Software Project Two Assignment

Background

One learning outcome of this class is the ability to develop and prototype a Kalman filter. In software project two, you will develop and test a Kalman filter. For this project, you will do your own work.

Project description

Implement the discrete-time, time-invariant Kalman filter as a MATLAB® or Python function. The function should take in parameter settings as well as a sequence of *K* observations. It should produce a sequence of state estimates, P matrices, and Kalman gain matrices.

The function should produce state estimates and P matrices for index values 0, 1,….K. The Kalman gain matrices should be produced for index values 1, 2 … K. To keep the time indices straight and simplify plotting, you may want to also output a list of times, e.g. [0,T,2T, etc.] for the state estimates and P matrices, and [T,2T, etc.] for the Kalman gain matrices. Note: you may want to modify your signal model if it doesn't provide a true state value at time 0 or if it doesn't provide a list of times.

Test your code for the same benchmark case as in software project one. Note that the textbook has a mismatch between what the KF should use to initialize xhat and P (m0 and Pi0) and what it actually uses in equations (9.398) and (9.399). Thus, your code should have additional inputs so you can model this mismatch. In addition, there is a typo in (9.398) in the textbook; the state estimate vector should be initialized to [0 0], not [1100 -100].

Specifically, run your signal model and Kalman filter for example 9.10 in textbook (only position is observed and produce two plots similar to Fig. 9.38 (this time INCLUDING the curve for "Estimate"). In MATLAB®, use "subplot()" to get both plots together.

Also produce a plot of the Kalman gain vs. time (like Fig. 9.39), except use a common y-axis range of 0 to 1.5. For this example, the Kalman gain is a 2x1 matrix, where the first element corresponds to how the scalar error (position error) is used to updated the first state (position) estimate and the second element corresponds to how the scalar error updates the second state (speed) estimate.

Also plot theoretical MSE (diagonal elements of P matrix) vs. time (like Fig. 9.40, y axis in log scale). NOTE: Fig. 9.40 in the textbook is measured (instantaneous) square error, not theoretical MSE, so your result will look different. Use the same y axis range for both plots, similar to the book.

Project deliverables

Provide a short presentation (e.g. Powerpoint) in PDF that includes the following slides.

1. Title slide: Provide a title, your name, date, and name of this course
2. Background: Restate the problem in your own words, using bullet items
3. Kalman filter prototype: Explain what you did. A block diagram would be hepful.
4. Interface: Provide an example function call and list of inputs and outputs and what they mean
5. Benchmark case: describe benchmark case

6. Benchmark results: Plot like Fig. 9.38 and comment on how they compare to textbook results (recommend using `subplot` in MATLAB® to get 2 plots together)
7. Benchmark results: Plot like Fig. 9.39 and comment on how they compare to textbook results (make y axis of both plots go from 0 to 1.5)
8. Benchmark results: Plot like Fig. 9.40, except theoretical MSE instead of measured, and comment on how they compare to textbook results
9. Conclusion. Summarize what you did and provide any insights.

In a separate file, provide the source code function(s) and script(s) you used to generate the results. For MATLAB® this could be one .m file (script + functions) or a zipped folder of files. For Python using a Google Colab notebook, upload the .ipynb file. In both cases, also provide a readme.txt file explaining how to run your code.

You need only provide what is asked for. The presence or absence of additional work has no grade impact.

Grading rubric: if you follow the instructions and provide all that was asked for in a clear manner, you get 100%. Points will be deducted if components are missing or wording is unclear.

Recommendations:

- In MATLAB® the expression $\mathbf{BA}^{-1}$ can be implemented as B*inv(A) or B/A. The latter is numerically preferred.
- You may want to store the P and Kalman gain matrices in a 3-D array (row, col, time). To plot an element vs. time you may need to use the reshape command in MATLAB®.