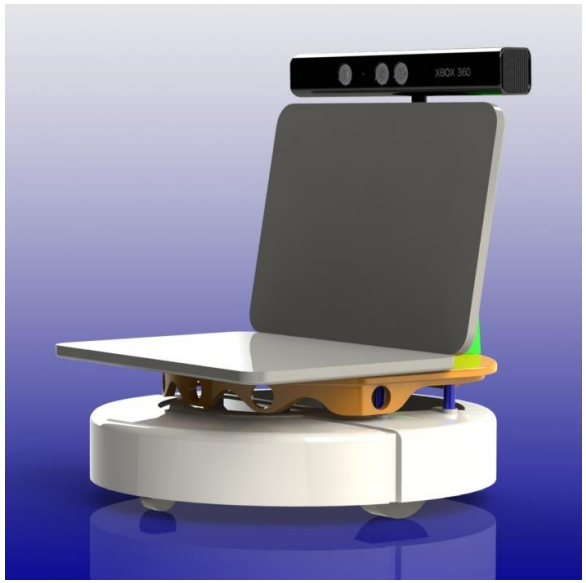
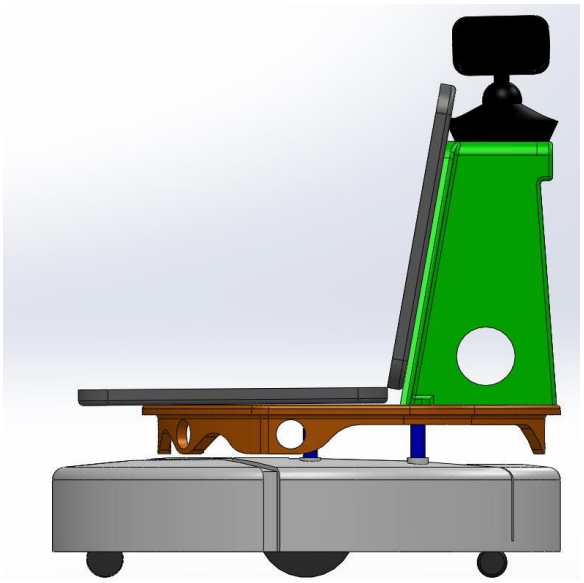
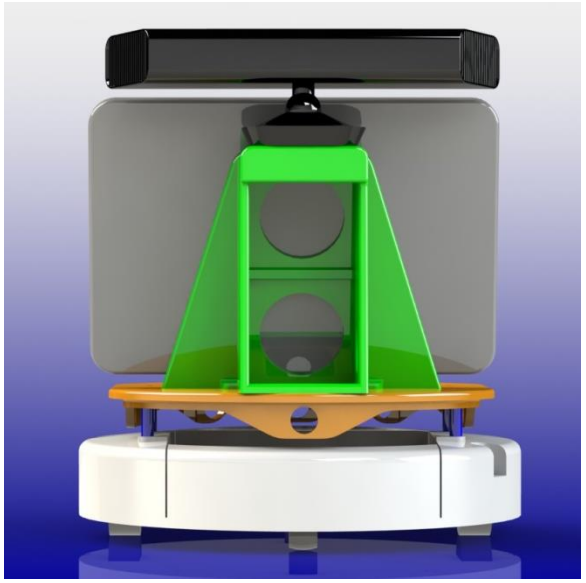
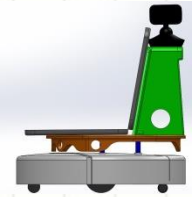
MAE 493G, CpE 493M, Mobile Robotics

8. Introduction to Kalman Filter









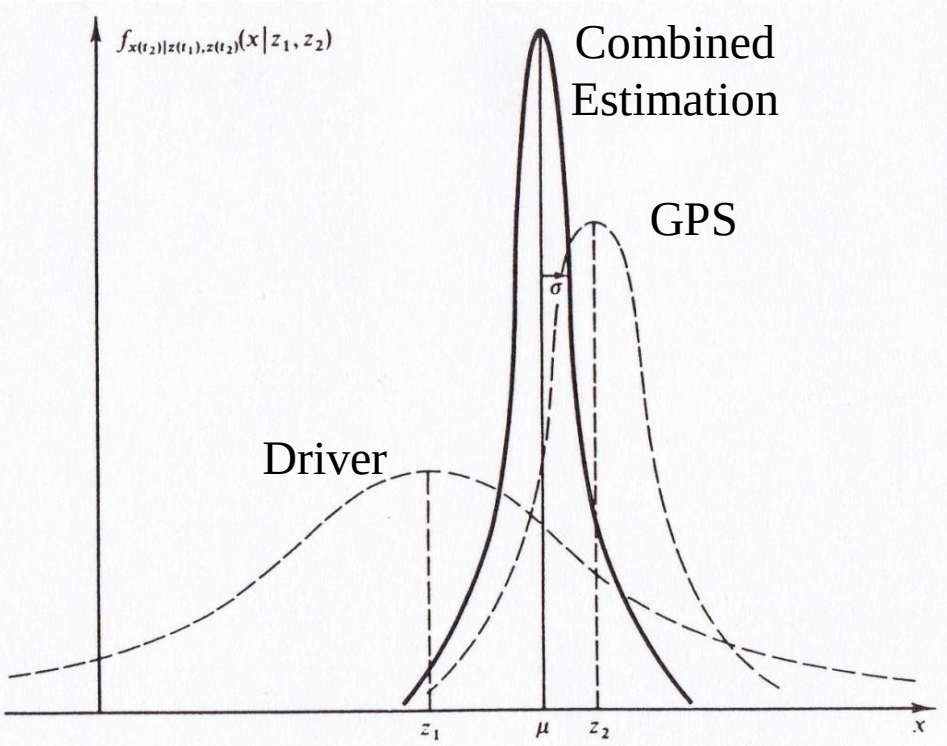
Kalman Filter

- The Kalman filter operates *recursively* on streams of noisy input data to produce a *statistically optimal* estimate of the underlying system state;
- Remember the question we had earlier about how to do the average if the measured parameter is time-varying?

- Kalman filter gives us the answer if you can formulate the time-varying parameter that you want to measure as a *state* of a *dynamic system*, which usually is not too difficult;
- Kalman filter is rooted from both *optimal control* and the *Bayers' rule*. Conceptually, it is about how to make a series of (educated) guesses based on both priori knowledge and current observation (Bayesian). It's optimal in the sense that it minimizes the *mean square error;*
- Kalman filter is widely used in navigation, object tracking, stock analysis, etc...

The Basic Idea

- Assuming that you are on the way to New York City;
- You have been there before and you *kind of know* how long you have to drive on each segment of the road;
- As you drive, you have a rough idea of where you at based on the *time* of travel;
- You also have a GPS unit, which is also *not very trustable*, because the map hasn't been updated for a few years;



- So, how do you *combine the information* from these two sources?

  Simple answer: use a *weighted average* of the two information.

  However, we need to figure out to how to calculate the weights...

Propagating Statistics Through Linear Functions

- For *independent* random variables X, Y, and constants a, b, and c:

$$E(aX+bY+c)=aE(X)+bE(Y)+c$$

$$\text{var}(aX+bY+c)=a^2\text{var}(X)+b^2\text{var}(Y)+0$$

- If X, and Y are not independent:

$$\text{var}(aX+bY+c)=a^2\text{var}(X)+b^2\,\text{var}(Y)+2ab\cdot\text{cov}(X,Y)$$

- For two $n\times1$ random vectors **X** and **Y** that are *independent* from each other, $n\times n$ constant matrices **A, B,** and $n\times1$ constant vector C**:**

$$E(\mathbf{AX}+\mathbf{BY}+\mathrm{C})=\mathbf{A}E(\mathbf{X})+\mathbf{B}E(\mathbf{Y})+\mathrm{C}$$

$$\text{cov}(\mathbf{AX}+\mathbf{BY}+\mathrm{C})=\mathbf{A}\text{cov}(\mathbf{X})\mathbf{A}^T+\mathbf{B}\text{cov}(\mathbf{Y})\mathbf{B}^T$$

- With these equations, we can predict the mean and variance (covariance) of random variables after transforming them through linear equations. The next step is to find these equations for specific engineering problems;
- *A Gaussian distribution will still be Gaussian after a linear transformation.*

State Space Representation

- In engineering, the mathematical model of a physical process is often described as differential equations. For example: position P is the second derivation of acceleration a: $\ddot{P} = a$
- A state space representation is a mathematical model of a physical system as a set of input, output and state variables related by first-order differential equations;
- The inputs **u**, outputs **y**, and states **x** are expressed as vectors. If the dynamical system is linear, the differential and algebraic equations may be written in matrix form;
- "State Space" refers to the space whose axes are the state variables. The state of the system can be represented as a vector within that space.
- An intuitive example is the x, y, and z position of a robot.
- State-space model creates a convenient platform for solving Multiple-Input-Multiple-Output (MIMO) problems.

