Lab Report A6
CS4300
Derek Heldt-Werle 2,4,6
Matthew Lemon 1,3,5

11/17/16

1. Introduction

For this Lab we developed several functions namely a Process model which models Linear or Projectile trajectory, a Sensor model which is used to sample the process model's position at certain time intervals , and a Kalman Filter which we used to track the trajectories.

With a higher amount of noise does the function produce a more jagged trajectory model? Does it take longer for the Kalman Filter to find the actual when the noise in the y is high?

2. Method

In this lab we began by creating the state transition matrix, the control matrix for the process equation, the linear transform for measurement equation matrix, the control vector, the state transition covariance matrix, and the sensor noise covariance matrix. By setting these up we are able to calculate the process, sensor, and Kalman Filter. By having our A, B, C, u , R, and Q matrices we can simply iterate over the number of given time steps and create the the actual, sensor, and Kalman filter graphs. We begin the loop by calling our process function that calculates x^t+1 through the use of the current xt and the A, B, and u matrices (`xtplus1 = A*xt + B*u + noise;`). We then add in noise (`sqrt(val)*randn;`) to each of the values to be returned. A similar approach is taken to accomplish our sensor function, but the returned value from our process is used as the input and multiplied by the linear transform for measurement equation with noise added in at the end (`xz = C * xa + noise;`). We finally accomplish the calculation of the Kalman filter under the strict guide of the algorithm found in Peter Norvig and Stuart J. Russell's book Artificial Intelligence: A Modern Approach.

3. Verification of Program

To verify the functionality of the CS4300_Sensor I ran it 1000 times at point [0,0] and it produced this set of data

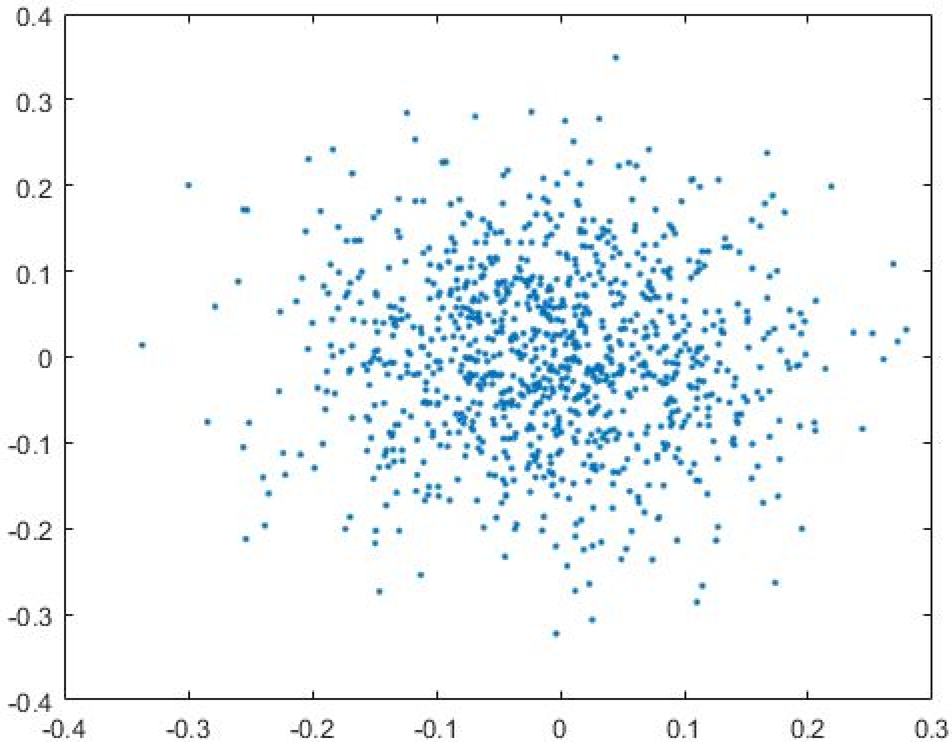

Figure 0-1: Sensor Data around [0,0]

As can be seen in Figure 0 CS4300_Sensor produces a Gaussian distribution centered around [0,0] as was expected.

