Introduction to Kalman Filtering

An Engineer's Perspective

Gilbert Gede

January 20, 2011

Outline

Measurement of a Dynamic System

Let's say we have a physical, dynamic system

- Normally, we want to measure the states
- This can be problematic, due to real-world limitations on sensors
- So we use filters and observers

Limitations of Sensors

Unfortunate things about the real world we are all familiar with

- Not all quantities can be directly measured
- Size & Cost
- Noise & Biases

Sensor Fusion

Sensor fusion is the process of combining multiple sensor readings to create a more useful measurement.
I do not believe all Kalman Fitlers are necessarily fusing multiple sensors.
Most do though.

History of the Kalman Filter

Developed around 1960 mainly by Rudolf E. Kalman. It was originally designed for aerospace guidance applications. While it is the optimal observer for system with noise, this only true for the linear case. A non-linear Kalman Filter can not be proven to be optimal.

Targeted at Mechanical Systems and MEMs Sensors

I'm really only interested in measuring mechanical systems, so most of what I describe and present will be with this focus. The sensors I have tried to build my understanding around are similar to the MEMs sensors we have been using in the lab.

What is Not Covered

I'm not going to go through the derivation of the filter, mainly because I haven't done it myself.
I'm also not going to discuss more than the linear and Extended Kalman Filters.
There are other versions, such as the continuous filter, or the continuous-discrete filter (for a continuous system with discrete measurement points), but I have not studied these yet.

Two Step, Discrete

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

I didn't completely understand the Kalman Filter until I though of it in a specific sense: A discrete/digital filter, with two different steps as part of each cycle.
This doesn't really define a Kalman Filter, but it is how I am thinking about it.

The Filter Cycle

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

As stated, this is a two-step filter.

- One step is based on the system dynamics
- The other is based on the sensor inputs
- These are tied together with 3 covariance matrices and the Kalman Gain

System Description

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

It is *very* important to note that z represents the sensor measurement, but calculated from the states.
Another way to say this is that you need to be able to calculate something that ideally would give the sensor measurement, from only the states.

Time Update

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

The states get updated based on the known system inputs and system dynamics. The state covariance matrix ($\mathbf{P}_k$) is updated by the state matrix and process noise covariance matrix ($\mathbf{Q}$).
It should be noted that if ($\mathbf{P}_k$) = 0, that it will have the process noise updated after this step.

Time Update: EKF

SystemDescription

$$\dot{x} = f(x, u)$$
$$z = h(x)$$

TimeUpdate

$$\dot{x} = f(x, u)$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

When dealing with the extended Kalman Filter, before the time update step, you would linearize the state space model to get the state matrix, **A**.
If the process noise covariance matrix, **Q**, is dependent on the states, then it needs to be calculated before the time update as well.
The measurement, u, will be the from the next step (we use the old one).

Measurement Update

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

The Kalman gain is calculated from state covariance matrix, **P**, observation matrix,**H** and the measurement noise covariance matrix, **R**.
The state is updated from the Kalman gain and the error between the calculated sensor output and the actual sensor output (measured at this point).
The state covariance matrix, **P**, is updated by the Kalman gain, **K**, and the observation matrix, **H**.

Measurement Update

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

I feel it is important to note that the Kalman gain, while calculated from the measurement noise covariance matrix, **R**, does not necessarily have to have non-zero entries in all rows.
In fact, **R** does not have to be the same size as the state matrix.
This can lead to not all states being affected during the measurement update step, which can lead to an unsuccessful filter implementation.

Measurement Update: EFK

SystemDescription

$$\dot{x} = f(x, u)$$
$$z = h(x)$$

TimeUpdate

$$\dot{x} = f(x, u)$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

When dealing with the extended Kalman Filter, before the measurement update step, you would linearize the observation function to get, **H**.

If the sensor noise covariance matrix, **R**, is dependent on the states, then it needs to be calculated before the time update as well.

I'm not really sure how often the sensor noise will be dependent on the states though.

