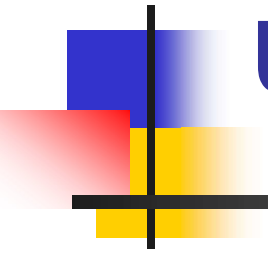
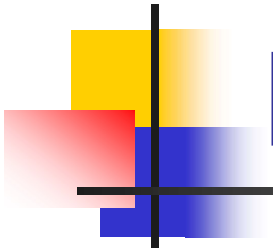


Mobile Robot Localization and Mapping using the Kalman Filter

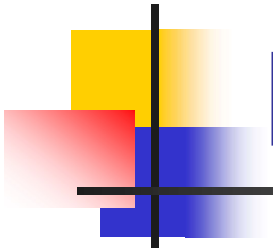
15-491 : CMRoboBits: Creating an Intelligent AIBO Robot

Paul E. Rybski



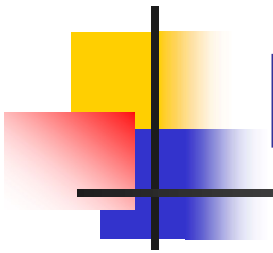
Mobile Robot Localization

- *Where am I?*
- Given a map, determine the robot's location
  - Landmark locations are known, but the robot's position is not
  - From sensor readings, the robot must be able to infer its most likely position on the field
  - Example : where are the AIBOs on the soccer field?



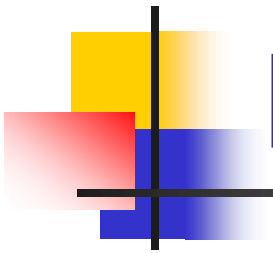
Mobile Robot Mapping

- *What does the world look like?*
- Robot is unaware of its environment
- The robot must explore the world and determine its structure
  - Most often, this is combined with localization
  - Robot must update its location wrt the landmarks
  - Known in the literature as Simultaneous Localization and Mapping, or Concurrent Localization and Mapping : SLAM (CLM)
  - Example : AIBOs are placed in an unknown environment and must learn the locations of the landmarks *(An interesting project idea?)*



Bayesian Filter

- Why should you care?
  - Robot and environmental state estimation is a fundamental problem!
- Nearly all algorithms that exist for spatial reasoning make use of this approach
  - If you're working in mobile robotics, you'll see it over and over!
  - Very important to understand and appreciate
- Efficient state estimator
  - Recursively compute the robot's current state based on the previous state of the robot
- *What is the robot's state?*



Bayesian Filter

- Estimate state *x* from data *d*
  - *What is the probability of the robot being at x?*
- *x* could be robot location, map information, locations of targets, etc…
- *d* could be sensor readings such as range, actions, odometry from encoders, etc…)
- This is a general formalism that does not depend on the particular probability representation
- Bayes filter **recursively** computes the posterior distribution: $Bel(x_T) = P(x_T \mid Z_T)$

What is a Posterior Distribution?

Derivation of the Bayesian Filter *(slightly different notation from before)*

Estimation of the robot's state given the data:

$$Bel(x_t) = p(x_t \mid Z_T)$$

The robot's data, Z, is expanded into two types: observations o_i and actions a_i

$$Bel(x_t) = p(x_t \mid o_t, a_{t-1}, o_{t-1}, a_{t-2}, \ldots, o_0)$$

Invoking the Bayesian theorem

$$Bel(x_t) = \frac{p(o_t \mid x_t, a_{t-1}, \ldots, o_0)\, p(x_t \mid a_{t-1}, \ldots, o_0)}{p(o_t \mid a_{t-1}, \ldots, o_0)}$$

Derivation of the Bayesian Filter

Denominator is constant relative to x_t

$$\eta = p(o_t \mid a_{t-1},\ldots,a_0)$$

$$Bel(x_t) = \eta p(o_t \mid x_t, a_{t-1},\ldots,o_0)\, p(x_t \mid a_{t-1},\ldots,o_0)$$

First-order Markov assumption shortens first term:

$$Bel(x_t) = \eta p(o_t \mid x_t)\, p(x_t \mid a_{t-1},\ldots,o_0)$$

Expanding the last term (theorem of total probability):

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1},\ldots,o_0) p(x_{t-1} \mid a_{t-1},\ldots,o_0) dx_{t-1}$$

Derivation of the Bayesian Filter

First-order Markov assumption shortens middle term:

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) p(x_{t-1} \mid a_{t-1}, \ldots, o_0) dx_{t-1}$$

Finally, substituting the definition of $Bel(x_{t-1})$:

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

The above is the probability distribution that must be estimated from the robot's data

Iterating the Bayesian Filter

- Propagate the motion model:

$$Bel\_(x_t) = \int P(x_t \mid a_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Compute the current state estimate before taking a sensor reading by integrating over all possible previous state estimates and applying the motion model

- Update the sensor model:

$$Bel(x_t) = \eta P(o_t \mid x_t) Bel\_(x_t)$$

Compute the current state estimate by taking a sensor reading and multiplying by the current estimate based on the most recent motion history

Localization

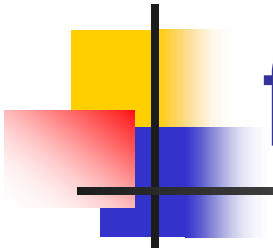
Initial state
detects nothing:

Moves and
detects landmark:

Moves and
detects nothing:

Moves and
detects landmark:

Bayesian Filter : Requirements for Implementation

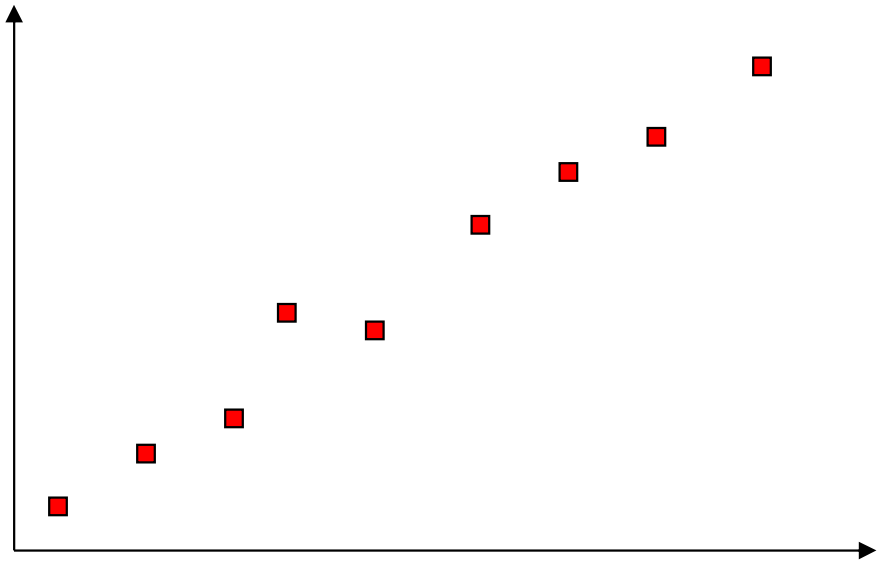


- Representation for the belief function
- Update equations
- Motion model
- Sensor model
- Initial belief state

