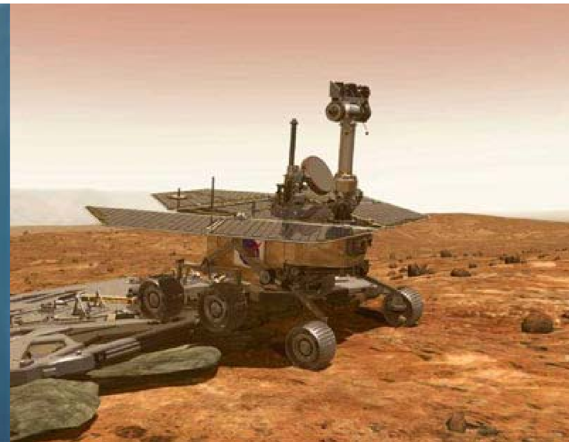
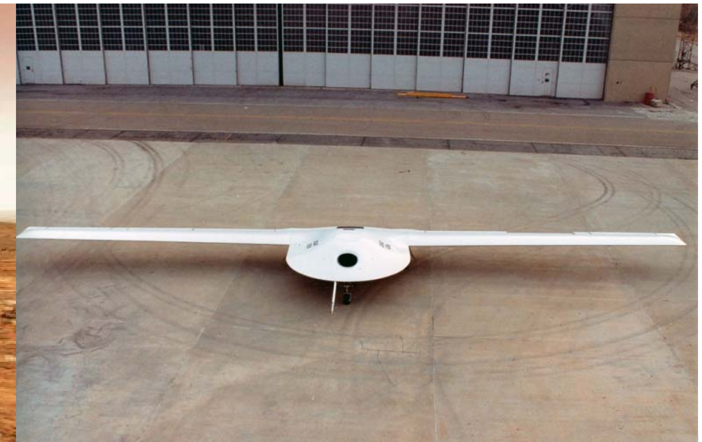
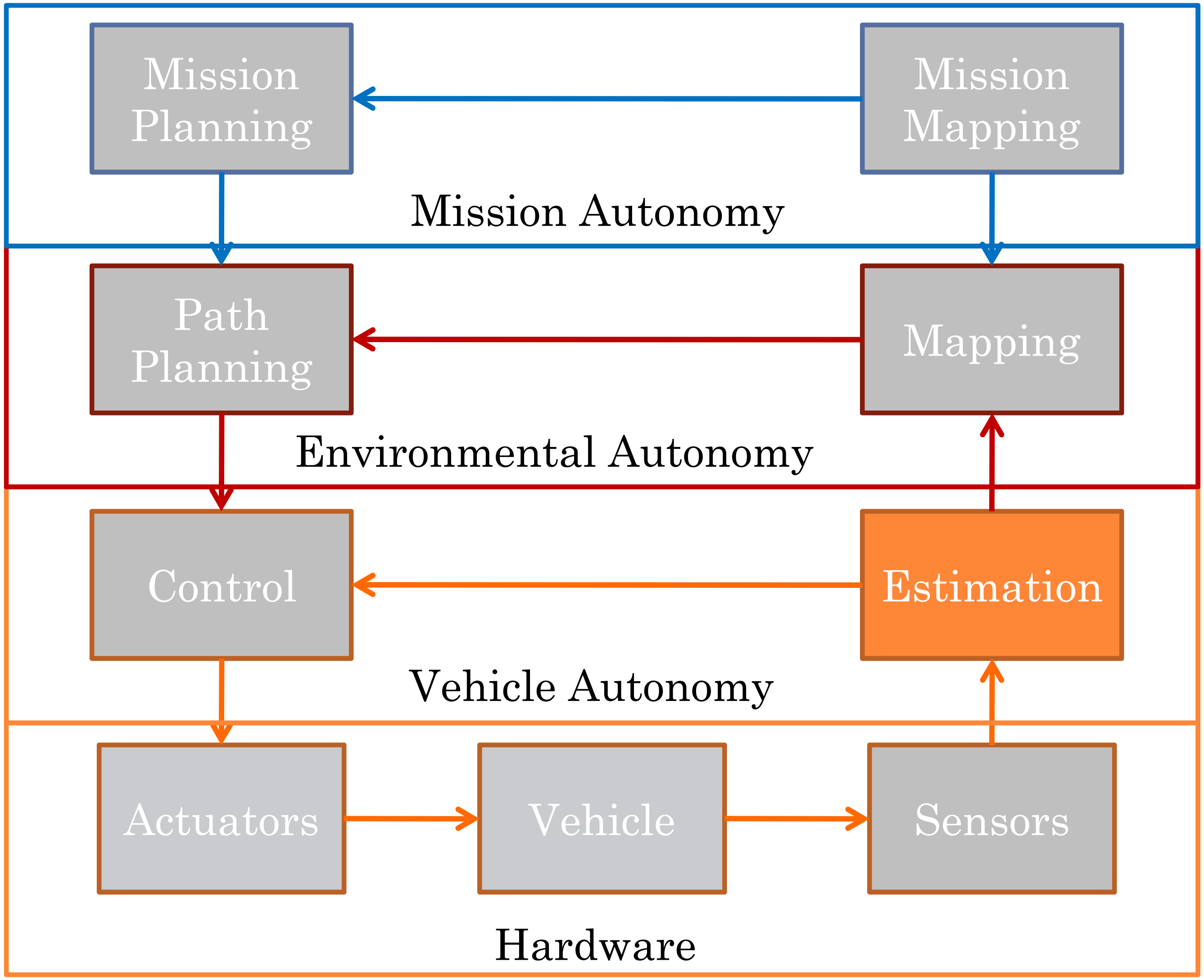





ME 597: Autonomous Mobile Robotics
Section 7 – Estimation I

Prof. Steven Waslander

Components


Mission Planning
Mission Mapping
Mission Autonomy
Path Planning
Mapping
Environmental Autonomy
Control
Estimation
Vehicle Autonomy
Actuators
Vehicle
Sensors
Hardware

Outline

- Bayes Filter Framework
- Kalman Filter
- Extended Kalman Filter
- Particle Filter

Bayes Filter

- The Bayes Filter forms the foundation for all other filters in this class
  - As described in background slides, Bayes rule is the right way to incorporate new probabilistic information into an existing, prior estimate
  - The resulting filter definition can be implemented directly for discrete state systems
  - For continuous states, need additional assumptions, additional structure to solve the update equations analytically

Bayes Filter

- State x_t
  - All aspects of the vehicle and its environment that can impact the future
  - **Assume the state is complete**
- Control inputs u_t
  - All elements of the vehicle and its environment that can be controlled
- Measurements y_t
  - All elements of the vehicle and its environment that can be sensed
- Note: sticking with Thrun, Burgard, Fox notation
  - Discrete time index t
  - Initial state is x_0
  - First, apply control action u_1
  - Move to state x_1
  - Then, take measurement y_1

Bayes Filter

- Motion Modeling
  - Complete state:
    - At each time t, x_{t-1} is a sufficient summary of all previous inputs and measurements

$$p(x_t \mid x_{0:t-1}, y_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

    - Application of Conditional Independence
      - No additional information is to be had by considering previous inputs or measurements
    - Referred to as the Markov Assumption
      - Motion model is a Markov Chain

Bayes Filter

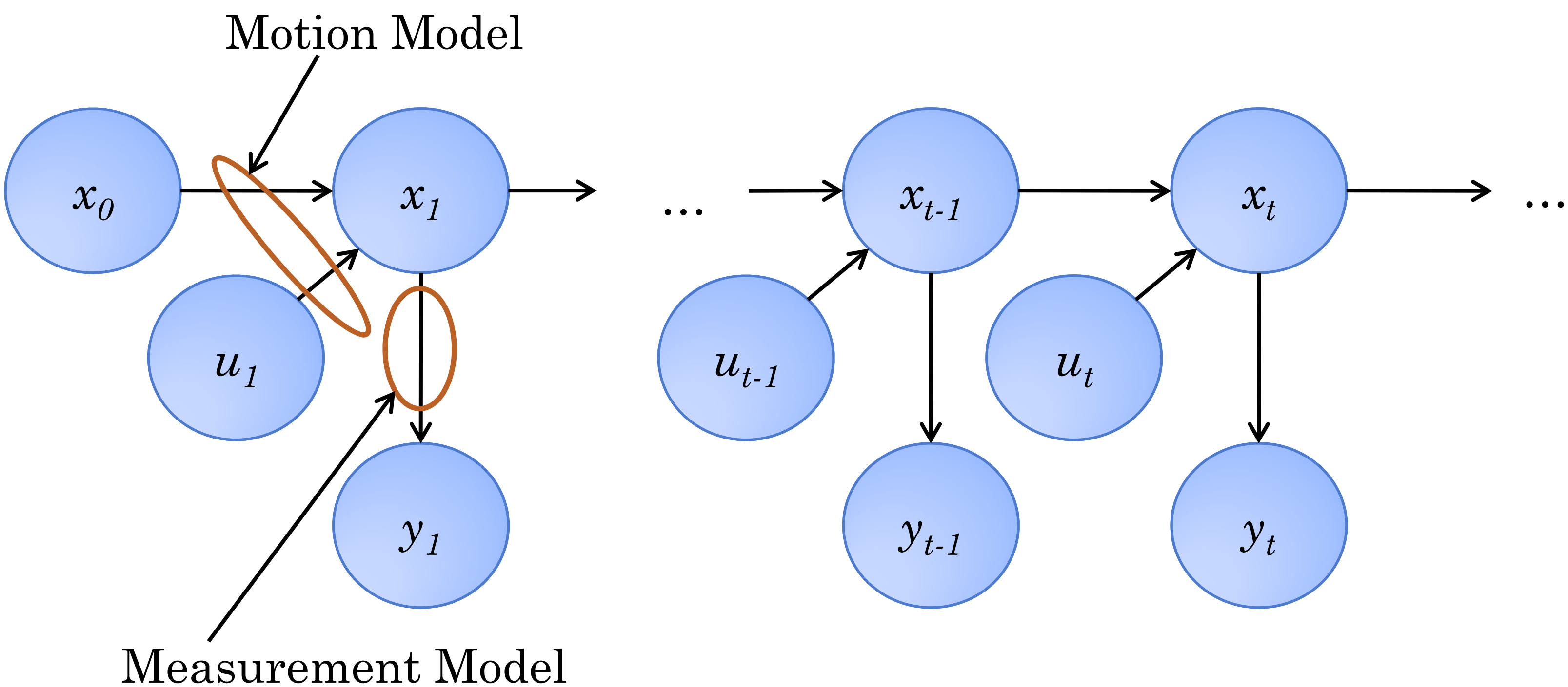
- Measurement Modeling
  - Complete state:
    - Current state is sufficient to model all previous states, measurements and inputs

$$p(y_t \mid x_{0:t}, y_{1:t-1}, u_{1:t}) = p(y_t \mid x_t)$$

    - Again, conditional independence
    - Recall, in standard LTI state space model, measurement model may also depend on the current input

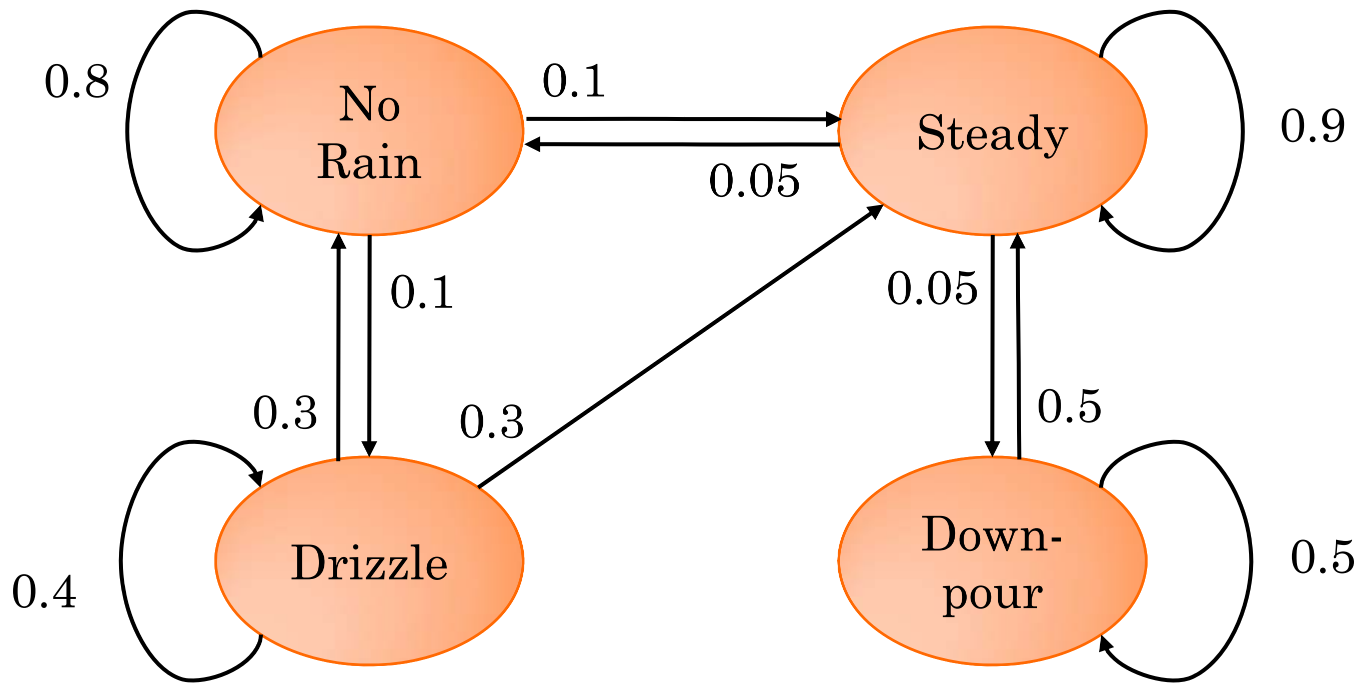
Bayes Filter

- Combined Model
  - Referred to as Hidden Markov Model (HMM) or Dynamic Bayes Network (DBN)



Bayes Filter

- Example Discrete State Motion Model:
  - States: *{No Rain, Drizzle, Steady, Downpour}*
  - Inputs: None



Bayes Filter

- For discrete states, the motion model can be written in matrix form
  - For each input u_t, the nXn motion model matrix is

$$p(x_t \mid u_t = u, x_{t-1}) =$$

$$\begin{bmatrix} p(x_t = x_1 \mid x_{t-1} = x_1) & p(x_t = x_1 \mid x_{t-1} = x_2) & \cdots \\ p(x_t = x_2 \mid x_{t-1} = x_1) & p(x_t = x_2 \mid x_{t-1} = x_2) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Each row defines the probabilities of transitioning to state x_t from all possible states x_{t-1}
- Each column defines the probabilities of transitioning to any state x_t from a specific state x_{t-1}
- Again, the columns must sum to 1

Bayes Filter

- Example:
  - Motion Model in Matrix Form
    - No inputs, one matrix

$$p(x_t \mid u_t, x_{t-1}) = \overbrace{\begin{bmatrix} 0.8 & 0.3 & 0.05 & 0 \\ 0.1 & 0.4 & 0 & 0 \\ 0.1 & 0.3 & 0.9 & 0.5 \\ 0 & 0 & 0.05 & 0.5 \end{bmatrix}}^{x_{t-1}} \Bigg\} x_t$$

Bayes Filter

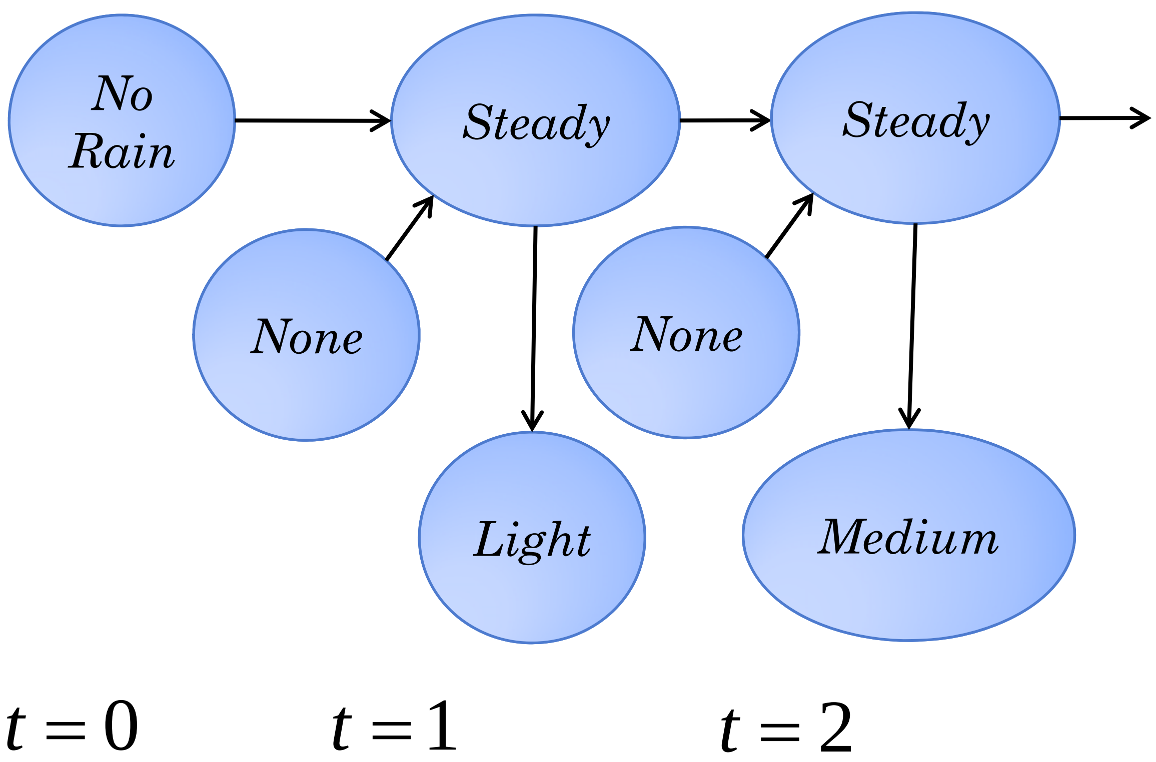
- Example Measurement Model:
  - States: *{No Rain, Drizzle, Steady, Downpour}*
  - Measurements: *{Dry, Light, Medium, Heavy}*

$$p(y_t \mid x_t) = \overbrace{\begin{bmatrix} 0.95 & 0.1 & 0 & 0 \\ 0.05 & 0.8 & 0.15 & 0 \\ 0 & 0.1 & 0.7 & 0.1 \\ 0 & 0 & 0.15 & 0.9 \end{bmatrix}}^{x_t} \left.\vphantom{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}\right\} y_t$$

  - Again, the columns sum to 1

Bayes Filter

- Example System Evolution



Bayes Filter

- Aim of Bayes Filter
  - To estimate the current state of the system based on all known inputs and measurements.

  - That is, to define a belief about the current state using all available information:

$$bel(x_t) = p(x_t \mid y_{1:t}, u_{1:t})$$

    - Known as belief, state of knowledge, information state
    - Depends on every bit of information that exists up to time t

  - Can also define a belief prior to measurement y_t

$$\overline{bel}(x_t) = p(x_t \mid y_{1:t-1}, u_{1:t})$$

    - Known as prediction, predicted state

Bayes Filter

- Problem Statement
  - Given a prior for the system state

$$p(x_0)$$

  - Given motion and measurement models

$$\overbrace{p(x_t \mid x_{t-1}, u_t)} \qquad \overbrace{p(y_t \mid x_t)}$$

  - Given a sequence of inputs and measurements

$$u_{1:t} = \{u_1, \ldots, u_t\}, \quad y_{1:t} = \{y_1, \ldots, y_t\}$$

  - Estimate the current state distribution (form a belief about the current state)

$$bel(x_t) = p(x_t \mid y_{1:t}, u_{1:t})$$

Bayes Filter

- Bayes Filter Algorithm
  - At each time step, t, for all possible values of the state x
    1. Prediction update (Total probability)

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

    2. Measurement update (Bayes Theorem)

$$bel(x_t) = \eta p(y_t \mid x_t) \overline{bel}(x_t)$$

      - η is a normalizing constant that does not depend on the state (will become apparent in derivation)

  - Recursive estimation technique

Bayes Filter

- Recall Bayes Theorem

$$p(a \mid b) = \frac{p(b \mid a)\, p(a)}{p(b)}$$

  - Terminology

$$posterior = \frac{likelihood \cdot prior}{evidence}$$

Bayes Filter

- Derivation
  - Proof by induction
    - Demonstrate that belief at time *t* can be found using belief at time *t-1*, input at *t* and measurement at *t*
  - Initially

$$bel(x_0) = p(x_0)$$

  - At time *t*, Bayes Theorem relates x_t, y_t

$$bel(x_t) = p(x_t \mid y_{1:t}, u_{1:t}) = p(x_t \mid y_t, y_{1:t-1}, u_{1:t})$$

1 2

$$= \frac{p(y_t \mid x_t, y_{1:t-1}, u_{1:t})\, p(x_t \mid y_{1:t-1}, u_{1:t})}{p(y_t \mid y_{1:t-1}, u_{1:t})}$$

3

Bayes Filter

- Derivation
  1. Measurement model simplifies first numerator term

$$p(y_t \mid x_t, y_{1:t-1}, u_{1:t}) = p(y_t \mid x_t)$$

  2. Second numerator term is definition of belief prediction

$$\overline{bel}(x_t) = p(x_t \mid y_{1:t-1}, u_{1:t})$$

  3. Denominator is independent of state, and so is constant for each time step. Define the normalizer,

$$\eta = p(y_t \mid y_{1:t-1}, u_{1:t})^{-1}$$

Bayes Filter

- Derivation
  - Summarizing the three substitutions

$$bel(x_t) = \eta p(y_t \mid x_t)\overline{bel}(x_t)$$

    - This is exactly the measurement update step
    - Requires the measurement y_t to be known

  - However, we now need to find the belief prediction
    - Done using total probability, over previous state

$$\overline{bel}(x_t) = p(x_t \mid y_{1:t-1}, u_{1:t})$$
$$= \int p(x_t \mid x_{t-1}, y_{1:t-1}, u_{1:t}) p(x_{t-1} \mid y_{1:t-1}, u_{1:t}) dx_{t-1}$$

1 2

BAYES FILTER

- Derivation
  1. This time, the motion model can be incorporated

$$p(x_t \mid x_{t-1}, y_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

  2. And we note that the control input at time *t* does not affect the state at time *t-1*

$$p(x_{t-1} \mid y_{1:t-1}, u_{1:t}) = p(x_{t-1} \mid y_{1:t-1}, u_{1:t-1}) = bel(x_{t-1})$$

Bayes Filter

- Derivation
  - And so the prediction update is defined

$$\overline{bel}(x_t) = \int p(x_t \mid x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$$

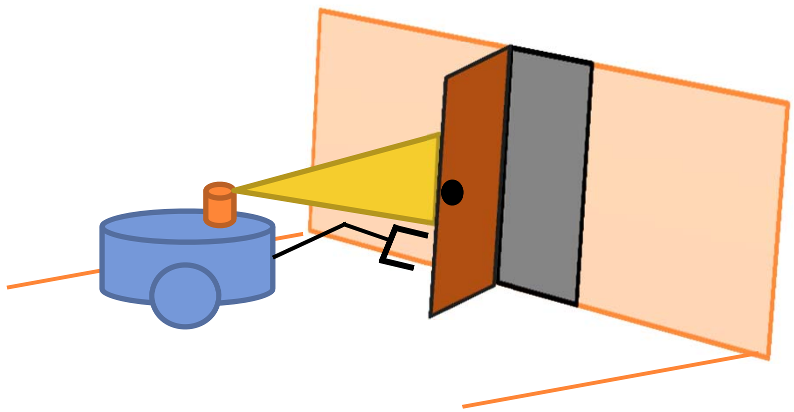
  - Which completes the proof by induction
    - For this step, we need the control input to define the correct motion model distribution
  - If state, measurements, inputs are discrete, can directly implement Bayes Filter
    - Prediction update is summation over discrete states
    - Measurement update is multiplication of two vectors

Bayes Filter

- If state, measurement, inputs are continuous, must define model or approximation to enable computation
  - Kalman Filter:
    - Linear motion models
    - Linear measurement models
    - Additive Gaussian disturbance and noise distributions
  - Extended Kalman Filter/Unscented Kalman Filter:
    - Nonlinear motion models
    - Nonlinear measurement models
    - Additive Gaussian disturbance and noise distributions
  - Particle Filter:
    - (Dis)continuous motion models
    - (Dis)continuous measurement models
    - General disturbance and noise distributions

Bayes Filter

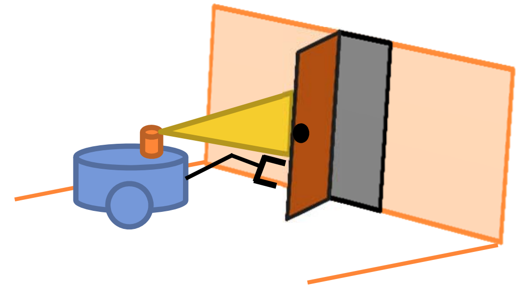
- Discrete Bayes Filter Example
  - Problem: Detect if a door is open/closed with a robot that can sense the door position and pull the door open