State Space (Cont.)

- *Continuous-time linear time invariant*:

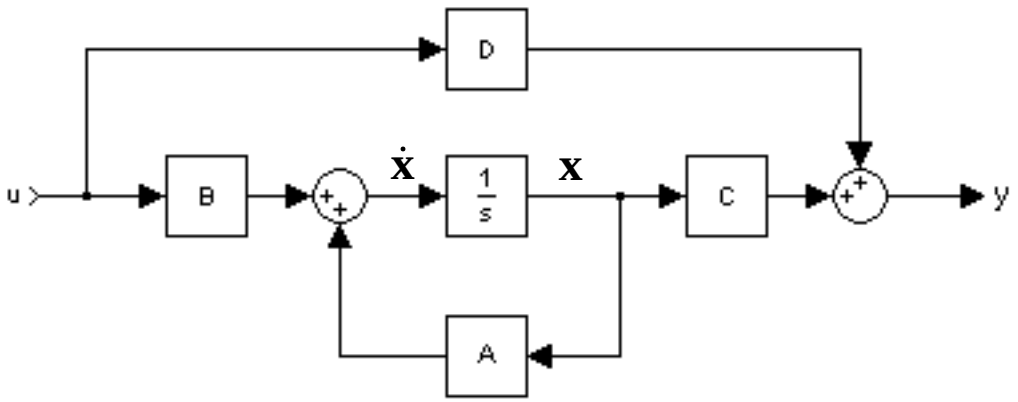
$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

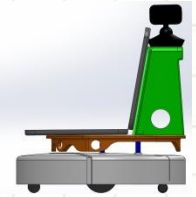
- *Discrete-time linear time invariant*:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$
$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$$

- Where **x** is a $n\times1$ state vector; **y** is a $q\times1$ output vector, **u** is a $p\times1$ input vector, **A** is a $n\times n$ state transition matrix, **B** is a $n\times p$ state input matrix, **C** is a $q\times n$ state output matrix, and **D** is a $q\times p$ feed-through matrix.
- The state space model does not have to be linear. For the nonlinear case, the relationships are just not in a matrix form; e.g. for continuous-time:

$$\dot{\mathbf{x}}(t) = f\left(t, \mathbf{x}(t), \mathbf{u}(t)\right)$$
$$\mathbf{y}(t) = h\left(t, \mathbf{x}(t), \mathbf{u}(t)\right)$$





State Space Example

- For the one-dimensional navigation problem we talked about earlier: $\ddot{P} = a$ We can breakdown this second order differential equation to two first order differential equations: $\dot{P} = V$, and $\dot{V} = a$ where V is the velocity.
- Let's define the state vector as: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} P \\ V \end{bmatrix}$ and the input as $u = a$
- Assume we also have a measurement of the position: $y = P_{measure}$
- We can have a continuous time state-space formulation as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- There are many tools to help us to understand systems like this;
- Keep in mind if a and $P_{measure}$ are provided by sensors, we will need add *noises* into the equations above.

Process and Measurement Noise

- Given the state space models we have seen so far, noises or uncertainties can show up at two places (at least):

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \qquad \mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + \mathbf{w}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{v}(t) \qquad \mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) + \mathbf{v}(t)$$

- We call **w** process noise and **v** measurement noise. We assume they are both *white* and *Gaussian*, with $\mathbf{w}(t) \sim N\left(0, \mathbf{Q}(t)\right), \quad \mathbf{v}(t) \sim N\left(0, \mathbf{R}(t)\right)$
- Here *N* stands for normal distribution, the zeros are the means and **Q** and **R** are covariance matrices;
- If **w** and **v** are time varying, then **Q** and **R** will be time varying as well;
- Measurement noise is easy to understand and easy to quantify;
- Process noise captures the uncertainties in the mathematical model, the parameters, the disturbance on the system, and the noises in the input signal;
- The property of the process noise is sometime tricky to determine due to multiple contributing factors.

Discretization

- Our computer does not run in continuous time so often we need to discretize our models before running a computer simulation.
- Starting from a continues time state-space model:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{v}(t)$$

  Where, **w** and **v** are Gaussian white noise with parameters:

$$\mathbf{w}(t) \sim N\left(0, \mathbf{Q}(t)\right), \quad \mathbf{v}(t) \sim N\left(0, \mathbf{R}(t)\right)$$

- It can be discretized to the following model:

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{u}(k) + \mathbf{w}_d(k)$$
$$\mathbf{y}(k) = \mathbf{C}_d\mathbf{x}(k) + \mathbf{D}_d\mathbf{u}(k) + \mathbf{v}_d(k)$$

  with $\mathbf{w}_d(k) \sim N\left(0, \mathbf{Q}_d(k)\right), \quad \mathbf{v}_d(k) \sim N\left(0, \mathbf{R}_d(k)\right)$

- If the sampling time is T_s, $\mathbf{A}_d$, $\mathbf{B}_d$, $\mathbf{C}_d$, $\mathbf{D}_d$ and $\mathbf{R}_d$ can be (approximately) calculated with: $\mathbf{A}_d = \mathbf{I} + \mathbf{A}T_s, \mathbf{B}_d = \mathbf{B}T_s, \mathbf{C}_d = \mathbf{C}, \mathbf{D}_d = \mathbf{D}, \mathbf{R}_d = \mathbf{R}$

  Where **I** is the identity matrix;

- The calculation of $\mathbf{Q}_d$ is trickier, but we won't need to do it most of the time.

1D Navigation Example

- For the continuous-time equation we had earlier, let's add in some uncertainties:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\left(u(t) + w_a(t)\right)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + v_P(t)$$

- We assume here the only uncertainties are introduced by the acceleration measurement noise $w_a \sim N(0, \sigma_a^2)$ and position measurement noise $v_P \sim N(0, \sigma_P^2)$
- The discretized system is:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ T_s \end{bmatrix}\left(u(k) + w_{da}(k)\right)$$

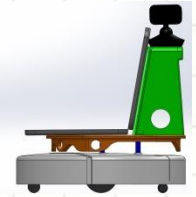
$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v_{dP}(k)$$

- The noise properties: $w_{da} \sim N(0, \sigma_{da}^2)$ and $v_{dP} \sim N(0, \sigma_{dP}^2)$ can be acquired by analyzing the collected sensor data.
- If we consider the total process noise: $w_d = T_s w_{da}$, then $w_d \sim N(0, T_s^2 \sigma_{da}^2)$

Kalman Filter

- A Kalman filter is a recursive estimator that is based on linear dynamic systems discretized in the time domain: $\mathbf{x}_k = \mathbf{A}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_{k-1} + \mathbf{w}_{k-1}$

  $\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{v}_k$
- Note that we have a slight change of notation here to make things a little less cluttered...
- The process and measurement noises are assumed to be *independent* from each other, *white*, and *Gaussian*, with $\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$, $\mathbf{v}_k \sim N(0, \mathbf{R}_k)$
- Instead of only propagating the states through the system dynamics, we will propagate the *distribution* of our estimates;
- This include the *state* (the expected value of the estimated distribution) and the *covariance matrix* (the confidence about the estimate). Remember? the Gaussian distribution can be fully described through these two (set of) parameters!
- This process takes two main steps: *prediction* and *update*.