Representation of the Belief Function

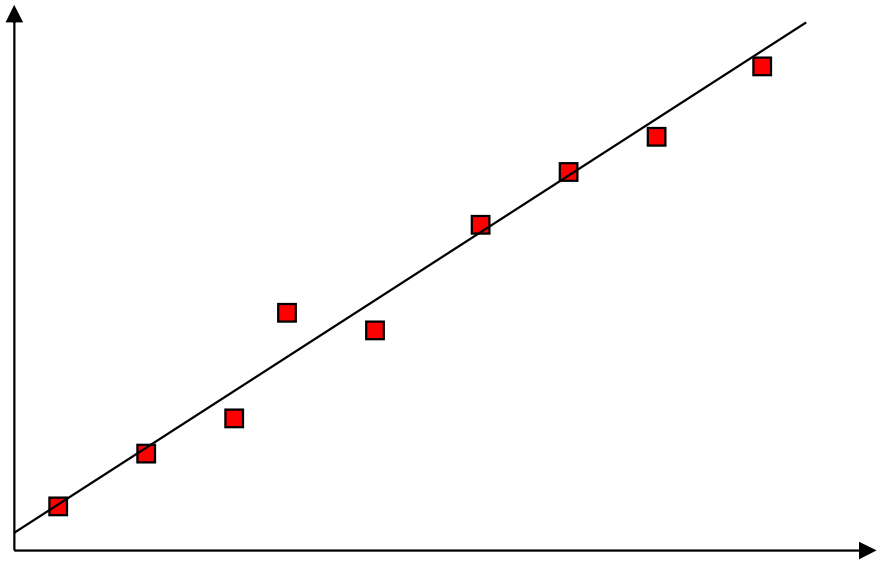
Sample-based representations

e.g. Particle filters



$(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots (x_n, y_n)$

Parametric representations



$y = mx + b$

Example of a Parameterized Bayesian Filter : Kalman Filter

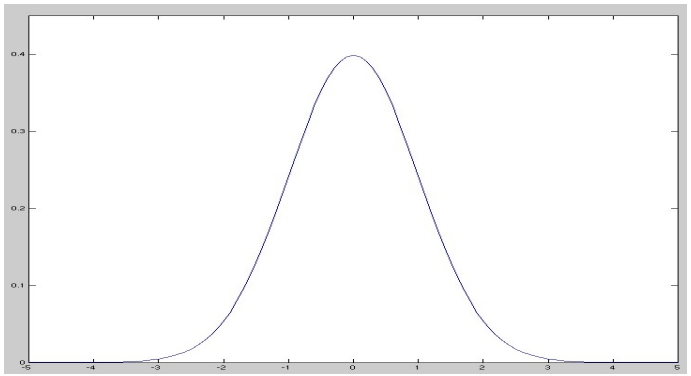
Kalman filters (KF) represent posterior belief by a Gaussian (normal) distribution

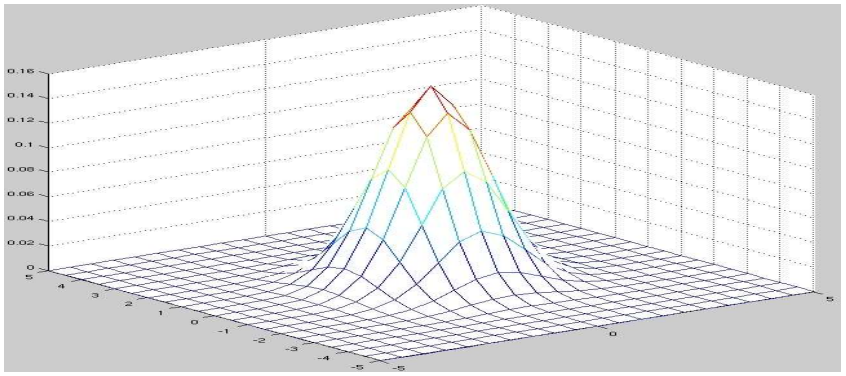
A 1-d Gaussian distribution is given by:

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

An n-d Gaussian distribution is given by:

$$P(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$





Kalman Filter : a Bayesian Filter

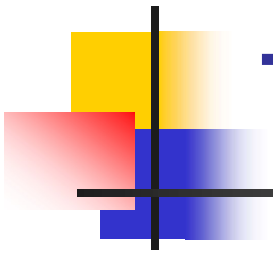
- Initial belief *Bel(x_0)* is a Gaussian distribution
  - *What do we do for an unknown starting position?*
- State at time *t+1* is a linear function of state at time *t:*

$$x_{t+1} = Fx_t + Bu_t + \varepsilon_{t(action)}$$

- Observations are linear in the state:

$$o_t = Hx_t + \varepsilon_{t(observation)}$$

- Error terms are zero-mean random variables which are normally distributed
- These assumptions guarantee that the posterior belief is Gaussian
  - The Kalman Filter is an efficient algorithm to compute the posterior
  - Normally, an update of this nature would require a matrix inversion (similar to a least squares estimator)
  - The Kalman Filter avoids this computationally complex operation



The Kalman Filter

- Motion model is Gaussian…
- Sensor model is Gaussian…
- Each belief function is uniquely characterized by its mean μ and covariance matrix Σ
- Computing the posterior means computing a new mean μ and covariance Σ from old data using actions and sensor readings
- *What are the key limitations?*

  1) Unimodal distribution
  2) Linear assumptions

What we know…
What we don't know…

- We know what the control inputs of our process are
  - We know what we've told the system to do and have a model for what the expected output should be if everything works right
- We don't know what the noise in the system truly is
  - We can only estimate what the noise might be and try to put some sort of upper bound on it
- When estimating the state of a system, we try to find a set of values that comes as close to the truth as possible
  - There will always be some mismatch between our estimate of the system and the true state of the system itself. We just try to figure out how much mismatch there is and try to get the best estimate possible

Minimum Mean Square Error

Reminder: *the expected value, or mean value, of a Continuous random variable* **x** *is defined as:*

$$E[x] = \int_{-\infty}^{\infty} xp(x)dx$$

Minimum Mean Square Error

What is the mean of this distribution? $P(x \mid Z)$

This is difficult to obtain exactly. With our approximations, we can get the estimate $\hat{x}$

…such that $E[(x-\hat{x})^2 \mid Z_t]$ is minimized.

According to the **Fundamental Theorem of Estimation Theory** this estimate is:

$$\hat{x}^{MMSE} = E[x \mid Z] = \int_{-\infty}^{\infty} xp(x \mid Z)dx$$

Fundamental Theorem of Estimation Theory

- The minimum mean square error estimator equals the expected (mean) value of x conditioned on the observations Z
- The minimum mean square error term is quadratic:

$$E[(x-\hat{x})^2 \mid Z_t]$$

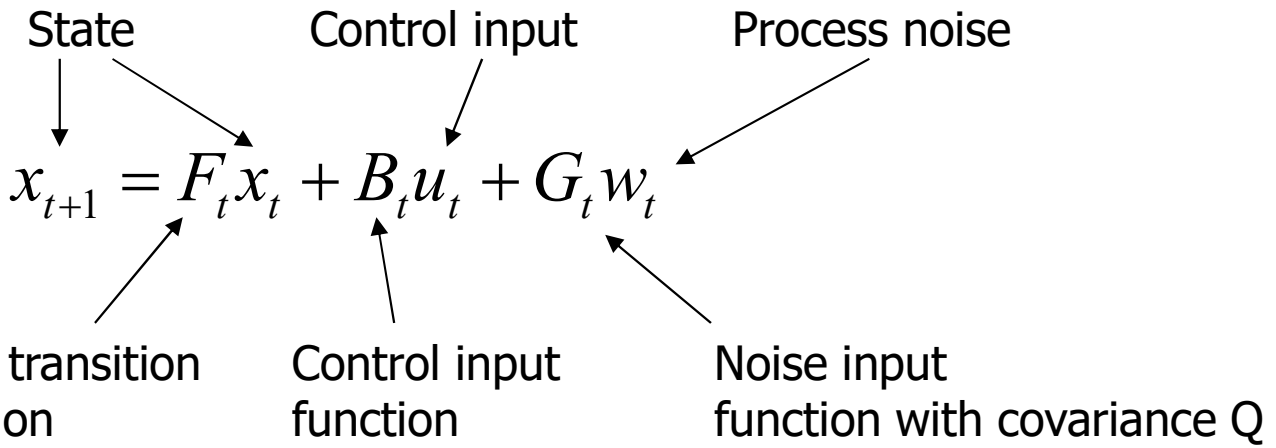
  - Its minimum can be found by taking the derivative of the function w.r.t x and setting that value to 0.

$$\nabla_x (E[(x-\hat{x})^2 \mid Z]) = 0$$

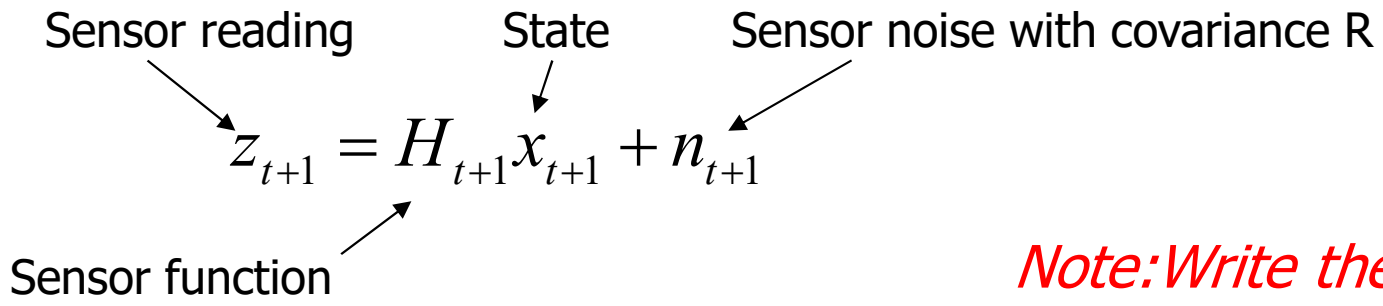
- It is interesting to note that when they use the Gaussian assumption, Maximum Posteriori estimators and MMSE estimators find the same value for the parameters.
  - This is because mean and the mode of a Gaussian distribution are the same.

