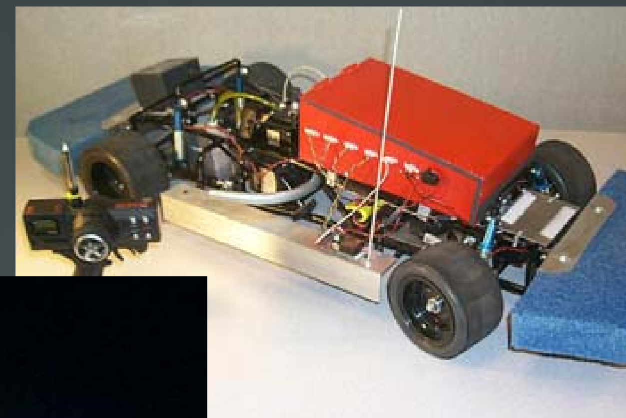
Kalman Filters for Robotics:
Getting the Most from your Sensors

Michael Prados
mpradosNOSPAM@gmail.com

The SWARM Project
http://orbswarm.com

What's Wrong With Me?



Vehicle Dynamics Research

SWARM

earthmine

SWARM – Positional Robotic Art

- Six Spherical Robots
  - 760mm (30in) Diameter, 40kg (90lbs) each
- Consumer Grade GPS- 3m accuracy
- Sparkfun 5-DOF IMU
  - ADXL330 3-DOF Accelerometer
  - IDG300 2-DOF Rate Gyro
- Wheel Encoder, Kinematic Model
- All Open Source!

Some Positional Sensors for Robotics

Sensors are a robot's window to the world!

- Inertial sensors
  - Accelerometers
  - Rate Gyros
  - Electrolytic Tilt Sensors
- Position Sensors
  - GPS
  - Wheel Encoders
  - Sonar

Problems with Positional Sensors

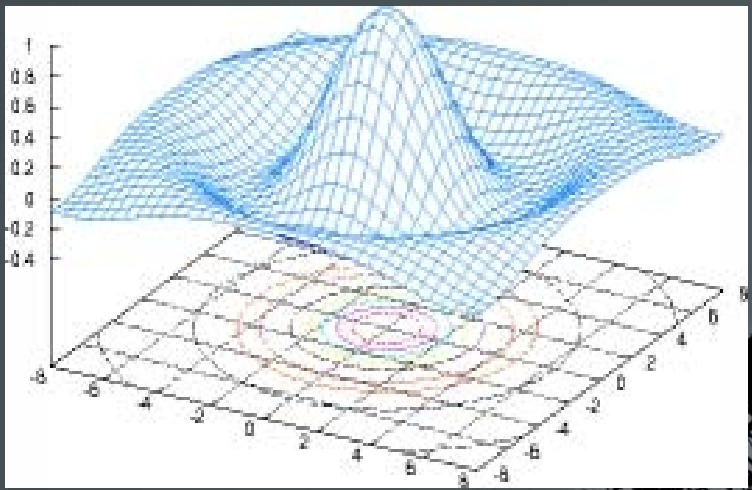
- Inertial sensors – DC Bias, Noise
- Position Sensors
  - GPS – outages, low frequency, noise
  - Wheel Encoders – relative position, slip
  - Sonar – limited range, noise

What is a Kalman Filter?
(aka Kalman Estimator, Kalman Observer)

- A Kalman filter is an optimal observer
  - It will trade off the relative strengths of your sensors, in order to provide an optimal estimate of your system states
- Always applied to a state space model of a system
  - Must derive a system model, but can often be simple
- Either discrete time or continuous
  - Usually must go to discrete time for implementation

Octave and Matlab

- Matlab is an expensive commercial product for matrix algebra and numerical processing
  - Great estimation and control system support
- Octave is a very similar, free open source alternative
  - Almost as good
  - Easier to share



What is a State Space Model?