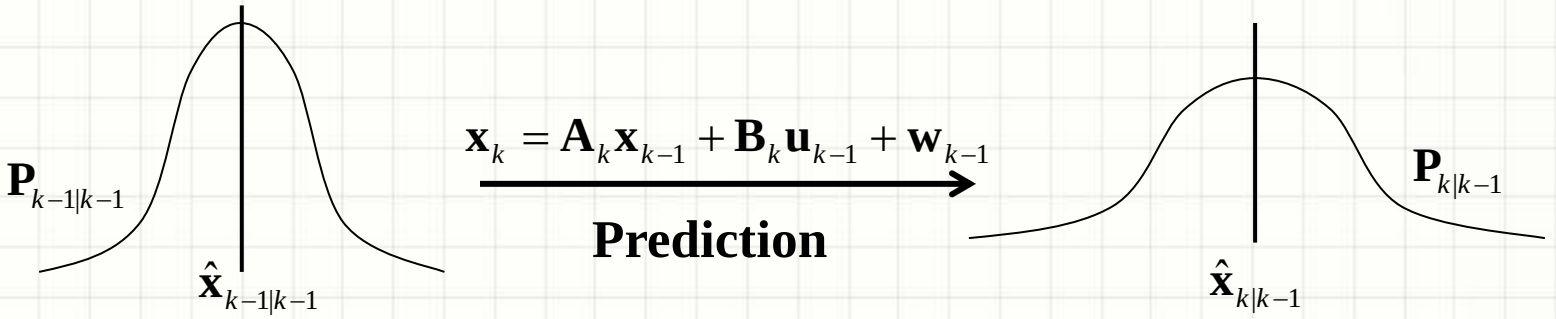


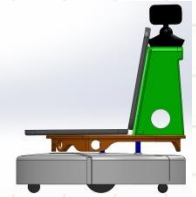
Prediction (in Time)

- Starting from the estimated state (the expected value of a distribution) $\hat{\mathbf{x}}_{k-1|k-1}$ and covariance (the confidence level) $\mathbf{P}_{k-1|k-1}$ from the previous time step (*k-1*), let's propagate these statistics through the linear system (difference) equations: $\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_{k-1}$

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

- We call $\hat{\mathbf{x}}_{k|k-1}$ and $\mathbf{P}_{k|k-1}$ *a priori* (with out seeing the measurement at time *k* yet!) *state estimate* and *estimation error covariance*;
- Remember a multivariate Gaussian distribution will remain Gaussian after a linear transformation? Only with different mean and covariance.
- We basically predicted what the distribution may look like using *prior knowledge* (system equation and property of the noises)





Measurement Update

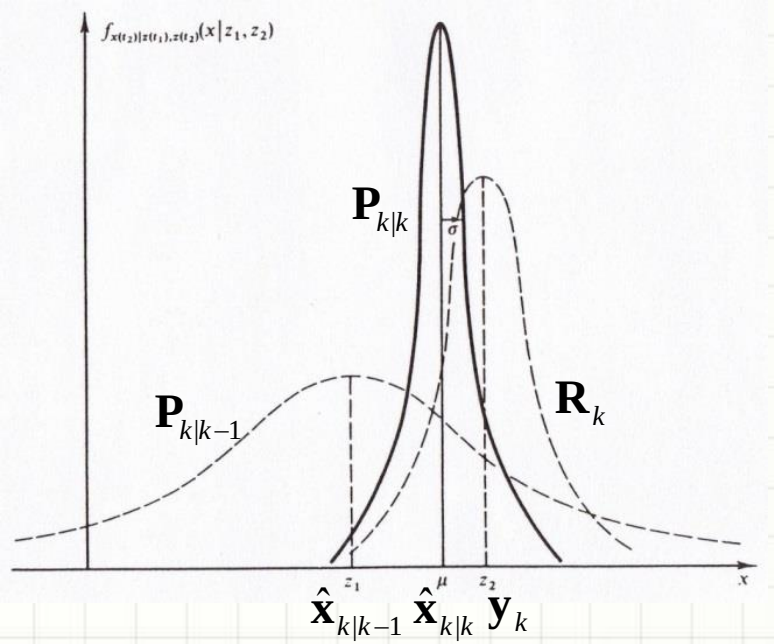
- However, we don't live in a perfect world and predictions are not always true...the good news is that we are also performing (noise corrupted...) measurements! Now we have two sources of information but which one to trust?
- The difference between measurement and predication is called *innovation* or *residual*: $\tilde{\mathbf{r}}_k = \mathbf{y}_k - \mathbf{C}_k\hat{\mathbf{x}}_{k|k-1}$ and we can also find out it's covariance with:

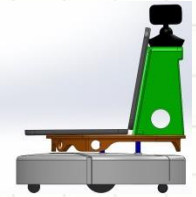
$$\mathbf{S}_k = \mathbf{C}_k\mathbf{P}_{k|k-1}\mathbf{C}_k^T + \mathbf{R}_k$$

- The rest is to update our distribution:
  Posterior estimate: $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{r}}_k$
  Posterior error covariance:

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

- Where $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{C}_k^T\mathbf{S}_k^{-1}$ is the optimal Kalman gain, which can be seeing as a time-varying weighting factor between prediction and measurements.





Putting it Together

Predication

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{A}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_{k-1}$$

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

Update

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{C}_k^T \left(\mathbf{C}_k \mathbf{P}_{k|k-1} \mathbf{C}_k^T + \mathbf{R}_k \right)^{-1}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{C}_k \hat{\mathbf{x}}_{k|k-1} \right)$$

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

It only takes five equations to have a Kalman filter!

1D KF Example for Navigation

- Let's figure out the one dimensional position of a robot from two measurements. The first one is a direct GPS position measurement, which is assumed to have a white and Gaussian noise with zero mean and standard deviation of 2m. The second measurement is from the accelerometer, which is assumed to have a white and Gaussian noise with zero mean and standard deviation of 0.1m/s^2. The sampling rate is 10Hz. Our goal here is to estimate the robot position.
- We already have the state-space model from the previous discussions:

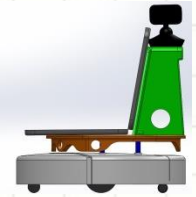
$$\mathbf{x}_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_k \\ V_k \end{bmatrix} + \begin{bmatrix} 0 \\ T_s \end{bmatrix} (a_k + w_a) \qquad \mathbf{y}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_k \\ V_k \end{bmatrix} + v_P$$

- Now we have some assumption of the noises:

$$\mathbf{w}_a \sim N(0, 0.01), \quad \mathbf{w} = \mathbf{w}_a \times T_s \sim N(0, 0.01 \times T_s^2), \quad \mathbf{v}_P \sim N(0, 4)$$

- Let's assume the initial conditions are (could be better or worse):

$$\mathbf{x}_0 = \begin{bmatrix} 10 \\ 2 \end{bmatrix}, \quad P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

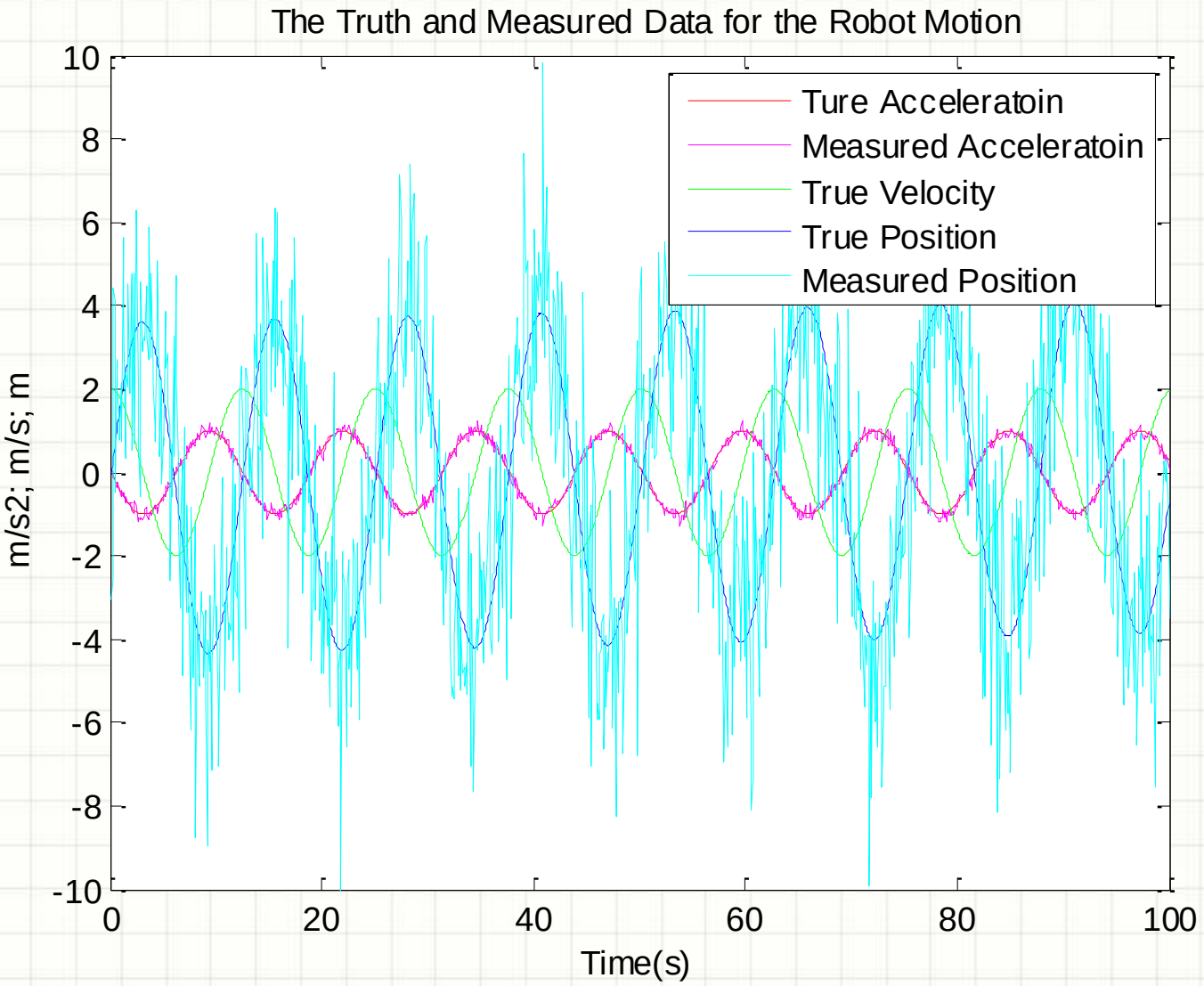


1D KF for Navigation (Cont.)

