Kristen Biermayer & Rebecca Schwartz
Professor Mark Mueller
MECENG 231b Experiential Advanced Controls II
4 May 2020

State Estimation Project

**What we Implemented**

We implemented an EKF as our state estimator to track the position and heading of a bike as it moved. We tried implementing both the EKF and UKF because they are both filters for nonlinear systems. We decided to use the EKF because it was computationally faster. We used the bicycle dynamics given to us and discretized them as follows:

q(x(k-1),v(k-1)) = [x1(k-1) + vel(k-1)*cos(x3(k-1))*dt + v1(k-1);
x2(k-1) + vel(k-1)*sin(x3(k-1))*dt + v2(k-1);
x3(k-1) + (vel(k-1)/B)*tan(gamma(k-1))*dt + v3(k-1)]

h(x(k),w(k)) = [x1(k)+ 0.5*B*cos(x3(k)) + w1(k);
x2(k)+ 0.5*B*sin(x3(k)) + w1(k);
x3(k) + w3(k)];

Here x1 is the x-position, y1 is the y-position, x3 is the angle theta, dt is the timestep, v1 (and v2 and v3) are the expected process noise, B is the length of the wheelbase, and w1 (and w2 and w3) are the expected measurement noise.

The Jacobian matrices A, H, M, and L were calculated as shown in the main function, estRun.

```
A = [1, 0, 0;
     0, 1, 0;
    -vel*dt*sin(Xmm(3)), vel*dt*cos(Xmm(3)), 1];

L = eye(3);

H = [1, 0, 0;
     0, 1, 0;
    -0.5*B*sin(Xmm(3)), 0.5*B*cos(Xmm(3)), 1];

M = eye(3);
```

**Design Decisions**

Initialization:

It was given that the initial positions of x and y would be near the origin, (0,0), so for the initialization function, 0 and 0 were chosen for x and y, respectively. It was given that theta

would be approximately north-east, so it was decided that the initial theta would be pi/4 north-east.

```
internalState.x = 0;
internalState.y = 0;
internalState.theta = pi/4; % rad, 'North East init'
```

Variances:

The initial variance at time zero, P0, was computed as:

```
P0 = diag([var_r + var_B, var_r + var_B, var_B]);
```

where var_r is the variance of r, the tire radius, which is computed to be gaussian and where var_B is the variance of B, the wheelbase length, which is also computed to be gaussian.

The rest of the initial variances, P0, for each time step are calculated to be Xmv, the measured variance, corresponding to when there was last a set of measurements given.

The variances for the sensor and process noise were chosen to be 0.1, given the myriad of uncertainties not defined.

Theta:

Since a measurement for theta was not given, but it is part of the state, theta was calculated to be the inverse tan of the y-measurement divided by the x-measurement.

```
theta_meas = atan(y_meas/x_meas);
```

When using the dynamics given for theta, the final error was -2.1 radians for run #1. To improve this, we changed the theta output to the inverse tan of the change in the y-position divided by the change in x-position. This adjustment improved the error for x and y slightly and improved the error of theta to be 0.5 radians.

```
theta = atan((internalStateOut.y -internalStateIn.y) / (internalStateOut.x-internalStateIn.x));
internalStateOut.theta = theta;
```

