The Kalman Filter

Khoa Do

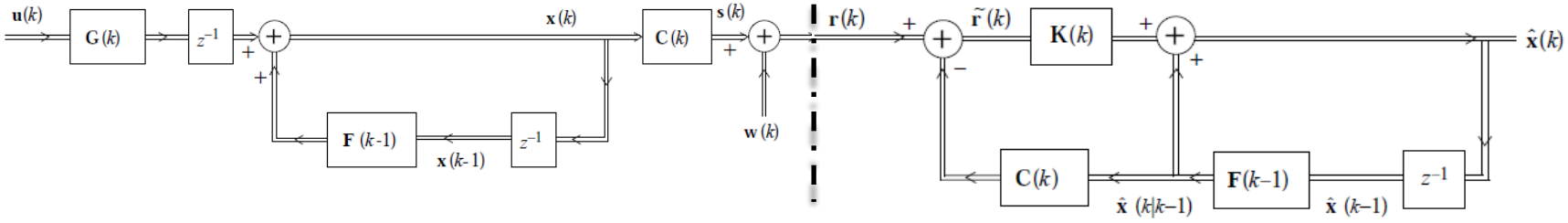
March 29, 2023

ECE 751 (Detection and Estimation Theory)

Background

- Kalman filter (KF)
  - One of the most popular filtering techniques
  - Wide range of applications, used network communication, signal processing, robotics, and more
- One outcome of ECE 751
  - Software project: developing and prototyping KF in MATLAB
- The project
  - Part I: developing state generator to generate states ("true" and noisy values) as the input to KF
  - Part II: developing discrete, time-invariant KF to estimate values closed to "true" values based on noisy values
  - Benchmarking both parts by running example 9.10 in [1] which models a one-dimensional motion (position and velocity over time)