Kalman Filter Components
(also known as: Way Too Many Variables…)

Linear discrete time dynamic system (motion model)



Measurement equation (sensor model)



Note:Write these down!!!

Computing the MMSE Estimate of the State and Covariance

Given a set of measurements: $Z_{t+1} = \{z_i, i \le t+1\}$

According to the Fundamental Theorem of Estimation, the state and covariance will be:

$$\hat{x}^{MMSE} = E[x \mid Z_{t+1}]$$

$$P^{MMSE} = E[(x - \hat{x})^2 \mid Z_{t+1}]$$

We will now use the following notation:

$$\hat{x}_{t+1|t+1} = E[x_{t+1} \mid Z_{t+1}]$$

$$\hat{x}_{t|t} = E[x_t \mid Z_t]$$

$$\hat{x}_{t+1|t} = E[x_{t+1} \mid Z_t]$$

Computing the MMSE Estimate of the State and Covariance

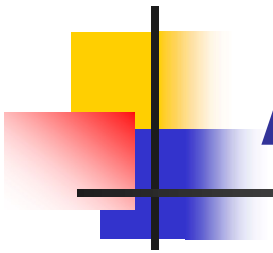
What is the **minimum mean square error** estimate of the system state and covariance?

$$\hat{x}_{t+1|t} = F_t \hat{x}_{t|t} + B_t u_t$$ Estimate of the state variables

$$\hat{z}_{t+1|t} = H_{t+1} \hat{x}_{t+1|t}$$ Estimate of the sensor reading

$$P_{t+1|t} = F_t P_{t|t} F_t^T + G_t Q_t G_t^T$$ Covariance matrix for the state

$$S_{t+1|t} = H_{t+1} P_{t+1|t} H_{t+1}^T + R_{t+1}$$ Covariance matrix for the sensors



At last! The Kalman Filter…

Propagation (motion model):

$$\hat{x}_{t+1/t} = F_t \hat{x}_{t/t} + B_t u_t$$

$$P_{t+1/t} = F_t P_{t/t} F_t^T + G_t Q_t G_t^T$$

Update (sensor model):

$$\hat{z}_{t+1} = H_{t+1} \hat{x}_{t+1/t}$$

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1}$$

$$S_{t+1} = H_{t+1} P_{t+1/t} H_{t+1}^T + R_{t+1}$$

$$K_{t+1} = P_{t+1/t} H_{t+1}^T S_{t+1}^{-1}$$

$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$$

$$P_{t+1/t+1} = P_{t+1/t} - P_{t+1/t} H_{t+1}^T S_{t+1}^{-1} H_{t+1} P_{t+1/t}$$

…but what does that mean in English?!?

Propagation (motion model):

$$\hat{x}_{t+1/t} = F_t \hat{x}_{t/t} + B_t u_t$$

- State estimate is updated from system dynamics

$$P_{t+1/t} = F_t P_{t/t} F_t^T + G_t Q_t G_t^T$$

- Uncertainty estimate *GROWS*

Update (sensor model):

$$\hat{z}_{t+1} = H_{t+1} \hat{x}_{t+1/t}$$

- Compute expected value of sensor reading

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1}$$

- Compute the difference between expected and "true"

$$S_{t+1} = H_{t+1} P_{t+1/t} H_{t+1}^T + R_{t+1}$$

- Compute covariance of sensor reading

$$K_{t+1} = P_{t+1/t} H_{t+1}^T S_{t+1}^{-1}$$

- Compute the Kalman Gain (how much to correct est.)

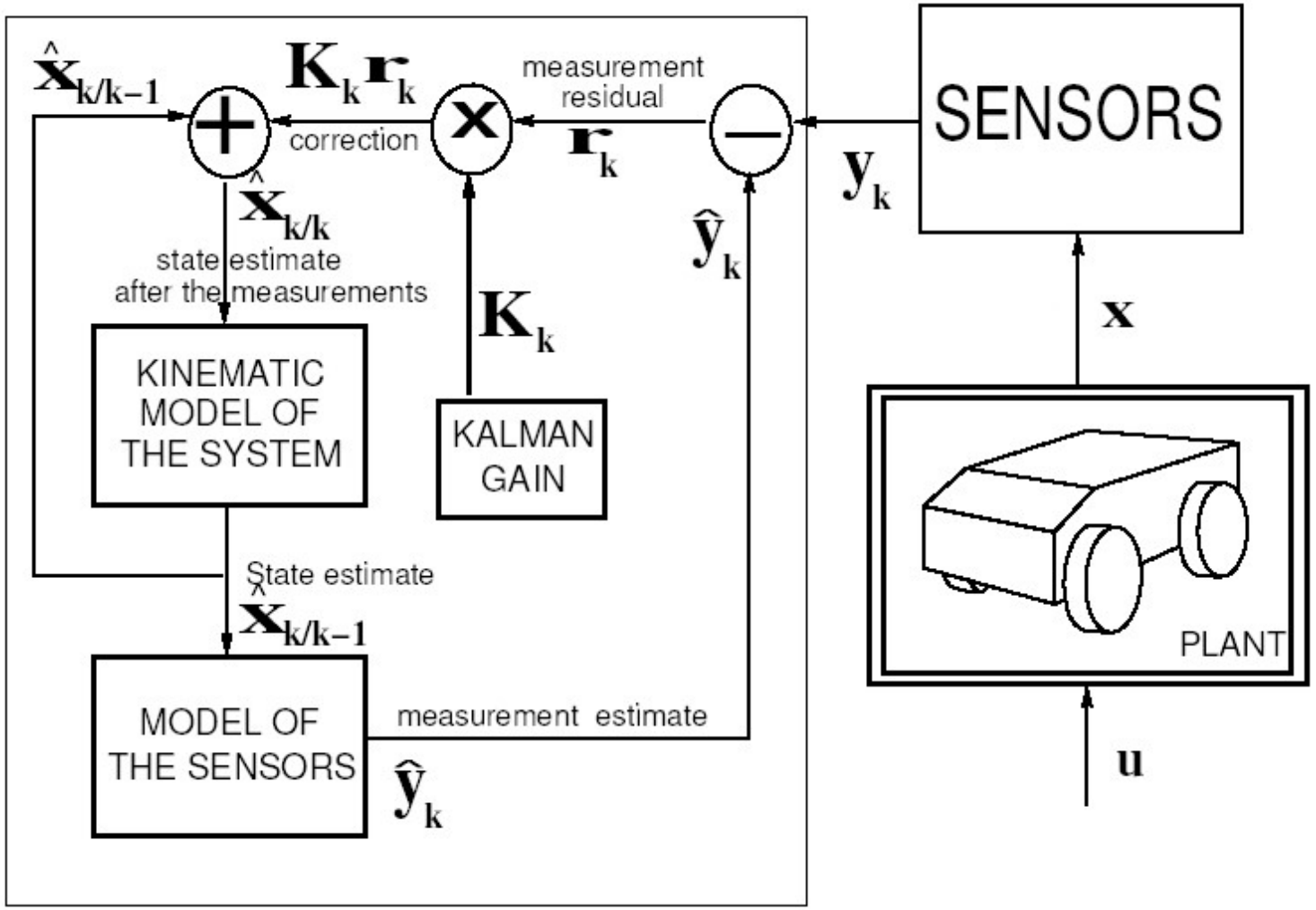
$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$$