To verify CS4300_Process I ran the function without noise which should produce a very expected set of data (e.g. A linear line in the linear case whose slope is the yv/xv). As can be seen in Figure 0-2 when running the Process with no noise on a linear model we get a linear line as expected

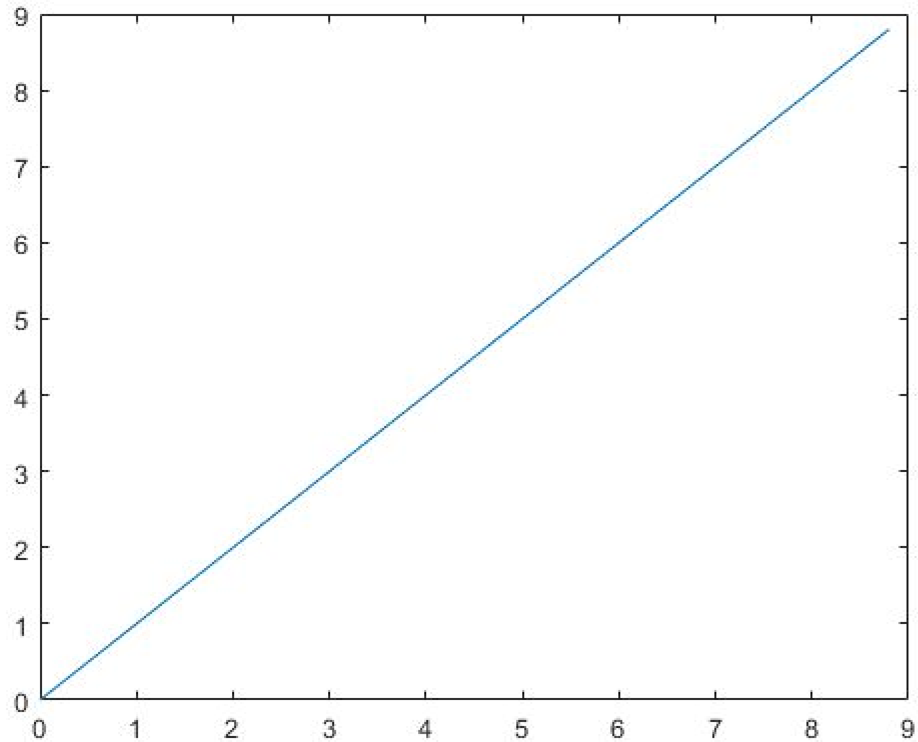

Figure 0-2: Linear Process model no noise

4. Data and Analysis

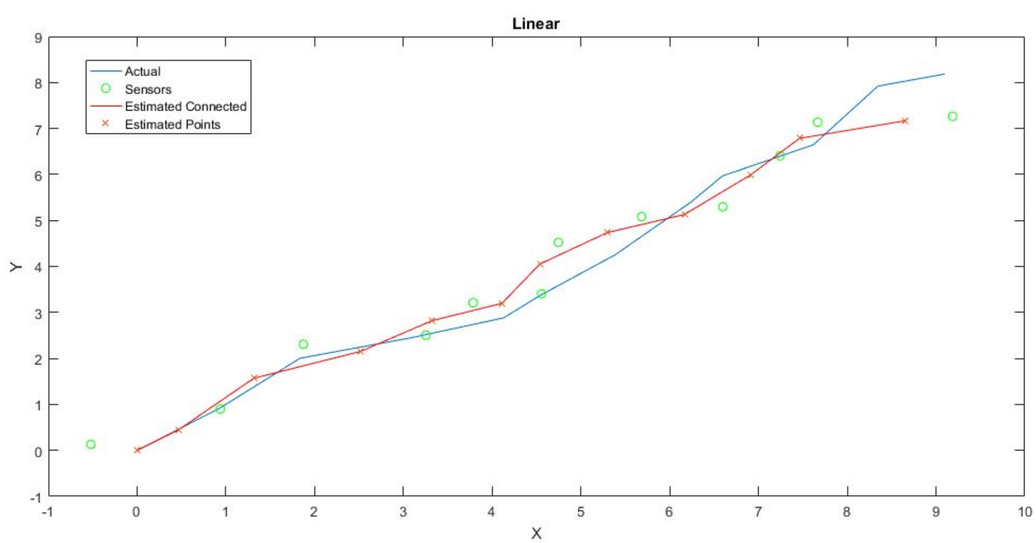

Figure 1: Results of Linear Trajectory KF

(Call CS4300_A5_driver_lin(0,0,8,8,1,0.1,10) with u = [0;0] R = 0.1 * eye(4) Q = 0.1 * eye(4))