The Three Covariance Matrices

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

As seen in the filter equations, there are three covariance matrices:

- $\mathbf{P}_k$, the state covariance matrix
- $\mathbf{Q}$, the process covariance matrix
- $\mathbf{R}$, the measurement covariance matrix

What is a Covariance Matrix?

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

This raises the question, what is a covariance matrix?

- If we have a value, x_i, there is an expected value, $E(x_i) = \mu_i$.
- The variance is defined as $var(x_i) = E[(x_i - \mu_i)^2]$.
- Covariance is used to describe the relationship between two variables.
- $cov(x_j, x_k) = E[(x_j - \mu_j)(x_k - \mu_k)]$

Remember, standard deviation is $\sigma = \sqrt{E[(x - \mu)^2]}$

What is a Covariance Matrix?

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

As you can imagine, you could provide a standard deviation for every state, parameter, and sensor measurement. One would think this would allow a straightforward calculation of these covariance matrices.
This would be incorrect though, as not all variables are correlated. This is where a lot of the work in creating a successful filter is as far as I can tell.

The Measurement Noise Covariance Matrix

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

I believe this is the simplest to understand. It represents the covariance of all the sensors.

Usually, our sensors will not be related, so the matrix takes a form similar to this:

$$R = \begin{vmatrix} \sigma_1^2 & & & 0 \\ & \sigma_2^2 & & \\ & & \ddots & \\ 0 & & & \sigma_n^2 \end{vmatrix}$$

There will probably be some constant values in there too

The Process Noise Covariance Matrix

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

The process noise covariance matrix is a little more difficult. I'm not sure I am in the best position to describe it, but it is basically describing the error in the state matrix. There is a more correct way to define, but I will leave that at the end.

The State Covariance Matrix

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

In an informal sense, similar to my previous description of the process covariance matrix, the state covariance matrix represents the estimated error.
It is different however, in that it if updated along with the state at each step.
We can look at these values and then get an idea for how accurate our current estimation is.

The State Covariance Matrix

SystemDescription

$$\dot{x} = \mathbf{A}x + \mathbf{B}u$$
$$z = \mathbf{H}x$$

TimeUpdate

$$x_{k+1} = \mathbf{A}x + \mathbf{B}u$$
$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q}$$

MeasurementUpdate

$$\mathbf{K} = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1}$$
$$x_{k+1} = x_k + \mathbf{K}(y_k - \mathbf{H}x_k)$$
$$\mathbf{P}_{k+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k$$

I have not yet completely explored all of the information that this matrix represents, and have instead just left it alone.
One important observation: for initialization, setting it to 0 seems to work, as the update steps seem to correct it quickly.
From what I have read however, this will not always be the case (especially for the EKF).

Example System

Now I will go through an example problem. The system will be a particle moving in a plane, with 4 states and 2 inputs.

$$\mathbf{x} = \begin{bmatrix} x \\ u \\ y \\ v \end{bmatrix}, \dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

$$\mathbf{z} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

It is basically a double integrator, with acceleration as an input, and positions which can be measured by sensors.

Example System

So, for this example, we have two sets of sensors: accelerometers and position sensors (like GPS).
These sensors will have simulated noise, just like real sensors.
We will use the Kalman Filter to estimate fuse the sensor readings and estimate the position.

Discretization

The first step is to rewrite the continuous state space model as a discrete model.

$$\dot{\mathbf{x}}_{k+1} = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \Delta t^2/2 & 0 \\ \Delta t & 0 \\ 0 & \Delta t^2/2 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

$$\text{where } \mathbf{x} = \begin{bmatrix} x \\ u \\ y \\ v \end{bmatrix}$$

Covariance Matrices

The next thing we have to provide to the filter description is the sensor & measurement covariance matrices. We will start with the sensor covariance matrix.

Sensor Covariance Matrix

The sensor noise covariance matrix is easiest to form; it is:

$$\mathbf{R} = \begin{bmatrix} \sigma^2_{gps_x} & 0 \\ 0 & \sigma^2_{gps_y} \end{bmatrix}$$

The value is simply the standard deviation of the sensor squared, or the variance of the sensor.
There is no coupling (in this example) between the x and y positions as reported by the "GPS" sensor.
I'm not sure if this is true in real life as well...