- A model of a dynamic system,
  using a system of differential equations
- In general:

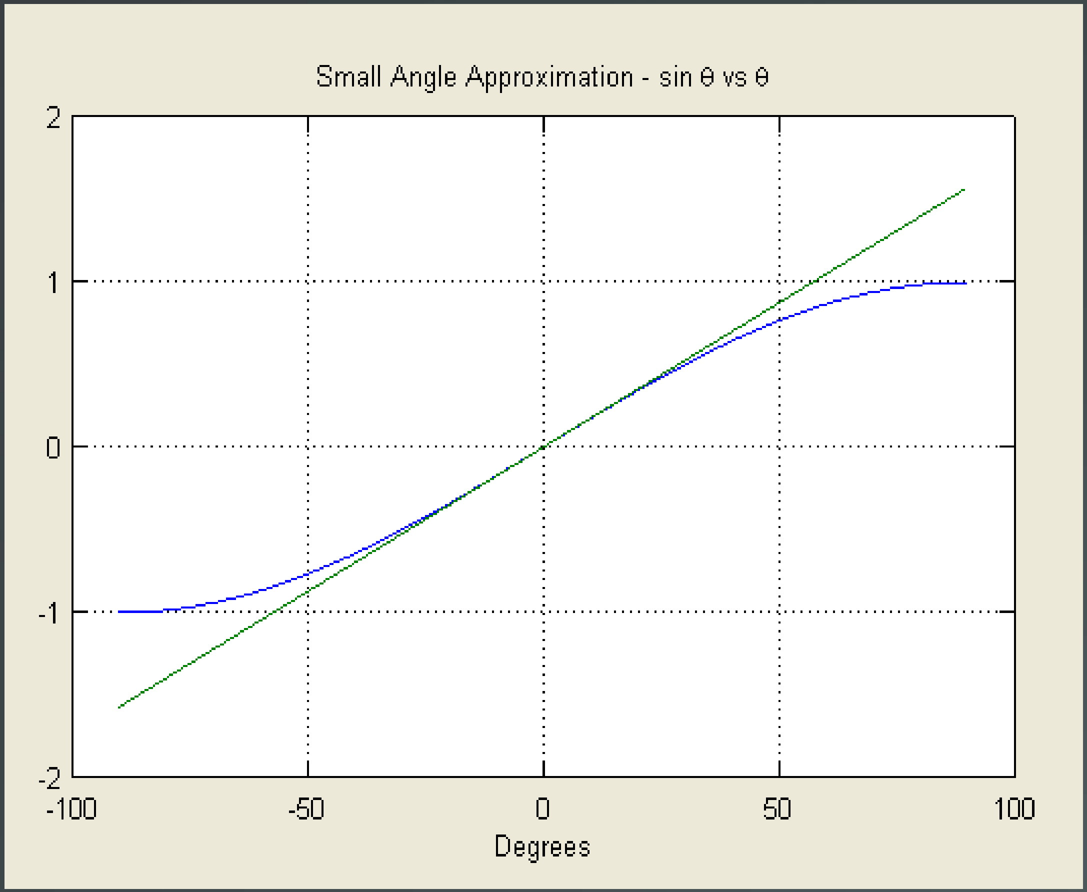
$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

- An example:

$$\begin{bmatrix} \dot{v} \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{matrix} v \\ x \end{matrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$

$$\begin{bmatrix} count_{encoder} \end{bmatrix} = \begin{bmatrix} 0 & k \end{bmatrix} \begin{bmatrix} v \\ x \end{bmatrix}$$

Small Angle Approximation



- Pretty good up to +/- 30 degrees
- Remember to implement using radians!

Continuous vs Discrete

- Continuous time is a mathematically elegant representation using derivatives
  - Relatively hard to code!
- Discrete time representation tells you directly how to get the next step from the current one
  - $x_{k+1} = f(x_k)$
  - For most systems, octave will do the conversion!

```
discrete_sys = c2d(continuous_sys, T);
```

The Observer

- For a system-

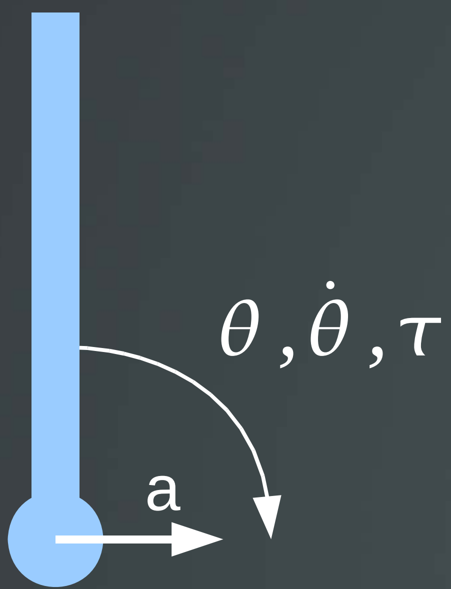
$$\begin{aligned} x_{k+1} &= A_d x_k + B_d u_k \\ y_k &= C_d x_k + D_d u_k \end{aligned}$$