  - State: *door={open, closed}*
  - State Prior (uniform):

$$p(x_0) = \begin{cases} p(open) = 0.5 \\ p(closed) = 0.5 \end{cases}$$

Bayes Filter



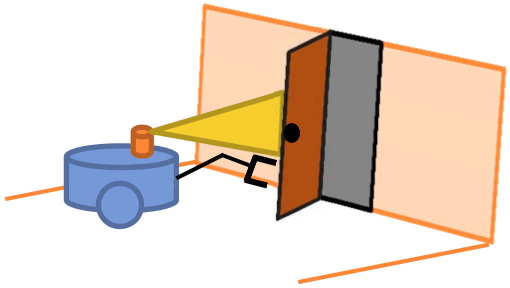
- Example
  - Inputs: *arm_command={none, pull}*
  - Motion Model
    - If input = *none*, do nothing:

$$p(x_t \mid u_t = none, x_{t-1}) \rightarrow \begin{cases} p(open_t \mid none, open_{t-1}) = 1 \\ p(closed_t \mid none, open_{t-1}) = 0 \\ p(open_t \mid none, closed_{t-1}) = 0 \\ p(closed_t \mid none, closed_{t-1}) = 1 \end{cases}$$

    - If input = *pull*, pull the door open

$$p(x_t \mid u_t = pull, x_{t-1}) \rightarrow \begin{cases} p(open_t \mid pull, open_{t-1}) = 1 \\ p(closed_t \mid pull, open_{t-1}) = 0 \\ p(open_t \mid pull, closed_{t-1}) = 0.8 \\ p(closed_t \mid pull, closed_{t-1}) = 0.2 \end{cases}$$

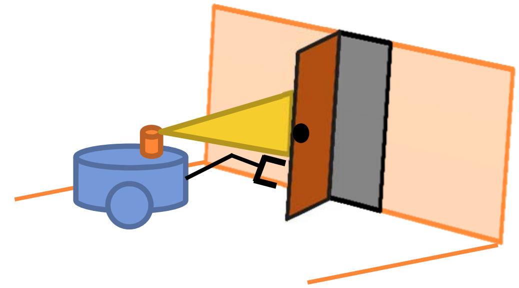
Bayes Filter



- Example
  - Measurements: *meas={sense_open, sense_closed}*
  - Measurement model (noisy door sensor):

$$p(y \mid x) \rightarrow \begin{cases} p(sense\_open \mid open) = 0.6 \\ p(sense\_open \mid closed) = 0.2 \\ p(sense\_closed \mid open) = 0.4 \\ p(sense\_closed \mid closed) = 0.8 \end{cases}$$

BAYES FILTER



- Example
  - At time step 1, input = *none*
  - Perform state prediction update

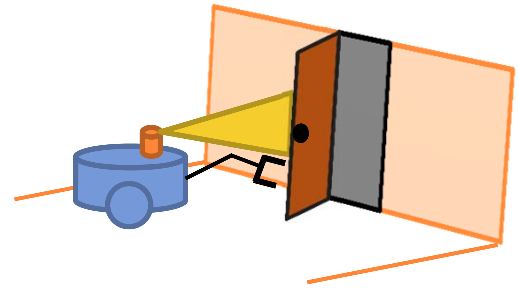
$$\overline{bel}(x_1) = \int p(x_1 \mid u_0, x_0) bel(x_0) dx_0$$
$$= \sum p(x_1 \mid u_0, x_0) p(x_0)$$

  - Calculate belief prediction for each possible value of state

$$\overline{bel}(open_1) = p(open_1 \mid none_1, open_0) bel(open_0)$$
$$+ p(open_1 \mid none_1, closed_0) bel(closed_0)$$
$$= 1 * 0.5 + 0 * 0.5 = 0.5$$

$$\overline{bel}(closed_1) = p(closed_1 \mid none_1, open_0) bel(open_0)$$
$$+ p(closed_1 \mid none_1, closed_0) bel(closed_0)$$
$$= 0 * 0.5 + 1 * 0.5 = 0.5$$

Bayes Filter



- Example
  - At time step 1,measurement $y_1 = sense\_open$
  - Measurement update

$$bel(x_1) = \eta\, p(y_1 \mid x_1)\overline{bel}(x_1)$$

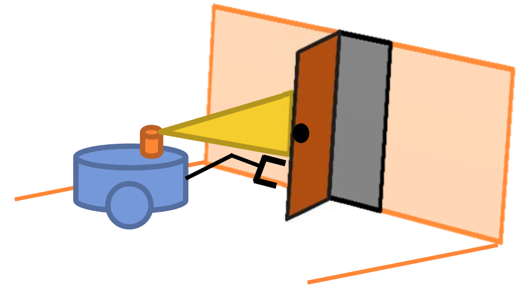
  - Calculate for each possible value of state

$$bel(open_1) = \eta\, p(sense\_open_1 \mid open_1)\overline{bel}(open_1)$$
$$= \eta 0.6 \cdot 0.5 = 0.3\eta$$
$$bel(closed_1) = \eta\, p(sense\_open_1 \mid closed_1)\overline{bel}(closed_1)$$
$$= \eta 0.2 \cdot 0.5 = 0.1\eta$$

  - Calculate normalizer and solve for posterior

$$\eta = \frac{1}{0.3 + 0.1} = 2.5 \quad \Rightarrow \quad bel(open_1) = 0.75, \quad bel(closed_1) = 0.25$$

Bayes Filter



- Example
  - At time step 2, a *pull* and a *sense_open*
  - Then state propagation

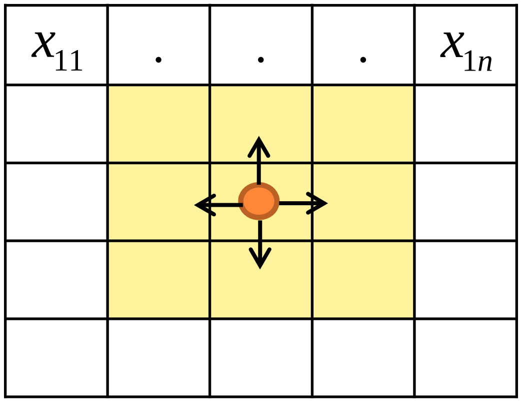
$$\overline{bel}(open_2) = 1 \cdot 0.75 + 0.8 \cdot 0.25 = 0.95$$
$$\overline{bel}(closed_2) = 0 \cdot 0.75 + 0.2 \cdot 0.25 = 0.05$$

  - And measurement update

$$bel(open_2) = \eta 0.6 \cdot 0.95 = 0.983$$
$$bel(closed_2) = \eta 0.2 \cdot 0.05 = 0.017$$

  - In summary:
    - Uniform prior, do nothing, measure open: $bel(open_1) = 0.75$
    - Pull open, measure open: $bel(open_2) = 0.983$

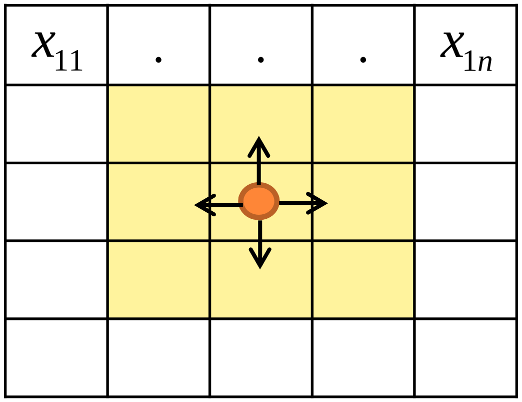
BAYES FILTER



- Example 2: Histogram Filter
  - Motion of robot in a *nXn* grid
    - State:
      - Position = $\{x_{11}, x_{12}, \ldots, x_{1n}, , \ldots, x_{nn}\}$
    - Input:
      - Move = {Up, Right, Down, Left}
      - 40% chance the move does not happen
      - Cannot pass through outer walls
    - Measurement: Accurate to within 3X3 grid

$$p\big(y(i-1:i+1, j-1:j+1) \mid x(i,j)\big) = \begin{bmatrix} .11 & .11 & .11 \\ .11 & .12 & .11 \\ .11 & .11 & .11 \end{bmatrix}$$

BAYES FILTER



- Example 2:
  - Prior over states
    - Assume no information, uniform
    - Vector of length n²

$$p(x_0) = \frac{1}{n^2}\boldsymbol{1}$$

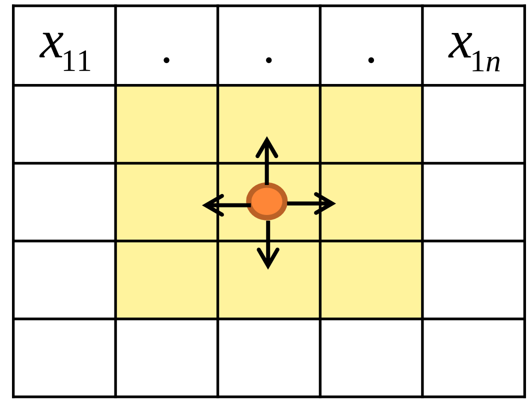
  - Motion model
    - Given a particular input and previous state, probability of moving to any other state
      - *nXn* state, one for each grid point
      - 4 input choices

$$p(x_t \mid x_{t-1}, u_t) \in [0,1]^{n^2 \times n^2 \times 4}$$

Bayes Filter

- Example 2
  - Measurement Model
    - Given any current state, probability of a measurement
    - Same number of measurements as states

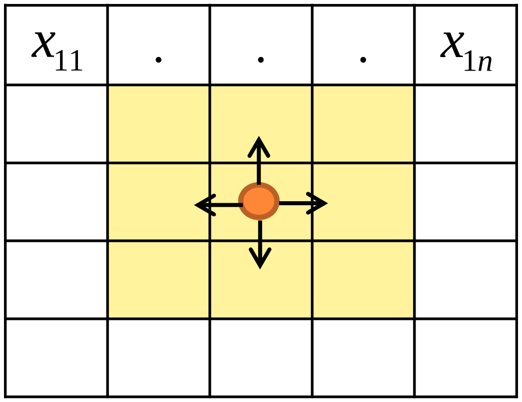
$$p(y_t \mid x_t) \in [0,1]^{n^2 \times n^2}$$



    - Same 3X3 matrix governs all interior points
    - Boundaries cut off invalid measurements and require normalization
    - Very simplistic and bloated model
      - Could replace with 2 separate states and measurements to perpendicular walls

Bayes Filter

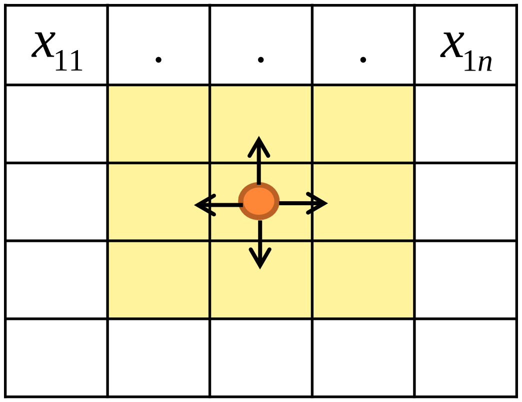
- Example 2 – Motion Model code




```
mot_mod = zeros(N,N,4);
for i=1:n
    for j=1:n
        cur = i+(j-1)*n;
        % Move up
        if (j > 1)
            mot_mod(cur-n,cur,1) = 0.6;
            mot_mod(cur,cur,1) = 0.4;
        else
            mot_mod(cur,cur,1) = 1;
        end
        % Move right
        if (i < n)
            mot_mod(cur+1,cur,2) = 0.6;
            mot_mod(cur,cur,2) = 0.4;
        else
            mot_mod(cur,cur,2) = 1;
        end
…
```

Bayes Filter

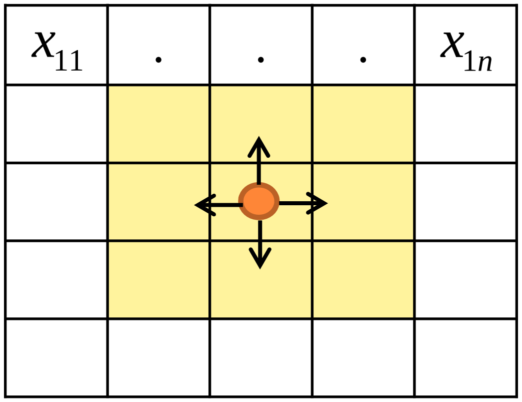
- Example 2 – Measurement Model




```
%% Create the measurement model
meas_mod_rel = [0.11 0.11 0.11;
                0.11 0.12 0.11;
                0.11 0.11 0.11];
% Convert to full measurement model
% p(y_t | x_t)
meas_mod = zeros(N,N);
% Fill in non-boundary measurements
for i=2:n-1
    for j=2:n-1
        cur = i+(j-1)*n;
        meas_mod(cur-n+[-1:1:1],cur) = meas_mod_rel(1,:);
        meas_mod(cur+[-1:1:1],cur) = meas_mod_rel(2,:);
        meas_mod(cur+n+[-1:1:1],cur) = meas_mod_rel(3,:);
    end
end
…
```

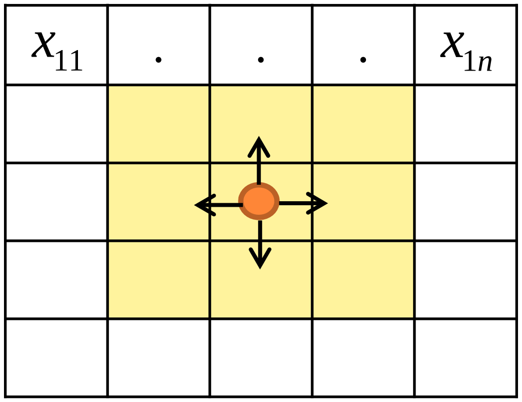
Bayes Filter

- Example 2 – Makin' movies!




```
videoobj=VideoWriter('bayesgrid.mp4','MPEG-4');
truefps = 1;
videoobj.FrameRate = 10; %Anything less than 10 fps fails.
open(videoobj);

figure(1);clf; hold on;
beliefs = reshape(bel,n,n);
imagesc(beliefs);
plot(pos(2),pos(1),'ro','MarkerSize',6,'LineWidth',2)
colormap(bone);
title('True state and beliefs')
F = getframe;
% Dumb hack to get desired framerate
for dumb=1:floor(10/truefps)
    writeVideo(videoobj, F);
end
…
```

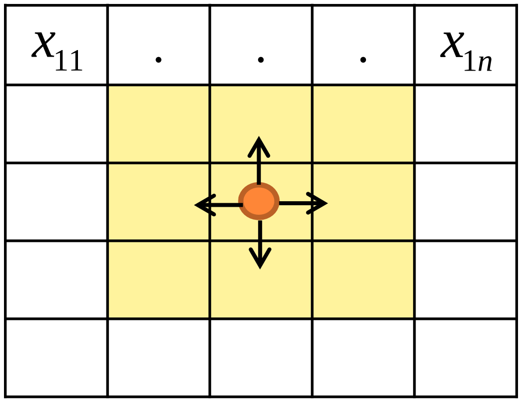
Bayes Filter



- Example 2 – Simulation code

```
%Main Loop
for t=1:T
    %% Simulation
    % Select motion input
    u(t) = ceil(4*rand(1));
    % Select a motion
    thresh = rand(1);
    new_x = find(cumsum(squeeze(mot_mod(:,:,u(t)))*x(:,t))>thresh,1);
    % Move vehicle
    x(new_x,t+1) = 1;
    % Take measurement
    thresh = rand(1);
    new_y = find(cumsum(meas_mod(:,:)*x(:,t+1))>thresh,1);
    y(new_y,t) = 1;
    % Store for plotting
    …
```

Bayes Filter

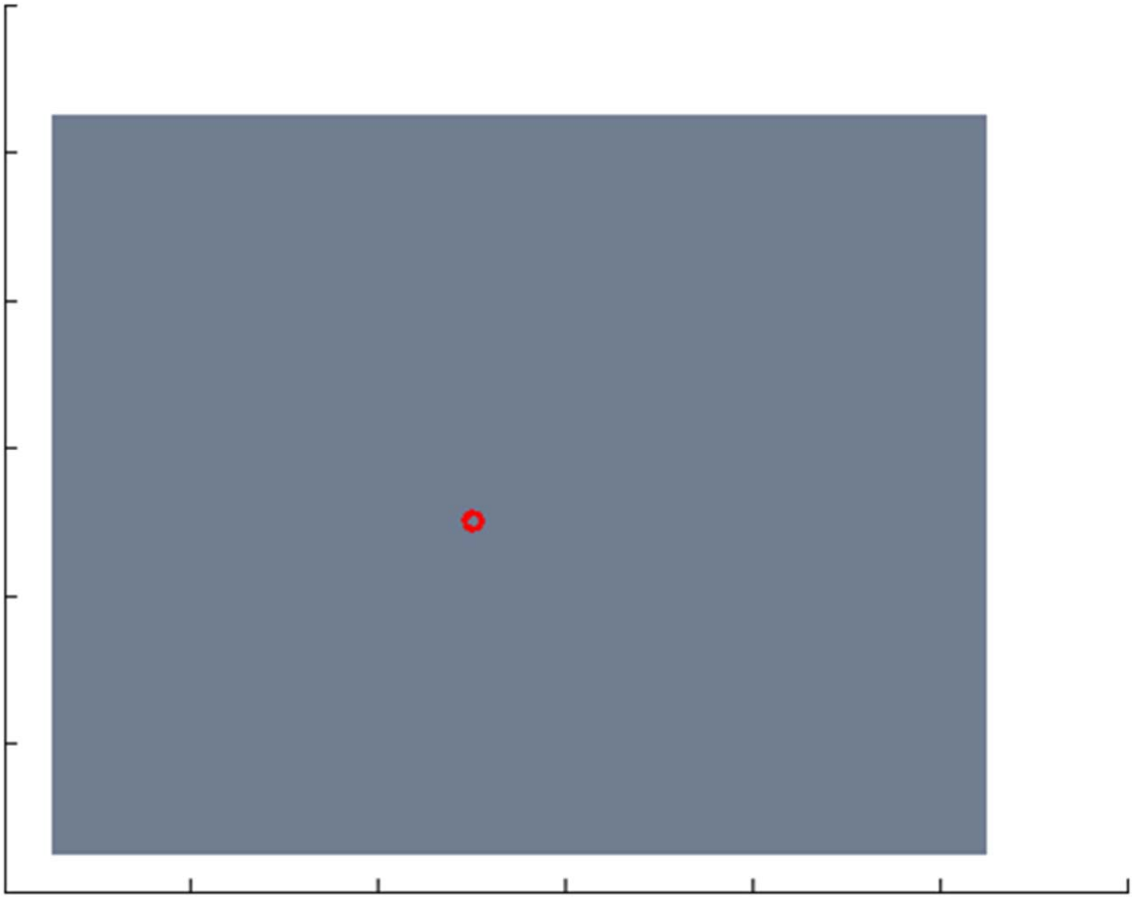


- Example 2 – Bayes Filter

```
…
%% Bayesian Estimation
% Prediction update
belp = squeeze(mot_mod(:,:,u(t)))*bel;

% Measurement update
bel = meas_mod(new_y,:)'.*belp;
bel = bel/norm(bel);

[pmax y_bel(t)] = max(bel);

%% Plot beliefs
…
```

Bayes Filter

- Example 2:
  - Results



Outline

- Bayes Filter Framework
- Kalman Filter
- Extended Kalman Filter
- Particle Filter

KALMAN FILTER


- Rudolf Kalman 1960 (BS, MS: MIT, PhD: Columbia)

The filter is named after Rudolf E. Kalman, though Thorvald Nicolai Thiele[1] and Peter Swerling developed a similar algorithm earlier. Stanley F. Schmidt is generally credited with developing the first implementation of a Kalman filter. It was during a visit of Kalman to the NASA Ames Research Center that he saw the applicability of his ideas to the problem of trajectory estimation for the Apollo program, leading to its incorporation in the Apollo navigation computer. The filter was developed in papers by Swerling (1958), Kalman (1960), and Kalman and Bucy (1961).

Wikipedia

- Discrete version (Kalman filter)
- Continuous version (Kalman-Bucy filter)
- Many other versions, improvements, modifications

Kalman Filter

- Kalman Filter Modeling Assumption
  - Continuous state, inputs, measurements
  - Prior over the state is Gaussian

$$p(x_0) \sim N(\mu_0, \Sigma_0)$$

  - Motion model, linear with additive Gaussian disturbances

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \qquad \varepsilon_t \sim N(0, R_t)$$

  - Often, robotics systems are more easily described in continuous domain
    - Convert to discrete time using matrix exponential
    - Matlab contains tools to perform this conversion (c2d, d2c)

Kalman Filter

- Kalman Filter Modeling Assumption
  - Measurement model also linear with additive Gaussian noise

$$y_t = C_t x_t + \delta_t \qquad \delta_t \sim N(0, Q_t)$$

  - Can add in input dependence to match up with controls literature

$$y_t = C_t x_t + D_t u_t + \delta_t$$

Kalman Filtering

- Full Model
  - State prior

$$p(x_0) \sim N(\mu_0, \Sigma_0)$$

  - Motion model

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \qquad \varepsilon_t \sim N(0, R_t)$$

  - Measurement model

$$y_t = C_t x_t + \delta_t \qquad \delta_t \sim N(0, Q_t)$$

KALMAN FILTER

- Assume belief is Gaussian at time *t*

$$bel(x_t) \sim N(\mu_t, \Sigma_t)$$

  - μ_t is the best estimate of the current state at time t
  - $\sum_t$ is the covariance, indicating the certainty in the current estimate

- Will be able to demonstrate the predicted belief at the next time step is Gaussian

$$\overline{bel}(x_{t+1}) \sim N(\bar{\mu}_{t+1}, \bar{\Sigma}_{t+1})$$

- And that the belief at next time step is also Gaussian

$$bel(x_{t+1}) \sim N(\mu_{t+1}, \Sigma_{t+1})$$

Kalman Filter

- Goal:
  - To find belief over state as accurately as possible given all available information
    - Minimize the mean square error of the estimate (MMSE estimator)

$$\min \quad E[(\mu_t - x_t)^2]$$

      - Same as least square problem

    - Using an unbiased estimator

$$E[\mu_t - x_t] = 0$$

      - On average, your estimate is correct!