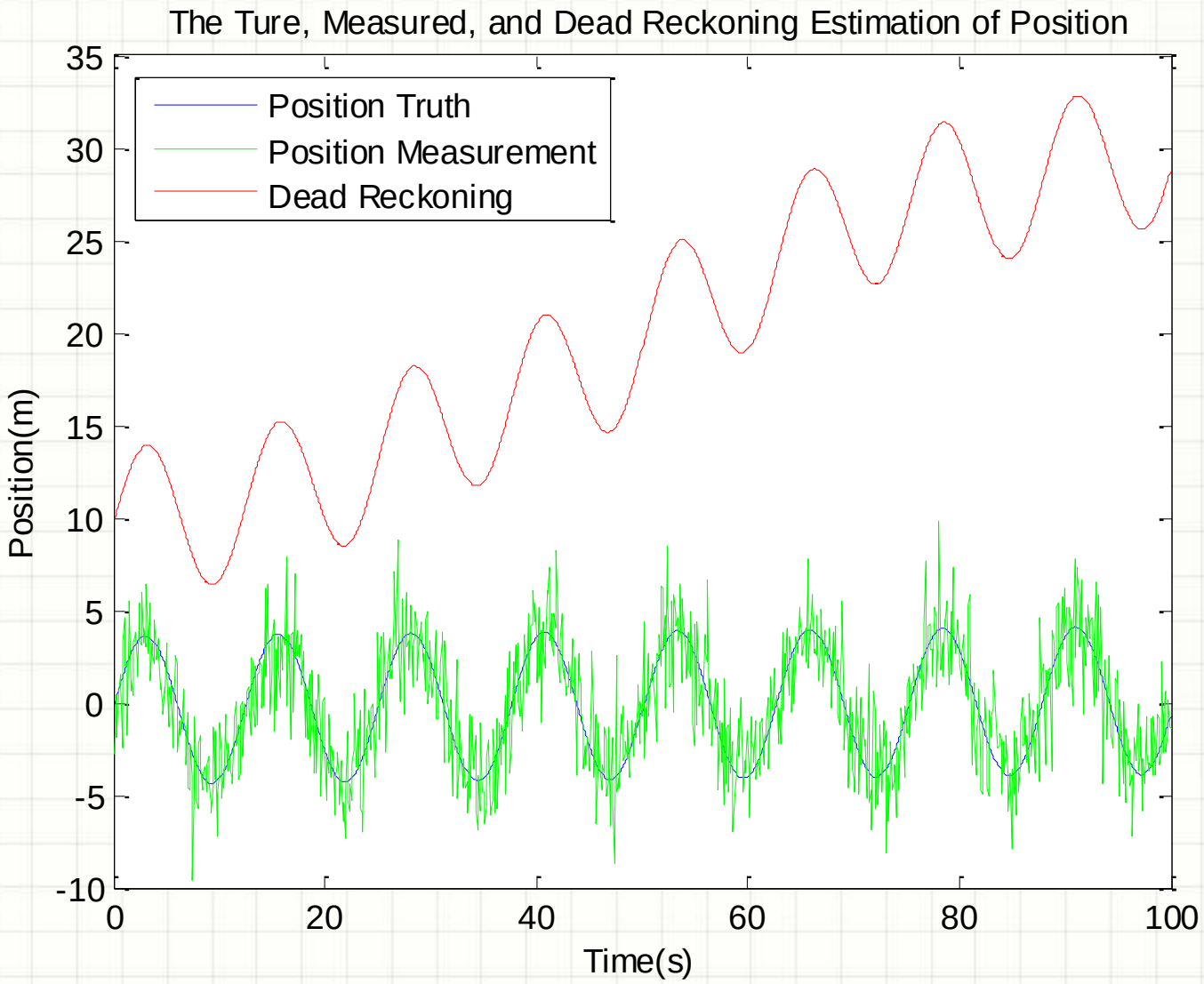
- The first step is to simulate the system without uncertainty and then add noises to the measurements:

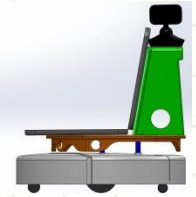
```
Ts=0.1;                                          % Sampling time
N=1000;                                          % Total steps of simulation
Time=zeros(N,1);                                 % The time vector
Position_T = zeros(N,1);                         % The truth for position
Velocity_T = 2*ones(N,1);                        % The truth for velocity
Acceleration_T=zeros(N,1);                       % The truth for acceleration
Position_M=zeros(N,1);                           % The measured position
Acceleration_M=zeros(N,1);                       % The measured acceleration
Position_E=zeros(N,1);                           % The estimated position
Velocity_E=zeros(N,1);                           % The estimated velocity

%% Generate the Truth Data and the Simulated Measurements
for k=1:N
    Time(k)=k*Ts;
    Acceleration_T(k)=-sin(Time(k)/2);           % Start with the acceleration
    if k>1
        Velocity_T(k)=Velocity_T(k-1)+Acceleration_T(k)*Ts;
        Position_T(k)=Position_T(k-1)+Velocity_T(k)*Ts;
    end
    Position_M(k)=Position_T(k)+normrnd(0,2);
    Acceleration_M(k)=Acceleration_T(k)+normrnd(0,0.1);
end
```

1D KF for Navigation (Cont.)



1D KF for Navigation (Cont.)

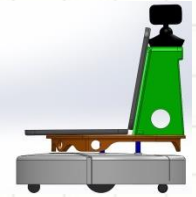




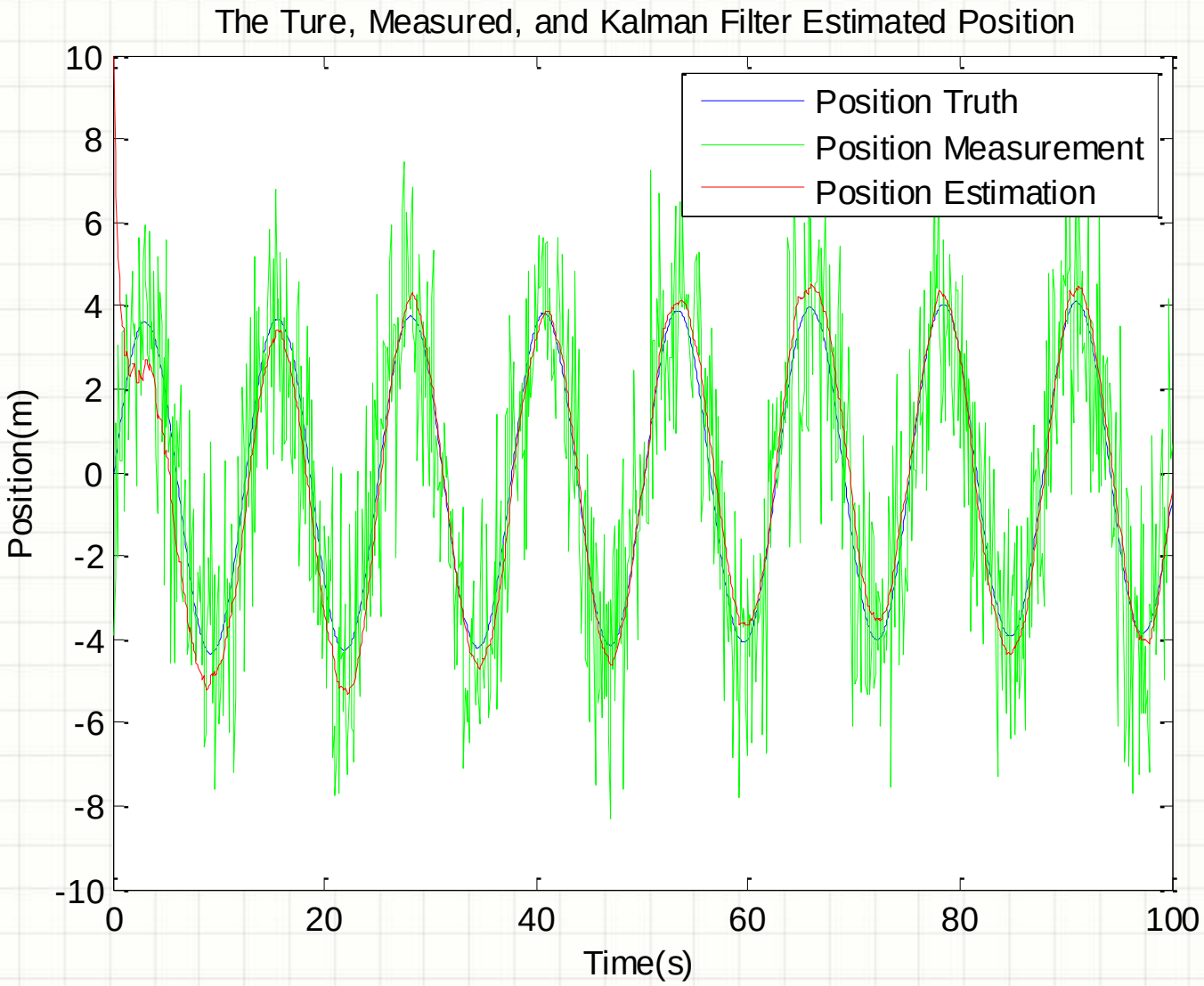
1D KF for Navigation (Cont.)