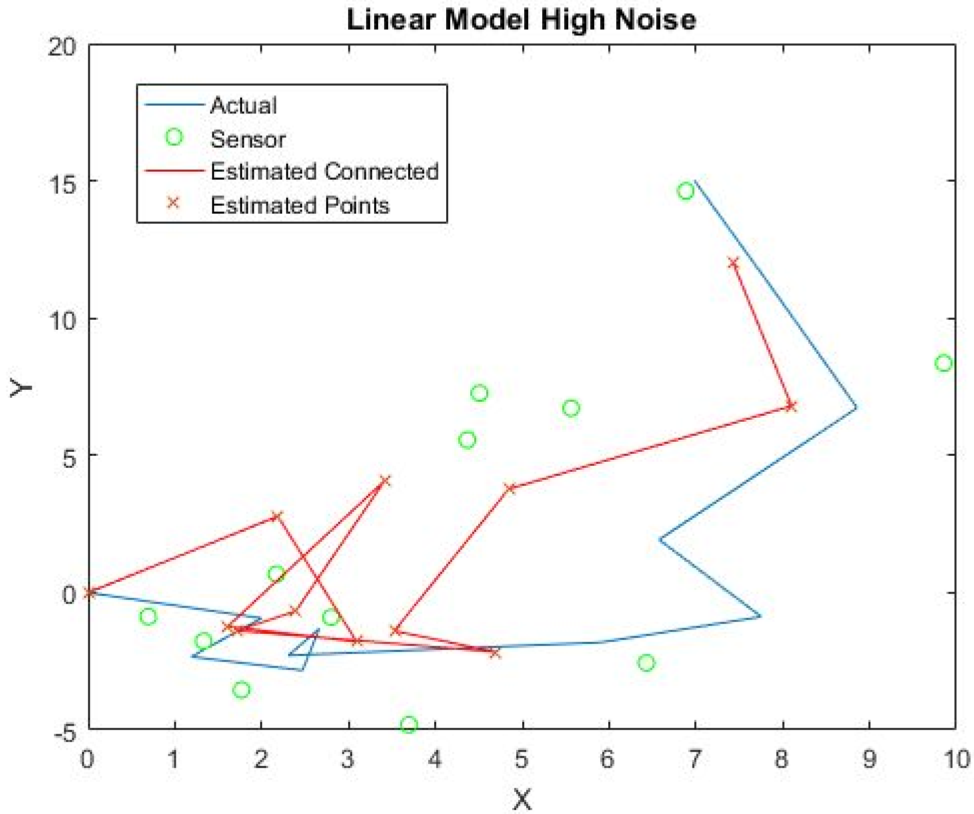

Figure 1-1: Linear Model with High Noise

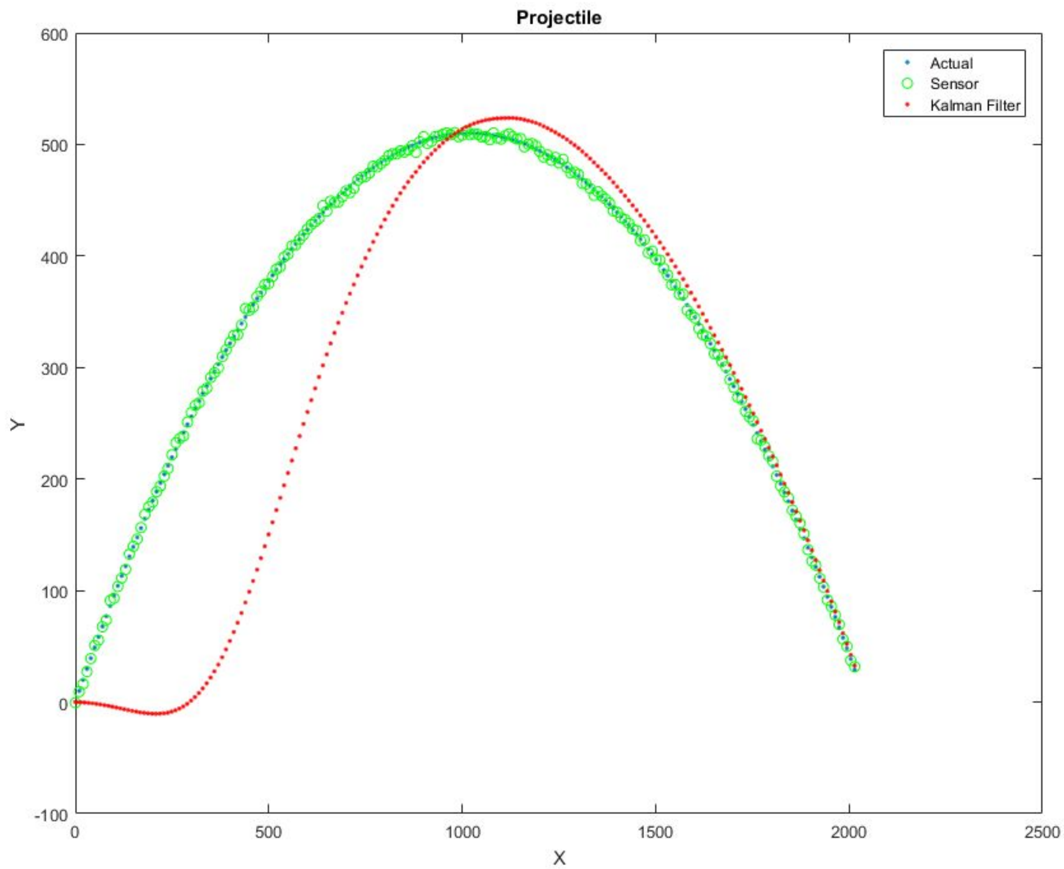

Figure 2: Results of Projectile Trajectory KF

(Call CS4300_A5_driver_lin(0,0,100,100,20,0.1,10) with u = [0;-9.8] R = 0.001 * eye(4)

Q = 0.001 * eye(4)

Q(2,2) = 6)