Kalman Filter

- Goal
  - The MMSE estimate can be written as

$$\min \quad E[(x_t - \mu_t)^T (x_t - \mu_t)]$$

  - And is equivalent to minimizing the trace of the error covariance matrix

$$\min \quad \mathrm{tr}(\Sigma_t)$$

  - Proof:

$$\begin{aligned} E[(x_t - \mu_t)^T (x_t - \mu_t)] &= E\left[\sum_i (x_{t,i} - \mu_{t,i})^2\right] \\ &= E[\mathrm{tr}\left((x_t - \mu_t)(x_t - \mu_t)^T\right)] \\ &= \mathrm{tr}\left(E[(x_t - \mu_t)(x_t - \mu_t)^T]\right) \\ &= \mathrm{tr}(\Sigma_t) \end{aligned}$$

Kalman Filter

- Kalman Filter Algorithm
  - At each time step, t, update both sets of beliefs
    1. Prediction update

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

    2. Measurement update

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (y_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

  - Kalman Gain, K_t
    - Blending factor between prediction and measurement

KALMAN FILTER

- Example
  - Temperature control
    - State is current temperature difference with outside
    - One dimensional example
    - Prior: fairly certain of current temperature difference

$$\mu_0 = 10$$
$$\Sigma_0 = 1$$

    - Motion Model: Decaying temperature + furnace input + disturbances (opening doors, outside effects)

$$dt = 0.1$$
$$x_t = 0.8x_{t-1} + 3u_t + r_t$$
$$A = 0.8, \quad B = 3$$
$$r_t \sim N(0, 2)$$

Kalman Filter

- Example
  - Measurement Model
    - Directly measure the current temperature difference

$$y(t) = x(t) + \delta_t$$
$$\delta_t \sim N(0,4)$$

  - Controller design
    - Bang bang control, based on current estimate of temperature difference

$$u(t) = \begin{cases} 1 & \mu_t < 2 \\ 0 & \mu_t > 10 \\ u(t-1) & \text{otherwise} \end{cases}$$

Kalman Filter

- Example
  - Simulation

```
for t=1:length(T)
% Select control action
    if (t>1) u(t)=u(t-1); end
    if (mu > 10)
        u(t) = 0;
    elseif (mu < 2);
        u(t) = 1;
    end

    % Update state
    e = sqrt(R)*randn(1);
    x(t+1) = A*x(t)+ B*u(t) + e;

    % Determine measurement
    d = sqrt(Q)*randn(1);
    y(t) = C*x(t+1) + d;
```

Kalman Filter

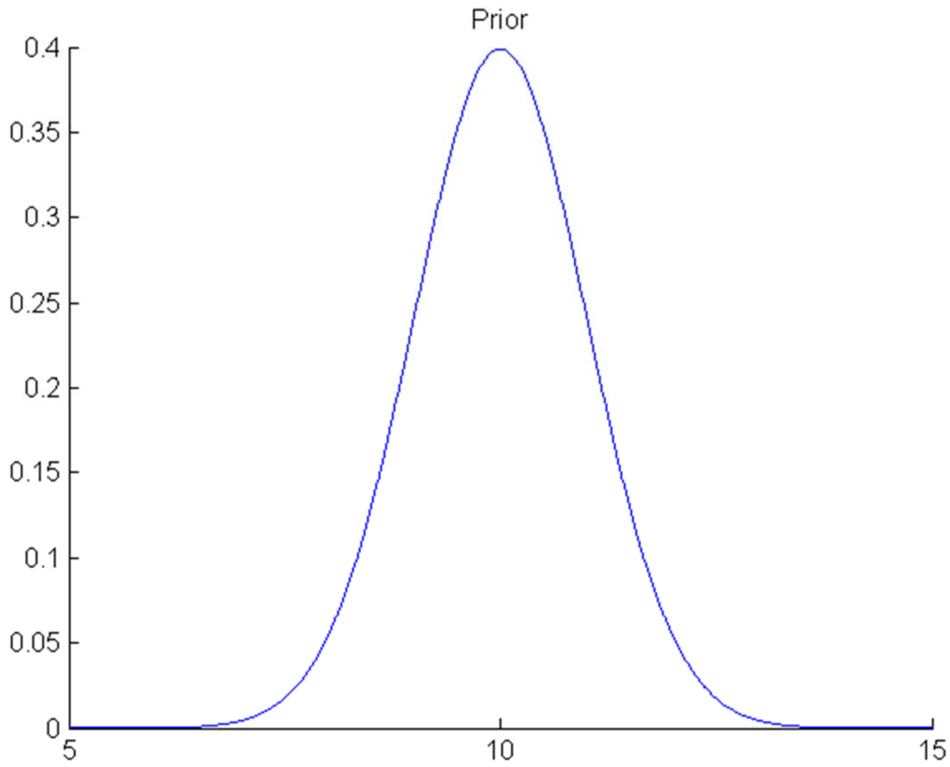
- Example
  - Estimation

```
% Prediction update
mup = A*mu + B*u(t);
Sp = A*S*A' + R;

% Measurement update
K = Sp*C'*inv(C*Sp*C'+Q);
mu = mup + K*(y(t)-C*mup);
S = (1-K*C)*Sp;
```

  - Matrix inverse $0(n^{2.4})$, matrix multiplication $O(n^2)$
  - When implementing in Matlab, inv( ) performs matrix inverse for you
  - For embeddded code, many libraries exist
    - Try Gnu Scientific Library, easy starting point

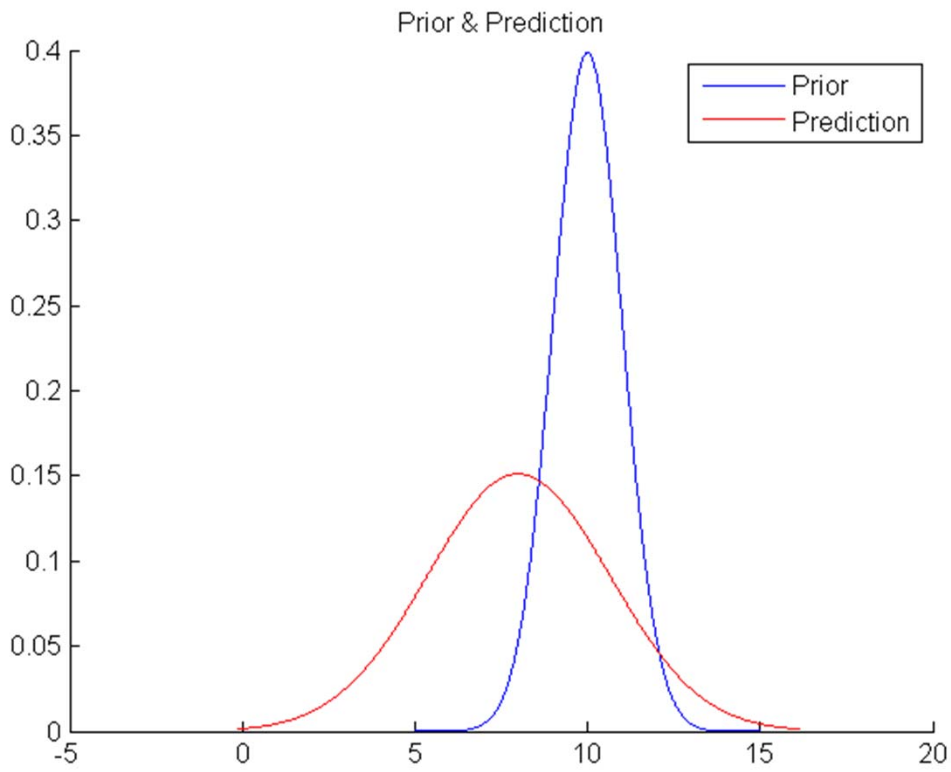
Kalman Filter

- Example
  - Beliefs during the first time step
    - Prior



Kalman Filter

- Example
  - Beliefs during the first time step
    - Prediction Update: increased variance, shifted mean



$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

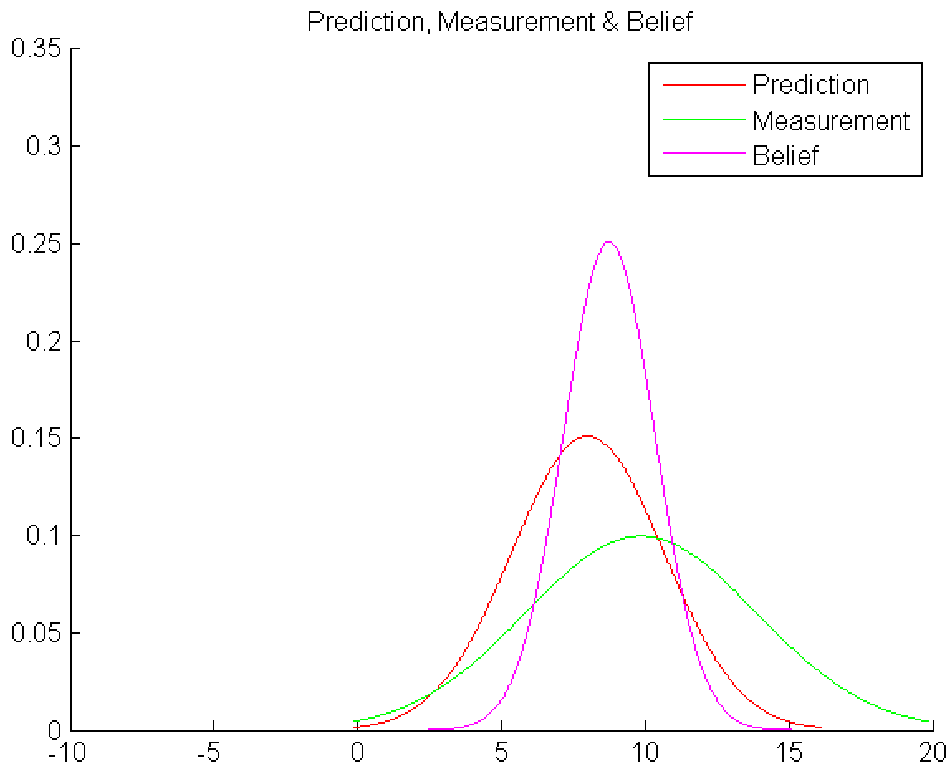
$$x_t = 0.8 x_{t-1} + 3 u_t + r_t$$

$$A = 0.8, \quad B = 3$$

$$r_t \sim N(0, 2)$$

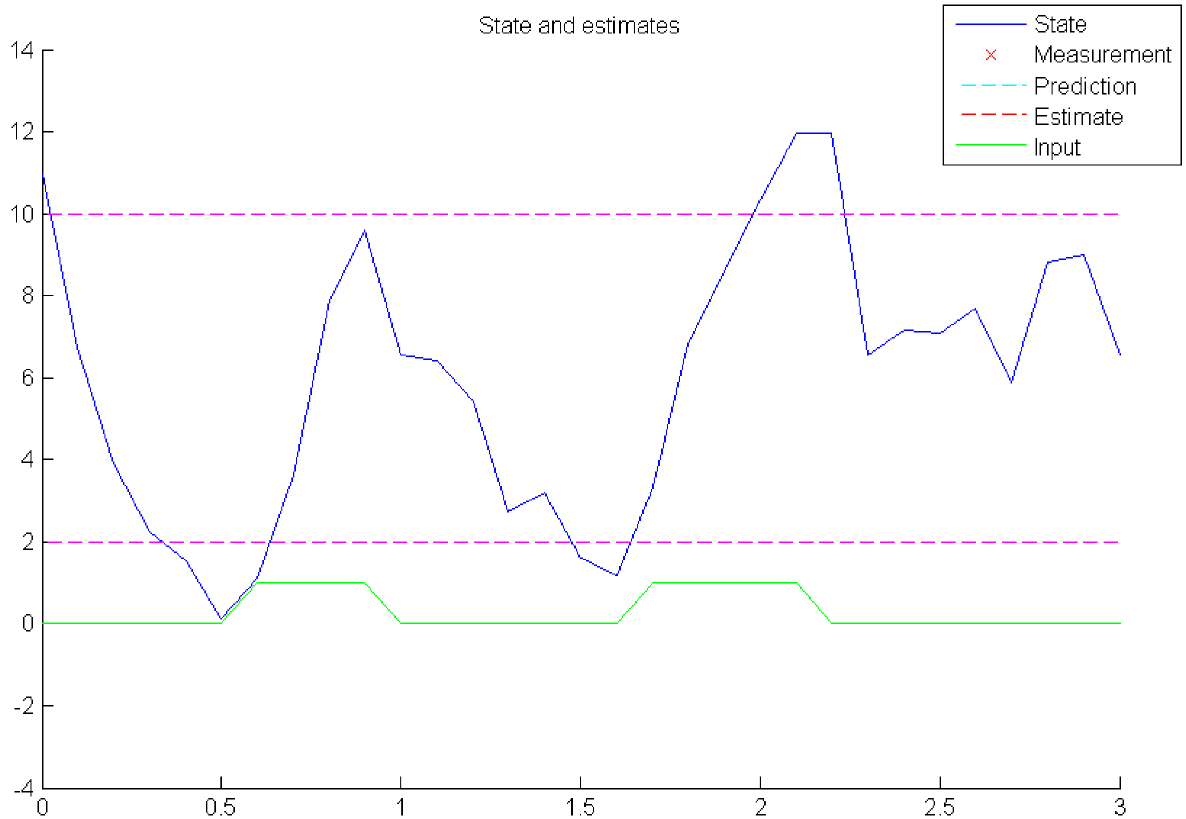
Kalman Filter

- Example
  - Beliefs after the first time step
    - Measurement update: decreased variance



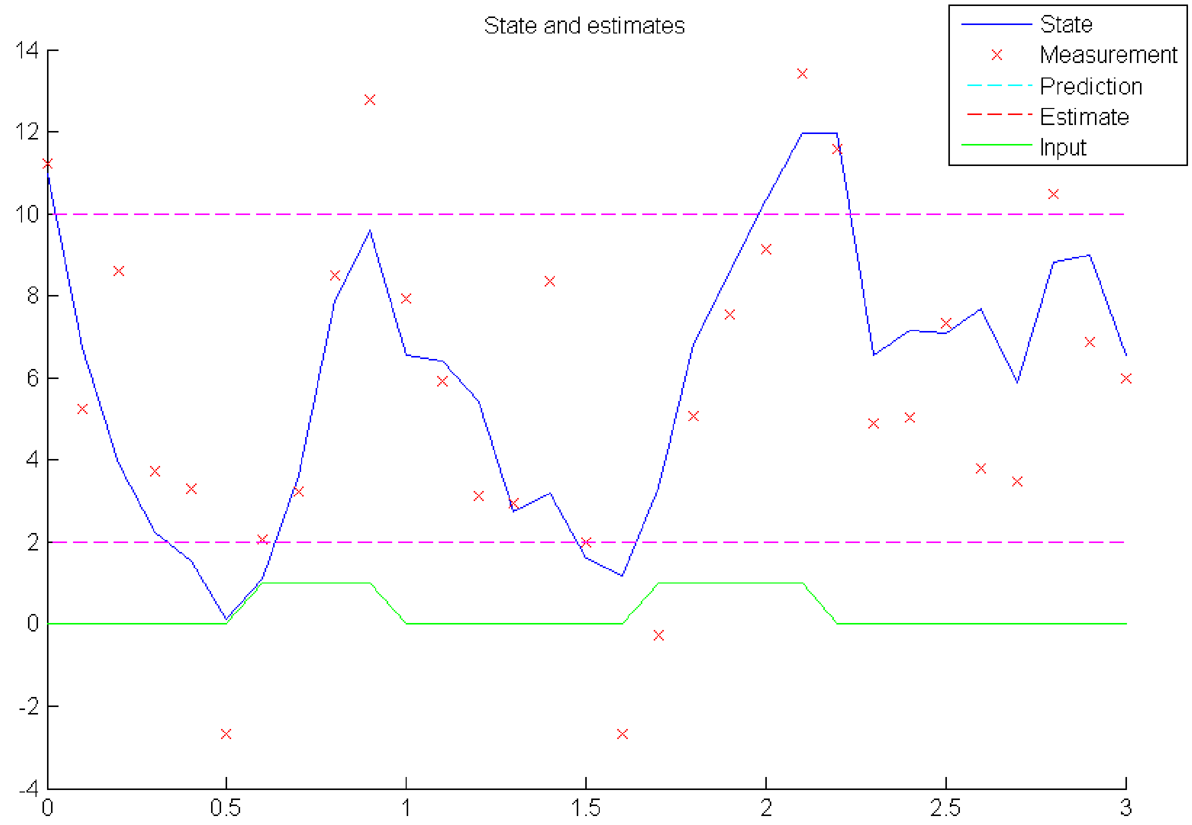
Kalman Filter

- Example
  - Trajectories over time



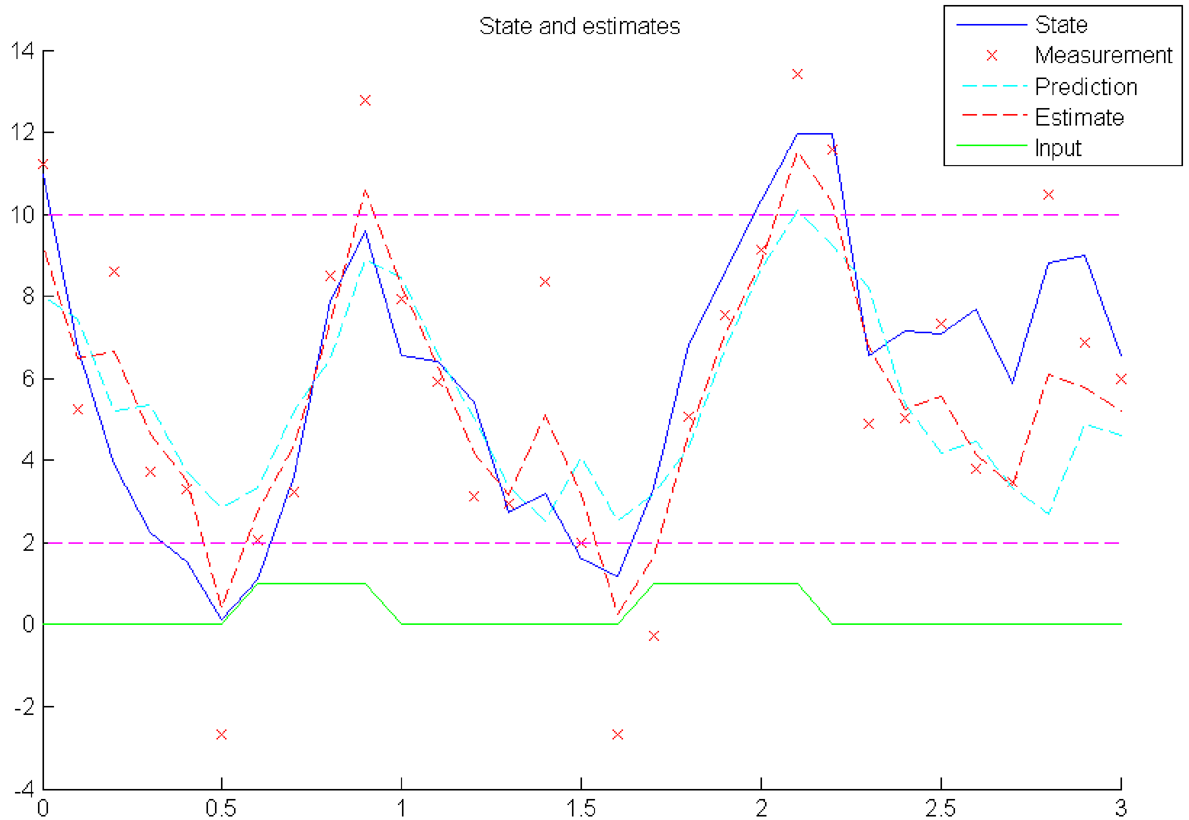
KALMAN FILTER

- Example
  - Measurements over time



Kalman Filter

- Example
  - Estimates over time



KALMAN FILTER

- Recall Goal:
  - To find belief over state as accurately as possible given all available information
    - Minimize the mean square error of the estimate (MMSE estimator)

$$\min \quad E[(\mu_t - x_t)^2]$$

      - Same as least square problem

    - Using an unbiased estimator

$$E[\mu_t - x_t] = 0$$

      - On average, your estimate is right!

Kalman Filter

- Derivation
  - Define the innovation
    - The difference between the measurement and the expected measurement given the predicted state and the measurement model

$$\text{Innovation}_t = y_t - C_t \bar{\mu}_t$$

  - Assume the form of the estimator is a linear combination of the predicted belief and the innovation
    - The following form turns out to be unbiased

$$\mu_t = \bar{\mu}_t + K_t (y_t - C_t \bar{\mu}_t)$$

Kalman Filter

- Steps

  - Prediction update
    - Find update rule for mean, covariance of predicted belief, given input and motion model

  - Measurement update
    - Solve MMSE optimization problem to find update rule for mean, covariance of belief given measurement model and measurement

Kalman Filter

- Prediction Update
  - Only new information is input u_t
  - Prediction update is a linear transformation of belief at previous time step
    - Motion model is

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

    - Motion noise, previous belief are Gaussian so this is an addition of Gaussian distributions

$$bel(x_{t-1}) \sim N(\mu_{t-1}, \Sigma_{t-1}) \qquad \varepsilon_t \sim N(0, R_t)$$

    - Therefore the predicted mean and covariance are

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t + 0$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

Kalman Filter

- Measurement update
  - First, lets define the form of the error covariance, substituting in the form of the mean update and the measurement model

$$\Sigma_t = E[(x_t - \mu_t)(x_t - \mu_t)^T]$$

  - But, by the assumption of the form of the estimator, and the measurement model

$$\begin{aligned} x_t - \mu_t &= x_t - \bar{\mu}_t - K_t(y_t - C_t\bar{\mu}_t) \\ &= x_t - \bar{\mu}_t - K_t(C_t x_t + \delta_t - C_t\bar{\mu}_t) \\ &= (I - K_t C_t)(x_t - \bar{\mu}_t) - K_t\delta_t \end{aligned}$$

Kalman Filter

- Measurement update
  - Next, reorganize terms of the covariance

$$\Sigma_t = E[\left((I - K_t C_t)(x_t - \bar{\mu}_t) - K_t \delta_t\right)\left((I - K_t C_t)(x_t - \bar{\mu}_t) - K_t \delta_t\right)^T]$$

$$= E[\left((I - K_t C_t)(x_t - \bar{\mu}_t)\right)\left((I - K_t C_t)(x_t - \bar{\mu}_t)\right)^T]$$

$$-2E[\left((I - K_t C_t)(x_t - \bar{\mu}_t)\right) K_t \delta_t]$$

$$+E[\left(K_t \delta_t\right)\left(K_t \delta_t\right)^T]$$

  - But the middle term is zero in expectation

$$\Sigma_t = E[\left((I - K_t C_t)(x_t - \bar{\mu}_t)\right)\left((I - K_t C_t)(x_t - \bar{\mu}_t)\right)^T]$$

$$+E[\left(K_t \delta_t\right)\left(K_t \delta_t\right)^T]$$

Kalman Filter

- Measurement Update
  - Recall multiplication by a constant yields

$$\begin{aligned} \mathrm{cov}(Ax) &= E[(Ax - A\mu)(Ax - A\mu)^T] \\ &= A\,\mathrm{cov}(X)A^T \end{aligned}$$

  - In the above, numerous constants

$$\begin{aligned} \Sigma_t &= E[\big((I - K_t C_t)(x_t - \bar{\mu}_t)\big)\big((I - K_t C_t)(x_t - \bar{\mu}_t)\big)^T] \\ &\quad + E[\big(K_t \delta_t\big)\big(K_t \delta_t\big)^T] \\ &= \mathrm{cov}\big((I - K_t C_t)(x_t - \bar{\mu}_t)\big) + \mathrm{cov}\big(K_t \delta_t\big) \end{aligned}$$

KALMAN FILTER

- Measurement Update
  - The resulting covariance is

$$\Sigma_t = (I - K_t C_t) E\left[(x_t - \bar{\mu}_t)(x_t - \bar{\mu}_t)^T\right](I - K_t C_t)^T$$
$$+ K_t E\left[\delta_t \delta_t^T\right] K_t^T$$

  - The expectations that remain are known quantities

$$\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t (I - K_t C_t)^T + K_t Q_t K_t^T$$
$$= \bar{\Sigma}_t - K_t C_t \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T K_t^T + K_t \left(C_t \bar{\Sigma}_t C_t^T + Q_t\right) K_t^T$$

  - Which leaves us with a quadratic equation in K_t

KALMAN FILTER

- Measurement update
  - We now have the covariance in a form that can be optimized

$$\min \quad \mathrm{tr}(\Sigma_t)$$

  - We need two identities to find this minimum
    - Differentiation of linear matrix expression

$$\frac{\partial \mathrm{tr}(\mathrm{AXB})}{\partial X} = \frac{\partial \mathrm{tr}(\mathrm{B}^T \mathrm{X}^T \mathrm{A}^T)}{\partial X} = A^T B^T$$

    - Differentiation of quadratic matrix expression

$$\frac{\partial \mathrm{tr}(\mathrm{XAX}^T)}{\partial X} = XA^T + XA$$

KALMAN FILTER

- Measurement update
  - The optimization is done by setting the derivative of the trace w.r.t the Kalman gain to 0

$$\min \quad \mathrm{tr}\left(\bar{\Sigma}_t - K_t C_t \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T K_t^T + K_t \left(C_t \bar{\Sigma}_t C_t^T + Q_t\right) K_t^T\right)$$

  - Taking the matrix derivative w.r.t K_t
    - Two linear terms and one quadratic

$$\frac{\partial}{\partial K_t} \mathrm{tr}(\Sigma_t) = -\bar{\Sigma}_t^T C_t^T - \bar{\Sigma}_t C_t^T$$

$$+ K_t \left(C_t \bar{\Sigma}_t C_t^T + Q_t\right)^T + K_t \left(C_t \bar{\Sigma}_t C_t^T + Q_t\right)$$

Kalman Filter

- Measurement update
  - Set the derivative to 0, noting that covariance is symmetric, and AXA^T preserves symmetry

$$\frac{\partial}{\partial K_t}\mathrm{tr}(\Sigma_t)=\left(-2\bar{\Sigma}_t C_t^T + 2K_t\left(C_t\bar{\Sigma}_t C_t^T + Q_t\right)\right)=0$$

  - Simplifying

$$K_t\left(C_t\bar{\Sigma}_t C_t^T + Q_t\right)=\bar{\Sigma}_t C_t^T$$

  - And finally, we arrive at the Kalman gain equation

$$K_t = \bar{\Sigma}_t C_t^T (C_t\bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

KALMAN FILTER

- Measurement Update
  - So far, we have found the optimal gain K_t which minimizes mean square error in the measurement update for the mean

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (y_t - C_t \bar{\mu}_t)$$

  - Next, we need to simplify the covariance update using this result for the Kalman gain

Kalman Filter

- Measurement Update
  - Recall the Covariance update was

$$\Sigma_t = \bar{\Sigma}_t - K_t C_t \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T K_t^T + K_t \left(C_t \bar{\Sigma}_t C_t^T + Q_t \right) K_t^T$$

  - Substituting in the Kalman gain gives

$$\begin{aligned}\Sigma_t = {} & \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} C_t \bar{\Sigma}_t \\ & - \bar{\Sigma}_t C_t^T \left(\bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \right)^T \\ & + \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \left(C_t \bar{\Sigma}_t C_t^T + Q_t \right) \left(\bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \right)^T\end{aligned}$$

Kalman Filter

- Measurement Update
  - Fortunately, almost everything cancels and we are left with

$$
\begin{aligned}
\Sigma_t &= \bar{\Sigma}_t - \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} C_t \bar{\Sigma}_t \\
&= (I - K_t C_t)\bar{\Sigma}_t
\end{aligned}
$$

KALMAN FILTER

- Kalman Filter Algorithm
  - At each time step, *t*, update both sets of beliefs
    1. Prediction update