**Results**

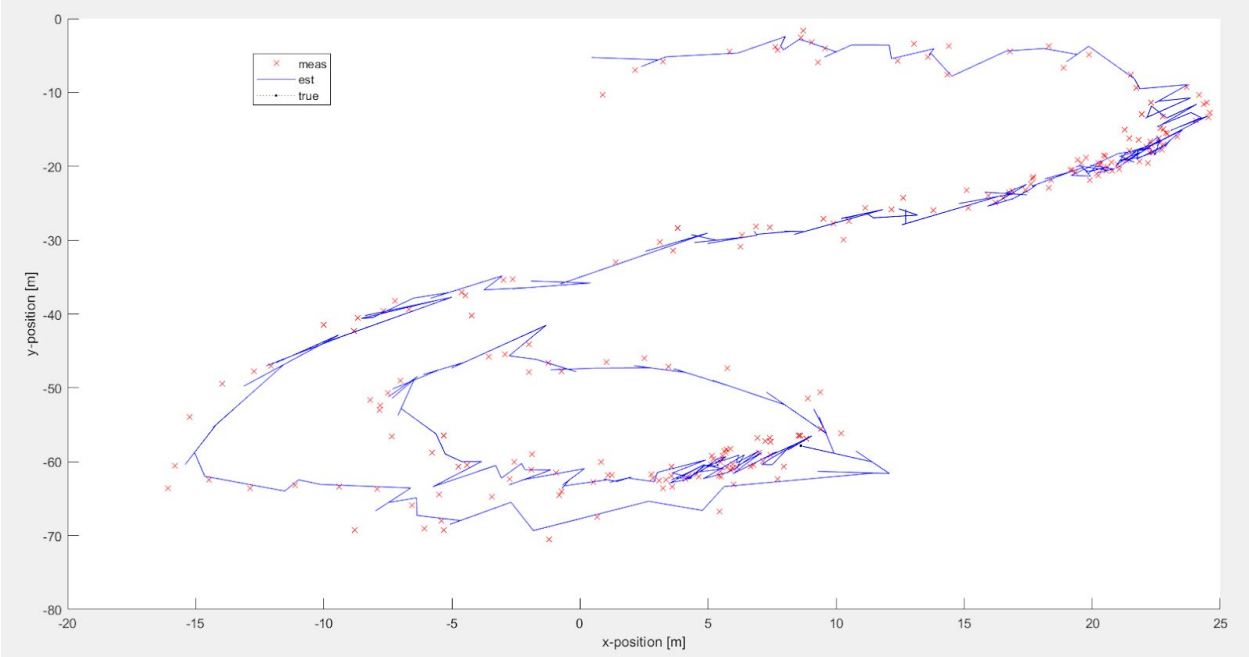
Based off of run #1:

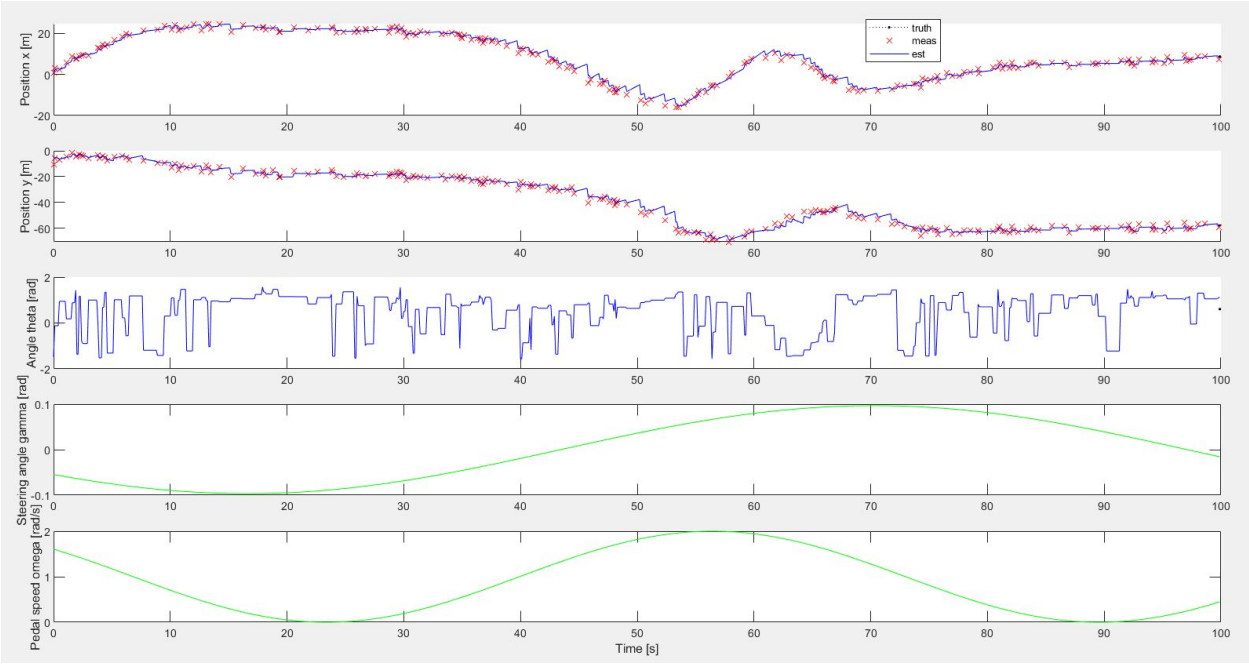
The final error was given as:

```
Final error:
   pos x = -0.46524 m
   pos y = -0.48785 m
   angle = 0.50782 rad
```

On an i7 machine, the entirety of the code ran in 0.44 seconds.

The plots as defined in the main function are below.




## Discussion of Results

The final error for the position was relatively close, being off by about 0.5 m for each position measurement. Theta was also off by about 0.5 radians. This is relatively close as well.

The error could be improved potentially with a different type of filter, although implementing another filter would be more computationally expensive.