KF Prototype

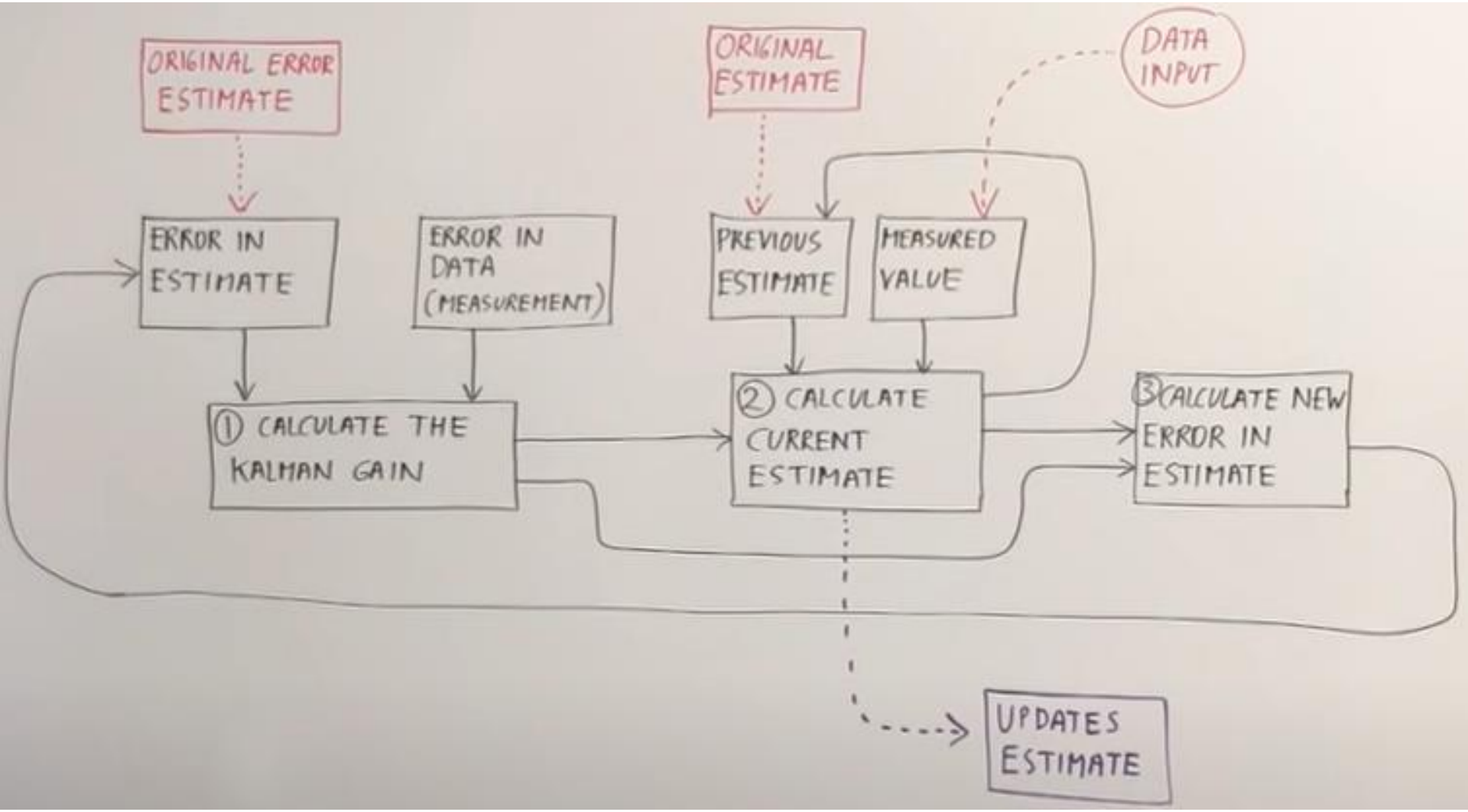


State Generator ***KF***

- Majority of part I kept (especially variables shown in above diagram), with few significant modifications and changes (*)(**)
  - *state_generator_input.m* => *kalman_filter_input.m* (file 1 or F1)
  - *state_generator.m* => *kalman_filter.m* (file 2 or F2)
  - F1: mainly for user-defined variables and plotting of returned values from function call in F2
    - Line 22-24: *x_est*, *Kg_plot*, *MSE_theo* initialized to hold returned values from F2
    - Line 29: function call *state_generator* changed to *kalman_filter*
    - Line 34-49: estimated values added to plots for position and velocity, axes scaling adjusted
    - Line 51-79: Kalman gain, theoretical MSE plots added for position and velocity, axes scaling adjusted
  - F2: state values generation and KF algorithm
    - Line 1: function adjusted to fit with function call from F1
    - Line 9, 10: state estimate vector and P matrix initialized with specific values for benchmarking
    - Line 22-32: equation 9.298-9.9304 from [1] added to for loop for prediction stage and Kalman gain calculation of KF
    - Line 36-39: results for estimates, Kalman gain, theoretical MSE of both position and velocity stored and returned to F1 for plotting (matching with line 22-24 of F1)

(*) Refer to part I's report for notations and details
(**) Refer to MATLAB codes for detailed explaining comments

- Summary of main steps done in part II of the project [2] (mainly matching line 22-32 of F2 shown in slide 3