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

    2. Measurement update

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (y_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

  - Kalman Gain, K_t
    - Blending factor between prediction and measurement

Kalman Filter

- Summary
  - Follows same framework as Bayes filter
  - Requires linear motion and Gaussian disturbance
  - Requires linear measurement and Gaussian noise
  - It is sufficient to update mean and covariance of beliefs, because they remain Gaussian
  - Prediction step involves addition of Gaussians
  - Measurement step seeks to minimize mean square error of the estimate
    - Expand out covariance from definition and measurement model
    - Assume form of estimator, linear combination of prediction and measurement
    - Solve MMSE problem to find optimal linear combination
    - Simplify covariance update once gain is found

Kalman Filtering

- Relation to Bayes Filter Problem Formulation
  - State prior

$$bel(x_0) = p(x_0) = N(\mu_0, \Sigma_0)$$

  - Motion model

$$p(x_t \mid x_{t-1}, u_t) = N(A_t x_{t-1} + B_t u_t, A_t \Sigma_{t-1} A_t^T + R_t)$$

  - Measurement model

$$p(y_t \mid x_t) = N(C_t x_t, Q_t)$$

  - Beliefs

$$bel(x_t) = N(\mu_t, \Sigma_t), \quad \overline{bel}(x_t) = N(\bar{\mu}_t, \bar{\Sigma}_t)$$

KALMAN FILTER

- Relation to Bayes Filter Algorithm
  1. Prediction update (Total probability)
     - Insert normal distributions

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

$$= \eta \int e^{-1/2(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)} e^{-1/2(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})} dx_{t-1}$$

- Separate out terms that depend on current state
- Manipulate remaining integral into a Gaussian pdf form of previous state
- Integrate over full range to get 1
- Manipulate remaining terms and solve for Kalman prediction equations.

$$\sim N(A_t \mu_{t-1} + B_t u_t, A_t \Sigma_{t-1} A_t^T + R_t)$$

- Refer to Thrun, Burgard & Fox Chap. 3 for details

Kalman Filter

- Relation to Bayes Filter Algorithm
  2. Measurement update (Bayes Theorem)

$$bel(x_t) = \eta p(y_t \mid x_t)\overline{bel}(x_t)$$

$$= \eta e^{-1/2(y_t - C_t x_t)^T Q_t^{-1}(y_t - C_t x_t)} e^{-1/2(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)}$$

  - Reorganize exponents and note it remains a Gaussian
  - For any Gaussian:
    - Second derivative of exponent is inverse of covariance
    - Mean minimizes exponent,
      - Set first derivative of exponent to 0 and solve
  - Use this to solve for mean and covariance of belief

$$= N(\bar{\mu}_t + K_t(y_t - C_t\bar{\mu}_t), (I - K_t C_t)\bar{\Sigma}_t)$$

    - where K_t is the Kalman gain as before

Kalman Filter

- Example
  - 3D Linear motion model for three thruster AUV (heading constant)
    - State
$$x = \begin{bmatrix} p_n \\ v_n \\ p_e \\ v_e \\ p_d \\ v_d \end{bmatrix}$$
    Input
$$u = \begin{bmatrix} T_n \\ T_e \\ T_d \end{bmatrix}$$

    - Continuous dynamics for

$$\dot{x}_n(t) = v_n(t)$$
$$m\dot{v}_n(t) = -bv_n(t) + T_n(t)$$

KALMAN FILTER

- Example – Omni-directional AUV
  - Discrete Dynamics from zero order hold, dt = 0.1s

$$x_t = \begin{bmatrix} 1 & .0975 & 0 & 0 \\ 0 & .9512 & 0 & 0 \\ 0 & 0 & 1 & .0975 \\ 0 & 0 & 0 & .9512 \end{bmatrix} x_{t-1} + \begin{bmatrix} .0025 & 0 \\ .0488 & 0 \\ 0 & .0025 \\ 0 & .0048 \end{bmatrix} u_t + \varepsilon_t$$

  - Disturbances

$$\varepsilon_t \sim N\left(0, R = \begin{bmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}\right)$$

KALMAN FILTER

- Example – Omni-directional AUV
  - Measurement Model
    - Can only measure position (relative to known objects)

$$y_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_t + \delta_t$$

  - With correlated measurement noise

$$\delta_t \sim N\left(0, Q = \begin{bmatrix} 0.4 & -0.1 \\ -0.1 & 0.1 \end{bmatrix}\right)$$

KALMAN FILTER

- Example
  - Control inputs
    - This time different frequencies of sinusoidal input

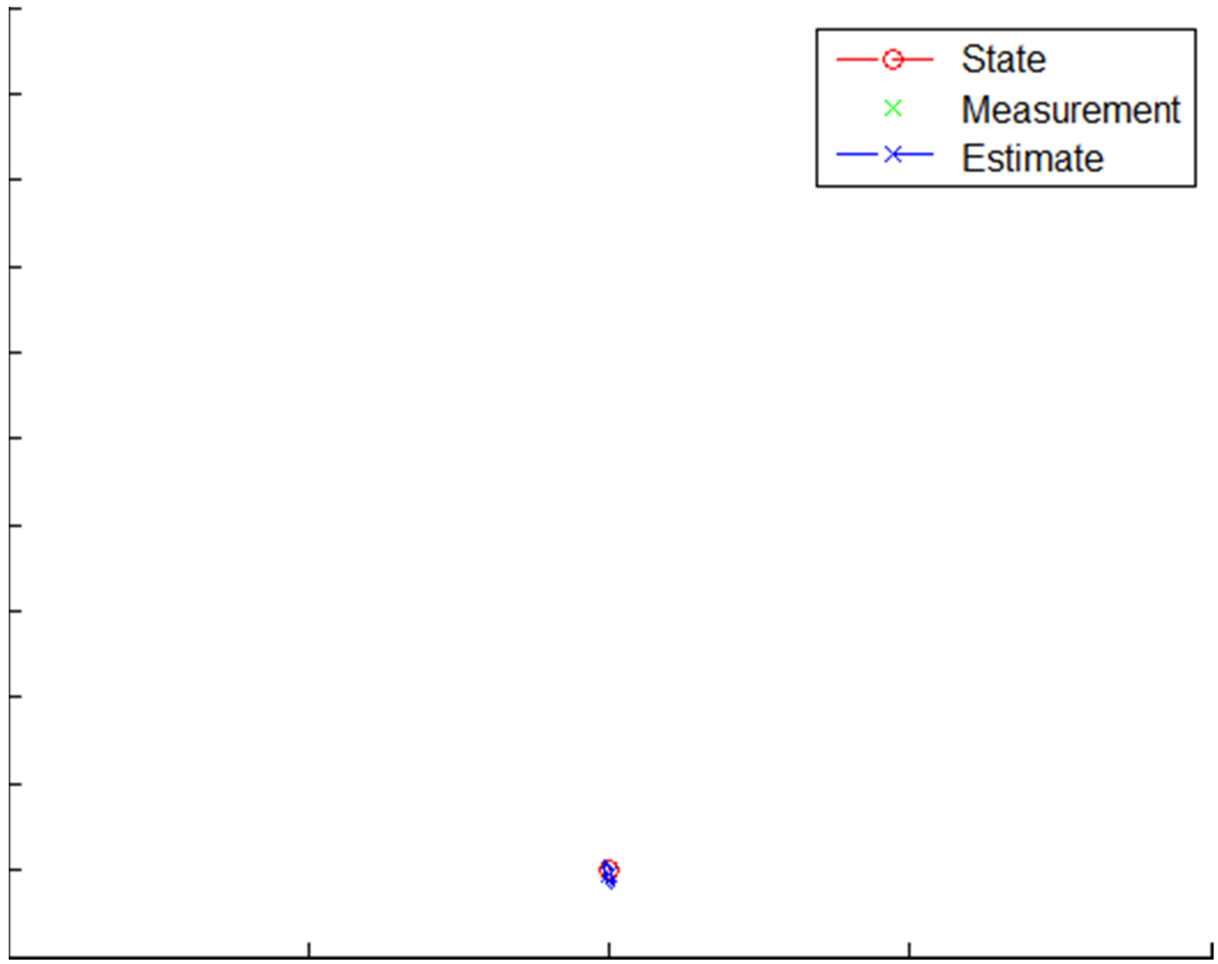
$$u_t = 10\begin{bmatrix}\sin(2t) \\ \cos(t)\end{bmatrix}$$

    - Simulation run for 10 seconds, or 101 time steps
  - Prior distribution
    - Fairly course initialization

$$bel(x_0) \sim N(0, I)$$

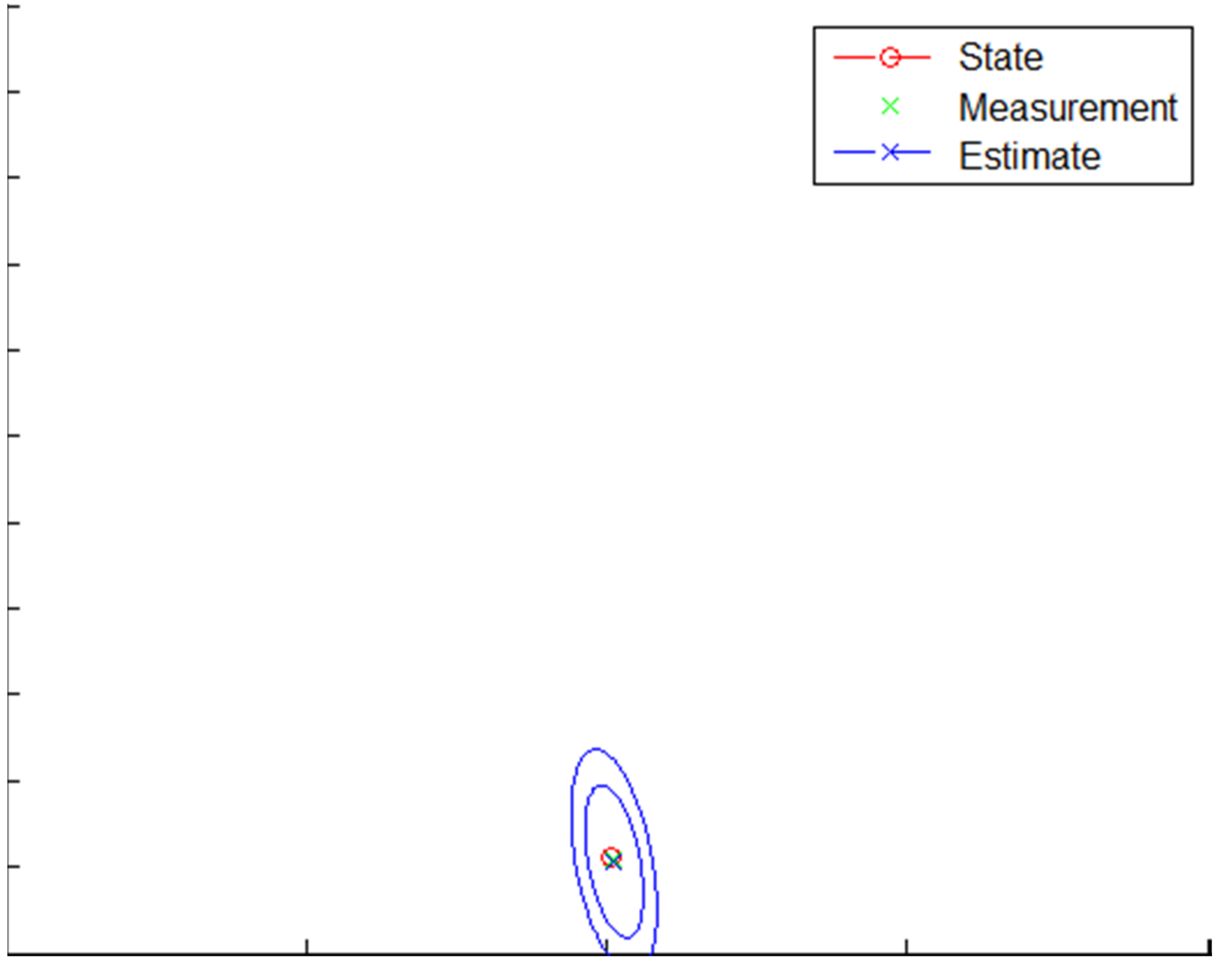
KALMAN FILTER

- Example
  - Ideal results: very low noise and disturbance levels



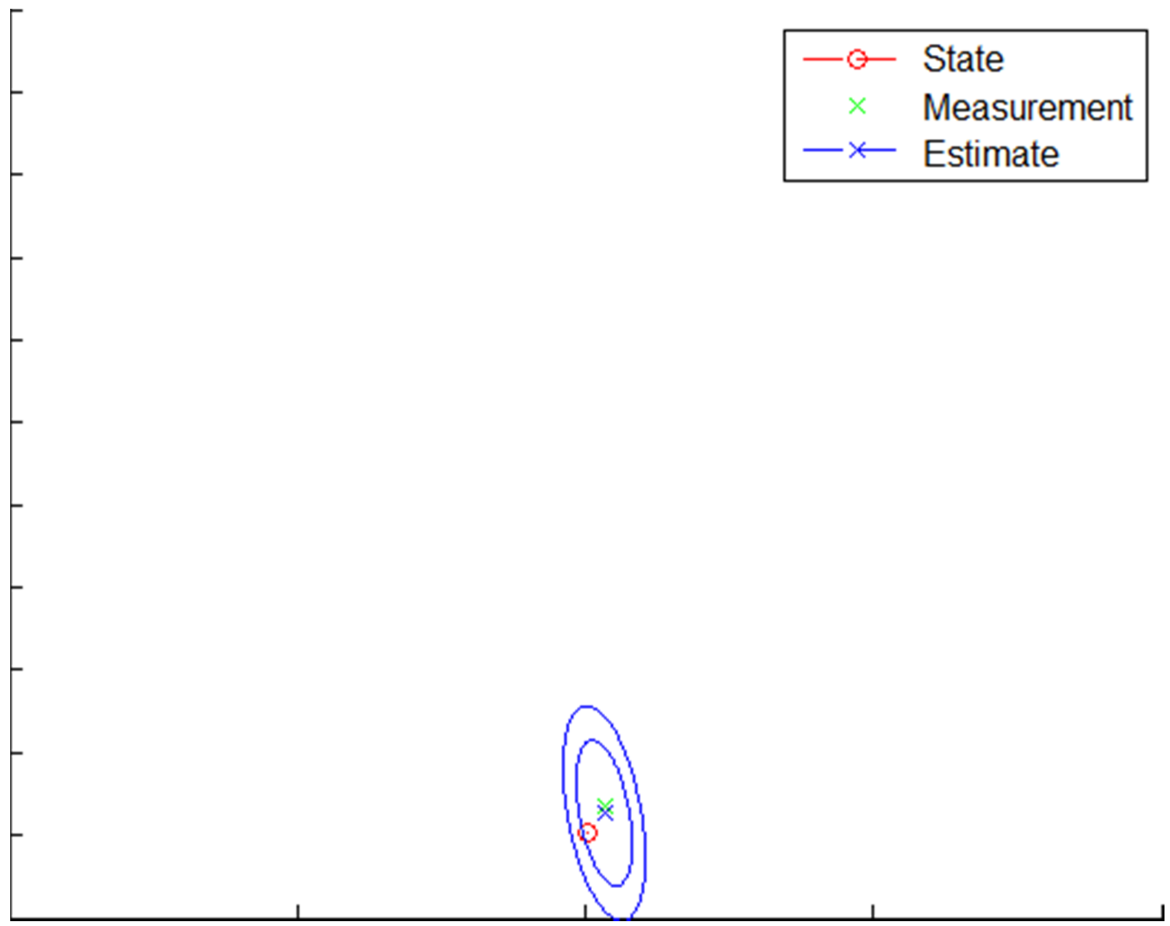
KALMAN FILTER

- Example
  - Results with larger noise and disturbances



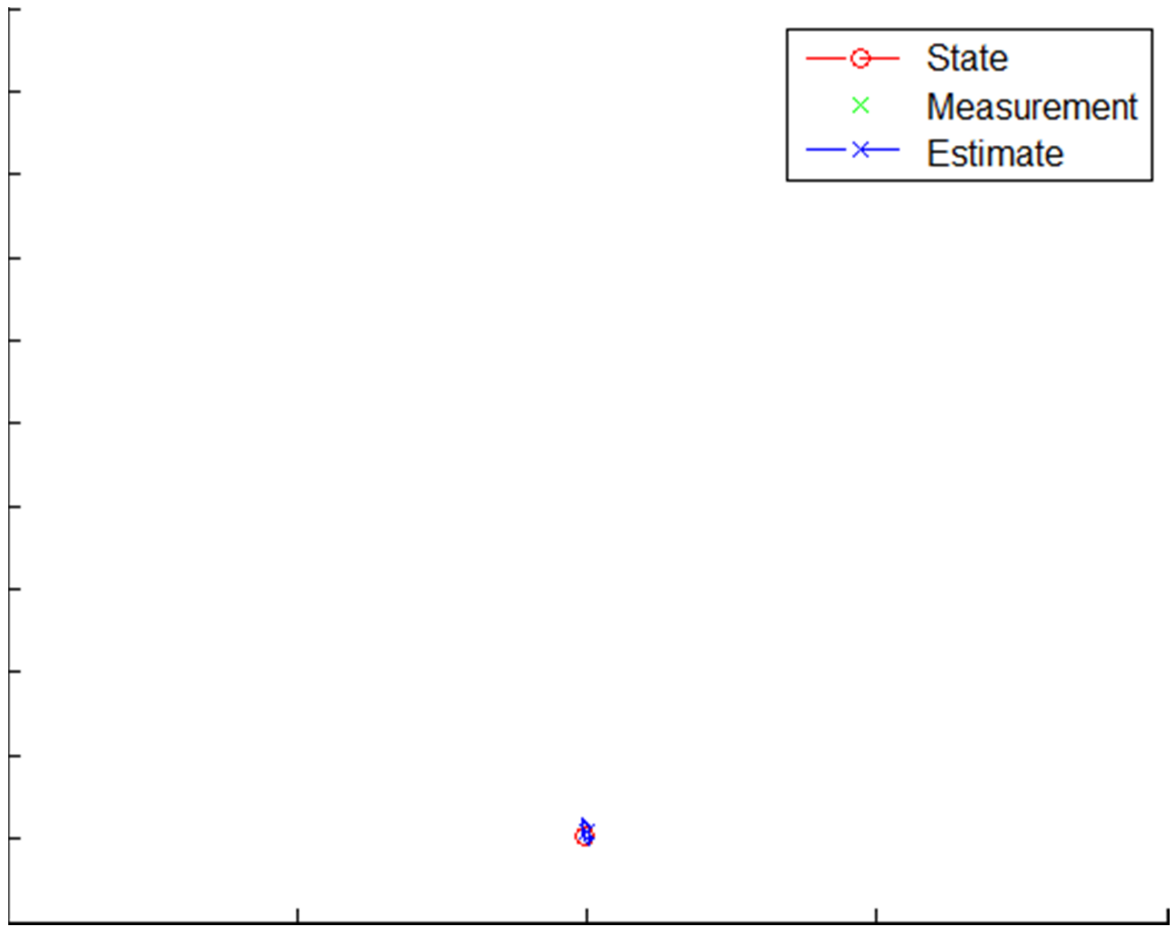
Kalman Filter

- Example
  - Effect of decreased motion disturbances



KALMAN FILTER

- Example
  - Effect of decreased measurement noise



Kalman Filter

- Belief mean is tradeoff between prediction and measurement

$$\mu_t = \bar{\mu}_t + K_t(y_t - C_t\bar{\mu}_t)$$

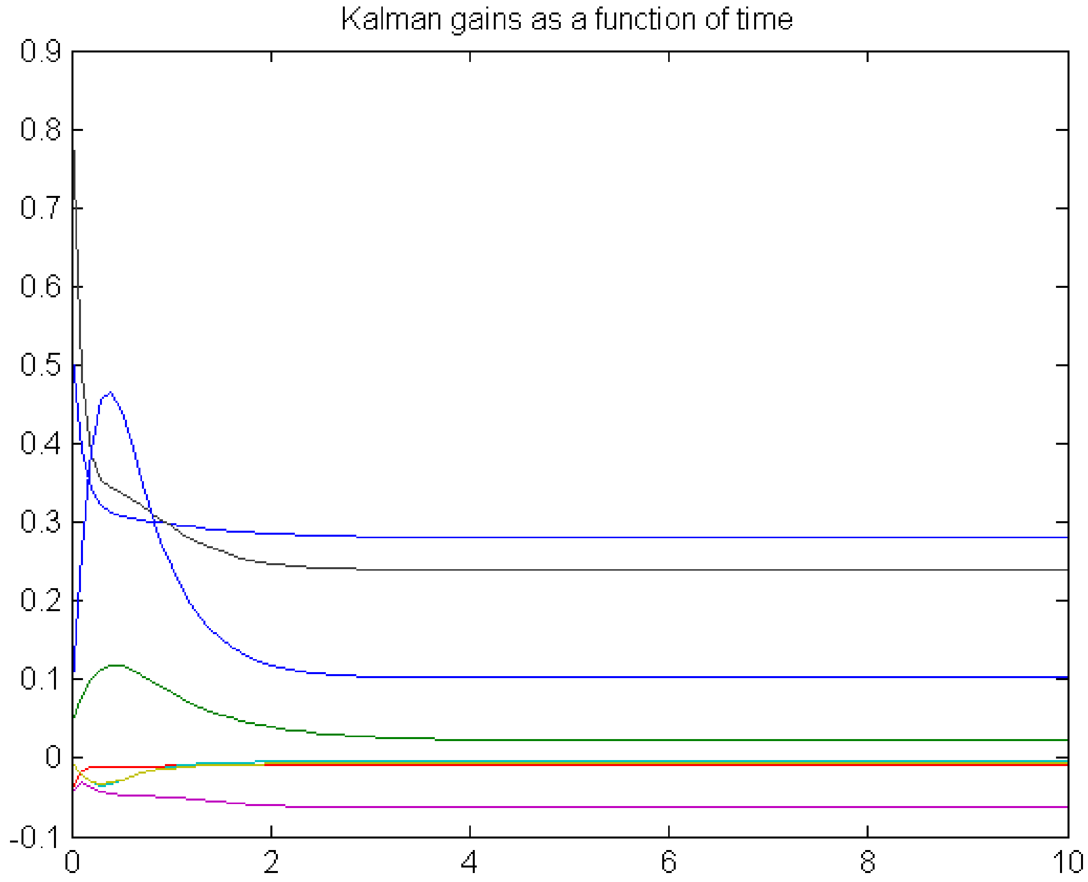
  - Kalman gain determines how to blend estimates

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

- If Q_t is large, inverse is small, so Kalman gain remains small
  - When measurements are high in covariance, don't trust them!
- If R_t is large, then so is predicted belief covariance, so Kalman gain becomes large
  - When model is affected by large unknown disturbances, don't trust the predicted motion!