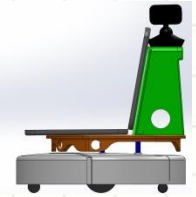
- Now the Kalman Filter Part:

```
%Initialization for the KF
A=[1   Ts
   0   1];                                  % State Transition Matrix
B=Ts*[0 1]';                                % Input Matrix
C=[1 0];                                    % Measurement Matirix
Q=[0   0
   0   0.01]*Ts^2;                          % Process noise covariance
R= 4;                                       % Measurement noise variance
I=eye(2);                                   % Identity matrix
x_P=[10 2]';                                % Initial estimate of the states
Position_E(1)=x_P(1);
Velocity_E(1)=x_P(2);
P_P=eye(2);                                 % Initial estimated error covariance
for k=2:N
    x_A=A*x_P+B*Acceleration_M(k);          % calculate the a priori states
    P_A=A*P_P*A'+Q;                         % calculate the a priori error covariance
    K_k=P_A*C'*inv(C*P_A*C'+R);             % calculate the Kalman gain
    x_P=x_A+K_k*(Position_M(k)-C*x_A);% calculate the posterior states
    P_P=(I-K_k*C)*P_A;                      % calculate the posterior error covariance
    % Get the states for plotting
    Position_E(k)=x_P(1);
    Velocity_E(k)=x_P(2);
end
```

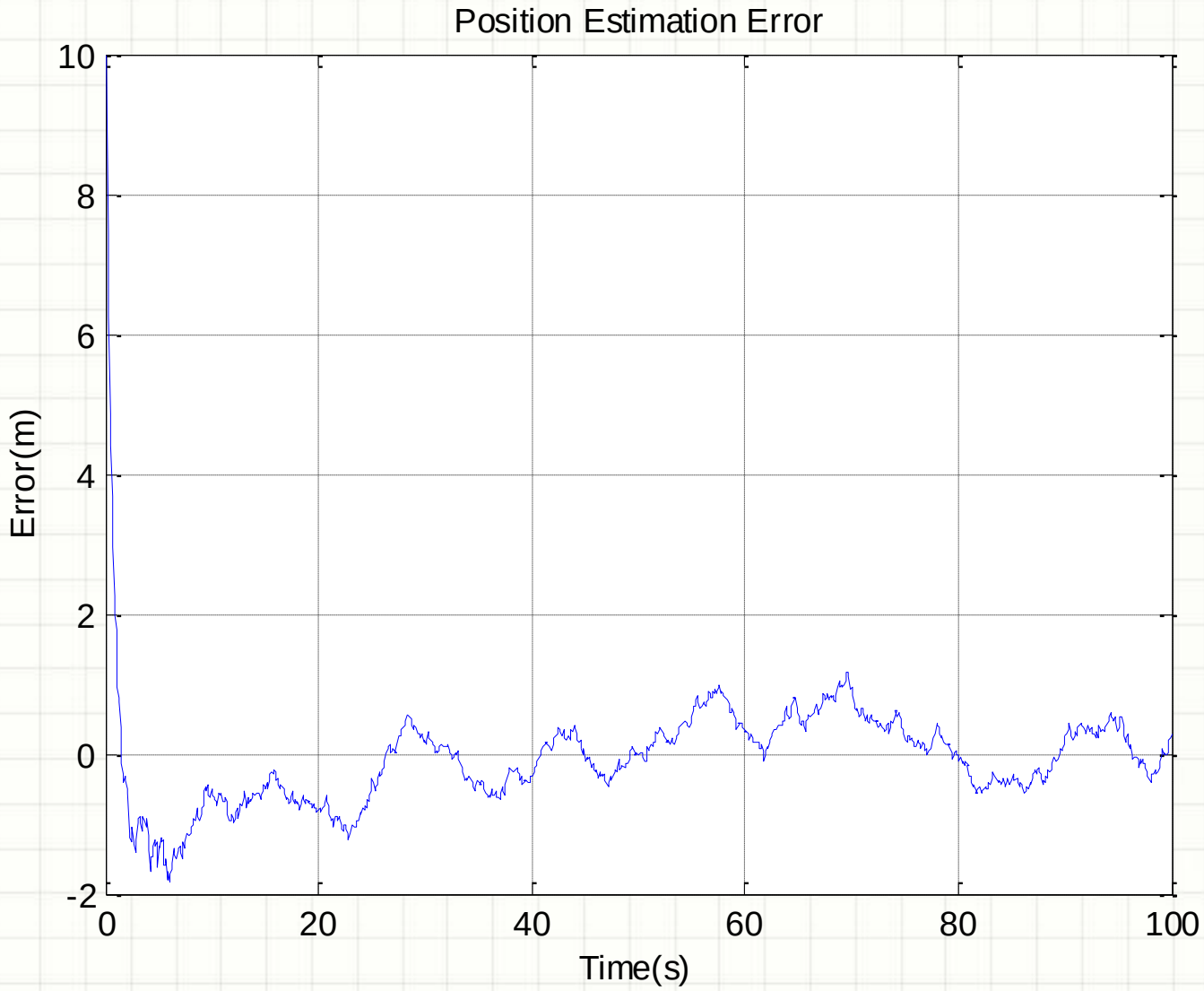


1D KF for Navigation (Cont.)





1D KF for Navigation (Cont.)



Implementation Issues

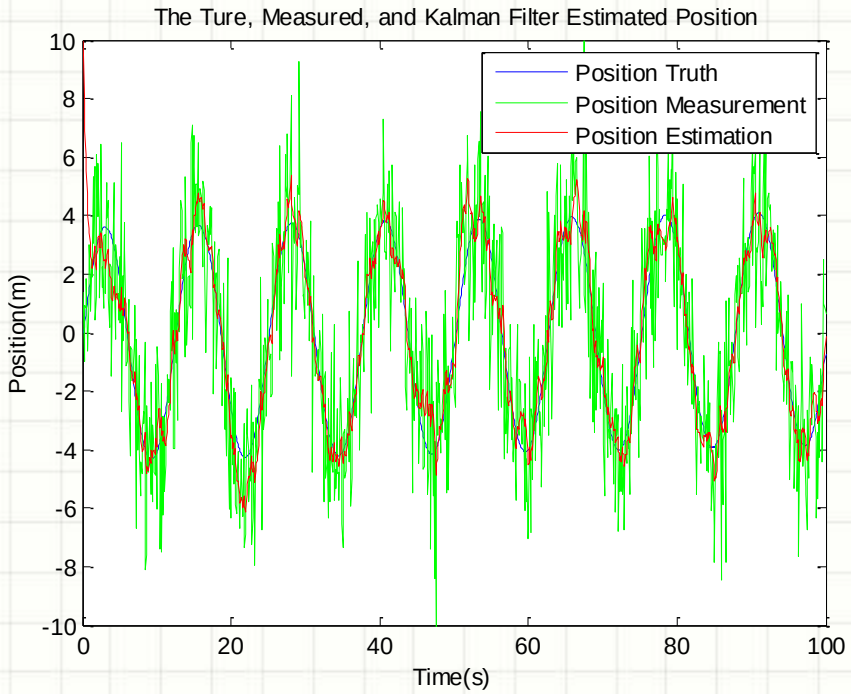
- *Initial values* of the state $\mathbf{x}_0$ and error covariance $\mathbf{P}_0$: the better your initial guess is, the quicker the Kalman filter will converge!
- Process noise: process noise covariance matrix **Q** is often hard to determine;
- *Tuning* can be used, especially if you can not find the right value of **Q** intuitively. Once you start tuning the filter, **Q** and **R** will start to lose their meaning as covariance matrices; they become tuning knobs...
- The higher the **Q** (with respect to **R**) value is, the more the filter will trust the measurement;
- The higher the **R** value is, the more the filter will trust the prediction;
- We can see the examples in the next page.