Interface

- Interfacing using MATLAB – inputs, outputs, and function call (F1 calls F2)

```
%----------------------------setting up variables, inputs from user----------------------------%
T = 0.1;                        % sampling interval, value 9.396 from book
N = 100;                        % no. of iterations, value from user (100 so that plots on same scale as in book fig 9.38)

sigma2_a = 40;                  % process (plant) noise's variance, value 9.395 from book
sigma2_r = 100;                 % measurement noise's variance, value 9.397 from book

F = [1 T; 0 1];                 % state transition matrix, equation 9.386 from book
G = [T^2/2; T];                 % input (control) transition matrix, equation 9.387 from book
C = [1 0];                      % output transition matrix, equation 9.390 from book

x_prev = [1000; -50];           % initialize [position(0); velocity(0)], values 9.393 & 9.394 from book
%----------------------------end---------------------------%


%----------------------------initilize variables for plotting----------------------------%
x_true = zeros(size(F,1), N);           % initialize matrix F_row-by-N to hold "grounth-truth" state vector values of postion and velocity with noise
x_noisy = zeros(size(G,2), N);          % initialize matrix G_column-by-N to hold position measurements (with noise)
x_est = zeros(size(F,1), N);            % initializw matrix F_row-by-N to hold estimated values of position and velocity
Kg_plot = zeros(size(F,1), N);          % initialize matrix F_row-by-N to hold Kalman gain values of postion and velocity
MSE_theo = zeros(size(F,1), N);         % initialize matrix F_row-by-N to hold theoretical MSE for position and velocuty, which is diagonal elements of P matrix
%----------------------------end---------------------------%


%----------------------------call function to generate state values----------------------------%
[x_true, x_noisy, x_est, Kg_plot, MSE_theo] = kalman_filter(N, sigma2_a, sigma2_r, F, G, C, x_prev);     % return values for plotting
%----------------------------end---------------------------%
```

F1 (kalman_filter_input.m)

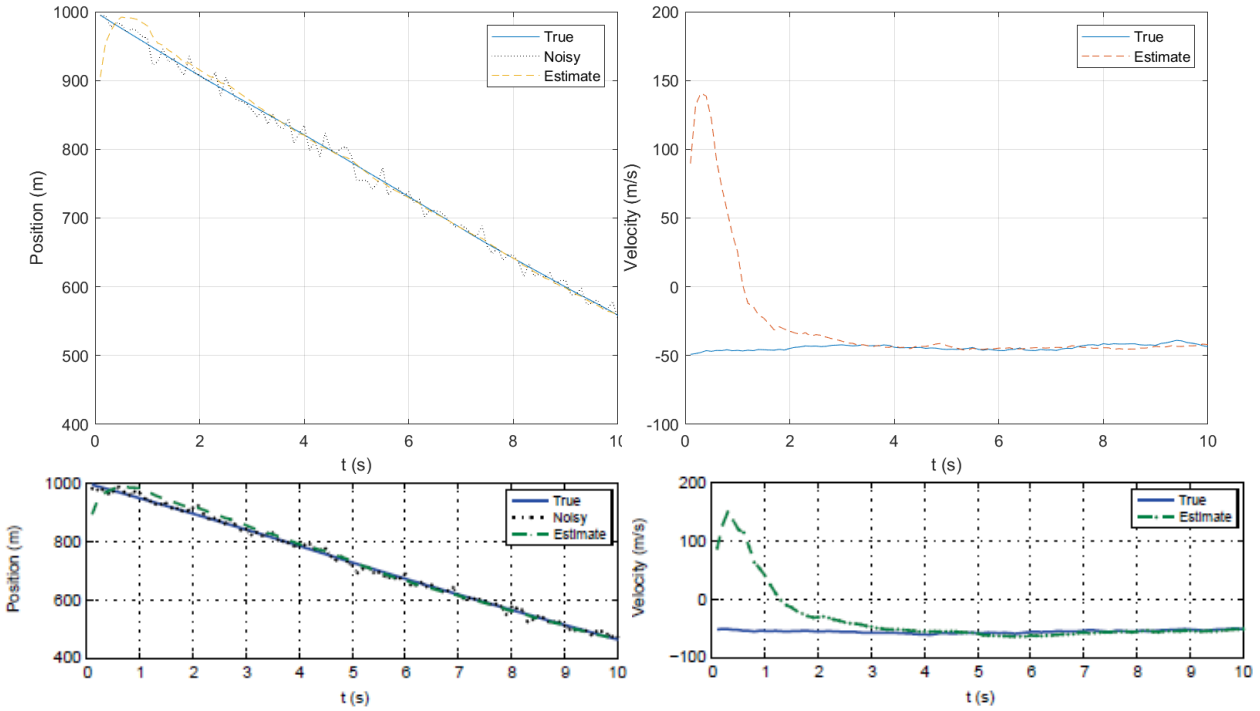
```
function [x_true, x_noisy, x_est, Kg_plot, MSE_theo] = kalman_filter(N, sigma2_a, sigma2_r, F, G, C, x_prev);
```