KALMAN FILTER

- Example
  - Evolution of Kalman Gain

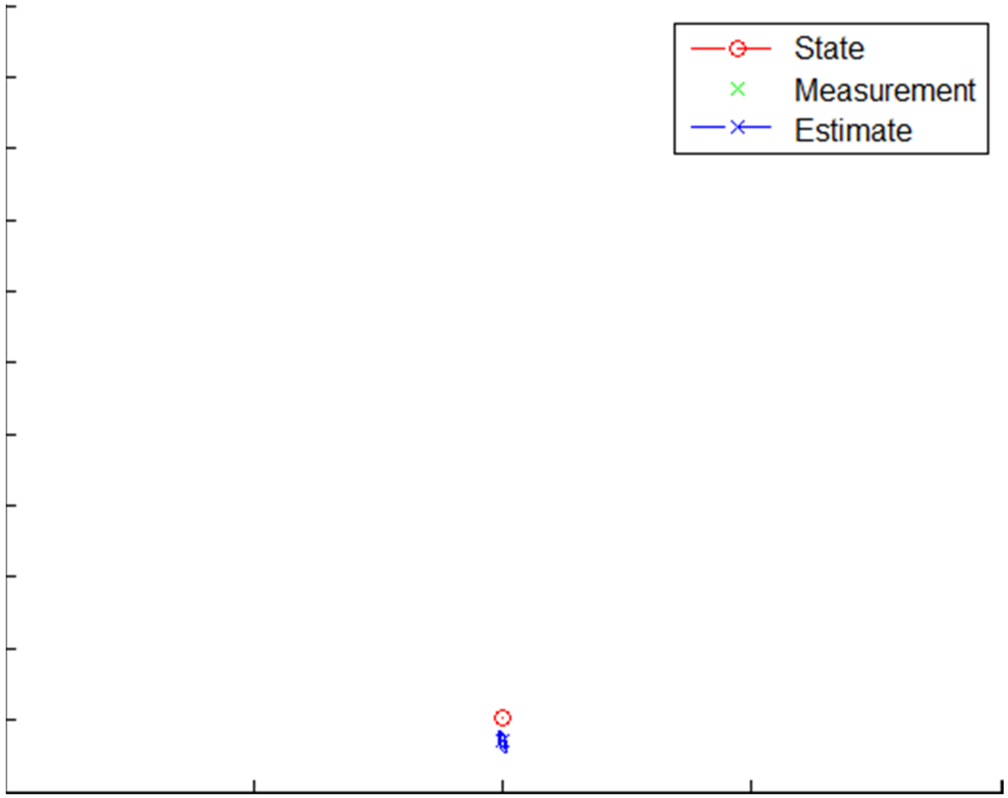


Kalman Filter

- Steady state Kalman Filter
  - For constant noise/disturbance models, it is possible to use steady state values for the Kalman gain
    - Set $\Sigma = \Sigma_t = \Sigma_{t-1}$ in the Kalman filter update equations and solve for Σ
    - Referred to as the Discrete Algebraic Ricatti Equation (DARE), Matlab will solve it for you
  - Can also run Kalman filter until convergence and then eliminate gain update step (matrix inversion)

KALMAN FILTER

- Example
  - Incorrect measurement distribution (covariance actually much larger)



  - Estimate tracks measurements too closely

Kalman Filter

- Multi-Rate Kalman Filter
  - At each time step, it is possible to use different measurement models
    - Time varying C_t and Q_t
  - Identify a base update rate
    - Find greatest common divisor of sample rates
      - e.g. GPS 5Hz, Sodar 12 Hz, Base rate 60 Hz
  - Create discretized motion model at base rate
  - At each timestep
    1. Perform prediction update
    2. If new measurements exist, perform measurement update for those measurements only
       - Select appropriate C_t and Q_t

KALMAN FILTER

- Example
  - Multi-rate Kalman Filter
    - Add in velocity measurements at 100 Hz
    - Base update rate 0.01 s
    - Create two separate types of measurement updates
      - Velocity only measurement for 9 time steps

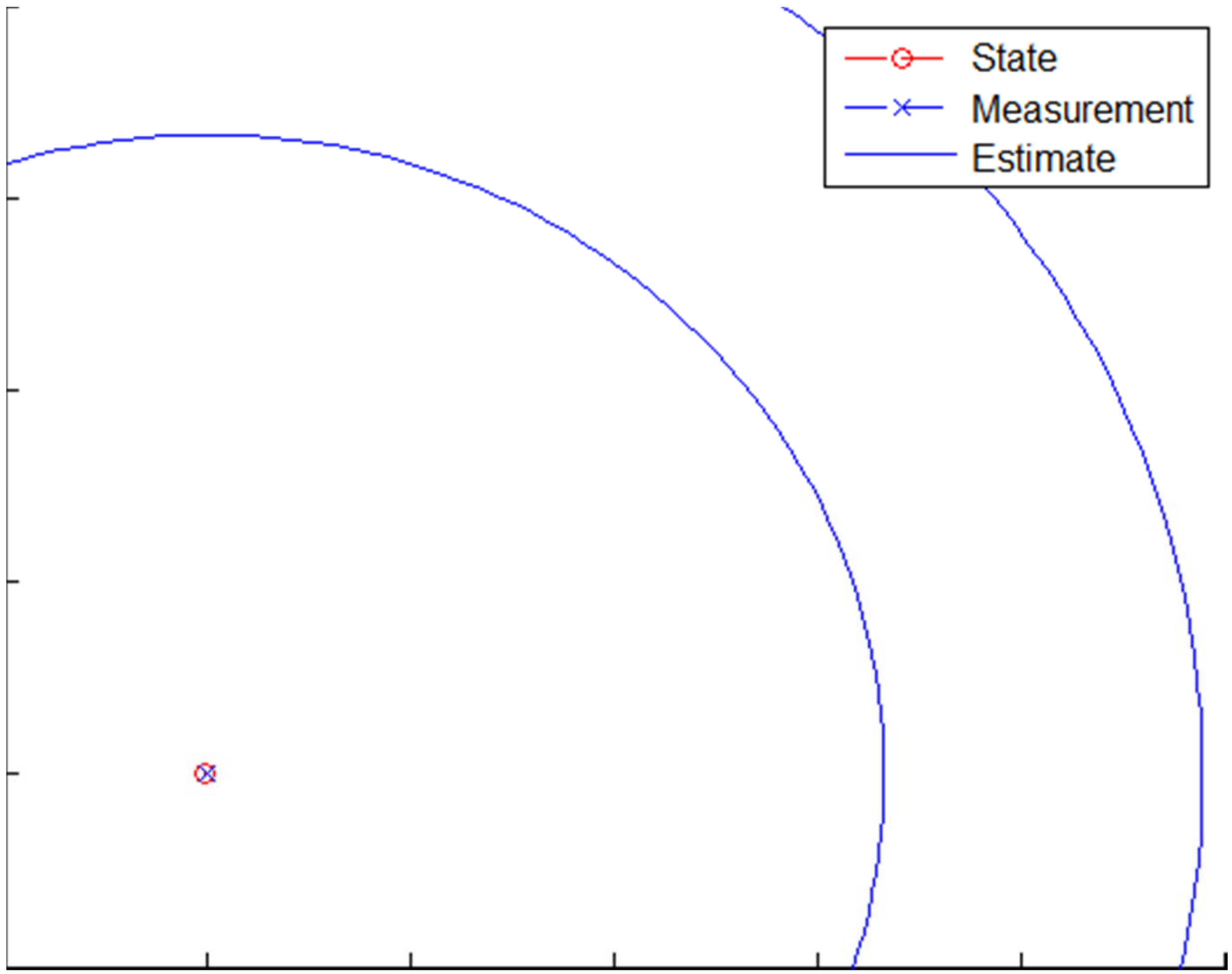
$$C_{1:9} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q_{1:9} = \begin{bmatrix} 0.1 & -0.01 \\ -0.01 & 0.05 \end{bmatrix}$$

      - Full state measurement on the 10^{th} time step

$$C_{10} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q_{10} = \begin{bmatrix} 0.004 & 0 & -0.001 & 0 \\ 0 & 0.1 & 0 & -0.01 \\ -0.001 & 0 & 0.001 & 0 \\ 0 & -0.01 & 0 & 0.05 \end{bmatrix}$$

KALMAN FILTER

- Example
  - Multi-rate estimation



Kalman Filter

- Alternate formulation: Information Filter
  - Provides possibility for computational savings when taking many redundant measurements
  - Based on information theory concepts (Fisher Information)
  - Define the Information Matrix as the inverse of the covariance

$$\Omega_t = \Sigma_t^{-1}$$

  - Define the Information vector as

$$\xi_t = \Sigma_t^{-1} \mu_t$$

Kalman Filter

- Information Filter
  - Substitution into the Kalman filter equation yields
    1. Prediction update

$$\bar{\xi}_t = \bar{\Omega}_t (A_t \Omega_{t-1}^{-1} \xi_{t-1} + B_t u_t)$$
$$\bar{\Omega}_t = (A_t \Omega_{t-1}^{-1} A_t^T + R_t)^{-1}$$

    2. Measurement update

$$\xi_t = C_t Q_t^{-1} y_t + \bar{\xi}_t$$
$$\Omega_t = C_t^T Q_t^{-1} C_t + \bar{\Omega}_t$$

Kalman Filter

- Information Filter
  - The matrix inversion is now embedded in the prediction update
    - Belief and predicted belief inverse depend on the number of states
    - For Kalman filter, gain inverse depends on the number of measurements
    - This can be a significant savings in some cases
  - To compute state estimate

$$\mu_t = \Omega_t^{-1} \xi_t$$

    - Covariance already calculated

OUTLINE

- Bayes Filter Framework
- Kalman Filter
- Extended Kalman Filter
- Particle Filter

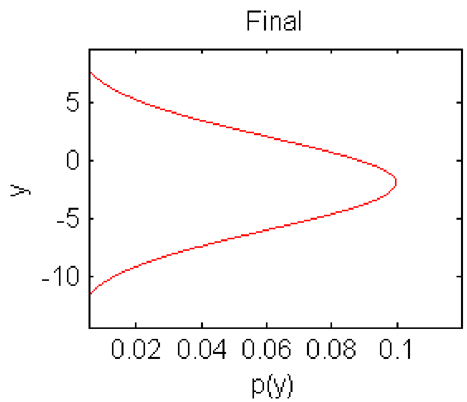
Extended Kalman Filter

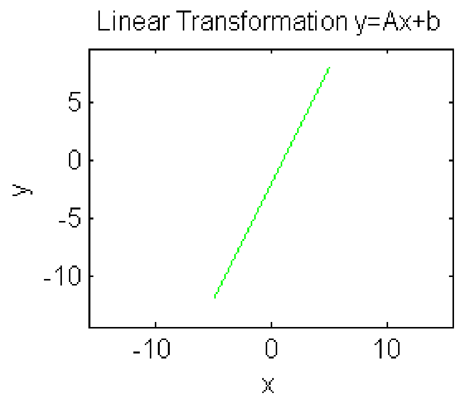
- Kalman Filter requires linear motion and measurement models
  - Results in compact, recursive estimation technique
  - Not very realistic for most applications
- Nonlinear models eliminate the guarantee that the belief distributions remain Gaussian
  - No longer able to simply track mean and covariance
  - No closed form solution to Bayes filter algorithm can be found for general nonlinear model

Extended Kalman Filter

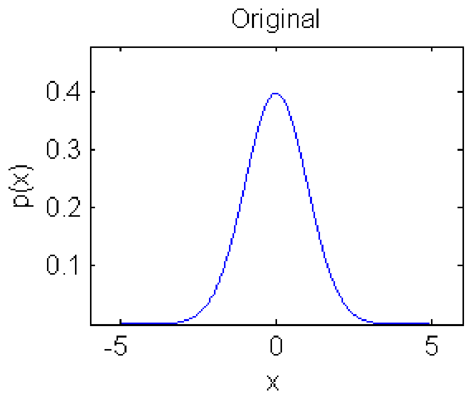
- Effect of nonlinearity on Gaussian distribution
  - Linear transformation

$$y = 2x - 2$$





$$y \sim N(-2,4)$$



$$x \sim N(0,1)$$

Extended Kalman Filter

- Arbitrary distribution generation
  - Take 5,000,000 samples of original Gaussian

$$x^i \sim N(0,1), \quad i = 1,\ldots,n$$

  - Apply nonlinear transformation to each sample

$$y^i = \tan^{-1}\left(x^i\right)$$

  - Create histogram with 100 bins and normalize counts

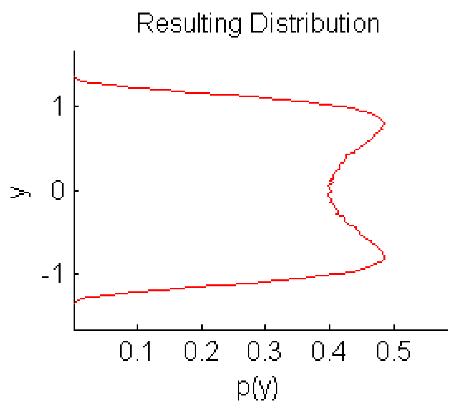
- Best Gaussian fit generation
  - Calculate mean and covariance of 5,000,000 transformed samples

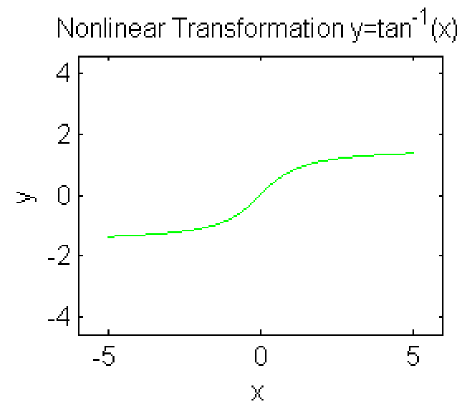
$$\mu_{BG} = \frac{1}{n}\sum_{i=1}^{n} y^i \qquad \Sigma_{BG} = \frac{1}{n-1}\sum_{i=1}^{n}\left(y^i - \mu_{BG}\right)\left(y^i - \mu_{BG}\right)^T$$

Extended Kalman Filter

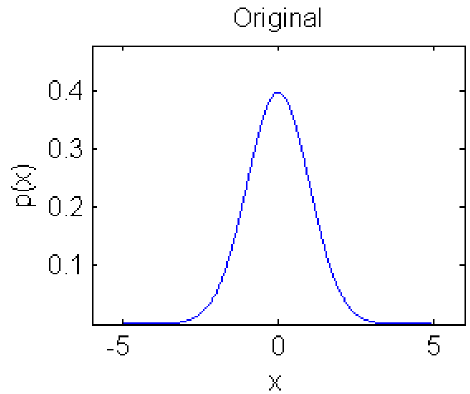
- Effect of nonlinearity on Gaussian distribution
  - Nonlinear transformation

$$y = \tan^{-1}(x)$$





$$y \sim ???$$

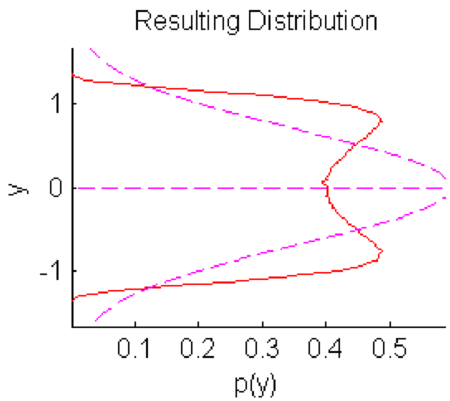


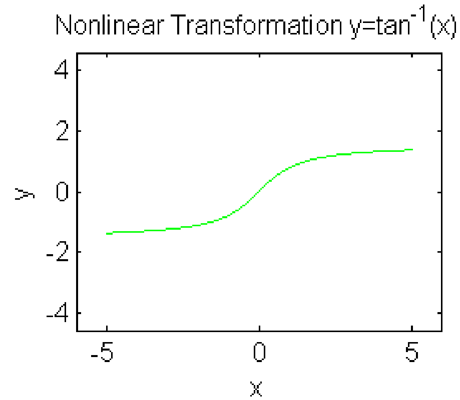
$$x \sim N(0,1)$$

Extended Kalman Filter

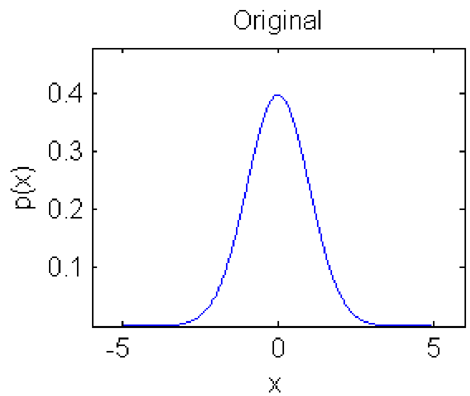
- Effect of nonlinearity on Gaussian distribution
  - Nonlinear transformation

$$y = \tan^{-1}(x)$$





$$y \sim N(\mu, \Sigma)??$$

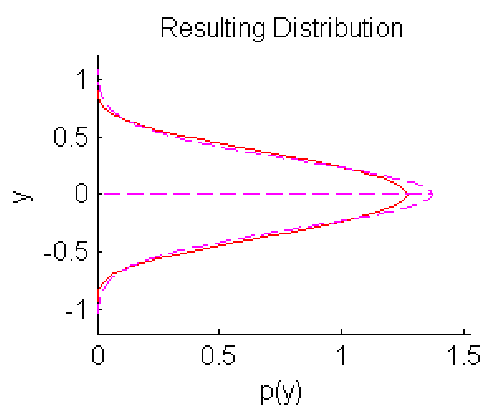


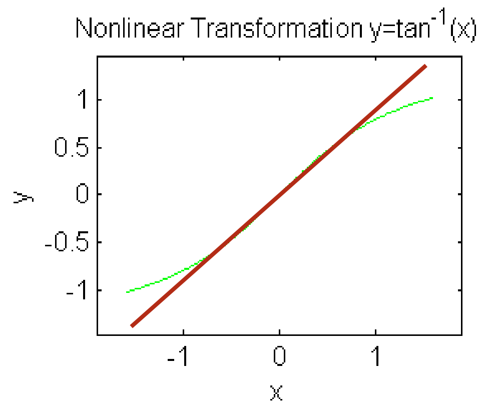
$$x \sim N(0,1)$$

Extended Kalman Filter

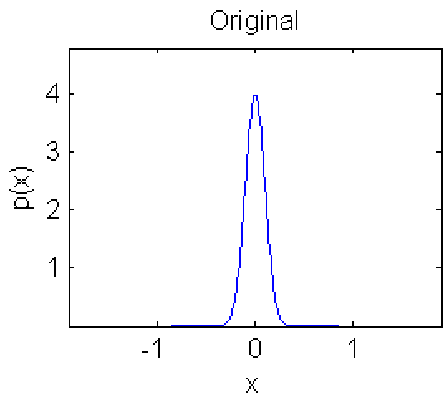
- Effect of nonlinearity on Gaussian distribution
  - Nonlinear transformation

$$y = \tan^{-1}(x)$$





$$y \sim N(\mu, \Sigma)$$



$$x \sim N(0, 0.1)$$

Extended Kalman Filter

- Extended Kalman Filter
  - A direct generalization of the Kalman filter to nonlinear motion and measurement models
    - Relies on linearization about current estimate
    - Works well when the problem maintains locally linear and Gaussian characteristics
    - Computationally similar to Kalman Filter
    - Covariance can diverge when approximation is poor!

Extended Kalman Filter

- Extended Kalman Filter Modeling Assumption
  - Prior over the state is Gaussian

$$p(x_0) \sim N(\mu_0, \Sigma_0)$$

  - Motion model, nonlinear but still with additive Gaussian disturbances

$$x_t = g(x_{t-1}, u_t) + \varepsilon_t \qquad \varepsilon_t \sim N(0, R_t)$$

  - Measurement model also nonlinear with additive Gaussian noise

$$y_t = h(x_t) + \delta_t \qquad \delta_t \sim N(0, Q_t)$$

  - Nonlinearity destroys certainty that beliefs remain Gaussian

Extended Kalman Filter

- Recall Kalman Filter Algorithm
  1. Prediction update

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

  2. Measurement update

$$K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (y_t - C_t \bar{\mu}_t)$$

$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

- B_t only enters predicted mean calculation
- A_t, C_t also affect covariance

Extended Kalman Filter

- How to update beliefs while maintaining Gaussian form of distribution?

  - Key idea of EKF

    - The mean can be propagated through the nonlinear model

    - The covariance can be updated with a locally linear approximation to the model

Extended Kalman Filter

- First Order Taylor Series Expansion

  - Motion model
    - Linearize about most likely state (the previous mean)

$$g(x_{t-1},u_t) \approx g(\mu_{t-1},u_t) + \frac{\partial}{\partial x_{t-1}} g(x_{t-1},u_t)\bigg|_{x_{t-1}=\mu_{t-1}} (x_{t-1}-\mu_{t-1})$$

$$= g(\mu_{t-1},u_t) + G_t \cdot (x_{t-1}-\mu_{t-1})$$

Extended Kalman Filter

- First Order Taylor Series Expansion
  - Measurement Model
    - Linearize about most likely state (the predicted mean)

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial}{\partial x_t} h(x_t)\bigg|_{x_t=\bar{\mu}_t} (x_t - \bar{\mu}_t)$$

$$= h(\bar{\mu}_t) + H_t \cdot (x_t - \bar{\mu}_t)$$

  - Both models are now linear
    - Only valid near point of linearization

Extended Kalman Filter

- Prediction Update
  - Only new information is input u_t
  - Prediction update is a linear transformation of belief at previous time step
    - Motion model is

$$x_t = g(\mu_{t-1}, u_t) + G_t \cdot (x_{t-1} - \mu_{t-1}) + \varepsilon_t$$

    - Motion disturbance, previous belief are Gaussian so this is remains addition of Gaussian distributions

$$bel(x_{t-1}) \sim N(\mu_{t-1}, \Sigma_{t-1}) \qquad \varepsilon_t \sim N(0, R_t)$$

    - Therefore the predicted mean and covariance are

$$\bar{\mu}_t = g(\mu_{t-1}, u_t) + 0 + 0$$
$$\bar{\Sigma}_t = 0 + G_t \Sigma_{t-1} G_t^T + R_t$$

EXTENDED KALMAN FILTER

- Measurement Update
  - Follows same arguments as Kalman Filter derivation
    - MMSE estimator
    - Assume form of measurement update (linear, Kalman Gain)
    - Substitute in approximate measurement and motion models
    - Mean update relies on nonlinear model
    - Gain, covariance update rely on linearization

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t (y_t - h(\bar{\mu}_t))$$

$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

Extended Kalman Filter

- Extended Kalman Filter Algorithm
  1. Prediction Update

$$G_t = \frac{\partial}{\partial x_{t-1}} g(x_{t-1}, u_t)\Big|_{x_{t-1}=\mu_{t-1}}$$

$$\bar{\mu}_t = g(\mu_{t-1}, u_t)$$

$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

  2. Measurement Update

$$H_t = \frac{\partial}{\partial x_t} h(x_t)\Big|_{x_t=\bar{\mu}_t}$$

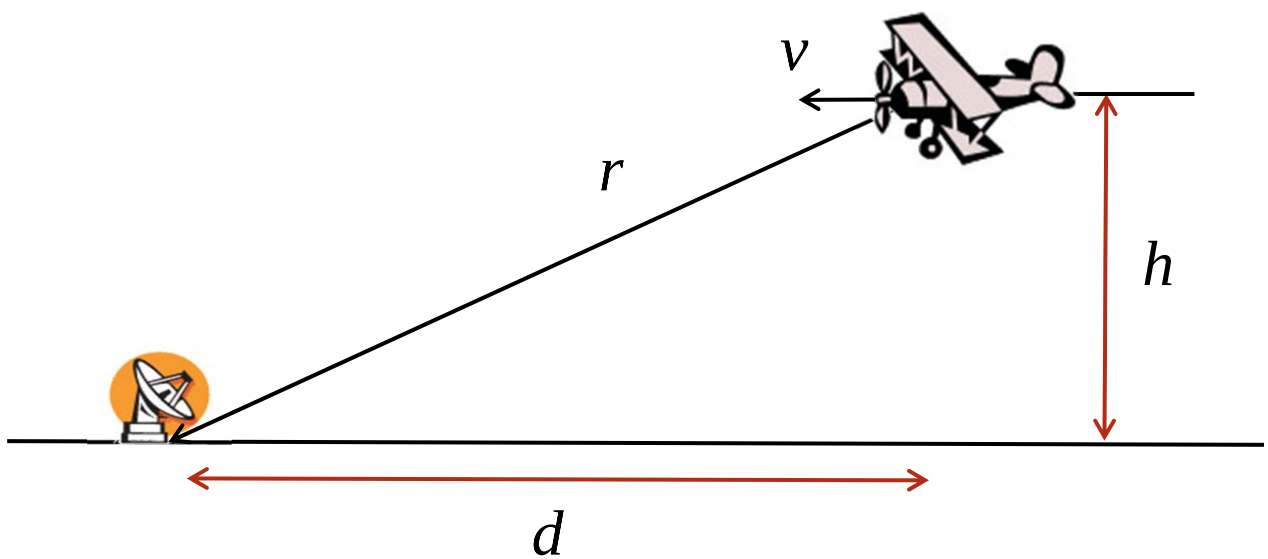
$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$

$$\mu_t = \bar{\mu}_t + K_t(y_t - h(\bar{\mu}_t))$$

$$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

Extended Kalman Filter

- Example
  - Radar measurement of an airplane position while flying at constant altitude and velocity



Extended Kalman Filter

- Example
  - State

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d \\ v \\ h \end{bmatrix}$$

Initial

$$x_0 = \begin{bmatrix} 20 \\ -2 \\ 3 \end{bmatrix}$$

  - Motion Model
    - Linear, no input (very simple)

$$x_t = g(x_{t-1}) + \varepsilon_t \quad \Rightarrow \quad \begin{aligned} x_{1,t} &= x_{1,t-1} + x_{2,t-1}dt \\ x_{2,t} &= x_{2,t-1} \\ x_{3,t} &= x_{3,t-1} \end{aligned}$$

Extended Kalman Filter

- Example
  - Measurement Model

$$r = \sqrt{d^2 + h^2} + \delta_t$$

- Using state variables

$$y_t = \sqrt{x_{1,t}^2 + x_{3,t}^2} + \delta_t \qquad h(x_t) = \sqrt{x_{1,t}^2 + x_{3,t}^2}$$

  - Linearization of measurement model

$$\frac{\partial h}{\partial x_1} = \frac{1}{2}\left(x_1^2 + x_3^2\right)^{-1/2} 2x_1 = x_1 \Big/ \sqrt{x_1^2 + x_3^2}$$

$$\frac{\partial h}{\partial x_3} = \frac{1}{2}\left(x_1^2 + x_3^2\right)^{-1/2} 2x_3 = x_3 \Big/ \sqrt{x_1^2 + x_3^2}$$

