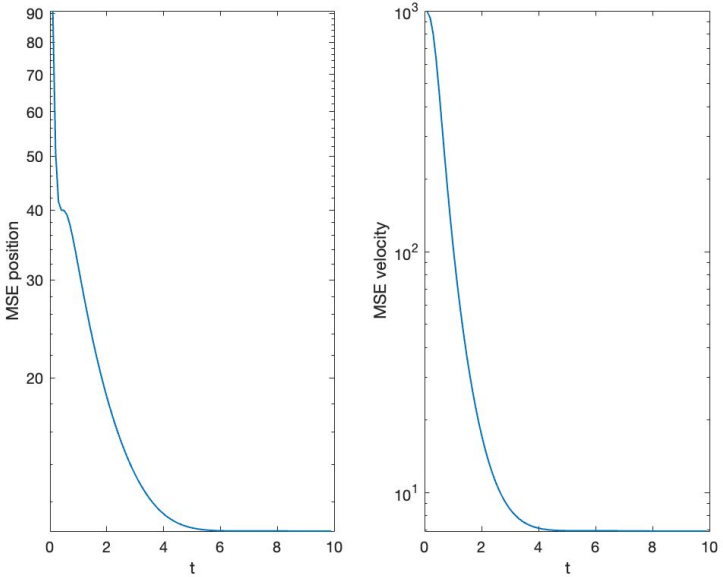


Figure 9.39: Kalman gain versus time.

Benchmark results:

Below are the results on the right from the textbook (Detection, estimation, and modulation theory by Harry L. Van Trees) Example 9.10. And on the left are the results from the MATLAB code. The results obtained from the do not exactly match the one in the textbook.This is because we are plotting the theoretical MSE (diagonal elements of P matrix) vs. time (like Fig. 9.40, y axis in log scale).



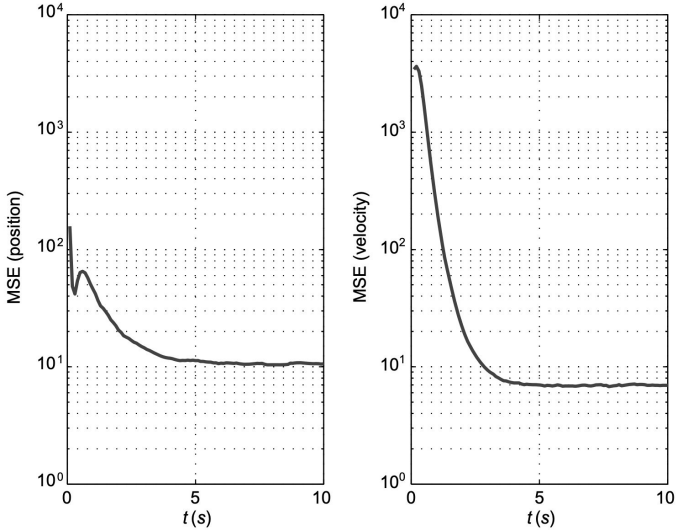


Figure 9.40: Mean-square error versus time.

Conclusion

- In conclusion, the software project required the implementation and testing of a Kalman filter prototype as a MATLAB function.
- The function takes in parameter settings and a sequence of K observations, producing a sequence of state estimates, P matrices, and Kalman gain matrices.
- The code was tested using the same benchmark case as in software project one, and plots were produced for the position and estimate, the Kalman gain vs. time, and theoretical MSE vs. time.