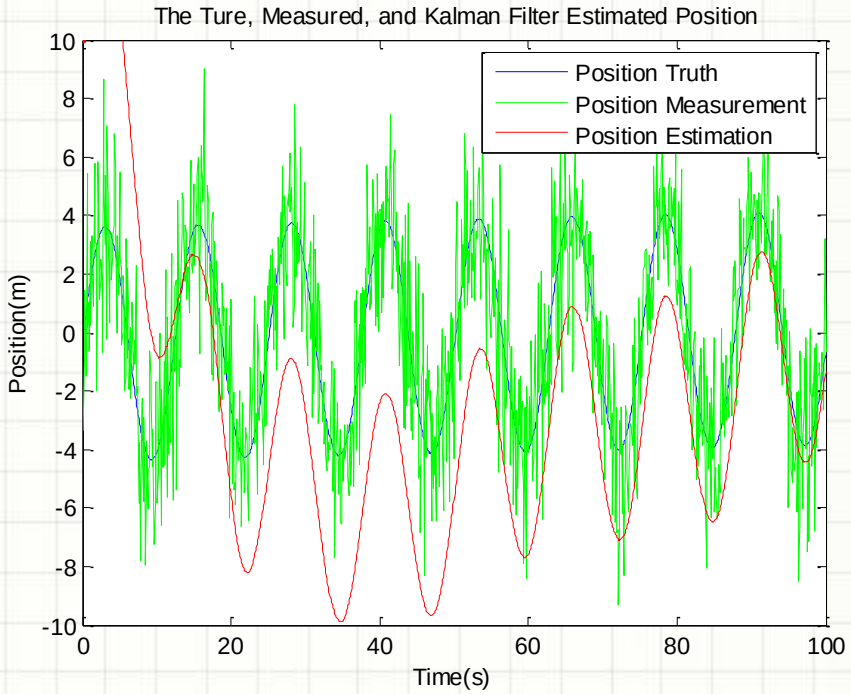
Kalman Filter Tuning

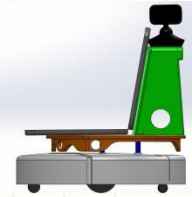
- The same filter that we had earlier but with different **Q** and **R**:

$$\mathbf{Q}_{new} = 1000\mathbf{Q}_{old}$$
$$\mathbf{R}_{new} = \mathbf{R}_{old}$$

$$\mathbf{Q}_{new} = \mathbf{Q}_{old}$$
$$\mathbf{R}_{new} = 1000\mathbf{R}_{old}$$

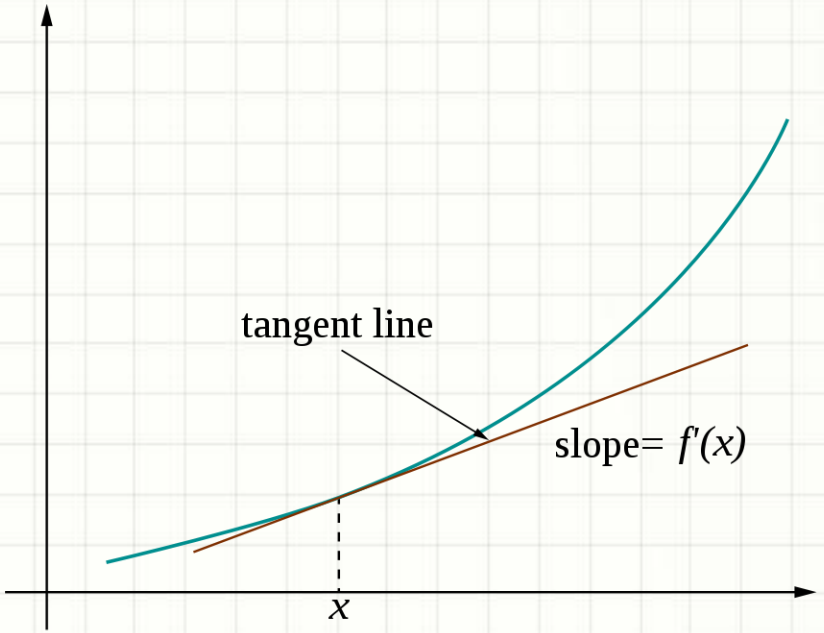


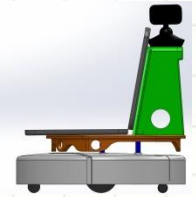




Nonlinear Kalman Filter

- If the dynamic system or the measurement equation is nonlinear (which is the case most of the time), things can get much harder;
- It's no longer easy to calculate the mean and covariance after transforming a random distribution through a nonlinear function;
- A Gaussian distribution may no longer be Gaussian after a nonlinear transformation as well;
- But for cases with relatively benign nonlinearity, we can make some simplifications;
- For example, we can try to linearize a nonlinear problem and then solve the linearized system with linear tools (works most of the time!).





Linearization

- Linearization finds the linear approximation to a nonlinear function at a given point;
- Linearization makes it possible to use tools for studying linear systems to analyze the behavior of a nonlinear function near a given point;
- The linearization of a function is the first order term of its *Taylor expansion* around the point of interest;
- For a system defined by the equation: $\dot{\mathbf{x}} = F(\mathbf{x}, t)$, it can be linearized around point $\mathbf{x}_0$ as: $\dot{\mathbf{x}} \approx F(\mathbf{x}_0, t) + DF(\mathbf{x}_0, t) \cdot (\mathbf{x} - \mathbf{x}_0)$
- DF($\mathbf{x}_0$,t) is the Jacobian matrix of F($\mathbf{x}$,t) evaluated at $\mathbf{x}_0$;
- Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function.
- Suppose F: $R^n \rightarrow R^m$ has *m* real-valued component functions: $F_1(x_1,...,x_n),...,F_m(x_1,...,x_n)$ the Jocobian **J** can be calculated with:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}$$

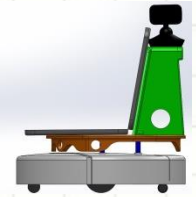
Extended Kalman Filter

- The Extended Kalman Filter (EKF) is an nonlinear descendent of the Kalman filter, which *linearizes about an estimate of the current mean*;
- For the following nonlinear state transition and observation functions:

$$\mathbf{x}_k = f\left(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}\right) + \mathbf{w}_{k-1}$$

$$\mathbf{y}_k = h\left(\mathbf{x}_k\right) + \mathbf{v}_k$$

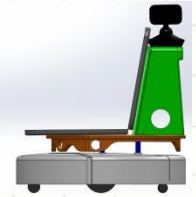
- The function *f* can be used to compute the predicted state from the previous estimate and similarly the function *h* can be used to compute the predicted measurement from the predicted state. However, *f* and *h* cannot be applied to the calculation of covariance directly. Instead the Jacobian matrices are computed;
- At each time step, the Jacobian is evaluated with current predicted states. These matrices can be used in the Kalman filter equations. This process essentially linearizes the non-linear function around the current estimate.