- Multiply residual times gain to correct state estimate

$$P_{t+1/t+1} = P_{t+1/t} - P_{t+1/t} H_{t+1}^T S_{t+1}^{-1} H_{t+1} P_{t+1/t}$$

- Uncertainty estimate SHRINKS

Kalman Filter Block Diagram



Example 1: Simple 1D Linear System

Given: F=G=H=1, u=0
Initial state estimate = 0
Linear system:

$$x_{t+1} = x_t + w_t$$

$$z_{t+1} = x_{t+1} + n_{t+1}$$

Unknown noise parameters

Propagation:

$$\hat{x}_{t+1/t} = \hat{x}_{t/t}$$

$$P_{t+1/t} = P_{t/t} + Q_t$$

Update:

$$\hat{z}_{t+1} = \hat{x}_{t+1/t}$$

$$r_{t+1} = z_{t+1} - \hat{x}_{t+1/t}$$

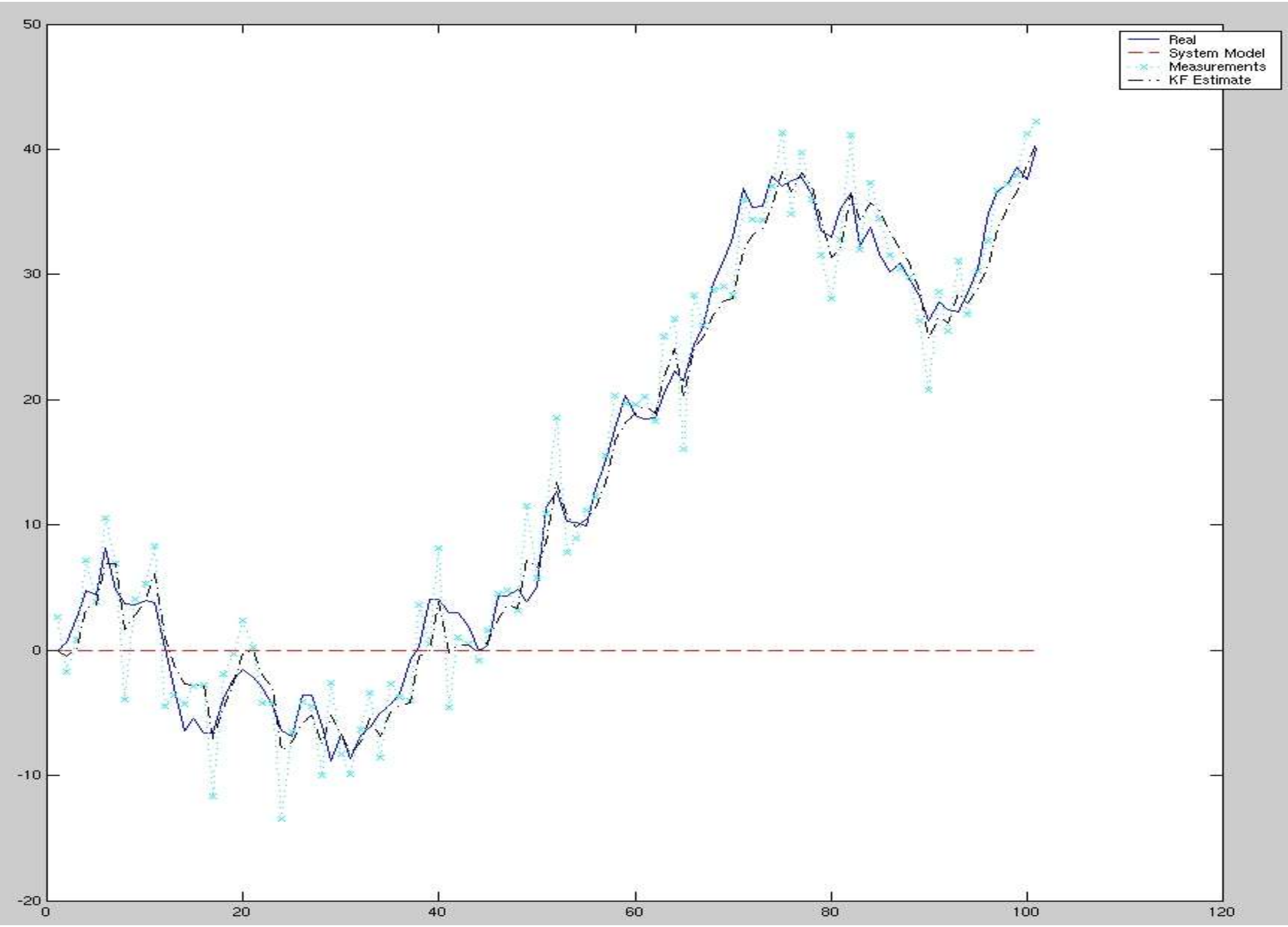
$$S_{t+1} = P_{t+1/t} + R_{t+1}$$

$$K_{t+1} = P_{t+1/t} S_{t+1}^{-1}$$

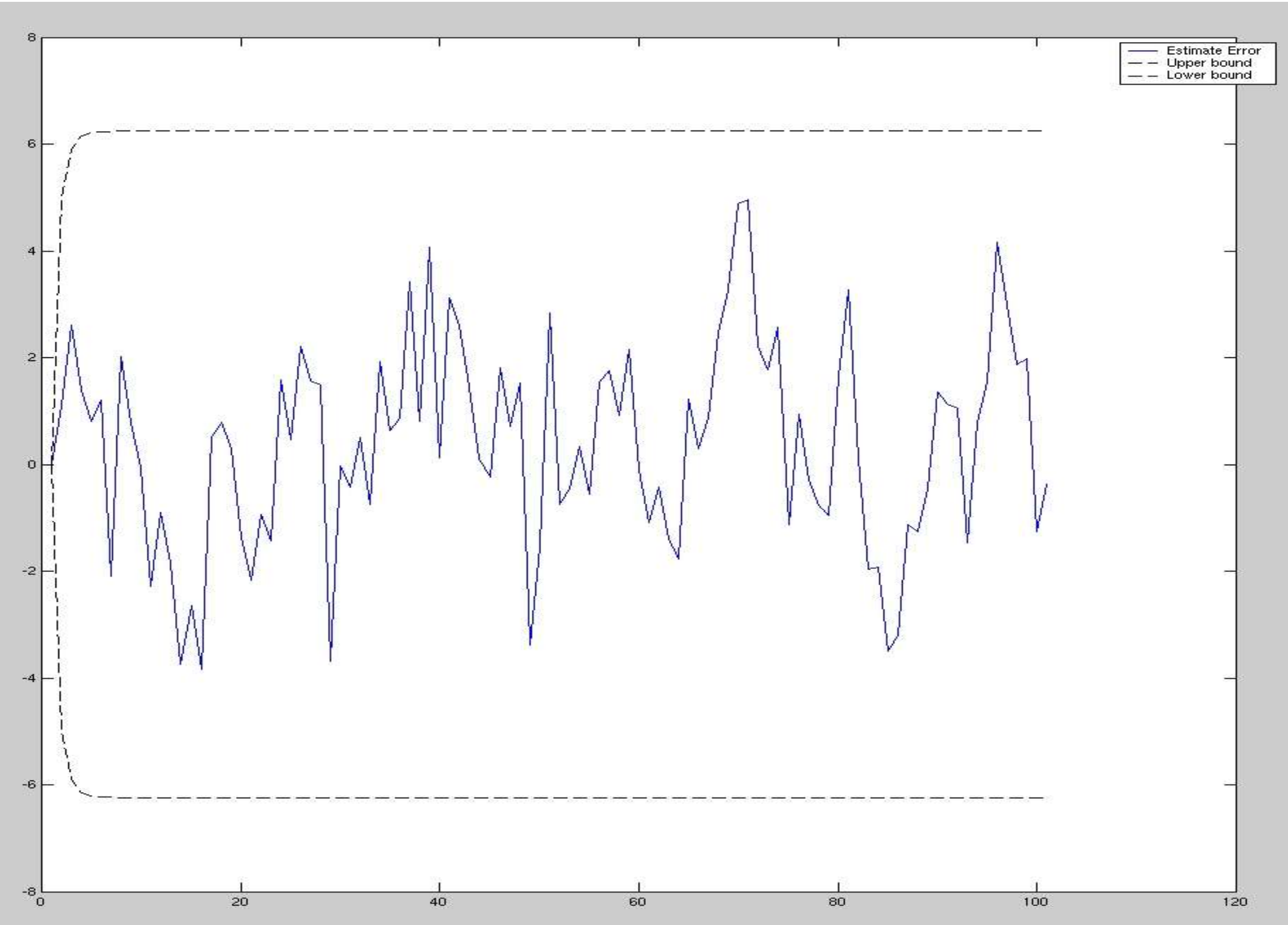
$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$$