F2 (kalman_filter.m)

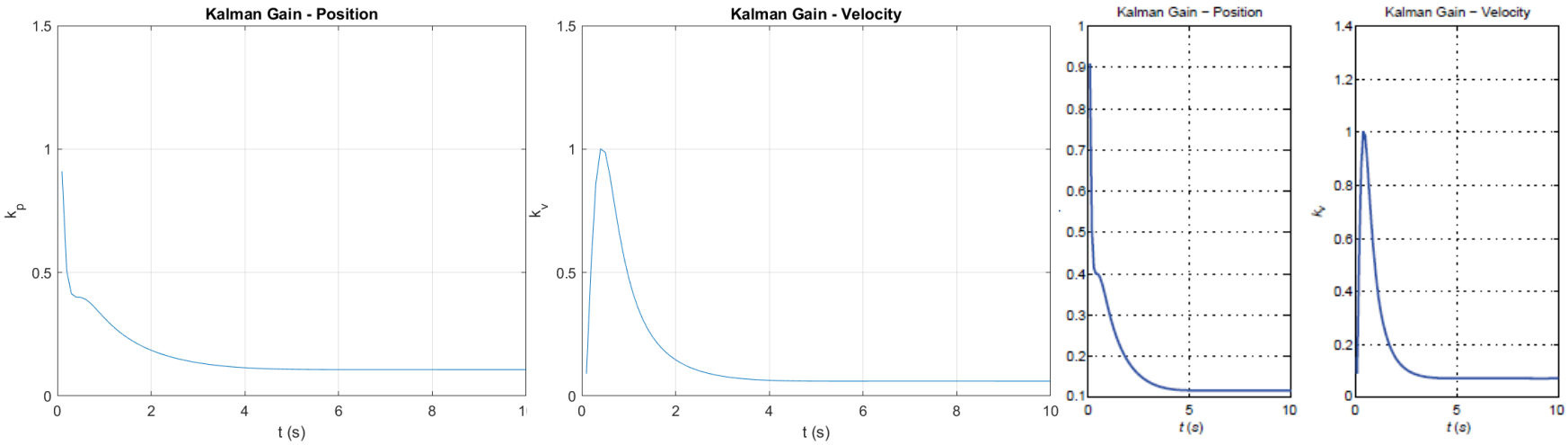
Benchmark Case

- Running MATLAB code on example 9.10 in [1], comparing to textbook plots
  - One-dimensional motion model with roughly constant velocity, meaning
    - Position is changed over time
    - Velocity is roughly constant over time
    - Acceleration is roughly 0 and can be omitted from benchmark
    - Numerical initialization (refer to line 5-15 of F1 in slide 4 or directly from example 9.10 in [1])
  - State generator generates “true” and noisy values for BOTH position and velocity (P&V)
    - “True”: ground-truth values + process noise
    - Noisy: measured values + measurement noise
  - KF predicts/estimates values closed to “true” values based on noisy values as the input
  - Plotting for observation and analysis: “true”/noisy/estimated + Kalman gain + theoretical MSE for P&V over time