EKF Equations

	EKF	KF
Models:	$\mathbf{x}_k = f\left(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}\right) + \mathbf{w}_{k-1}$	$\mathbf{x}_k = \mathbf{A}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_{k-1} + \mathbf{w}_{k-1}$
	$\mathbf{y}_k = h\left(\mathbf{x}_k\right) + \mathbf{v}_k$	$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{v}_k$
Predict:		
Predicted State Estimate:	$\hat{\mathbf{x}}_{k\|k-1} = f\left(\hat{\mathbf{x}}_{k-1\|k-1}, \mathbf{u}_{k-1}\right)$	$\hat{\mathbf{x}}_{k\|k-1} = \mathbf{A}_k\hat{\mathbf{x}}_{k-1\|k-1} + \mathbf{B}_k\mathbf{u}_{k-1}$
Predicted Error Covariance:	$\mathbf{P}_{k\|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1\|k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_k$	$\mathbf{P}_{k\|k-1} = \mathbf{A}_k\mathbf{P}_{k-1\|k-1}\mathbf{A}_k^T + \mathbf{Q}_k$
Update:		
Innovation or Residual:	$\tilde{\mathbf{r}}_k = \mathbf{y}_k - h\left(\hat{\mathbf{x}}_{k\|k-1}\right)$	$\tilde{\mathbf{r}}_k = \mathbf{y}_k - \mathbf{C}_k\hat{\mathbf{x}}_{k\|k-1}$
Innovation Covariance:	$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k\|k-1}\mathbf{H}_k^T + \mathbf{R}_k$	$\mathbf{S}_k = \mathbf{C}_k\mathbf{P}_{k\|k-1}\mathbf{C}_k^T + \mathbf{R}_k$
Kalman Gain:	$\mathbf{K}_k = \mathbf{P}_{k\|k-1}\mathbf{H}_k^T\mathbf{S}_k^{-1}$	$\mathbf{K}_k = \mathbf{P}_{k\|k-1}\mathbf{C}_k^T\mathbf{S}_k^{-1}$
Posterior State Estimate:	$\hat{\mathbf{x}}_{k\|k} = \hat{\mathbf{x}}_{k\|k-1} + \mathbf{K}_k\tilde{\mathbf{r}}_k$	$\hat{\mathbf{x}}_{k\|k} = \hat{\mathbf{x}}_{k\|k-1} + \mathbf{K}_k\tilde{\mathbf{r}}_k$
Posterior Error Covariance:	$\mathbf{P}_{k\|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k\|k-1}$	$\mathbf{P}_{k\|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\mathbf{P}_{k\|k-1}$

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

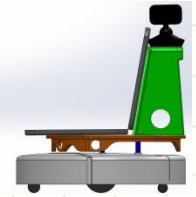


A Simple Pendulum w/ Friction

- A simple unforced pendulum can be modeled as: $ml^2\ddot{\theta}(t) = -mg\sin\left(\theta(t)\right)l - cl\dot{\theta}(t)$
- Where $\theta(t)$ is the angle of the pendulum with respect to the direction of gravity; $m=1kg$ is the mass of the pendulum (pendulum rod's mass is assumed to be zero); g is the gravitational acceleration; $c=0.5$ is coefficient of friction at the pivot point, and $l=0.5m$ is the radius of the pendulum (to the center of gravity of the mass m);
- If we define $x_1(t) = \theta(t),\ x_2(t) = \dot{\theta}(t)$,we will have the following relationships:

$$\dot{x}_1(t) = x_2(t), \quad \dot{x}_2(t) = \ddot{\theta}(t) = -\frac{g}{l}\sin\left(x_1(t)\right) - \frac{c}{m}x_2(t)$$

- The state space representation is then: $\dot{\mathbf{x}} = f_c(\mathbf{x}) = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin\left(x_1\right) - \frac{c}{m}x_2 \end{bmatrix}$
- If we measure the horizontal distance d between the pendulum and the vertical line through the pivot point, we will have a nonlinear measurement equation as well: $\mathbf{y} = h_c(\mathbf{x}) = l\sin\theta = l\sin\left(x_1\right)$



Simple Pendulum (Cont.)

- Discretize the state transition and measurement equations, we have:

$$\mathbf{x}(k) = f_d(\mathbf{x}) = \begin{bmatrix} x_1(k-1) + T_s x_2(k-1) \\ x_2(k-1) - T_s \frac{g}{l}\sin\left(x_1(k-1)\right) - T_s \frac{c}{m} x_2(k-1) \end{bmatrix}$$

$$\mathbf{y}(k) = h_d(\mathbf{x}) = l\sin\left(x_1(k)\right)$$

- Calculating the Jacobians:

$$\mathbf{F}_{k-1} = \left.\frac{\partial f}{\partial x}\right|_{\mathbf{x}_{k-1}} = \begin{bmatrix} 1 & T_s \\ -T_s \frac{g}{l}\cos\left(x_1(k-1)\right) & 1 - T_s \frac{c}{m} \end{bmatrix}$$

$$\mathbf{H}_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{\mathbf{x}}_k} = \begin{bmatrix} l\cos\left(x_1(k)\right) & 0 \end{bmatrix}$$

- Let's assume the process and measurement noises are independent from each other, white, and Gaussian, with:

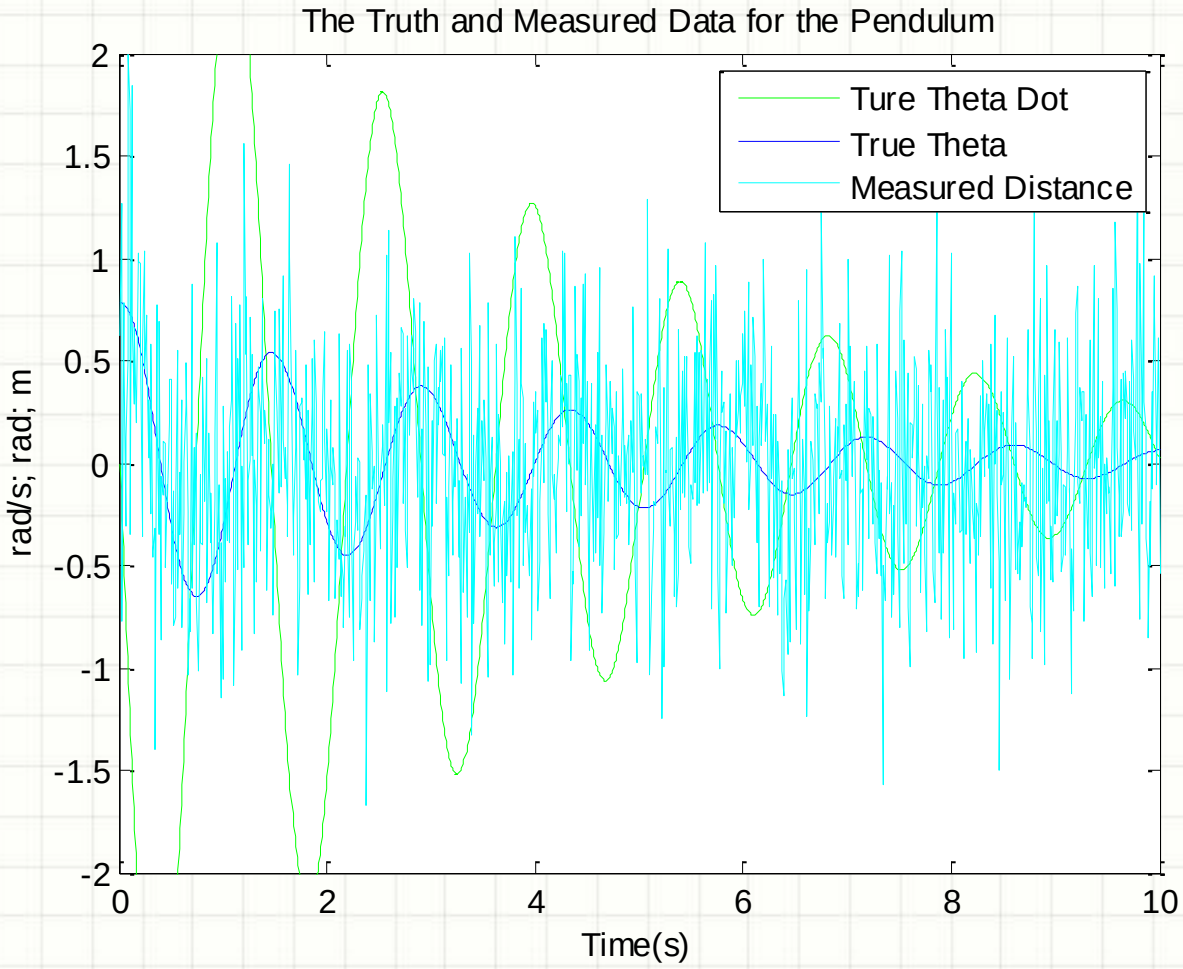
$$\mathbf{w} \sim N(0, \mathbf{Q}), \quad \mathbf{v} \sim N(0, \mathbf{R}), \quad \mathbf{Q} = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}, \quad \mathbf{R} = 0.25$$