$$P_{t+1/t+1} = P_{t+1} - P_{t+1/t} S_{t+1}^{-1} P_{t+1/t}$$

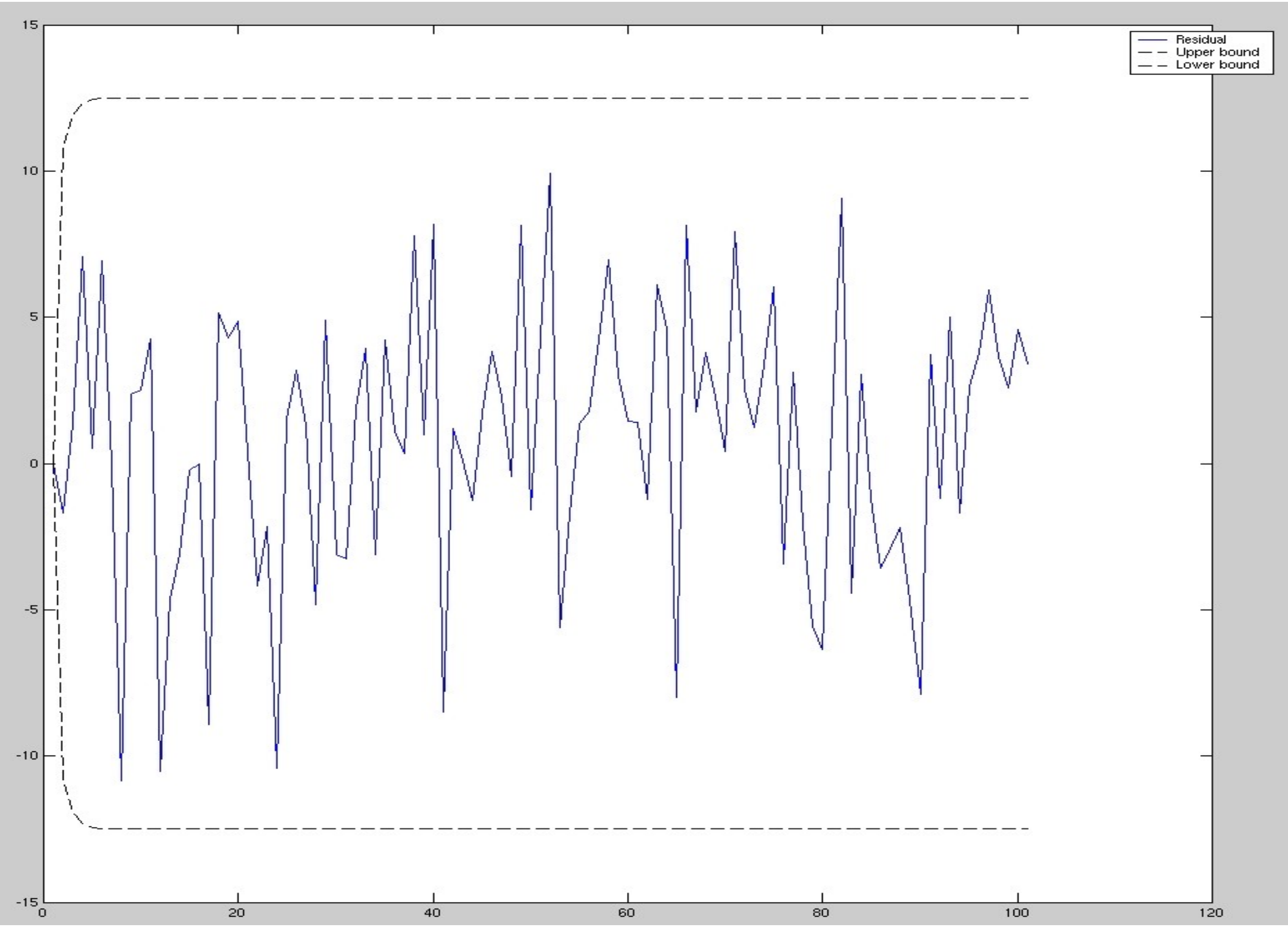
State Estimate



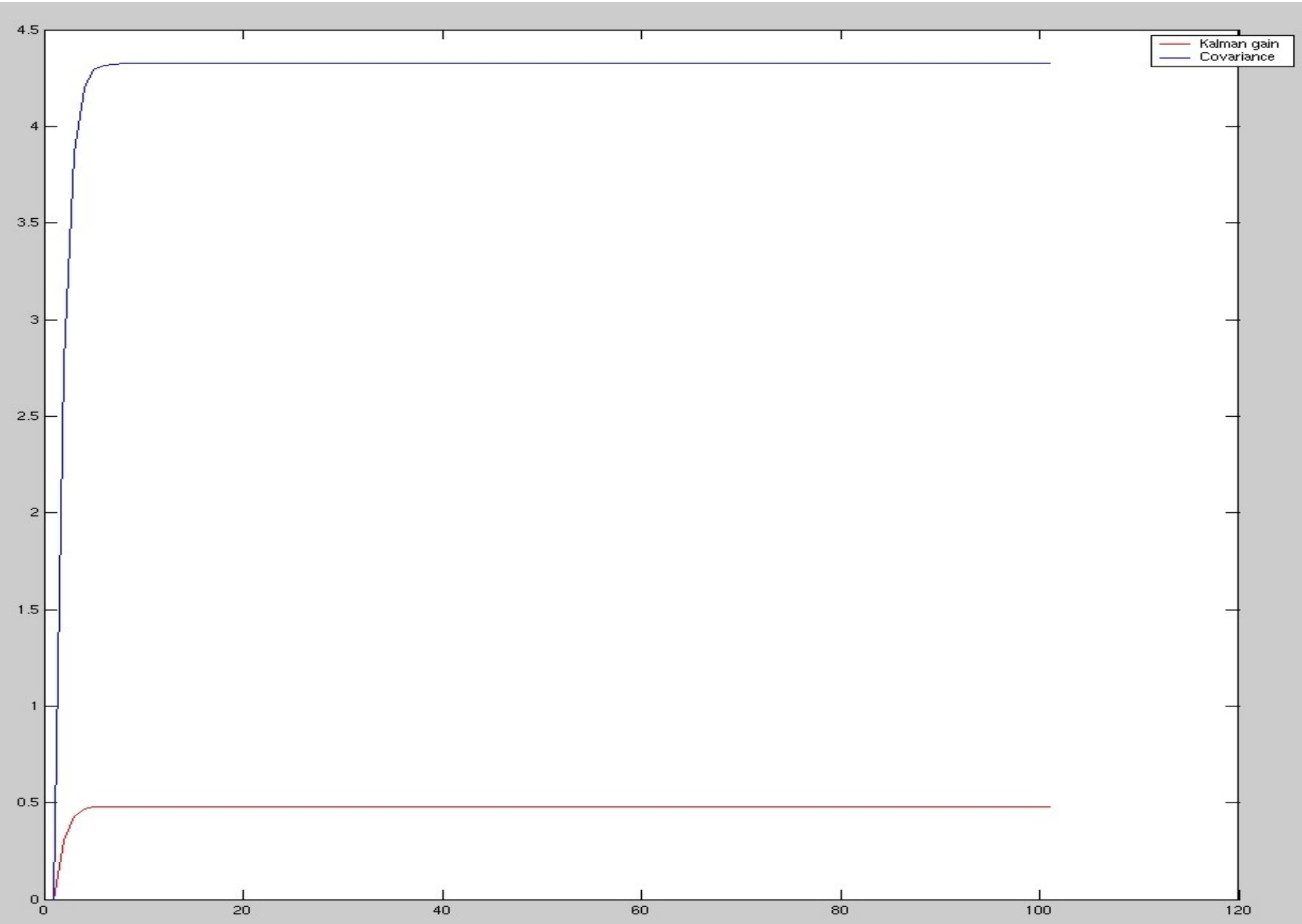
State Estimation Error vs 3σ Region of Confidence