Extended Kalman Filter

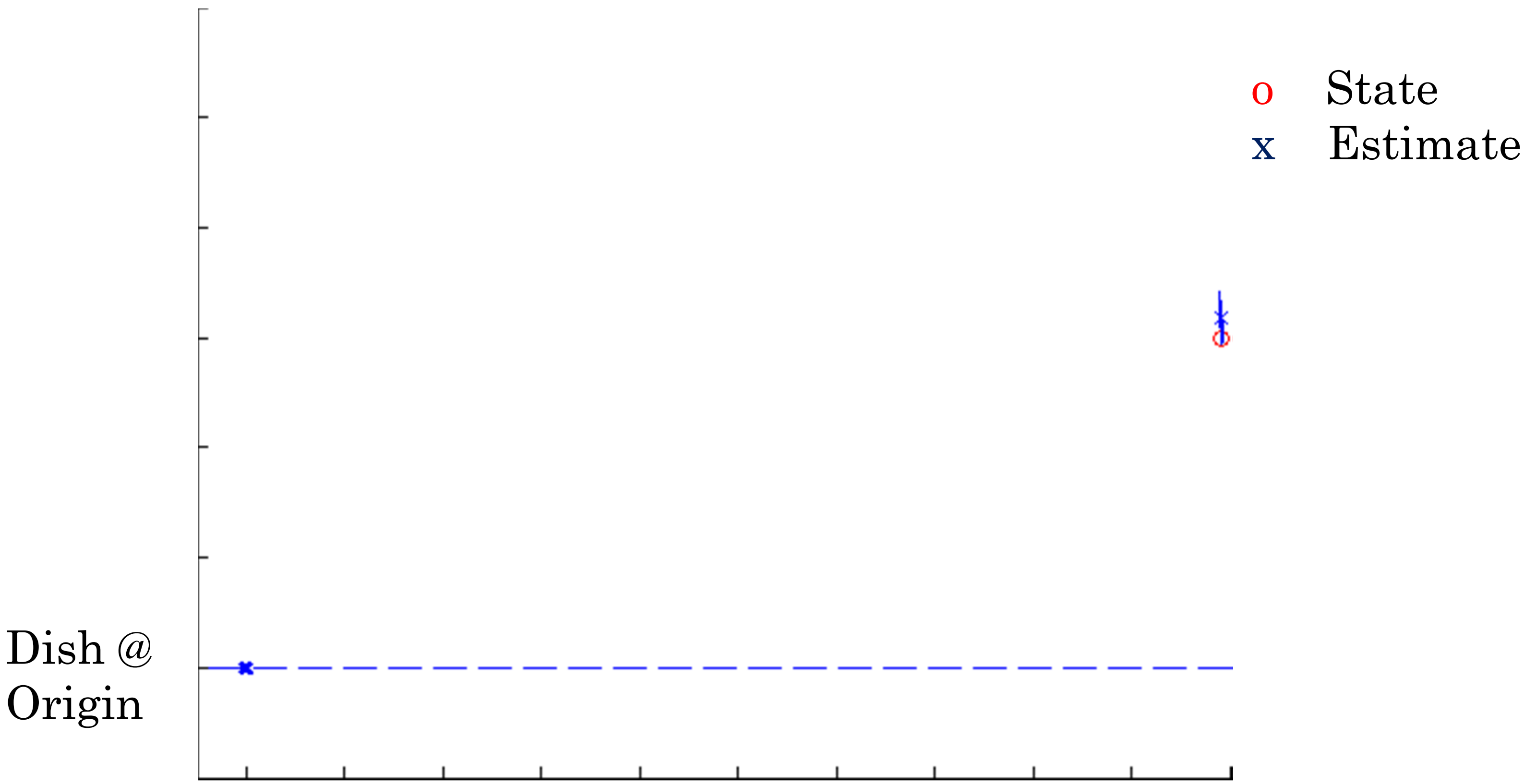
- Sample Code

```
%% Extended Kalman Filter Estimation
% Prediction update
mup = Ad*mu;
Sp = Ad*S*Ad' + R;

% Measurement update
Ht = [(mup(1))/(sqrt(mup(1)^2 + mup(3)^2));
      0;
      (mup(3))/(sqrt(mup(1)^2 + mup(3)^2))]';

K = Sp*Ht'*inv(Ht*Sp*Ht'+Q);
mu = mup + K*(y(:,t)-sqrt(mup(1)^2 + mup(3)^2));
S = (eye(n)-K*Ht)*Sp;
```

Extended Kalman Filter

- Results
  - Low noise, fairly accurate prior

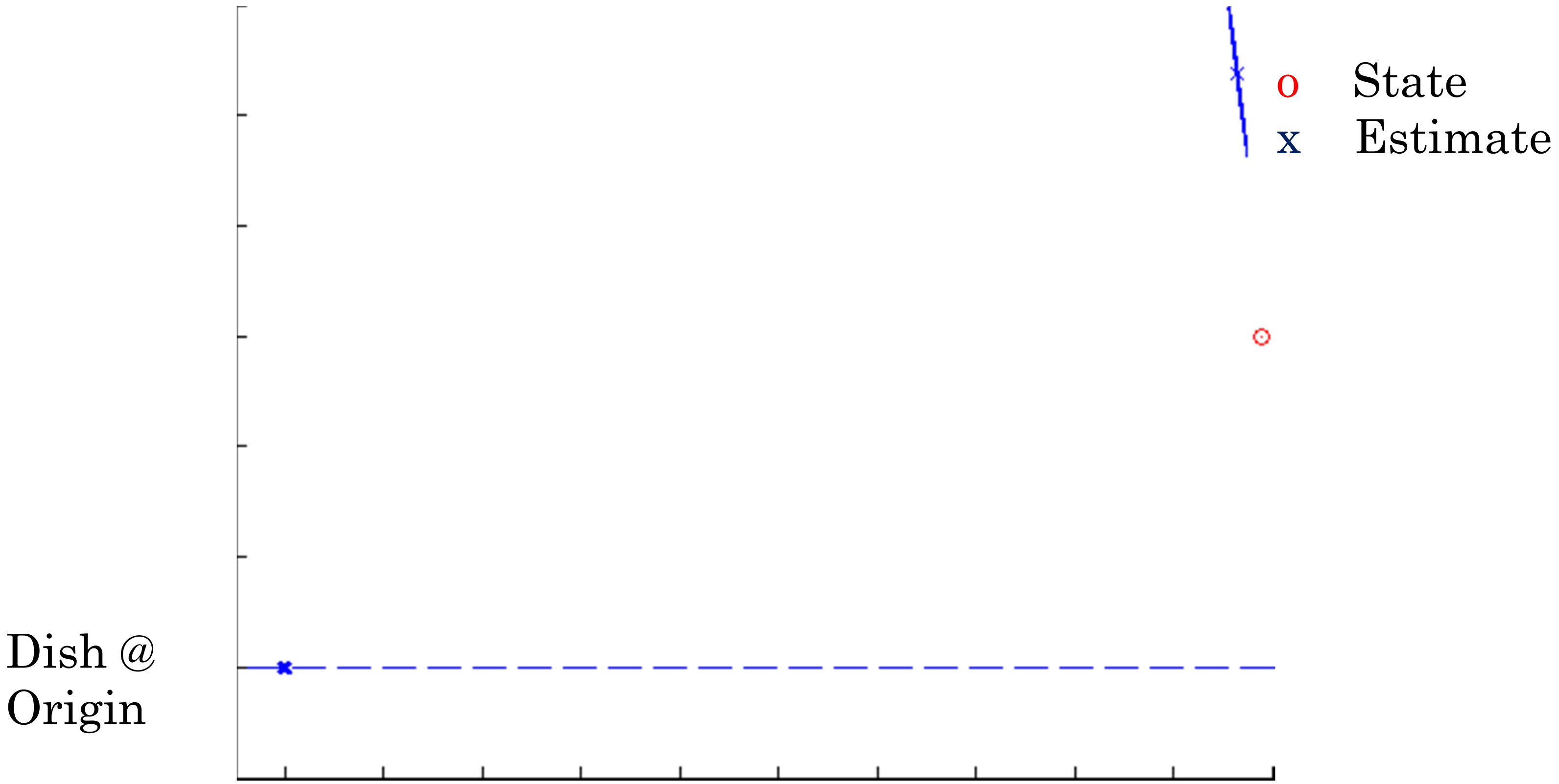
$$\mu_0 = [22 \quad -1.8 \quad 3.5]^T$$



EXTENDED KALMAN FILTER

- Results
  - Low noise, incorrect prior

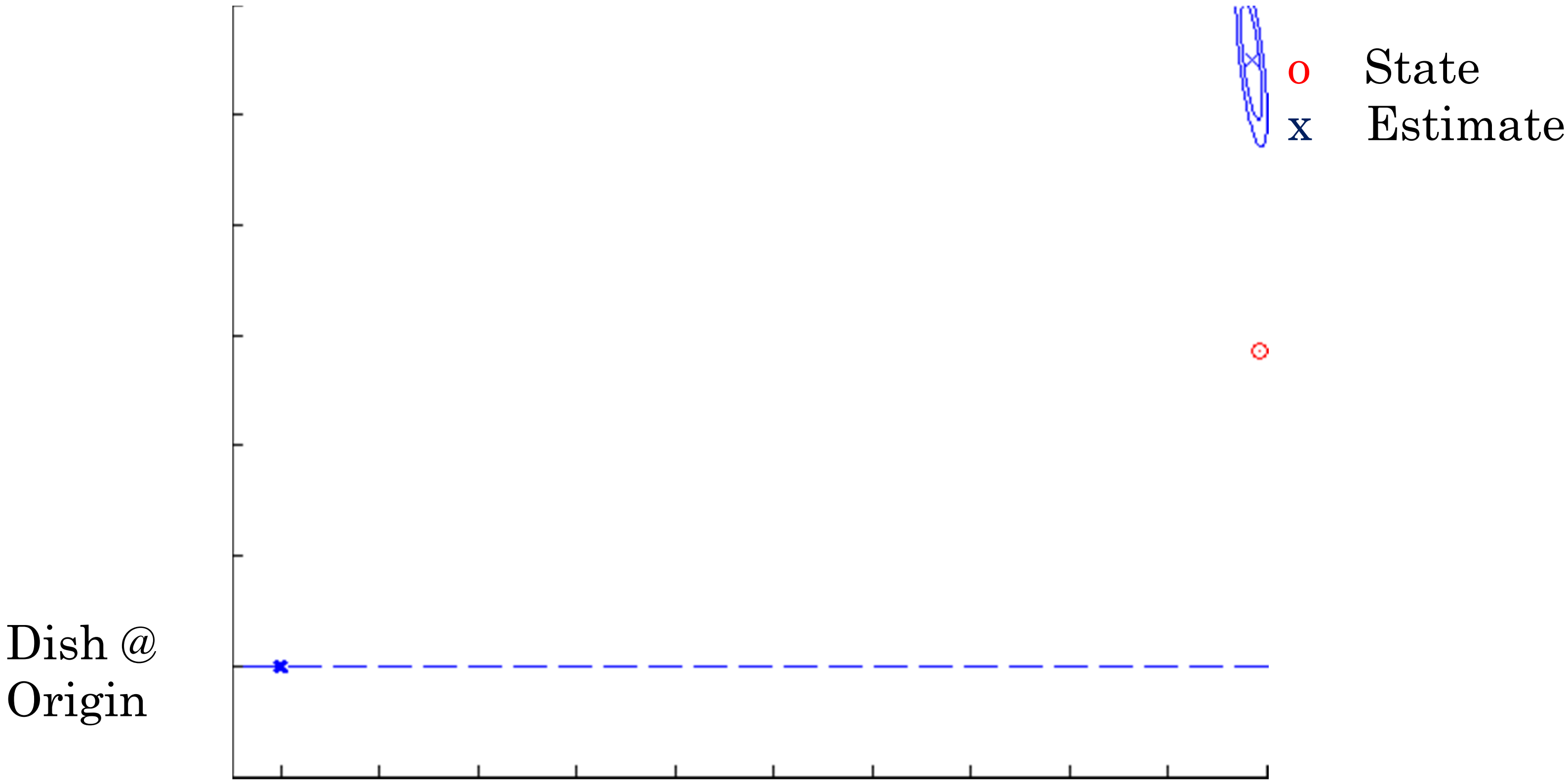
$$\mu_0 = [22 \quad -1.8 \quad 6]^T$$



Extended Kalman Filter

- Results
  - Noisy noise, big disturbances, incorrect prior

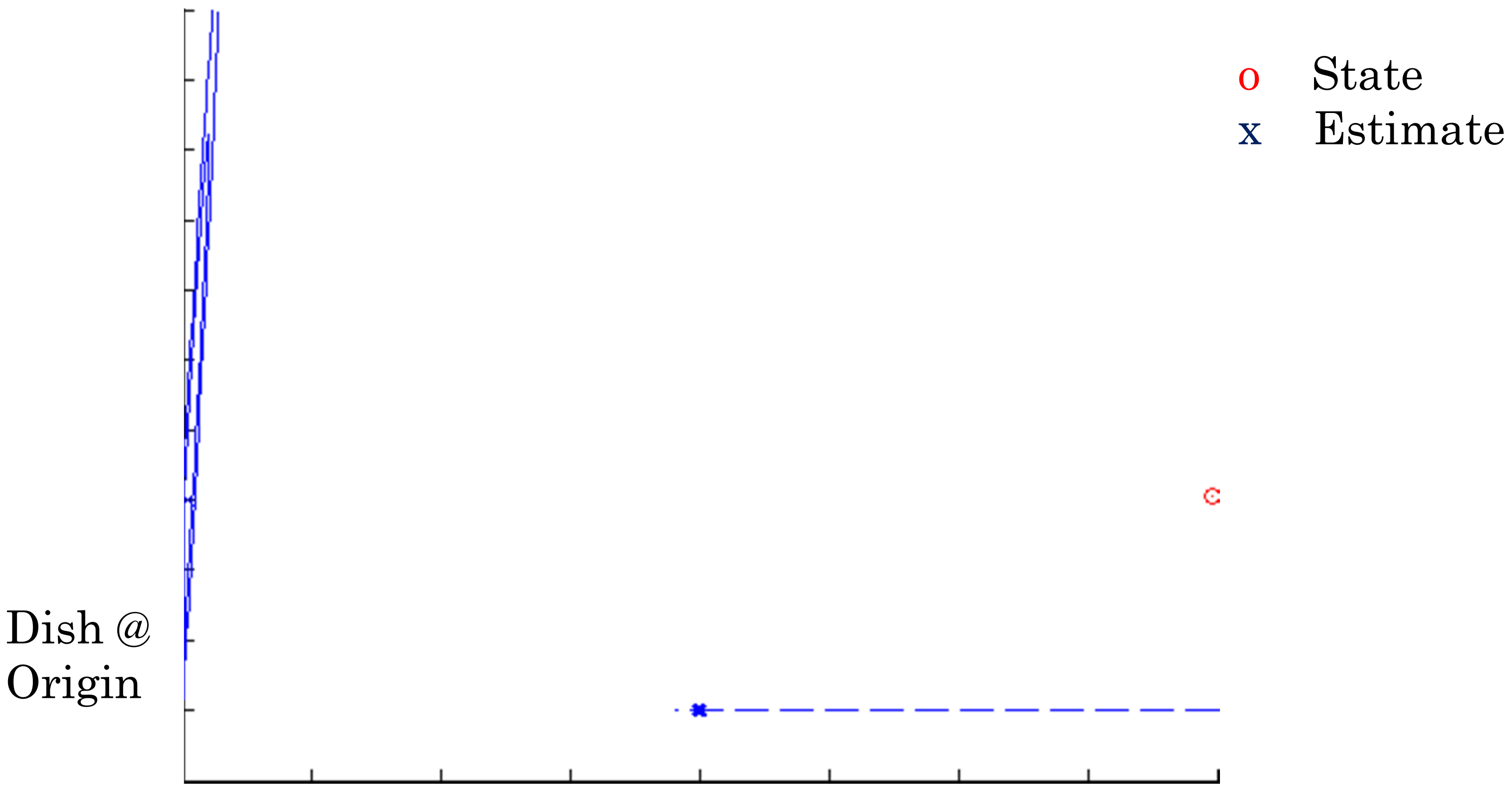
$$\mu_0 = [22 \quad -1.8 \quad 6]^T$$



Extended Kalman Filter

- Results
  - Symmetrically incorrect prior

$$\mu_0 = [-20 \quad 2 \quad 3]^T$$



EXTENDED KALMAN FILTER

- Summary
  - Direct extension of KF to nonlinear models
  - Use Taylor series expansion to find locally linear approximations
  - No longer optimal
  - Most effective when covariance is low
    - Local linear approximation more likely to be accurate over range of distribution
  - Covariance update may diverge

Extra Slides

Kalman Filter

- Reminder on generating multivariate random noise samples
  - Define two distributions, the one of interest and the standard normal distribution

$$\delta \sim N(\mu, \Sigma) \qquad \omega \sim N(0, I)$$

  - If the covariance is full rank, it can be diagonalized
    - Symmetry implies positive semidefiniteness

$$\begin{aligned}\Sigma &= E\lambda E^T \\ &= E\lambda^{1/2} I \lambda^{1/2} E^T \\ &= HIH^T\end{aligned}$$

  - Can now relate the two distributions (linear identity)

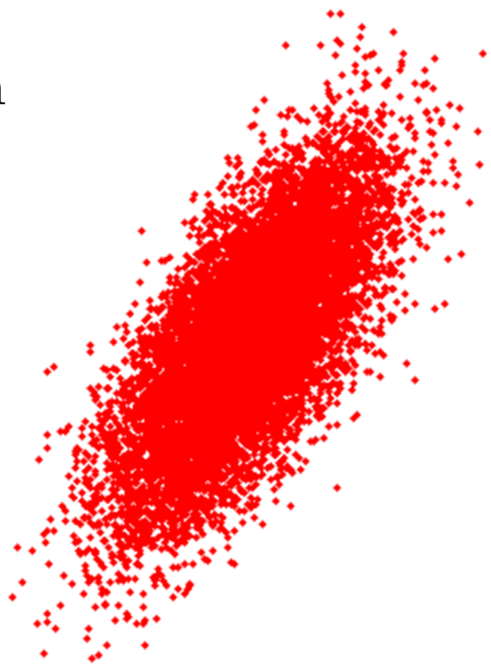
$$\delta \sim N(\mu, HIH^T)$$
$$\delta = \mu + Hw$$

Kalman Filter

- To implement this in Matlab for simulation purposes
  - Define $\mu,\sum$
  - Find eigenvalues , λ, and eigenvectors, E of $\sum$
  - The noise can then be created with

$$\delta = \mu + E\lambda^{1/2}\text{randn(n,1)}$$

$$\Sigma = \begin{bmatrix} 4 & 4 \\ 4 & 8 \end{bmatrix}$$

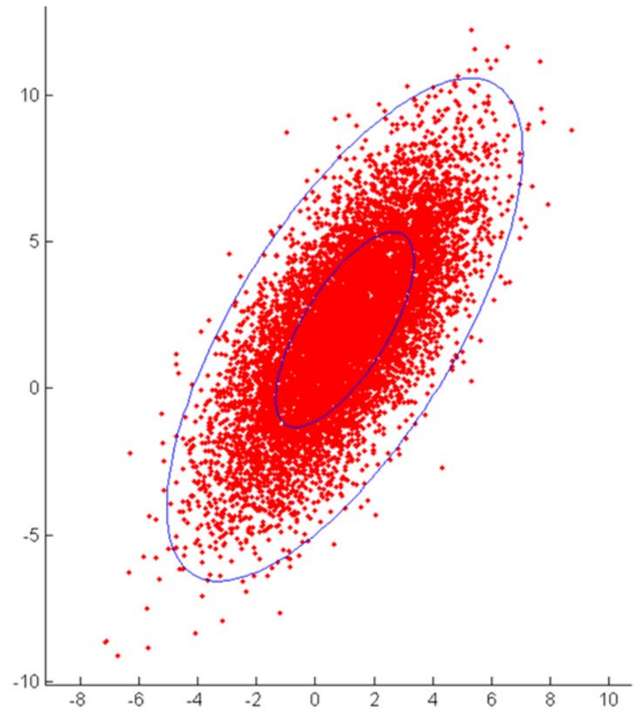


KALMAN FILTER

- Note on confidence ellipses
  - Lines of constant probability
    - Found by setting pdf exponent to a constant
    - Principal axes are eigenvectors of covariance
    - Magnitudes depend on eigenvalues of covariance

$$\mu = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 4 & 4 \\ 4 & 8 \end{bmatrix}$$

50%, 99% error ellipses
Not easily computed,
code provided



Unscented Kalman Filter

- The EKF used linearization about the predicted/previous state estimate to update the mean and covariance of the current estimate
  - Approximation of a nonlinear transformation of a Gaussian distribution by linear transformation of the mean and covariance

- There are other ways to approximate this transformation
  - Unscented transform leads to better estimates of resulting mean and covariance in some cases
  - Relies on a set of samples known as sigma points or particles, that get transformed directly
  - UKF first published in 1997, still being discussed, extended, solidified.

Unscented Kalman Filter

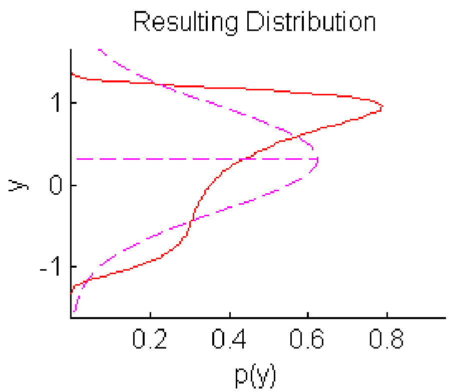
- Key idea: Unscented transform
  - It is more accurate to approximate a distribution using samples than it is to approximate an arbitrary nonlinear function through linearization.

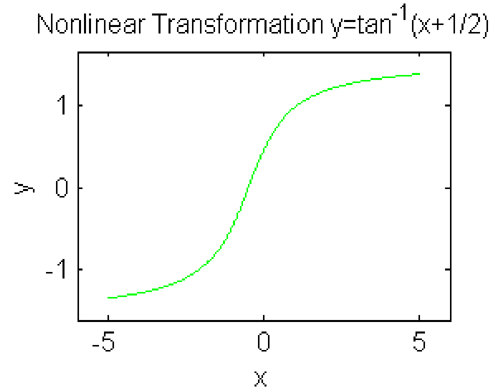
  - Let's first go back to the nonlinear function of a Gaussian and see what the EKF is doing.