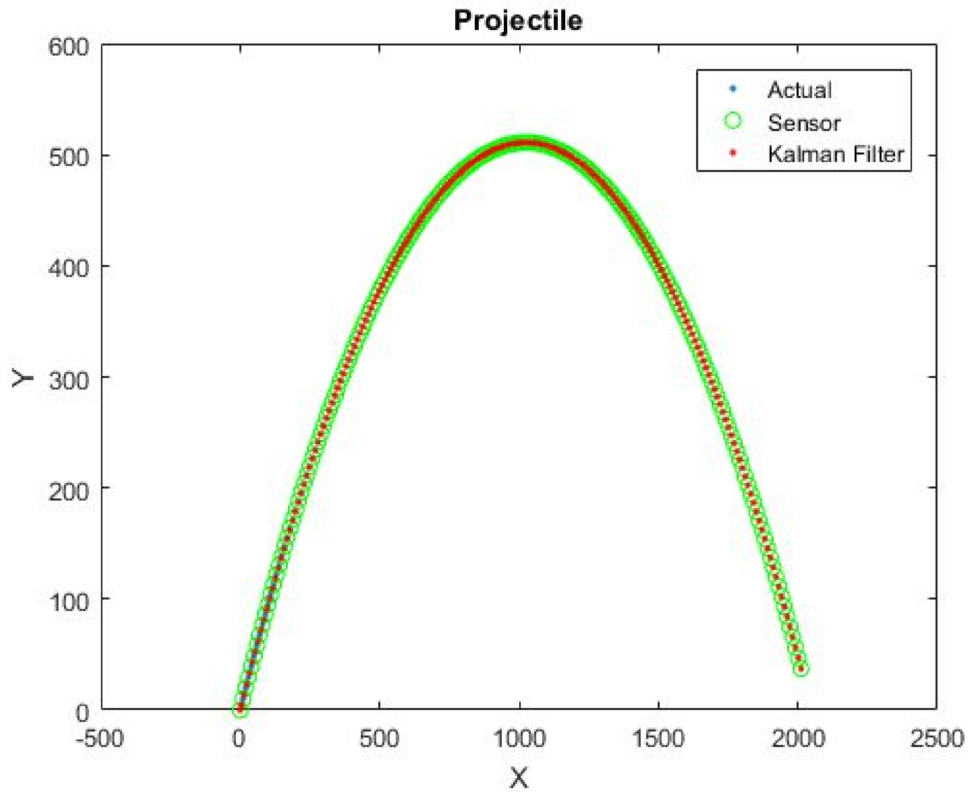

Figure 2-1: Projectile low noise

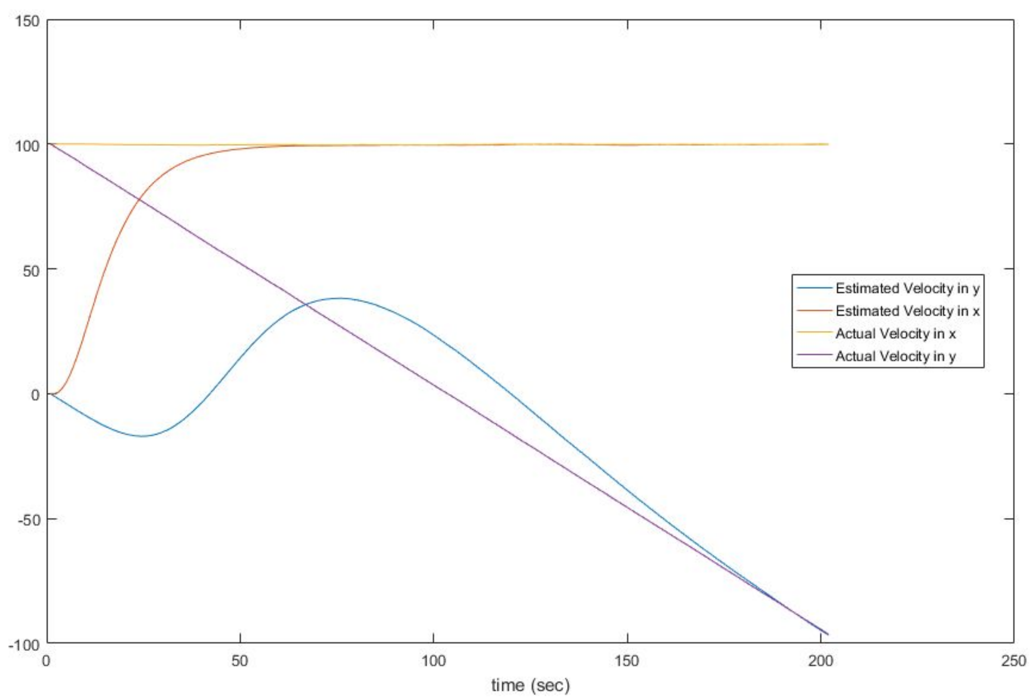

Figure 3: Results of Projectile KF Velocity Tracking.

5. Interpretation

For the projectile model displayed in Figure 2 we set the noise in the y to be 6 and in Figure 2-1 we set the noise to be 0.001. As can be seen between the two graphs given a larger noise in the y it takes the Kalman Filter longer to zero in on the actual.

We set the noise for the linear process model to be 10*eye(4) for the process noise and 10*eye(2) for the sensor noise. As can be seen in Figure 1-1 when given a large amount of noise we produce an extremely erratic set of data.

6. Critique

During this lab we both learned a great deal about how to create a valid process, sensor, and kalman filter model. Prior to this, neither of us had any knowledge of what any of those entailed, so it was a great learning experience. Through the use of the fall model sensor provided to us we were able to have a solid starting point to begin the proper modelling of both the linear and projectile case. This assignment did a great job at introducing us to Kalman Filter and state estimation, and provided us with just enough information to force us to learn the material without it taking too long to understand.

7. Log

Author Matthew Lemon & Derek Heldt-Werle
Coding Portion (Worked together): 12
Report (Derek): 2
Report (Matt): 2