Sensor Residual vs 3σ Region of Confidence



Kalman Gain and State Covariance



Example 2: Simple 1D Linear System with Erroneous Start

Given: F=G=H=1, u=cos(t/5)
Initial state estimate = 20

Linear system:

$$x_{t+1} = x_t + \cos(t/5) + w_t$$

$$z_{t+1} = x_{t+1} + n_{t+1}$$

Unknown noise parameters

Propagation:

$$\hat{x}_{t+1/t} = \hat{x}_{t/t} + \cos(t/5)$$

$$P_{t+1/t} = P_{t/t} + Q_t$$

Update: *(no change)*

$$\hat{z}_{t+1} = \hat{x}_{t+1/t}$$

$$r_{t+1} = z_{t+1} - \hat{x}_{t+1/t}$$

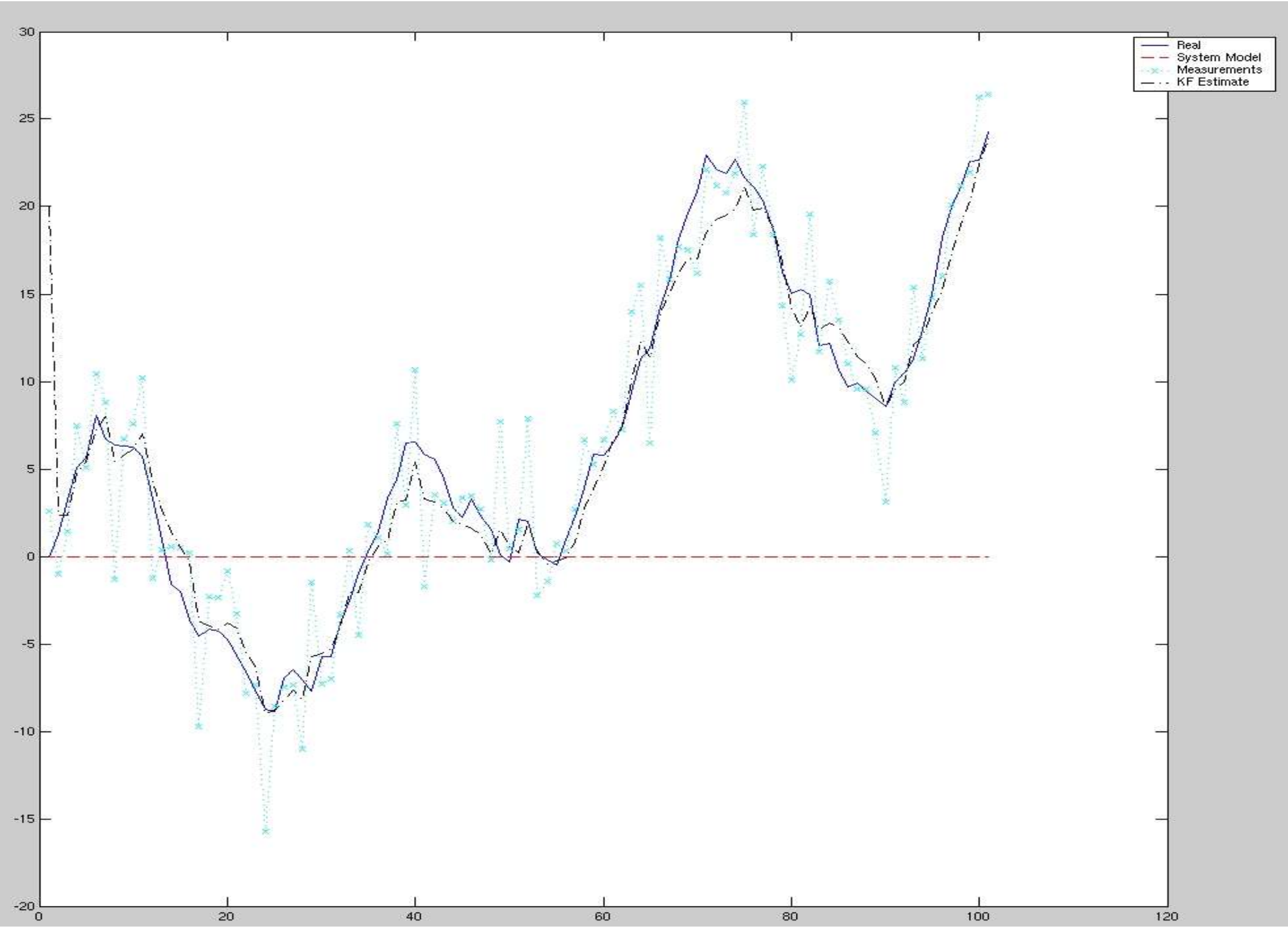
$$S_{t+1} = P_{t+1/t} + R_{t+1}$$

$$K_{t+1} = P_{t+1/t} S_{t+1}^{-1}$$

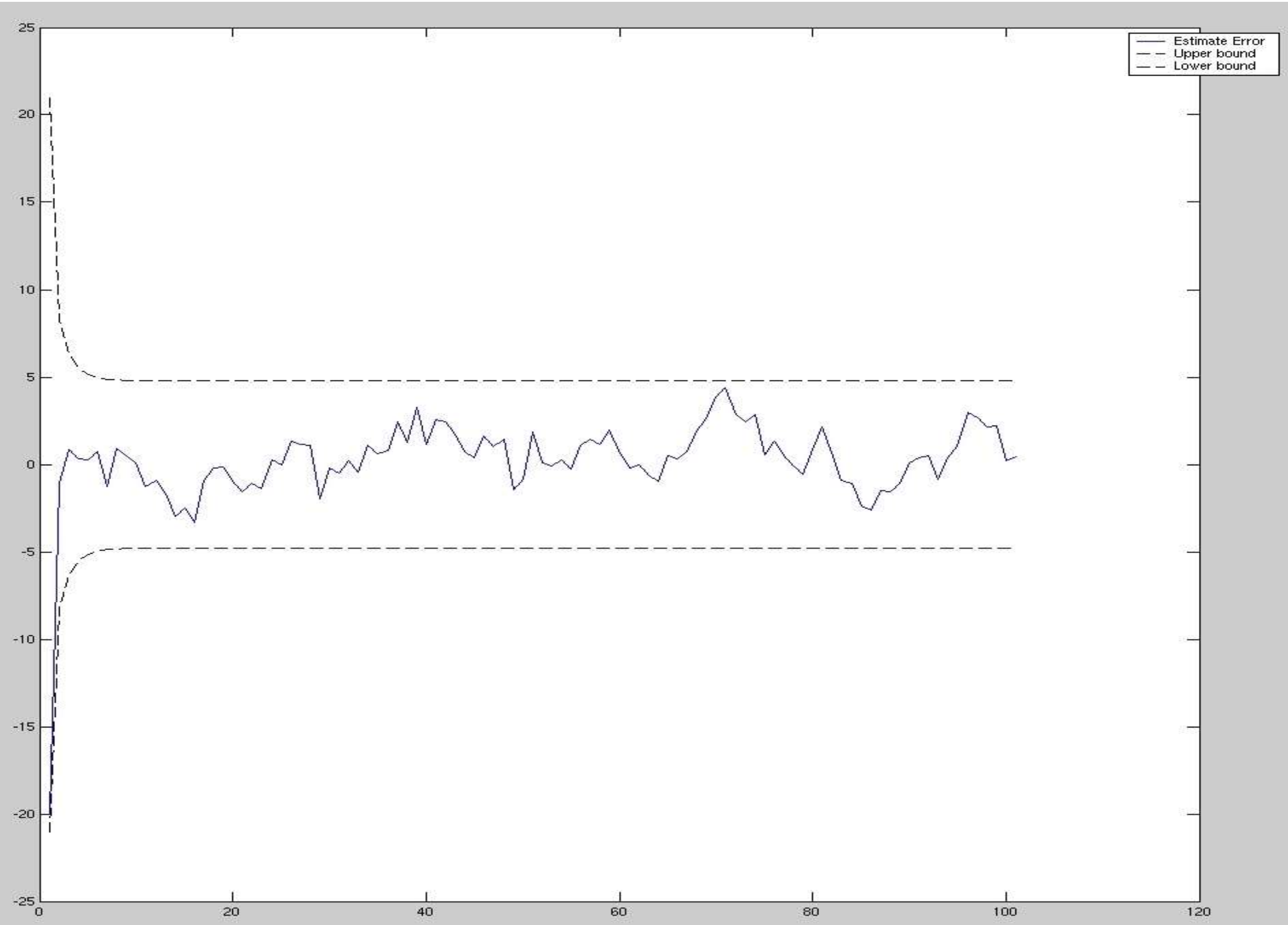
$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$$