Unscented Transform

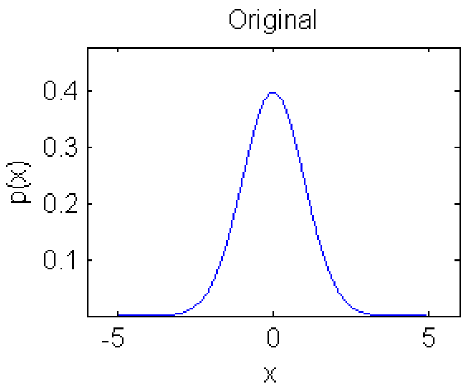
- Effect of nonlinearity on Gaussian distribution
  - Nonlinear transformation

$$y = \tan^{-1}(x + 1/2)$$





$$y \sim N(\mu, \Sigma)??$$



$$x \sim N(0,1)$$

UNSCENTED TRANSFORM

- Nonlinear distribution generation
  - Take 5,000,000 samples of original Gaussian

$$x^i \sim N(0,1), \quad i = 1, \ldots, n$$

  - Apply nonlinear transformation to each sample

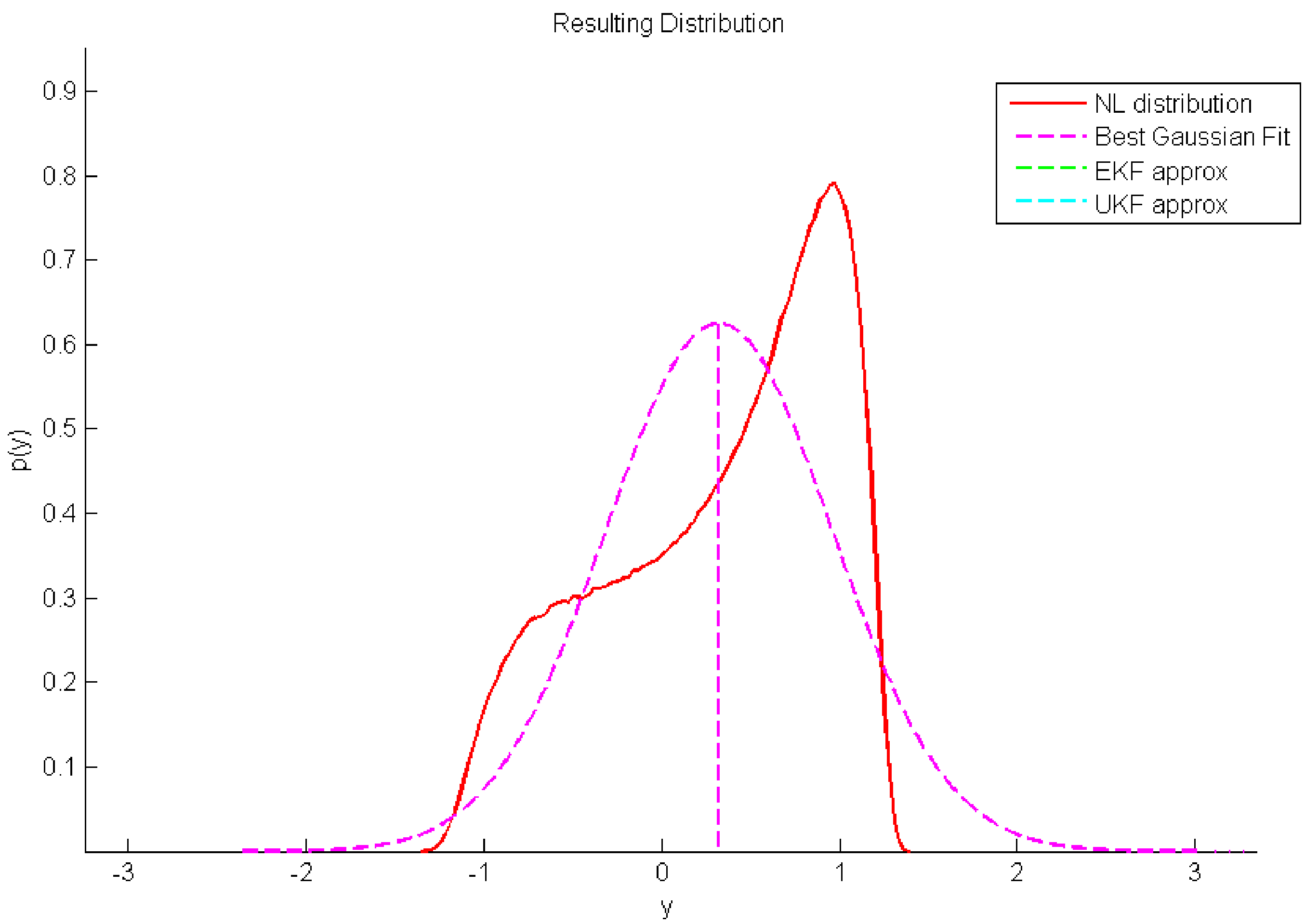
$$y^i = \tan^{-1}\left(x^i + 1/2\right)$$

  - Create histogram with 100 bins and normalize counts

- Best Gaussian fit generation
  - Calculate mean and covariance of 5,000,000 transformed samples

$$\mu_{BG} = \frac{1}{n}\sum_{i=1}^{n} y^i \qquad \Sigma_{BG} = \frac{1}{n-1}\sum_{i=1}^{n}\left(y^i - \mu_{BG}\right)\left(y^i - \mu_{BG}\right)^T$$

Unscented Transform



UNSCENTED TRANSFORM

- Extended Kalman Filter approximation generation
  - Linearize nonlinear function about mean

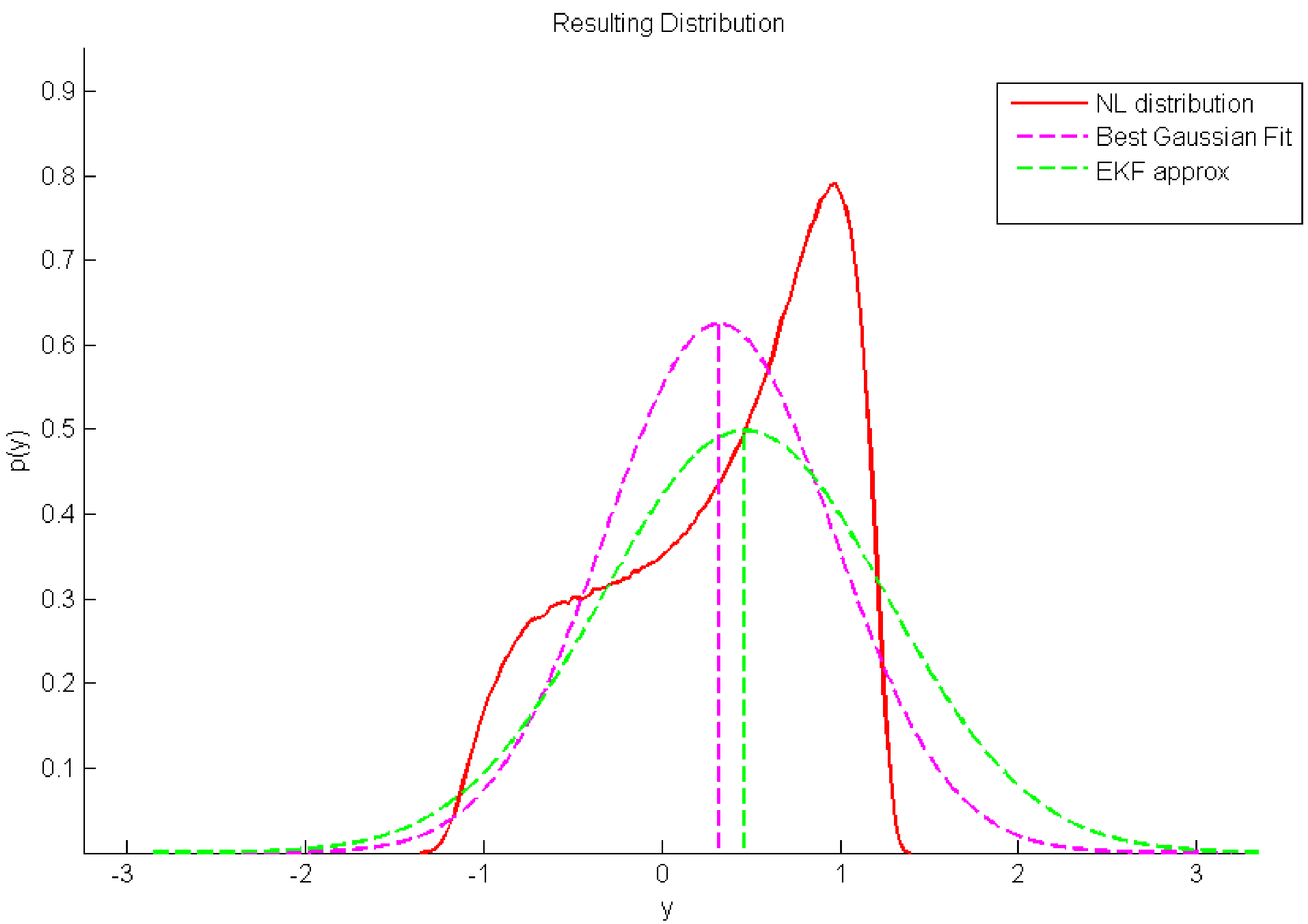
$$G = \frac{\partial}{\partial x}\left(\tan^{-1}\left(x + 1/2\right)\right)\Bigg|_{x=\mu}$$

$$= \frac{1}{\left(\mu + 1/2\right)^2 + 1}$$

  - Propagate mean through nonlinear function and covariance through linearized function

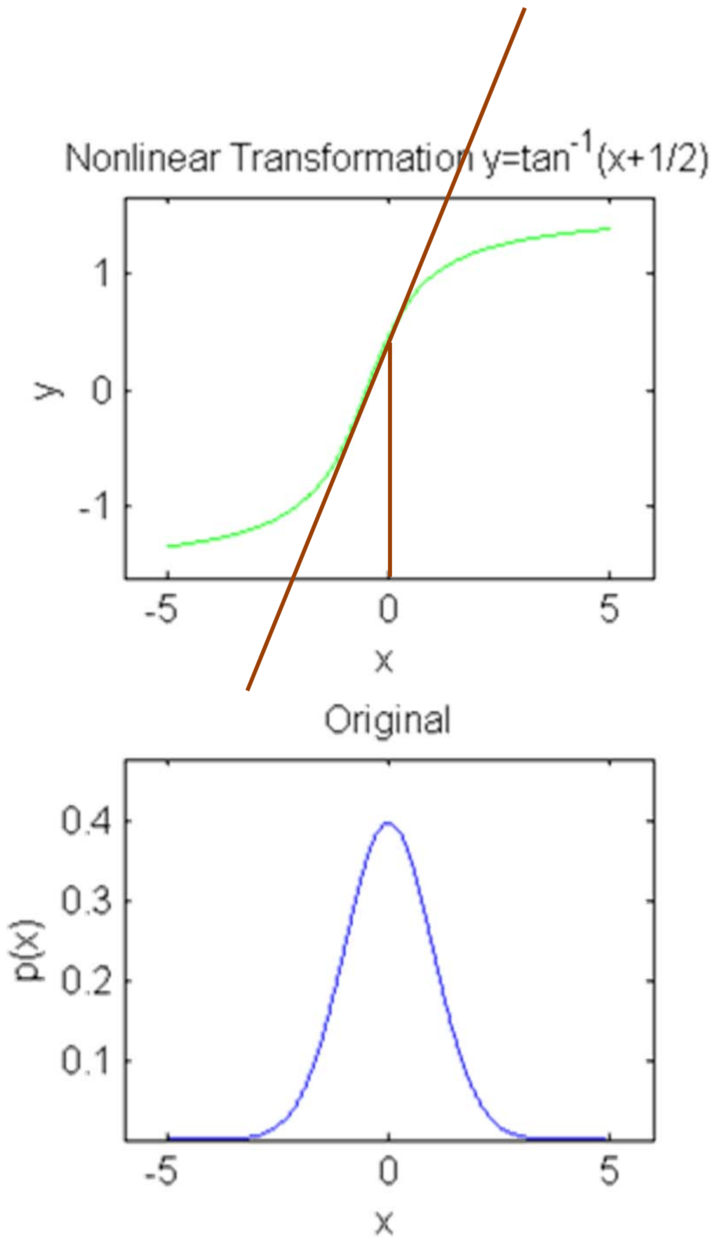
$$\mu_{EKF} = \tan^{-1}\left(\mu + 1/2\right) \qquad \Sigma_{EKF} = G\Sigma G^T$$

UNSCENTED KALMAN FILTER



LINEARIZATION

- Linearization over-predicts mean shift
  - Assumes symmetry of atan
- Covariance over-predicted as well
  - atan has effect of piling up tails at +/- 1.57



Unscented Transform

- The unscented transform can also be used
  - Linearization is a first order approximation
  - The unscented transform is second order accurate, and can be tuned to reduce fourth order errors
- The transform relies on a set of specially chosen samples known as sigma points
  - 2n+1 points chosen to capture the transformation of the distribution

UNSCENTED TRANSFORM

- In 1D case, the unscented transform select 3 points

$$\chi^{[0]} = \mu$$
$$\chi^{[1]} = \mu + \sigma$$
$$\chi^{[2]} = \mu - \sigma$$

- And we select weights so that we can recover the original mean and variance

$$\mu = \sum_{i=0}^{2} w_m^{[i]} \chi^{[i]} \qquad \sigma^2 = \sum_{i=0}^{2} w_c^{[i]} \left(\chi^{[i]} - \mu \right)^2$$

Unscented Transform

- We then pass the sigma points through the nonlinear function

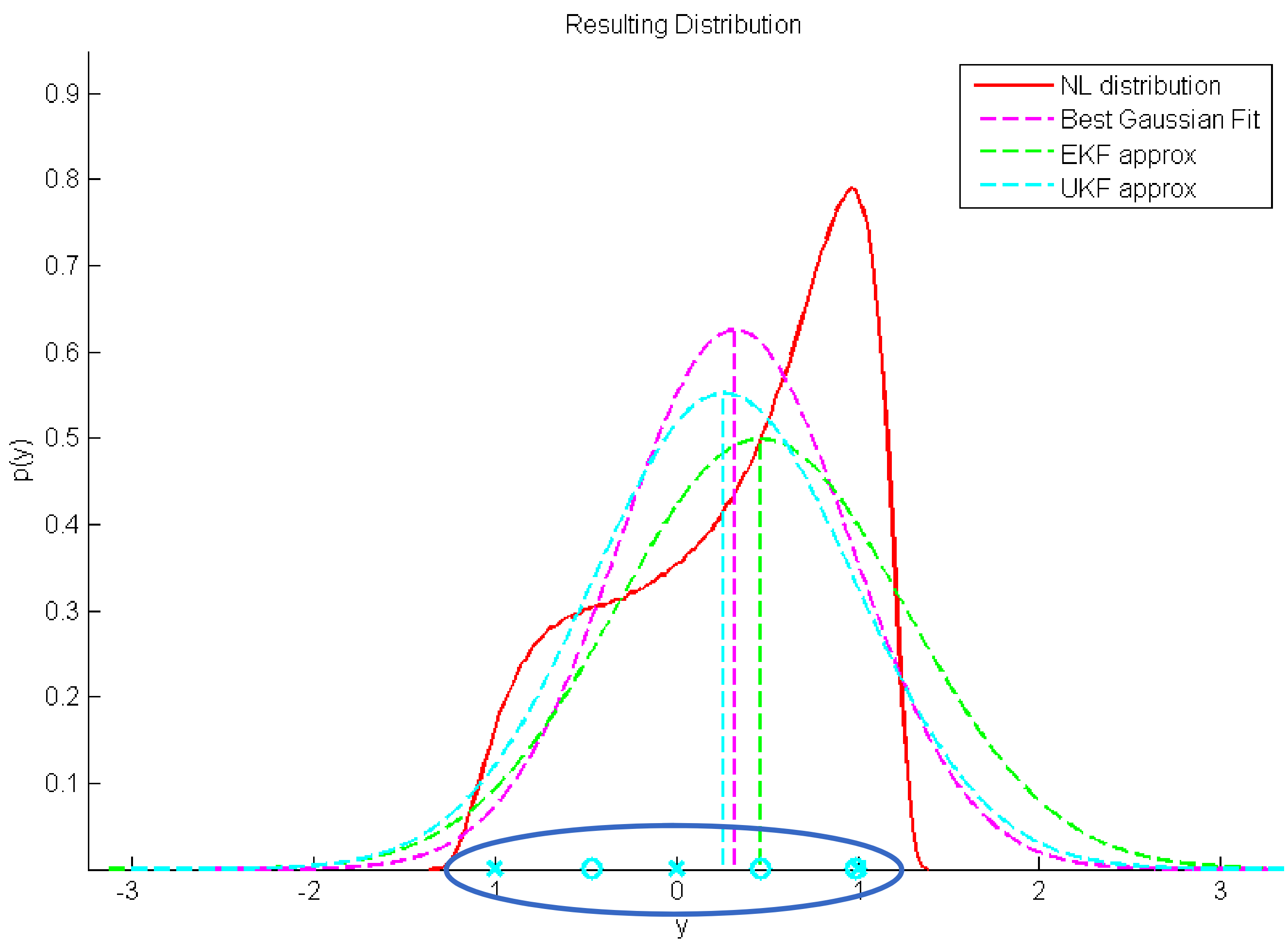
$$\Upsilon^{[i]} = f(\chi^{[i]})$$

- And construct the new mean and variance using the same weights

$$\mu_{\Upsilon} = \sum_{i=0}^{2} w_m^{[i]} \Upsilon^{[i]}$$

$$\sigma_{\Upsilon}^2 = \sum_{i=0}^{2} w_c^{[i]} \left(\Upsilon^{[i]} - \mu_{\Upsilon} \right)^2$$

UNSCENTED TRANSFORM



Unscented Transform

- In general, the sigma points are chosen as follows

$$\chi^{[0]} = \mu$$

$$\chi^{[i]} = \mu + \left(\sqrt{(n+\lambda)\Sigma}\right)_i, \; i = 1,\ldots,n$$

$$\chi^{[n+i]} = \mu - \left(\sqrt{(n+\lambda)\Sigma}\right)_i, \; i = 1,\ldots,n$$

  - Generalized Std. Dev. is square root of covariance
  - Here the square root of the covariance matrix is ambiguous, but must satisfy $A = \sqrt{B} \Rightarrow A^T A = B$
    - Can use sqrtm, which returns the unique solution with non-negative eigenvalues,
    - Or use chol, the cholesky decomposition, which returns an upper triangular square root and is very efficient
      - Assumes symmetry

Unscented Transform

- The parameter λ defines the weights to use for generating the mean and covariance, can be tuned

$$\lambda = \alpha^2 (n + \kappa) - n$$

  - α governs the spread of the sigma points about the mean
    - the larger the α the larger the spread of sigma points
    - Usually, $0 \leq \alpha \leq 1$
  - κ ensures positive semi-definiteness if $\kappa \geq 0$
    - Can be left at 0 safely (ignored)
    - Also affects the spread of sigma points

Unscented Transform

- The sigma points are then propagated through the nonlinear function

$$\Upsilon^{[i]} = f(\chi^{[i]})$$

- And a mean and covariance is extracted using special weights for the sigma points

$$\mu_{\Upsilon} = \sum_{i=0}^{2n} w_m^{[i]} \Upsilon^{[i]}$$

$$\Sigma_{\Upsilon} = \sum_{i=0}^{2n} w_c^{[i]} \left(\Upsilon^{[i]} - \mu_{\Upsilon}\right)\left(\Upsilon^{[i]} - \mu_{\Upsilon}\right)^T$$

UNSCENTED TRANSFORM

- The weights are defined as

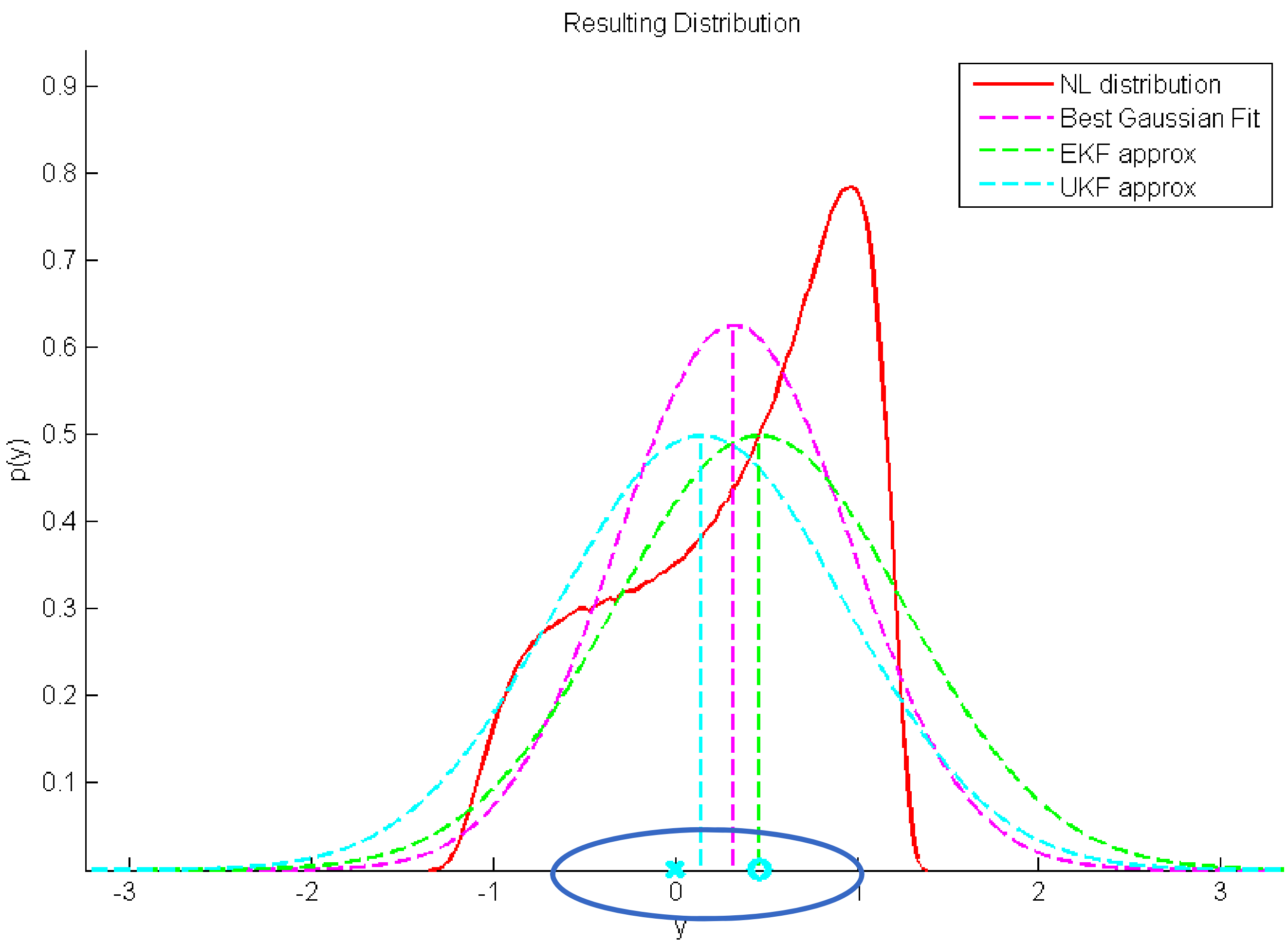
$$w_m^{[0]} = \frac{\lambda}{n+\lambda} \qquad w_c^{[0]} = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta$$

$$w_m^{[i]} = \frac{1}{2(n+\lambda)}, i = 1,\ldots,2n \qquad w_c^{[i]} = \frac{1}{2(n+\lambda)}, i = 1,\ldots,2n$$

- With another tunable parameter β,
  - Can be ignored
  - Or set to 2
    - reduces errors in some of the fourth order terms for a Gaussian prior