- Rewrite the state space models with noises: $\mathbf{x}(k) = f_d(\mathbf{x}) + \mathbf{w}$

$$\mathbf{y}(k) = h_d(\mathbf{x}) + \mathbf{v}$$

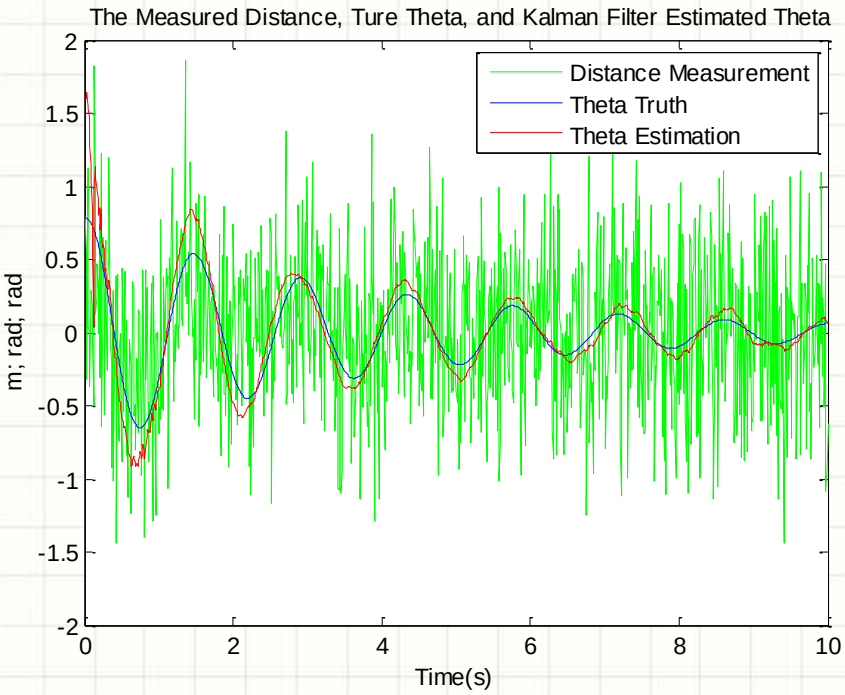
Simple Pendulum Simulation

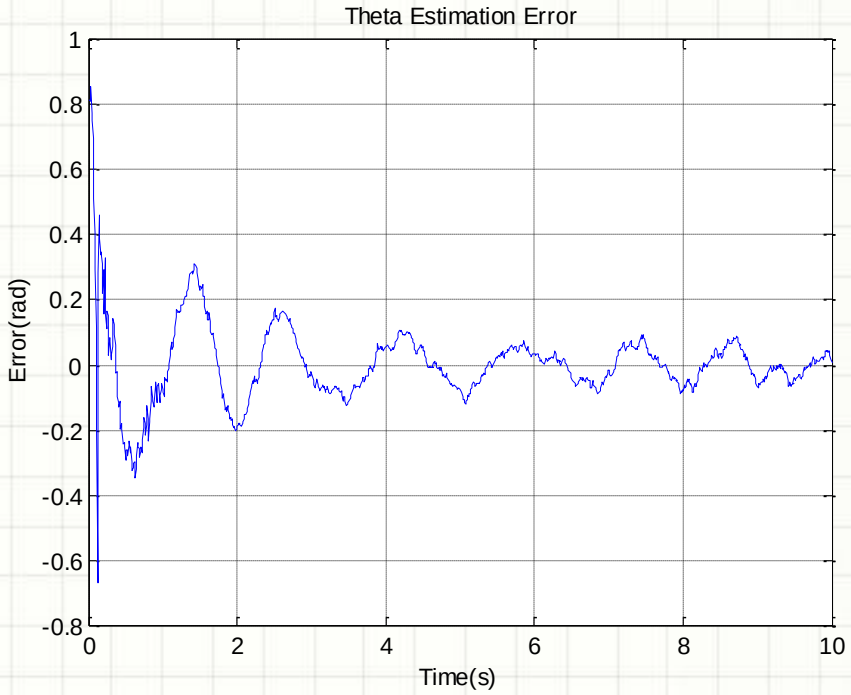
- The true angle, angular rate, and the measured distance:



Simple Pendulum Estimation

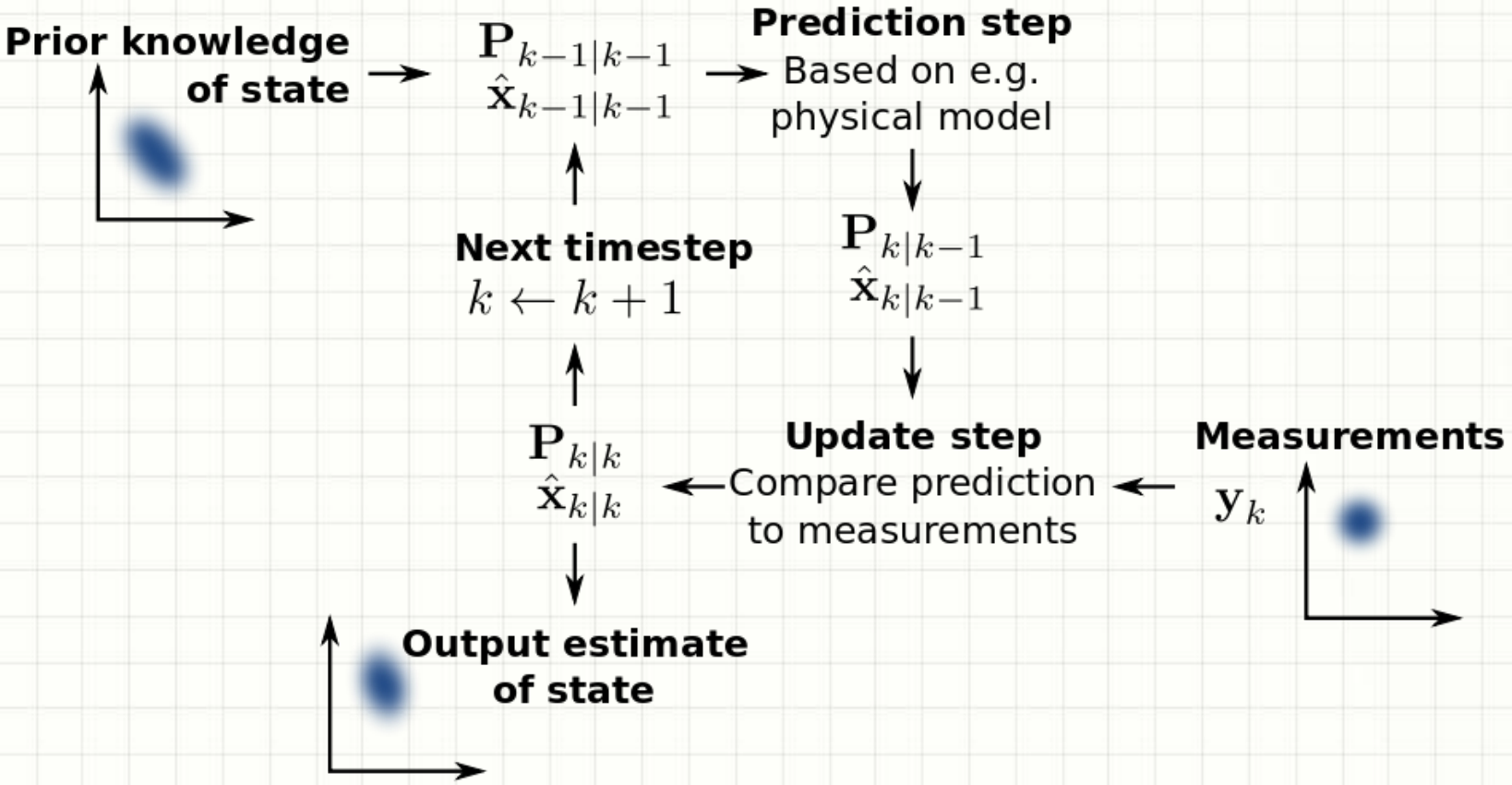
- Initial Conditions: $x_1 = 1.6\text{rad}, \quad x_2 = 0.0\text{rad/s}$
- EKF is not guaranteed to be stable! Sometimes if there is too much nonlinearity in the system, the error assumptions are wrong, or the initial guess is way off, the EKF may diverge!
- EKF is not a optimal state estimator as well.





Kalman Filter Reviewed

- (Linear) Kalman Filter is a optimal recursive stochastic state estimator that works with a linear dynamic system with the process and measurement noises assumed to be independent from each other, white, and Gaussian.



Summary

- Now we know how to average out errors if the measured parameter is time-varying;
- This is achieved through a clever use of prior knowledge, such as system dynamic equations (model of the system) and sensor noise statistics (stochastic model of sensor error);
- EKF allows us to tackle nonlinear problems but with some penalty (added computation requirement, lost optimality, and stability issues sometimes);
- EKF is a standard tool for robot, aircraft, and spacecraft navigation.

Further Reading

- An Introduction to the Kalman Filter, http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf
- Search Wikipedia for keywords: ‘State Space Representation’, ‘Discretization’, ‘Linearization’, ‘Kalman Filter’, and ‘Extended Kalman Filter’.