$$P_{t+1/t+1} = P_{t+1} - P_{t+1/t} S_{t+1}^{-1} P_{t+1/t}$$

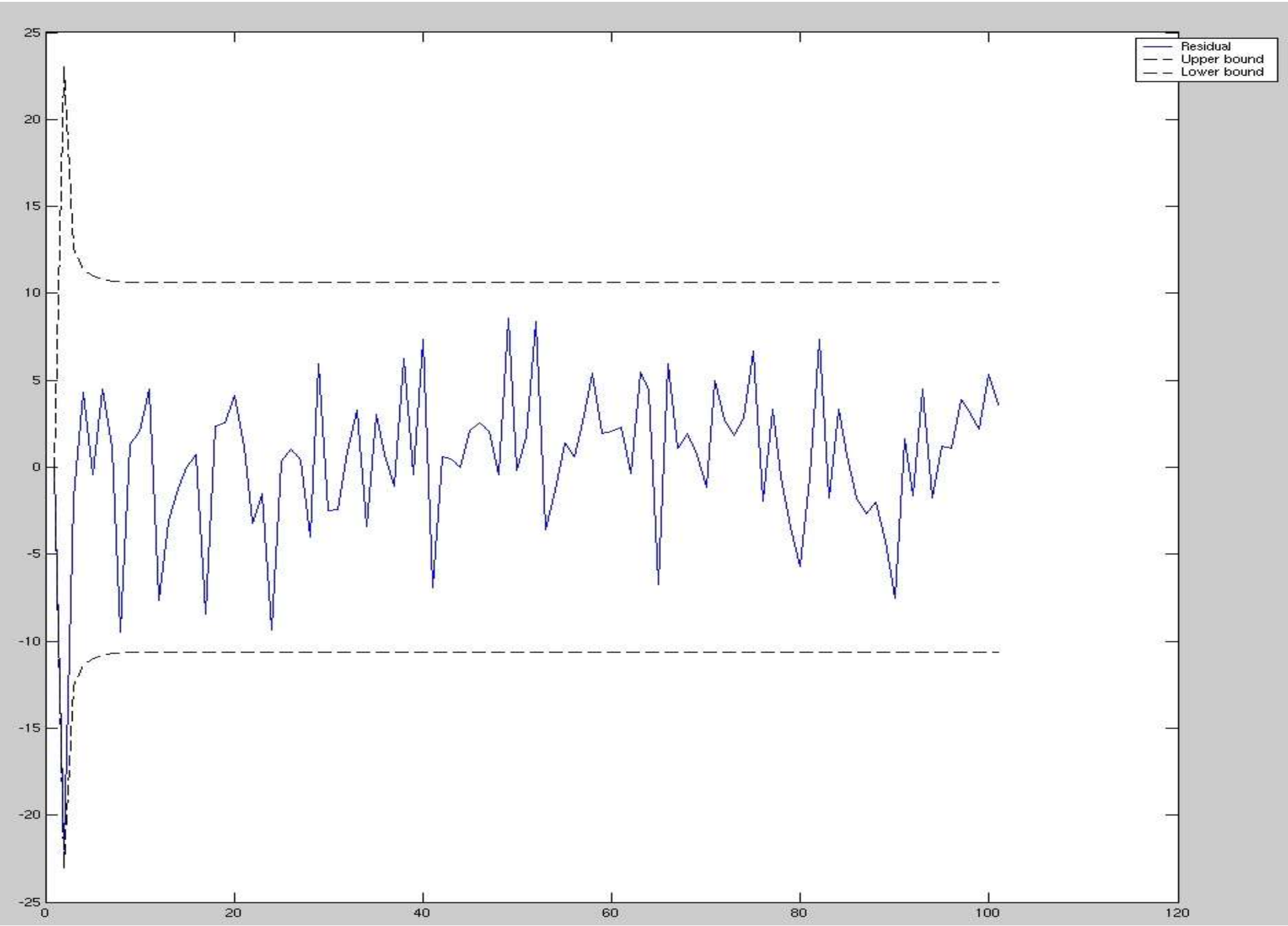
State Estimate



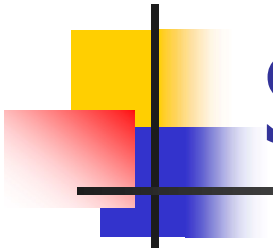
State Estimation Error vs 3σ Region of Confidence



Sensor Residual vs 3σ Region of Confidence



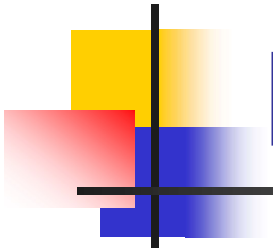
Kalman Gain and State Covariance



Some observations

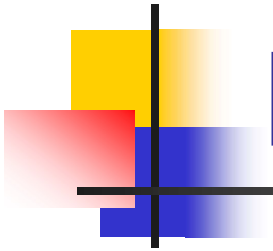
- The larger the error, the smaller the effect on the final state estimate
  - If *process* uncertainty is larger, *sensor* updates will dominate state estimate
  - If *sensor* uncertainty is larger, *process* propagation will dominate state estimate
- Improper estimates of the state and/or sensor covariance may result in a rapidly diverging estimator
  - As a rule of thumb, the residuals must always be bounded within a $\pm 3\sigma$ region of uncertainty
  - This measures the "health" of the filter
- *Many* propagation cycles can happen between updates

Using the Kalman Filter for Mobile Robots



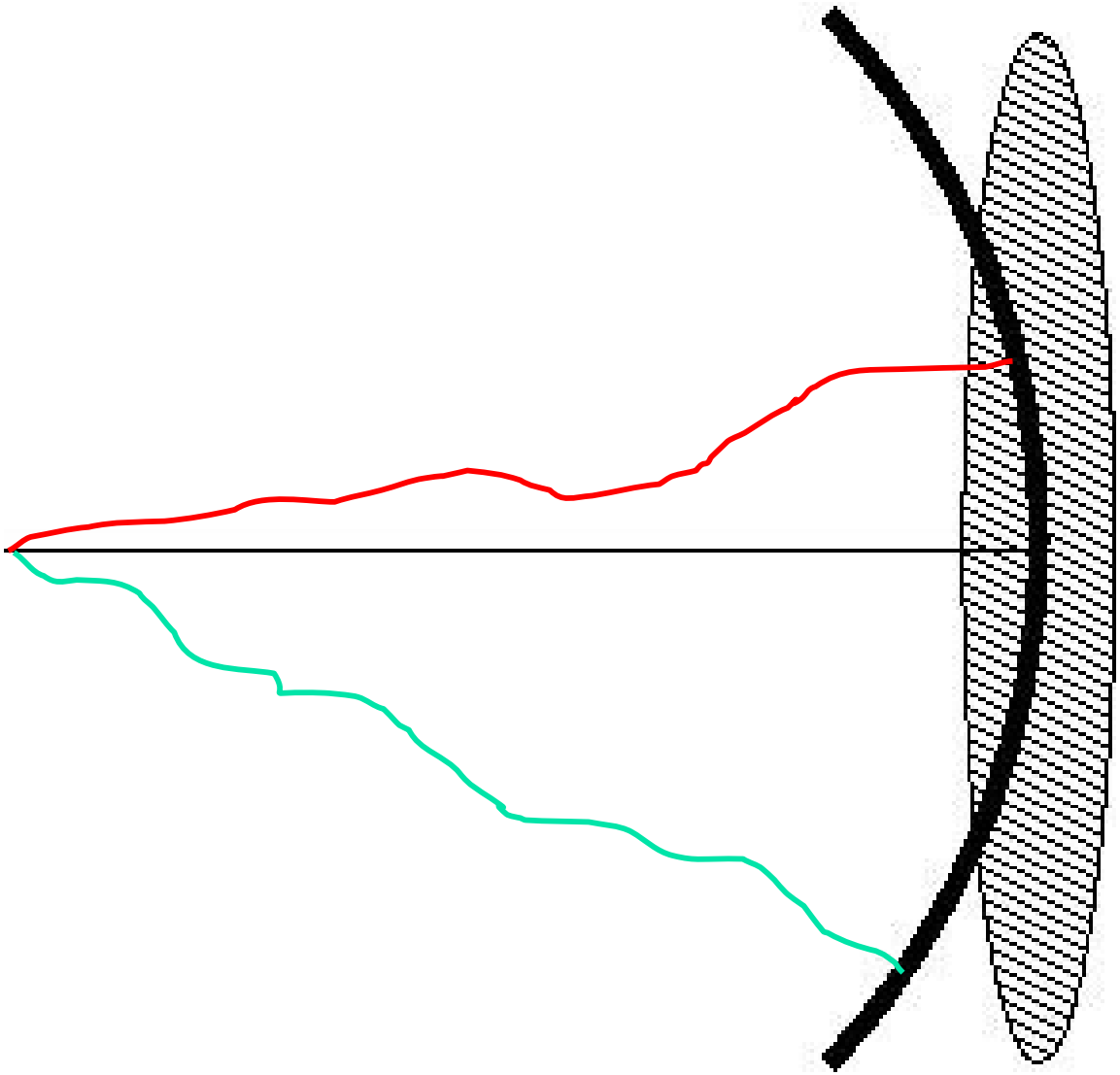
- Sensor modeling
  - The odometry estimate is not a reflection of the robot's control system is rather treated as a sensor
  - Instead of directly measuring the error in the state vector (such as when doing tracking), the error in the state must be estimated
  - This is referred to as the Indirect Kalman Filter
- State vector for robot moving in 2D
  - The state vector is 3x1: $[x,y,\theta]$
  - The covariance matrix is 3x3
- Problem: Mobile robot dynamics are NOT linear