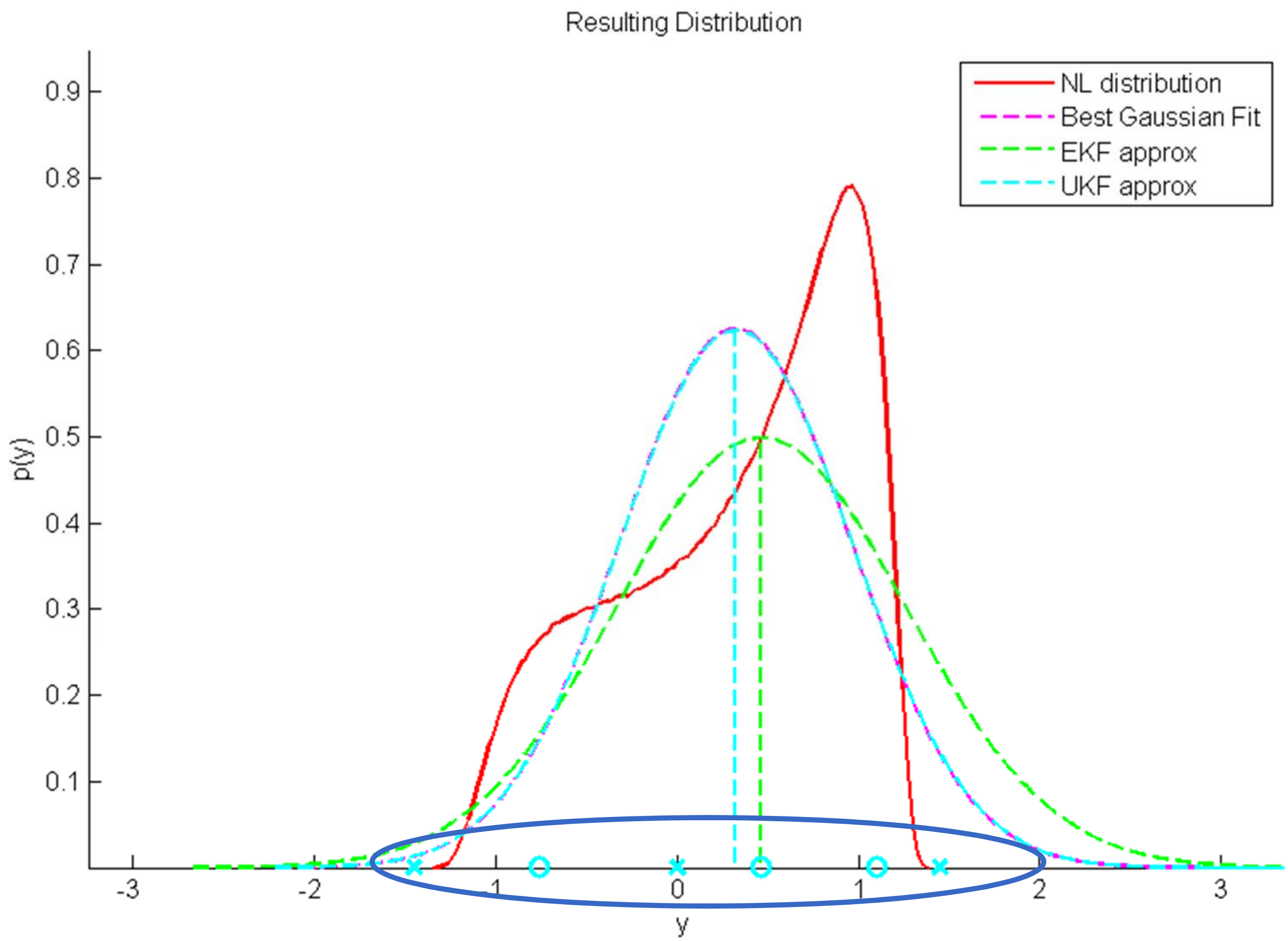
Unscented Transform

- Select $\alpha = 0.01, \kappa = 0, \beta = 0$



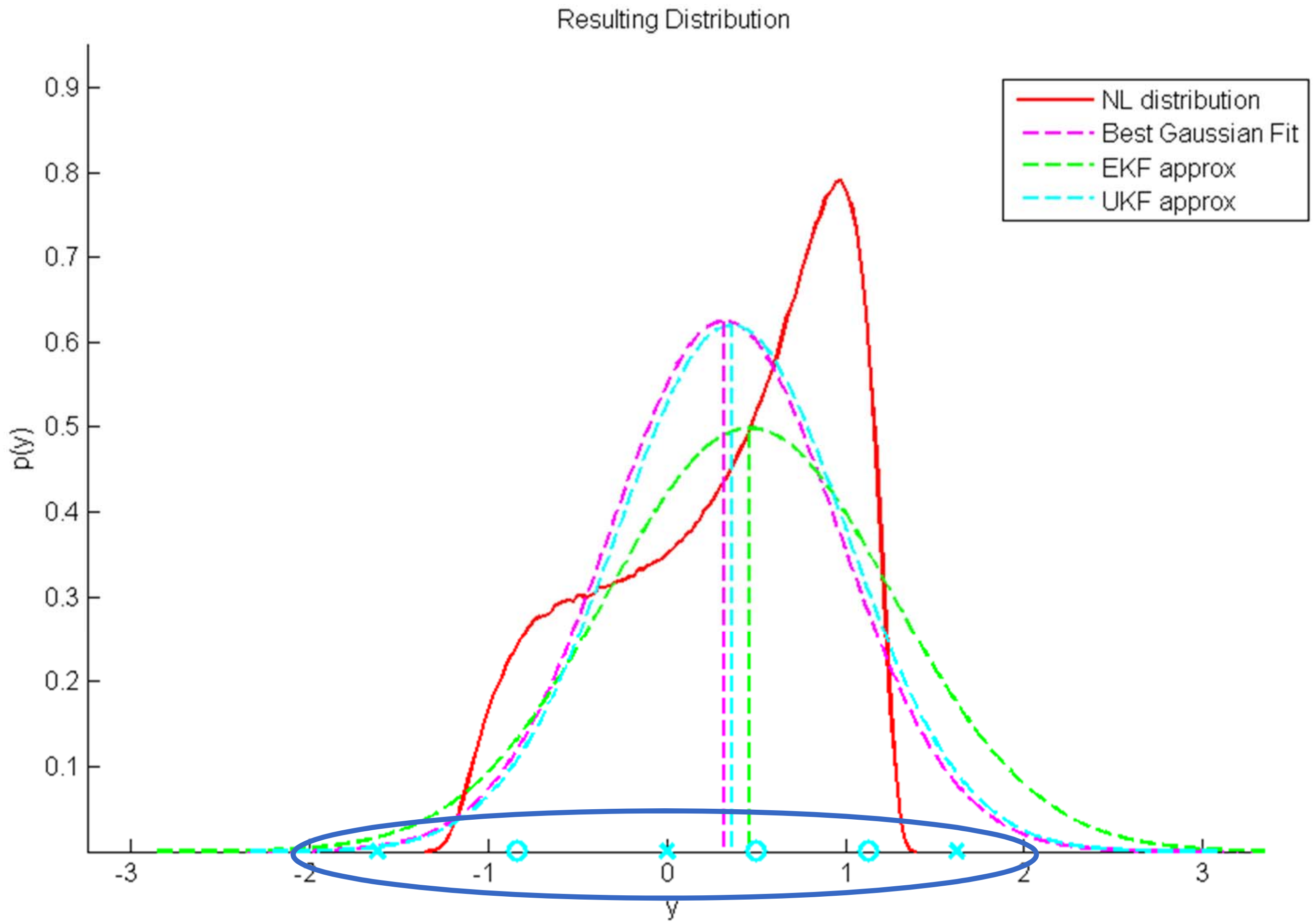
UNSCENTED TRANSFORM

- Select $\alpha = 1.45, \kappa = 0, \beta = 0$



UNSCENTED TRANSFORM

- Select $\alpha = 1.63, \kappa = 0, \beta = 2$



Unscented Kalman Filter

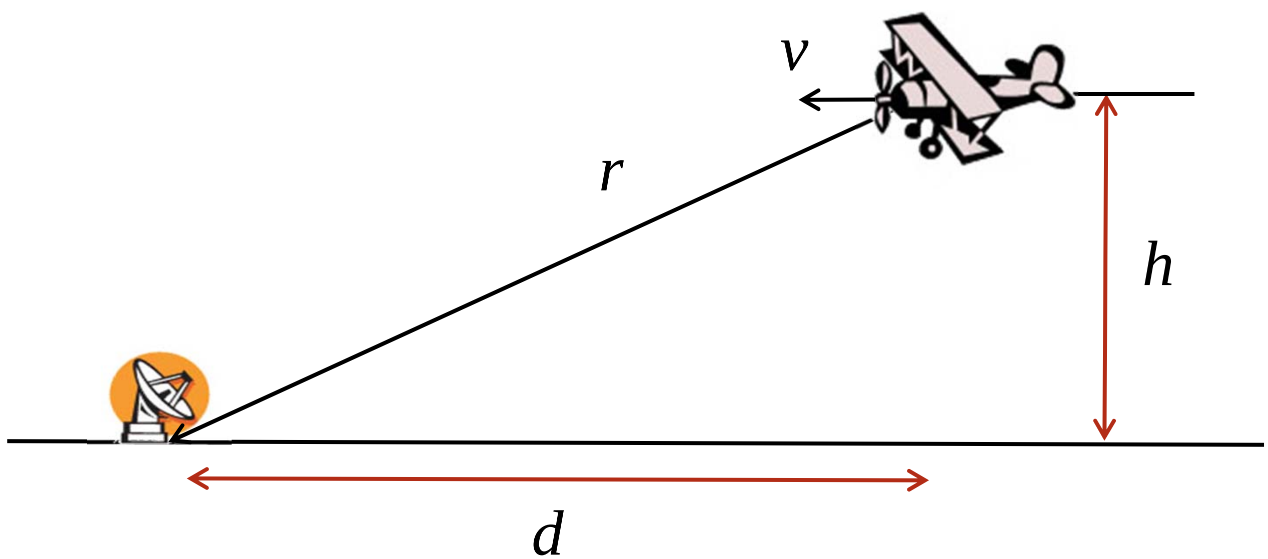
- Incorporating this method of distribution transformation into the Bayesian framework is possible

  - There are two nonlinear functions to deal with
    - Two unscented transforms are needed per timestep

  - The measurement model depends on the state we are trying to estimate
    - The state is augmented by the measurement noise states and a joint probability density function is updated

Unscented Kalman Filter

- Example repeat
  - Radar measurement of an airplane position while flying at constant altitude and velocity



Unscented Kalman Filter

- Example
  - State

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} d \\ v \\ h \end{bmatrix}$$

Initial

$$x_0 = \begin{bmatrix} 20 \\ -2 \\ 3 \end{bmatrix}$$

  - Motion Model

$$x_{1,t} = x_{1,t-1} + x_{2,t-1}dt$$

$$x_{2,t} = x_{2,t-1}$$

$$x_{3,t} = x_{3,t-1}$$

  - Measurement Model

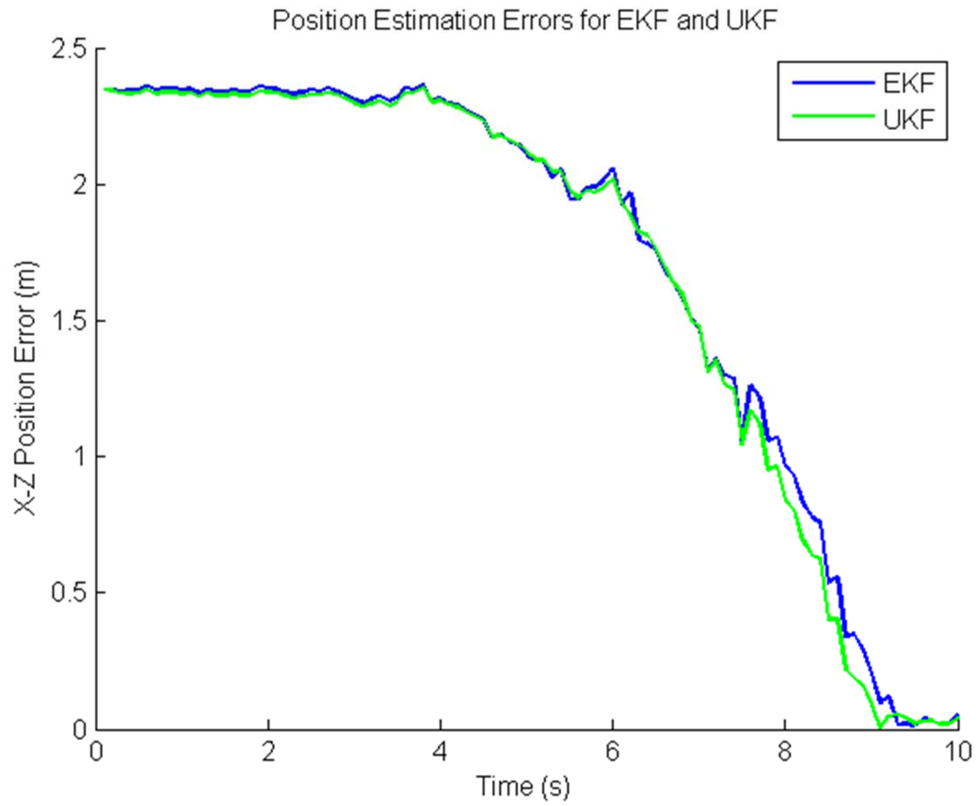
$$y_t = \sqrt{x_{1,t}^2 + x_{3,t}^2} + \delta_t$$

Unscented Kalman Filter

- Simulation results- low disturbances, noise

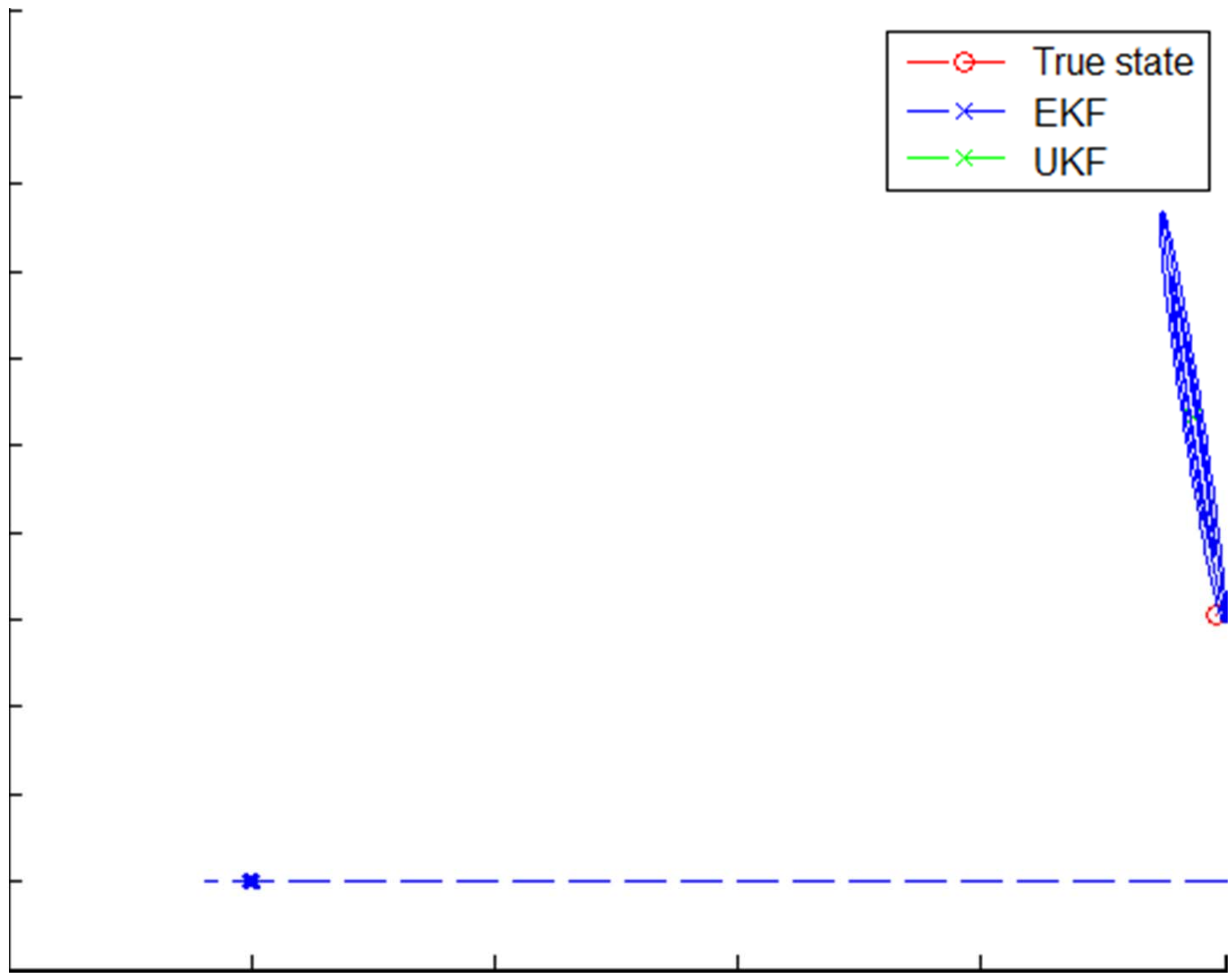
UNSCENTED KALMAN FILTER

- Error plot for position error



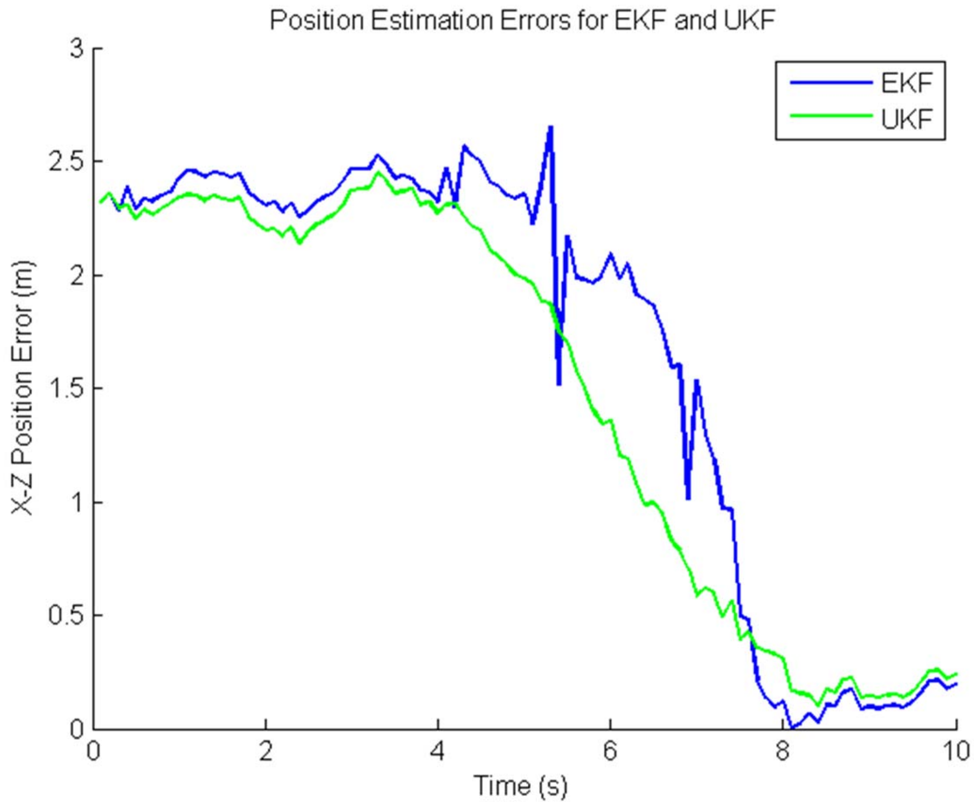
UNSCENTED KALMAN FILTER

- Simulation results, higher disturbances



UNSCENTED KALMAN FILTER

- Error plot for position error



UNSCENTED KALMAN FILTER

- Unscented Kalman Filter Modeling Assumptions
  - Prior over the state is Gaussian

$$p(x_0) \sim N(\mu_0, \Sigma_0)$$

  - Motion model, nonlinear but still with additive Gaussian disturbances

$$x_t = g(x_{t-1}, u_t) + \varepsilon_t \qquad \varepsilon_t \sim N(0, R_t)$$

  - Measurement model also nonlinear with additive Gaussian noise

$$y_t = h(x_t) + \delta_t \qquad \delta_t \sim N(0, Q_t)$$

  - Nonlinearity destroys certainty that beliefs remain Gaussian

Unscented Kalman Filter

- Prediction step
  - Propagation of belief at *t-1* through motion model
    - Pick sigma points

$$\chi_{t-1}^{[0]} = \mu_{t-1}$$

$$\chi_{t-1}^{[i]} = \mu_{t-1} + \left(\sqrt{(n+\lambda)\Sigma_{t-1}}\right)_i, \; i = 1,\ldots,n$$

$$\chi_{t-1}^{[n+i]} = \mu_{t-1} - \left(\sqrt{(n+\lambda)\Sigma_{t-1}}\right)_i, \; i = 1,\ldots,n$$

    - Propagate through motion model

$$\bar{\chi}_t^{[i]} = g\left(\chi_{t-1}^{[i]}, u_t\right)$$

Unscented Kalman Filter

- Prediction step
  - Unscented prediction step
    - Calculate mean and covariance, adding motion covariance to result

$$\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\chi}_t^{[i]}$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\chi}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\chi}_t^{[i]} - \bar{\mu}_t\right)^T + R_t$$

Unscented Kalman Filter

- Measurement step
  - Recall from Bayes filter, we are trying to define

$$bel(x_t) = \eta p(y_t \mid x_t)\overline{bel}(x_t)$$

  - We have the mean and covariance of predicted belief
  - We need to propagate this belief through another unscented transform
    - To do this, we need to look at the joint unscented transform

Unscented Kalman Filter

- Joint transform
  - with additive noise in model $\delta \sim N(0, Q)$

$$\begin{bmatrix} x \\ y = h(x) + \delta \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{xy} & \Sigma_y \end{bmatrix}\right)$$

  - Mean and covariance is found as before
    - Generate sigma points
    - Propagate through model
    - Find mean as before and covariance, cross-covariance as

$$\Sigma_{\Upsilon} = \sum_{i=0}^{2n} w_c^{[i]} \left(\Upsilon^{[i]} - \mu_{\Upsilon}\right)\left(\Upsilon^{[i]} - \mu_{\Upsilon}\right)^T + Q$$

$$\Sigma_{\chi\Upsilon} = \sum_{i=0}^{2n} w_c^{[i]} \left(\chi^{[i]} - \mu_{\chi}\right)\left(\Upsilon^{[i]} - \mu_{\Upsilon}\right)^T$$

UNSCENTED KALMAN FILTER

- Magic trick (Schur's complement)
  - If

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim N\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{xy} & \Sigma_y \end{bmatrix} \right)$$

  - Then

$$p(x \mid y) = N\left(\mu_x + \Sigma_y \Sigma_{xy}{}^{-1} \left(y - \mu_y \right), \Sigma_x - \Sigma_y \Sigma_{xy}{}^{-1} \Sigma_y{}^T \right)$$

  - Can in fact derive KF updates using this as well

UNSCENTED KALMAN FILTER

- Measurement Step
  - Form the joint distribution of x_t, y_t given all inputs and all but the latest measurement

$$p\left(x_t, y_t \mid y_{1:t-1}, u_{1:t}\right) = N\left(\begin{bmatrix}\mu_{x_t} \\ \mu_{y_t}\end{bmatrix}, \begin{bmatrix}\Sigma_{x_t} & \Sigma_{x_t y_t} \\ \Sigma_{x_t y_t} & \Sigma_{y_t}\end{bmatrix}\right)$$

  - Solve for all components and apply Schur's complement
  - But some of these we know already

$$p\left(x_t \mid y_{1:t-1}, u_{1:t}\right) = \overline{bel_t} \quad \Rightarrow \quad \begin{aligned} \mu_{x_t} &= \bar{\mu}_t \\ \Sigma_{x_t} &= \bar{\Sigma}_t \end{aligned}$$

UNSCENTED KALMAN FILTER

- Measurement step
  - The rest we can approximate with the unscented transform
  - Generate new sigma points from the predicted belief

$$\bar{\chi}_t^{[0]} = \bar{\mu}_t$$

$$\bar{\chi}_t^{[i]} = \bar{\mu}_t + \left(\sqrt{(n+\lambda)\bar{\Sigma}_t}\right)_i, \; i = 1,\ldots,n$$

$$\bar{\chi}_t^{[n+i]} = \bar{\mu}_t - \left(\sqrt{(n+\lambda)\bar{\Sigma}_t}\right)_i, \; i = 1,\ldots,n$$

- Propagate through measurement model

$$\Upsilon_t^{[i]} = h\left(\bar{\chi}_t^{[i]}\right)$$

Unscented Kalman Filter

- Measurement step
  - Then the measurement terms and cross terms can be approximated as

$$\mu_{y_t} \approx \sum_{i=0}^{2n} w_m^{[i]} \Upsilon_t^{[i]}$$

$$\Sigma_{y_t} \approx \sum_{i=0}^{2n} w_c^{[i]} \left(\Upsilon^{[i]} - \mu_{y_t}\right)\left(\Upsilon^{[i]} - \mu_{y_t}\right)^T + Q_t$$

$$\Sigma_{x_t y_t} \approx \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\chi}_t^{[i]} - \mu_{x_t}\right)\left(\Upsilon_t^{[i]} - \mu_{y_t}\right)^T$$

UNSCENTED KALMAN FILTER

- Measurement Step
  - Finally, applying Schur's complement

$$p(x \mid y) = N\left(\mu_x + \Sigma_y \Sigma_{xy}{}^{-1}\left(y - \mu_y\right), \Sigma_x - \Sigma_y \Sigma_{xy}{}^{-1} \Sigma_y{}^T\right)$$

  - To the above joint distribution

$$p\left(x_t \mid y_{1:t}, u_{1:t}\right) = bel(x_t) = N\left(\mu_t, \Sigma_t\right)$$

  - And therefore,

$$\mu_t = \bar{\mu}_t + \Sigma_{y_t} \Sigma_{x_t y_t}{}^{-1}\left(y_t - \mu_{y_t}\right)$$

$$\Sigma_t = \bar{\Sigma}_t - \Sigma_{y_t} \Sigma_{x_t y_t}{}^{-1} \Sigma_{y_t}{}^T$$

UNSCENTED KALMAN FILTER

- Summary
  - Prediction Step

$$\chi_{t-1}^{[0]} = \mu_{t-1}$$

$$\chi_{t-1}^{[i]} = \mu_{t-1} + \left(\sqrt{(n+\lambda)\Sigma_{t-1}}\right)_i, \; i = 1,\ldots,n$$

$$\chi_{t-1}^{[n+i]} = \mu_{t-1} - \left(\sqrt{(n+\lambda)\Sigma_{t-1}}\right)_i, \; i = 1,\ldots,n$$

Select sigma points

$$\bar{\chi}_t^{[i]} = g\left(\chi_{t-1}^{[i]}, u_t\right)$$

Apply motion model

$$\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\chi}_t^{[i]}$$

$$\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\chi}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\chi}_t^{[i]} - \bar{\mu}_t\right)^T + R_t$$

Extract mean and covariance

Unscented Kalman Filter

- Summary
  - Measurement step

$$\bar{\chi}_t^{[0]} = \bar{\mu}_t$$

$$\bar{\chi}_t^{[i]} = \bar{\mu}_t + \left(\sqrt{(n+\lambda)\bar{\Sigma}_t}\right)_i, \; i = 1,\ldots,n$$

$$\bar{\chi}_t^{[n+i]} = \bar{\mu}_t - \left(\sqrt{(n+\lambda)\bar{\Sigma}_t}\right)_i, \; i = 1,\ldots,n$$

Select sigma points

$$\Upsilon_t^{[i]} = h\left(\bar{\chi}_t^{[i]}\right)$$

Apply measurement model

$$\mu_{y_t} \approx \sum_{i=0}^{2n} w_m^{[i]} \Upsilon_t^{[i]}$$

$$\Sigma_{y_t} \approx \sum_{i=0}^{2n} w_c^{[i]} \left(\Upsilon^{[i]} - \mu_{y_t}\right)\left(\Upsilon^{[i]} - \mu_{y_t}\right)^T + Q_t$$

Extract mean and covariance

UNSCENTED KALMAN FILTER

- Summary
  - Measurement Step

$$\Sigma_{x_t y_t} \approx \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\chi}_t^{[i]} - \mu_{x_t}\right)\left(\Upsilon_t^{[i]} - \mu_{y_t}\right)^T$$

Extract cross-covariance

$$\mu_t = \bar{\mu}_t + \Sigma_{y_t} \Sigma_{x_t y_t}^{-1} \left(y_t - \mu_{y_t}\right)$$

$$\Sigma_t = \bar{\Sigma}_t - \Sigma_{y_t} \Sigma_{x_t y_t}^{-1} \Sigma_{y_t}^{T}$$

Update belief using Schur's complement

Unscented Kalman Filter

- Summary
  - Similar computation time to EKF
    - longer due to square root and inverse
  - Potentially capable of reducing errors in propagation of beliefs through nonlinear functions
  - Tuning effects unclear, can lead to strange results
  - Benefit minimal when nonlinearities are modest, or uncertainty is low