Process Covariance Matrix

The process covariance matrix is slightly more complicated. A more detailed version of the state space model needs to be used.

$$\dot{\mathbf{x}} = \mathbf{A}x + \mathbf{B}u + v$$

Understanding that the noise in this step is introduced from the accelerometer we can rewrite the system as follows:

$$\dot{\mathbf{x}}_{k+1} = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \Delta t^2/2 & 0 \\ \Delta t & 0 \\ 0 & \Delta t^2/2 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} a_x + v_x \\ a_y + v_y \end{bmatrix}$$

Process Covariance Matrix

We can then take the derivative of the state vector with regards to the noises.
This will be defined as $\mathbf{F} = \frac{\delta \mathbf{x}}{\delta v}$.
It can be seen that this will simply be the $\mathbf{B}$ matrix, due to linearity.

Process Covariance Matrix

If we define $\mathbf{R}_v = E[(v_i - \overline{v})(v_i - \overline{v})]$, where v is the noise vector, we would get a 2x2 matrix.
This matrix should actually be diagonal though, because the sensor error will not be coupled.
This gives $\mathbf{R}_v = \begin{bmatrix} \sigma^2_{acc_x} & 0 \\ 0 & \sigma^2_{acc_y} \end{bmatrix}$.

Process Covariance Matrix

Finally, we can define $\mathbf{Q}$.

$$\mathbf{Q} = \mathbf{F}\mathbf{R}_v\mathbf{F}^T = \sigma_{acc}^2 \begin{bmatrix} \Delta t^4/4 & \Delta t^3/2 & 0 & 0 \\ \Delta t^3/2 & \Delta t^2 & 0 & 0 \\ 0 & 0 & \Delta t^4/4 & \Delta t^3/2 \\ 0 & 0 & \Delta t^3/2 & \Delta t^2 \end{bmatrix}$$

When dealing with the EKF, $\mathbf{F}$ might need to be linearized at each time step. This would happen at the same time as the linearization of $\mathbf{A}$.

Simulation Parameters

We are now ready to simulate this system.
We need to specify the simulation parameters first.

- $\Delta t = .1sec$
- $\sigma_{gps} = 2m$
- $\sigma_{acc} = .5m$

Simulation Inputs

The inputs to the model follow:

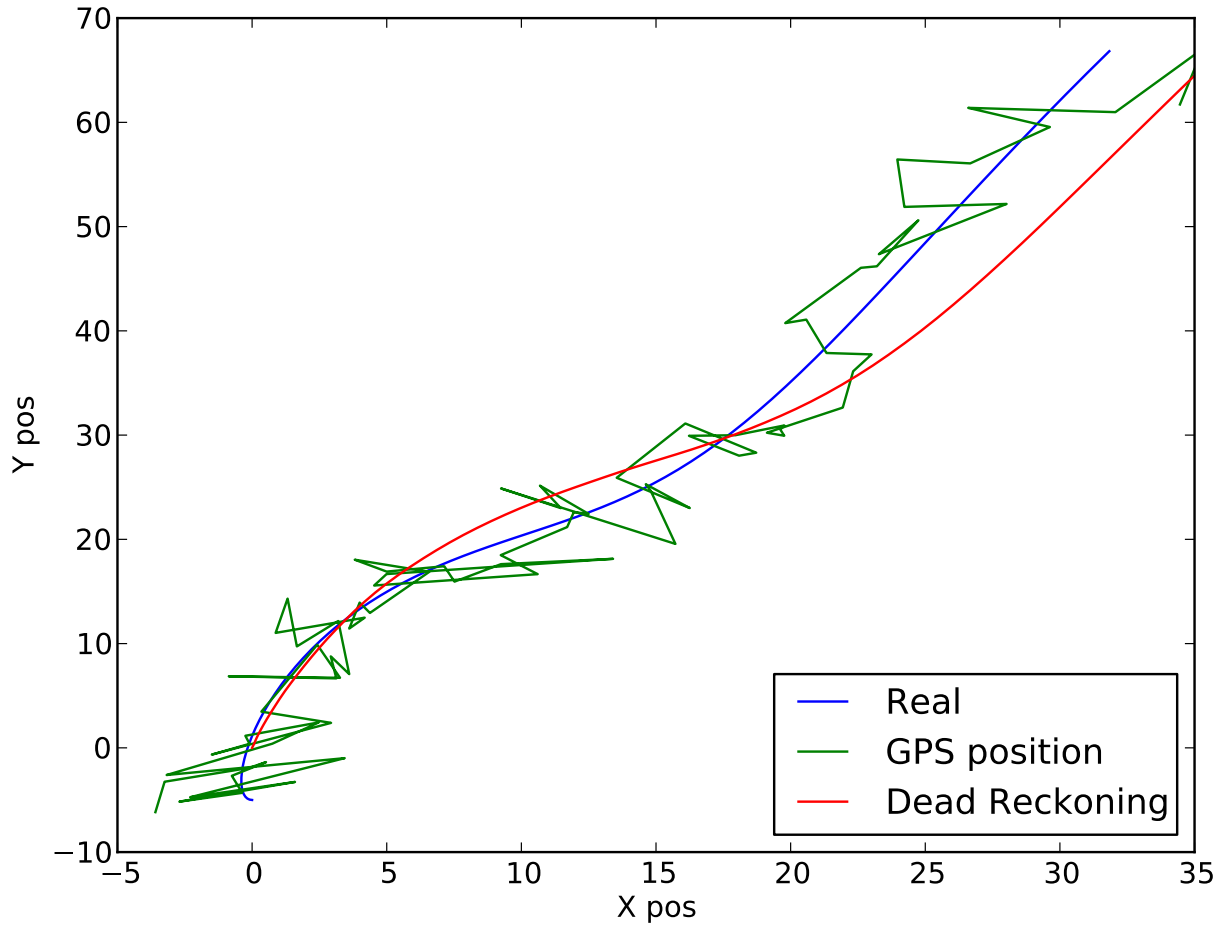
- $acc_x = 1 + sin(t)$
- $acc_y = 2 + 5sin(t)$
- $x = t^2/2 - sin(t)$
- $y = t^2 - 5cos(t)$

The appropriate noises are added on top of these measurements.