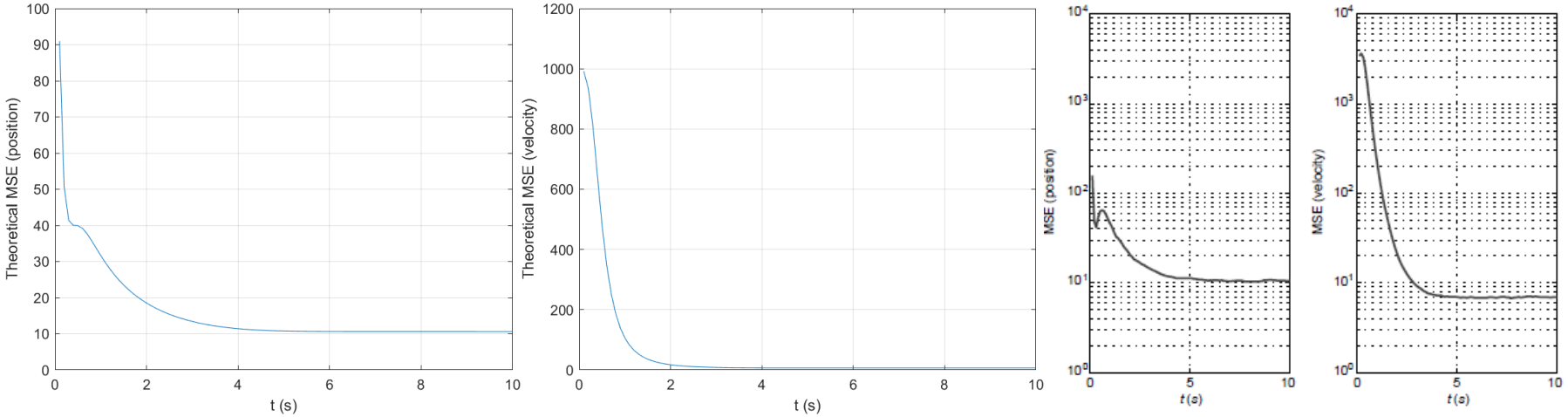
Benchmark Results



- "True," noisy, and estimated P&V over time
  - Benchmark results (top row) look similar to [fig. 9.38] textbook results (bottom row) in terms of shape and trend lines though exact values seems to be slightly different due to different random seeds for noise used
  - Position changes linearly and velocity roughly stays constant



- Kalman gain of P&V over time
  - Benchmark results (first two plots from left) look similar to [fig. 9.39] textbook results (last two plots) in terms of shape and trend lines though exact values seems to be different due to different random seeds for noise used at the beginning
  - Kalman gain decreases over time, meaning high uncertainty in measurements (noisy values or data) => higher error in data => more weight on the estimate for new estimates



- Theoretical MSE of P&V over time
  - Benchmark results (first two plots from left) look similar to [fig. 9.40] textbook results (last two plots) in terms of shape and trend lines though exact values seems to be different because theoretical MSE is used in the benchmark (diagonal elements of P matrix) as compared to the textbook, which uses instantaneous MSE
  - MSE decreases over time for both position and velocity
    - Position: converging to about 10 meters, which seems low mathematically but huge in practice due to noisy measurements
    - Velocity: converging to 0 m/s which is ideal due to roughly constant velocity

Conclusion

- Project done in MATLAB
  - Part I: developing state generator to generate states (“true” and noisy values) as the input to KF
  - Part II: developing discrete, time-invariant KF to estimate values closed to “true” values based on noisy values
  - Benchmarking both parts by running example 9.10 in [1] which models a one-dimensional motion (position and velocity over time)
  - Comparing benchmark results to ones from textbook
- Insights from benchmark results
  - Working KF prototype as benchmark results are similar to textbook results
  - Simple KF implementation shown to be efficiently estimating technique
- General insights
  - KF is a powerful filtering technique beside Particle Filter
  - Derivations such as UKF, EKF, PKF, and many others for more complicated estimating problems

Reference

[1] V. T. H. L., K. L. Bell, and Z. Tian, Detection estimation and modulation theory. Oxford: Wiley-Blackwell, 2013.

[2] “Special Topics - The Kalman Filter” by Michel van Biezen.