- We estimate the states like this-

$$\begin{aligned} \hat{x}_{k+1} &= A_d \hat{x}_k + L(y_k - \hat{y}_k) + B_d u_k \\ \hat{y}_k &= C_d \hat{x}_k + D_d u_k \end{aligned}$$

Inverted Pendulum

- Inverted pendulum model
- Rate gyro, and accelerometer tilt sensor



$$\begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix} + \begin{bmatrix} 1/I_m \\ 0 \end{bmatrix} \tau$$

$$\begin{bmatrix} \omega \\ a \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & g \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \theta \end{bmatrix}$$

In Octave...

- Create system

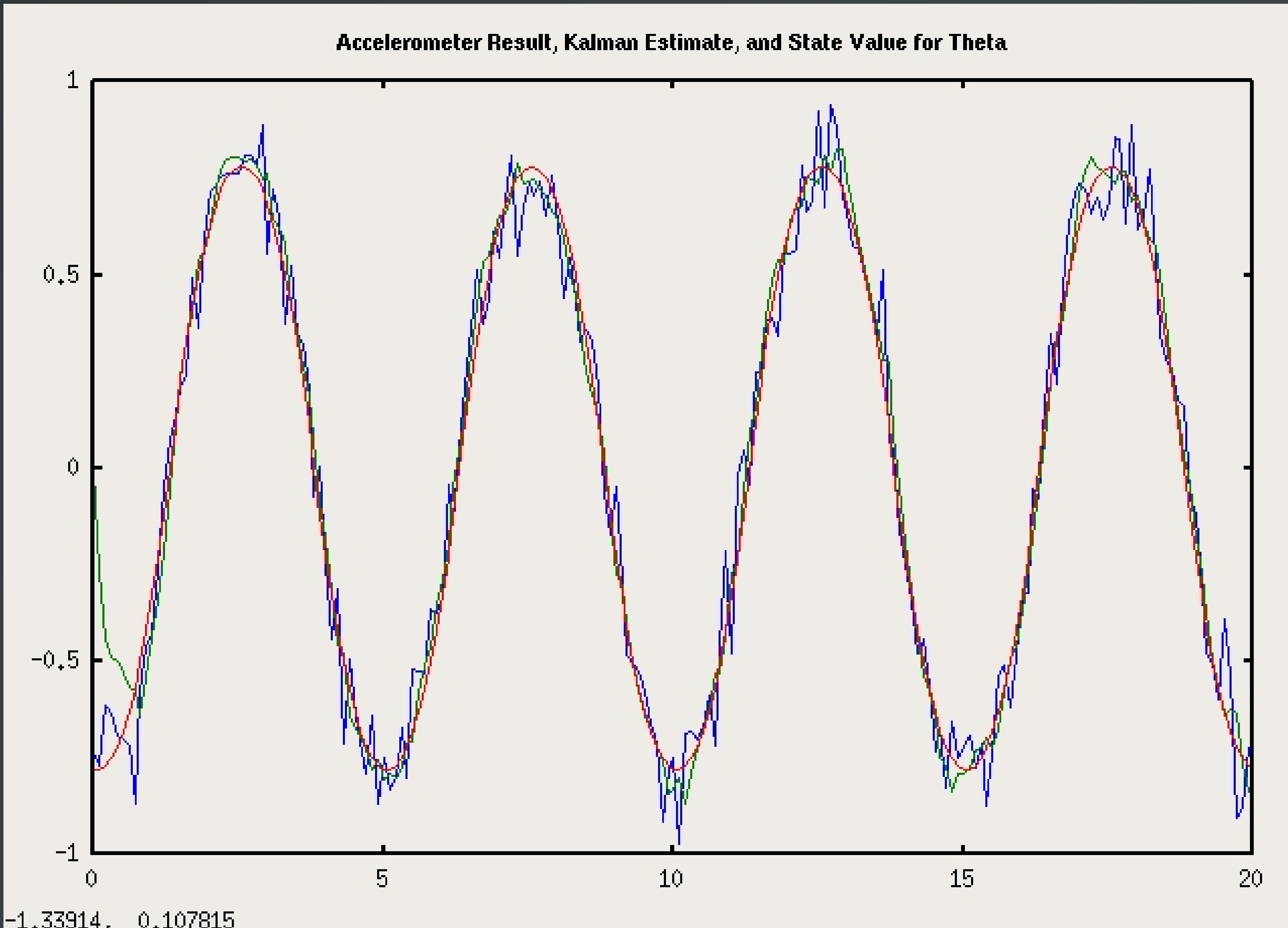
with A,B,C,D

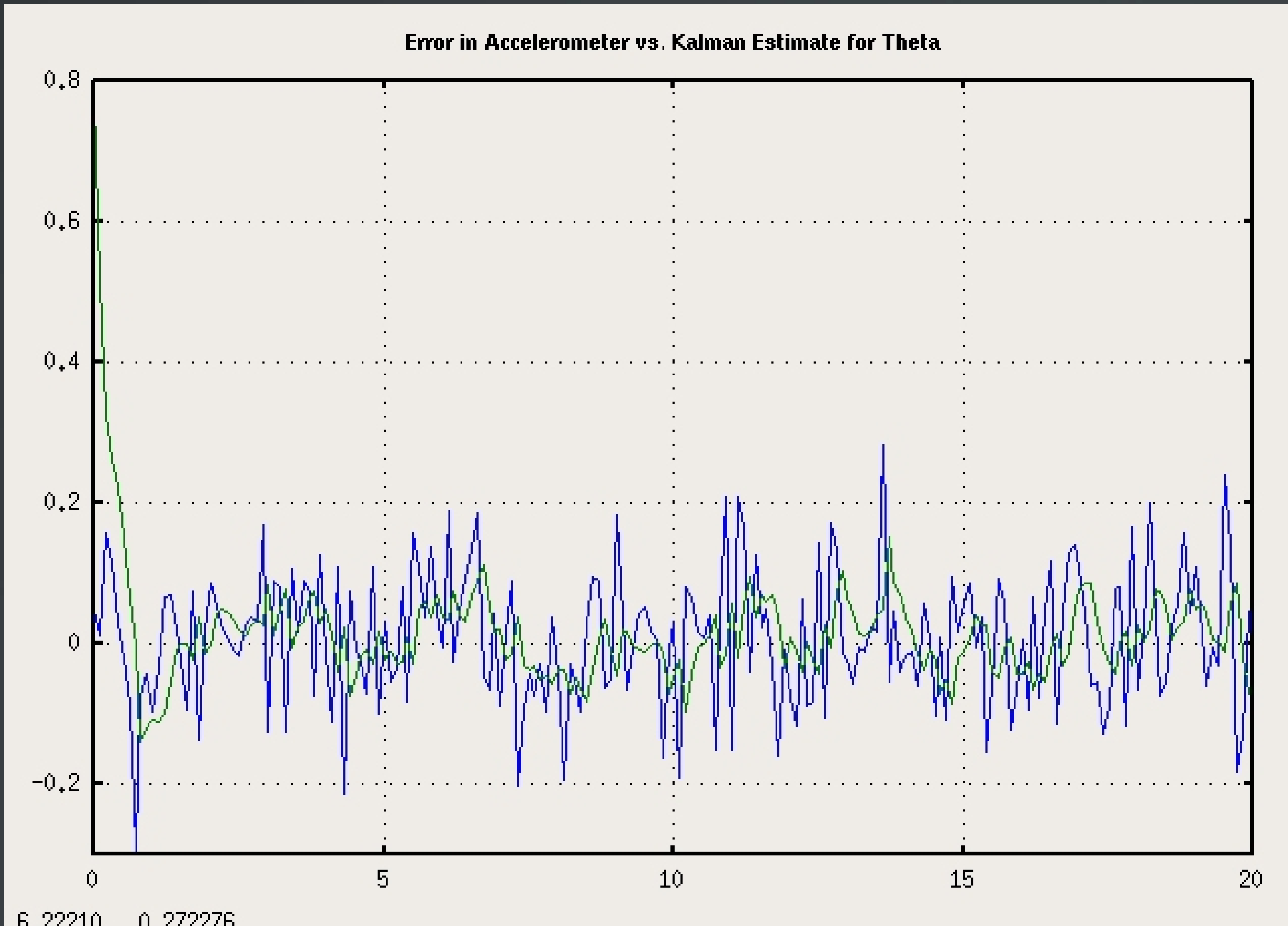
- Use c2d
- Set QW, RV
- Invoke dkalman

$$\begin{bmatrix}\ddot{\theta}\\ \dot{\theta}\end{bmatrix} = \overset{A}{\begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix}} \begin{bmatrix}\dot{\theta}\\ \theta\end{bmatrix} + \overset{B}{\begin{bmatrix}1/I_m\\ 0\end{bmatrix}} \tau + W$$

$$\begin{bmatrix}\omega\\ a\end{bmatrix} = \overset{C}{\begin{bmatrix}1 & 0\\ 0 & g\end{bmatrix}} \begin{bmatrix}\dot{\theta}\\ \theta\end{bmatrix} + V$$

Presto, optimal observer!


Accelerometer Result, Kalman Estimate, and State Value for Theta
1
0.5
0
-0.5
-1
0
5
10
15
20
-1.33914, 0.107815



Error in Accelerometer vs. Kalman Estimate for Theta