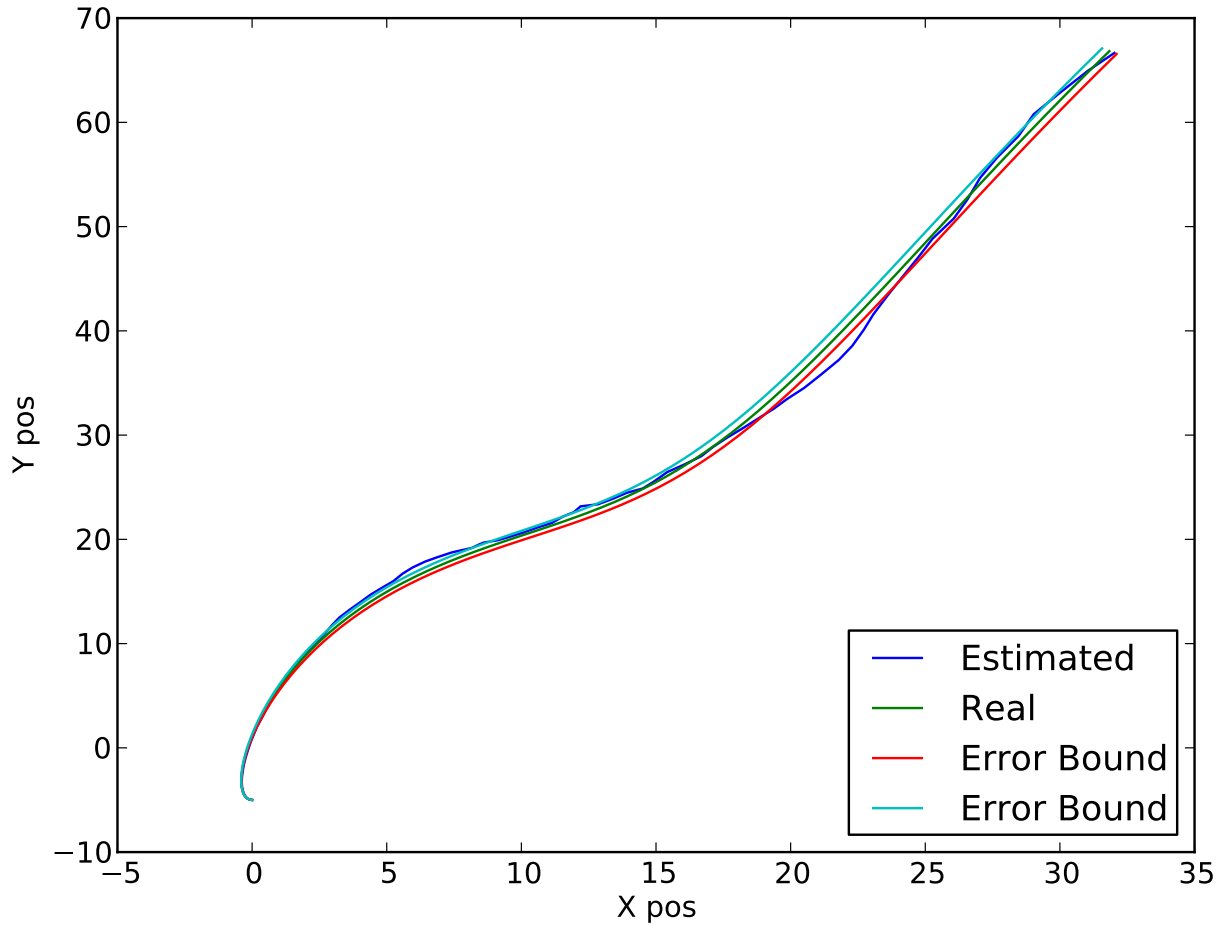
Simulation Results

The system was simulated for 8 seconds.
Results follow.

Simulation Results



Simulation Results



Simulation Results

We can see that the results with the filter are much better. The calculated standard deviations for each method are:

- $\sigma_{kalman} \approx 0.6m$
- $\sigma_{gps} \approx 1.6m$
- $\sigma_{acc} \approx 2.5m$

Obviously, for longer simulation times the error gets much larger with the dead reckoning case.

Success?

I think it is safe to say that this filter implementation is successful.
I believe we can also say it is optimal.

Next Steps

With this simple example, we could next try estimating sensor biases.
I did not include any here, but estimating biases usually seems to be successful.
This would be done by adding an extra state, assuming it was constant in time updates, and ensuring that there were proper covariances describing it.

Next Steps

Exploration of other Kalman Filter types could be useful too. There exists the continuous filter and continuous-discrete filter; we have discussed the EKF, but there also exists the Unscented Kalman Filter for highly nonlinear systems. There are also square root forms and triangular forms.

References

References:

1. Fiorenzani T., Manes C, Oriolo G., Peliti P. Comparative Study of Unscented Kalman Filter and Extended Kalman Filter for Position/Attitude Estimation in Unmanned Aerial Vehicles, IASI-CNR, R. 08-08, 2008
2. Sabatini, A.M. Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing. IEEE Transactions on Biomedical Engineering 53, 1346-1356 (2006).
3. "Kalman Filter." Wikipedia, the Free Encyclopedia. 20 Jan 2011. `http://en.wikipedia.org/wiki/Kalman_filter`.
4. An Introduction to the Kalman Filter, SIGGRAPH 2001 Course, Greg Welch and Gary Bishop. `http://www.cs.unc.edu/~tracker/media/pdf/SIGGRAPH2001_Course`