ECE 751 Software Project 1

Project Partner: Naga Suryadevara

March 5, 2021

1 Background

This project issued the challenge of implementing the descrete-time, time-invariant Kalman signal model into a MATLAB function. This was used to help further the understanding of the course goals to have the ability to develop and prototype a Kalman filter by the end of this course. The function needed to output two sequences both of N length. The first sequence would have N_r measurements and the second sequence would have N_x measurements. Another parameter was the input of the function would have to be set in order to test the function of the filter.

2 Interface Design

To begin with this problem, the model needed to be chosen to replicate within MATLAB. Based on the instructions, it required a dynamic Kalman model since the goal was to track trajectory. There were two leading equations that had to be implemented within the function.

$$x(k) = F * x(k-1) + G * u(k-1), k = 1, 2, 3... \tag{1}$$

$$r(k) = C * x(k) + W(k), k = 1, 2, 3... \tag{2}$$

For this project, the function is called by using the DiscreteKalmanModel that was created within MATLAB. There are two outputs and 7 inputs that this function requires. These inputs are specific for this problem itself. Example of a function call: [x_{in} noise] = DiscreteKalmanModel(F, x_true, G, Q, R_w, C, Nmax)

For the inputs, we have the following in order that they are listed:

1. F: A 2x2 matrix that is dependent on T, the time period.
2. x_{true}: This is the initial values of x(0). In this instance, it is a 2x1 vector with the first row representing initial position and the second row being the initial velocity.
3. G: A 2x1 vector that is dependent on T. It applies the effect of the noise to the state vector.

4. Q: This is the variance of a normal distribution. For this function, it is treated as some scalar value.

5. R_w: This value is the variance of the velocity estimate. In this problem, it is treated as a scalar.

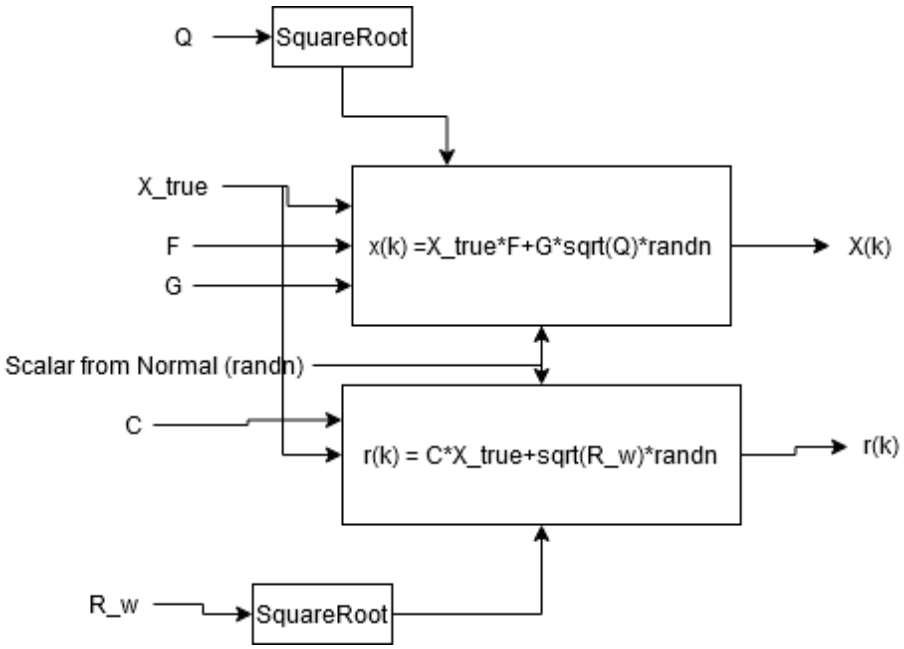
6. C: A vector of observable values. In this problem, the observable values are either 0 or 1, and a 1x2 matrix of these values are used.

7. Nmax: This is the total number of time steps the loop will go through.

The outputs are listed in order of appearance as follows:

1. x_{in}: This is a 2xN matrix that contains the true values of the position and velocity based on equation 1. Row 1 contains the position measurements and row 2 contains the velocity measurements.

2. noise: This is a single row vector of length 1xN that contains the noisy position data only. It is calculated using equation 2.



3 Benchmark Example

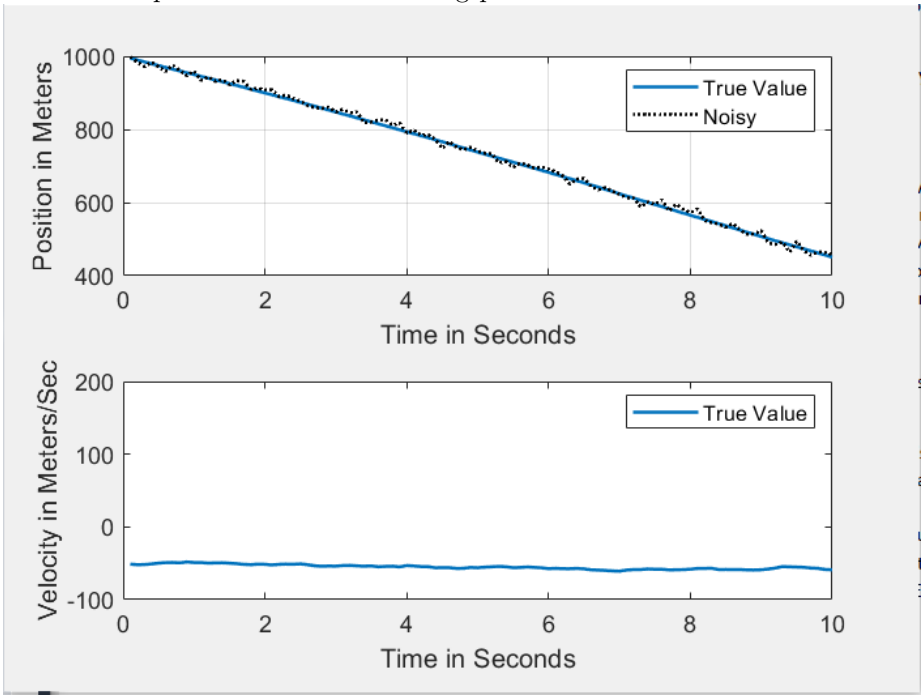
For this benchmark example, the parameters were chosen according to those found in example 9.10 found in the textbook, Detection, Estimation, and Modulation Theory Part 1 [1]. The values for the parameters were as follows:

1. F=$\begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$

2. x_{true}= $\begin{bmatrix}1000\\-50\end{bmatrix}$ The initial position is 1000 meters and intial velocity is -50 m/s.
3. G= $\begin{bmatrix}\mathrm{T}^2/2\\T\end{bmatrix}$
4. Q= 60
5. R_w=100
6. C$\begin{bmatrix}1\\0\end{bmatrix}$
7. Nmax= 100
8. T = 0.01

The definition of these values are available in the previous section. The only new value is T which represents the time period of each step. The final value within the function was some random noise. This was added using MATLAB's randn function which returns a random scalar drawn from a standard normal distribution.

Once these parameter settings were added, the function was called to produce the output vectors. The following plots show the results from the code.



We can see from the results that the velocity remains roughly constant throughout the process with a little bit of bounce. The first graph shows what we expect

in terms of position with constant negative velocity. The predicted position is going down towards zero over time but there is some slight noise that may cause the predicted position to not quite be perfect.

Now, we can look at my partner's function output to confirm this benchmark example. Theoretically, our plots and outputs should look very similar. Here are my partner's plots:

4 Conclusion

The project was successful and the results resembled those found in example 9.10. By building a MATLAB function, this process was condensed into a single loop that allowed us to calculate the new position value as well as the noise. We could also see that the velocity plot appears to be roughly constant as expected.

These results were further compared to my partner's plots and outputs. Although some areas are slightly different, this is due to the noise not following the same seed value so the random noise is truly random.

5 Appendix

Function Code:

```
function [x_in,noise] = DiscreteKalmanModel(F, x_true, G, Q, Rw, C, Nmax)
%ECE 751 Software Project 1 Luke Martin
%Inputs:
%F: A 2x2 Matrix that is [0 T; 0 1] where T is some discrete time
%x_true: A 2x1 matrix of initial distance and velocity values.
%e.g. x_true = [1000;-50]
%G: A 2x1 matric of values: [T^2/2; T]
%Q: Variance of a normal distribution, some scalar. e.g. 40
%Rw: Variance of a scalar, 0-mean discrete white process. e.g. 100
%C: 1x2 matrix of observable ranges. e.g. [1 0]
%Nmax: Number of steps. NOTE: x_in and noise should be initialized the same
%way both in the function and before the call to ensure no data is lost.

%Outputs:
%x_in: This is a 2xN matrix. Row 1 contains the true Position values while
%Row 2 contains the True Velocity Values.
%noise: This is a 1xN matrix. It contains only the Noisy Position values and should be plotted with Row 1 of x_in
x_in = zeros(2,Nmax);
noise = zeros(1,Nmax);
for n = 1:Nmax
    x_true = F * x_true + G * sqrt(Q) * randn;
    r = C * x_true + sqrt(Rw)*randn;

    x_in(:,n) = x_true;
    noise(n) = r;
end

end
```

Test Code used for creating Plots/testing Functions:

```
%%
%ECE 751 Software Project 1
%Luke Martin

%Kalman Filtering

%Parameters according to 9.10
%Equations following that of example 9.10
clear all
close all
%Initial Parameters
T = 0.1;

sigma_a = 60;
sigma_r = 100;

F = [1, T; 0 1];
G = [T^2/2; T];
C = [1 0];

Q = sigma_a;
Rw = sigma_r;

%Initial Values
x_true = [1000; -50];
x = [0; 0];
P = 10*Rw;
%For plotting and stuff
Nmax = 100;

Nt = [1:Nmax]'*T;

x_in = zeros(2,Nmax);
noise = zeros(1,Nmax);

[x_in, noise] = DiscreteKalmanModel(F, x_true, G, Q, Rw, C, Nmax);
%Plot the appropriate figures like in the textbook (9.38)
figure
set(0,'DefaultLineLineWidth', 1.5, 'DefaultAxesFontSize',12, 'DefaultTextFontSize',12)
subplot(2,1,1)
plot(Nt, x_in(1,:), Nt, noise,':k')
legend('True Value','Noisy')
xlabel('Time in Seconds')
ylabel('Position in Meters')
grid
subplot(2,1,2)
plot(Nt, x_in(2,:))
axis([0, 10, -100, 200]);
legend('True Value')
xlabel('Time in Seconds')
ylabel('Velocity in Meters/Sec')
```

6 References

[1] Van, Trees, Harry L., and Kristine L. Bell. Detection Estimation and Modulation Theory, Part I : Detection, Estimation, and Filtering Theory, John Wiley

Sons, Incorporated, 2013. ProQuest Ebook Central,
https://ebookcentral.proquest.com/lib/ncsu/detail.action?docID=1166811.