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
6.22210, 0.272276

Limitations of the Kalman Filter

- For Good Performance
  - No strong nonlinearities
    - Definitely no trig with greater than +/- 90 degree motion!
  - Zero-mean noise – No DC offset
- Necessary conditions to be optimal,

but often less important-

  - Noise has gaussian distribution
  - Noise is white
  - Noise is not auto-correlated

DC Offset

- Common for inertial sensors
  - Rate Gyro's and accelerometers
  - Need to debias
- Add state to system-

$$\begin{bmatrix} \ddot{\theta} \\ \dot{\theta} \\ \dot{bias}_{gyro} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \theta \\ bias_{gyro} \end{bmatrix} + \begin{bmatrix} 1/I_m \\ 0 \\ 0 \end{bmatrix} \tau + W$$

$$\begin{bmatrix} \omega \\ a \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & g & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \theta \\ bias_{gyro} \end{bmatrix} + V$$

Extended Kalman Filter

- Necessary for systems with strong nonlinearity
  - Large angle motion- certainly greater that 180 degrees
  - Squares, cubes, exponentials

Predict

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k)$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

Update

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$$

where the state transition and observation matrices are defined to be the following Jacobians

$$\mathbf{F}_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k}$$

$$\mathbf{H}_k = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}}$$

SWARM Extended Kalman Filter

- 13 states

$$\dot{v}, v, \dot{\phi}, \phi, \theta, \psi, x, y, x_{ab}, y_{ab}, z_{ab}, x_{rb}, z_{rb}$$

- 10 sensors

$$x_a, y_a, z_a, x_r, z_r, x_{gps}, y_{gps}, \psi_{gps}, v_{gps}, \omega$$

- Kinematic model – too big for slide.

- Work in progress! *info@orbswarm.com*

Implementation

- Translation to C code
  - Easier with some matrix manipulation libraries
- Fixed vs floating point
  - Usually possible in fixed point, but often quite laborious
  - For more than 2-4 states, probably worth floating point
    - Emulation – 8 bit micro, ARM
    - FPU – x86, PPC, some ARM (caution!)
    - DSP – great platform, steep learning curve

Where to Go

Good Optimization and Control Texts:

- Design of Feedback Control Systems
  - Stefani, Shahian, Savant, Hostetter – Oxford University Press
- Digital Control of Dynamic Systems
  - Franklin, Powell, and Workman – Addison Wesley

More advanced:

- Applied Optimal Estimation
  - Gelb – M.I.T. Press
- Optimal Control and Estimation
  - Stengel – Dover Publications

Thanks!

?