Problems with the Linear Model Assumption

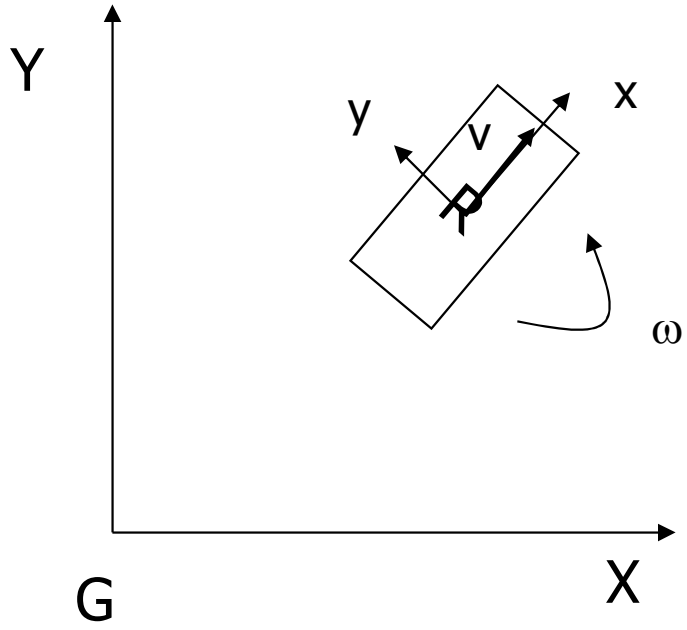


- Many systems of interest are highly non-linear, such as mobile robots
- In order to model such systems, a linear process model must be generated out of the non-linear system dynamics
- The Extended Kalman filter is a method by which the state propagation equations and the sensor models can be linearized about the current state estimate
- Linearization will increase the state error residual because it is not the best estimate

Approximating Robot Motion Uncertainty with a Gaussian



Linearized Motion Model for a Robot



From a robot-centric perspective, the velocities look like this:

$$\dot{x}_t = V_t$$
$$\dot{y}_t = 0$$
$$\dot{\phi}_t = \omega_t$$

From the global perspective, the velocities look like this:

$$\dot{x}_t = V_t \cos\phi_t$$
$$\dot{y}_t = V_t \sin\phi_t$$
$$\dot{\phi}_t = \omega_t$$

The discrete time state estimate (including noise) looks like this:

$$\hat{x}_{t+1} = \hat{x}_t + (V_t + w_{V_t})\delta t \cos\hat{\phi}_t$$
$$\hat{y}_{t+1} = \hat{y}_t + (V_t + w_{V_t})\delta t \sin\hat{\phi}_t$$
$$\hat{\phi}_{t+1} = \hat{\phi}_t + (\omega_t + w_{\omega_t})\delta t$$

Problem! We don't know linear and rotational velocity errors. The state estimate will rapidly diverge if this is the only source of information!

Linearized Motion Model for a Robot

Now, we have to compute the covariance matrix propagation equations.

The indirect Kalman filter derives the pose equations from the estimated error of the state:

$$x_{t+1} - \hat{x}_{t+1} = \tilde{x}_{t+1}$$
$$y_{t+1} - \hat{y}_{t+1} = \tilde{y}_{t+1}$$
$$\phi_{t+1} - \hat{\phi}_{t+1} = \tilde{\phi}_{t+1}$$

In order to linearize the system, the following small-angle assumptions are made:

$$\cos\tilde{\phi} \cong 1$$
$$\sin\tilde{\phi} \cong \tilde{\phi}$$

Linearized Motion Model for a Robot

From the error-state propagation equation, we can obtain the State propagation and noise input functions *F* and *G*:

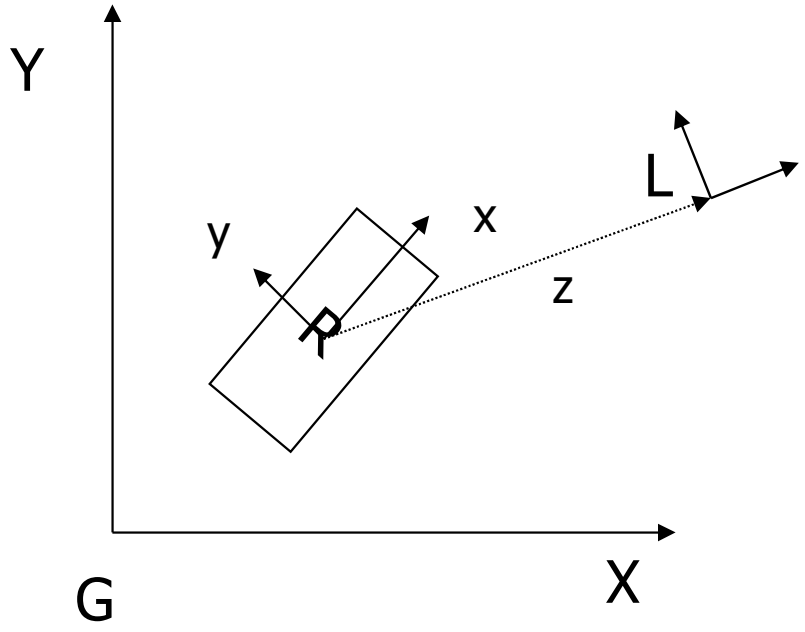
$$\begin{bmatrix} \tilde{x}_{t+1} \\ \tilde{y}_{t+1} \\ \tilde{\phi}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -V_m \delta t \sin\hat{\phi} \\ 0 & 1 & V_m \delta t \cos\hat{\phi} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_t \\ \tilde{y}_t \\ \tilde{\phi}_t \end{bmatrix} + \begin{bmatrix} -\delta t \cos\phi_R & 0 \\ -\delta t \sin\phi_R & 0 \\ 0 & -\delta t \end{bmatrix} \begin{bmatrix} w_{V_t} \\ w_{\omega_t} \end{bmatrix}$$

$$\tilde{X}_{t+1} = F_t \tilde{X}_t + G_t W_t$$

From these values, we can easily compute the standard covariance propagation equation:

$$P_{t+1/t} = F_t P_{t/t} F_t^T + G_t Q_t G_t^T$$

Sensor Model for a Robot with a Perfect Map



From the robot, the measurement looks like this:

$$z_{t+1} = \begin{bmatrix} x_{L_{t+1}} \\ y_{L_{t+1}} \\ \phi_{L_{t+1}} \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_\phi \end{bmatrix}$$

From a global perspective, the measurement looks like:

$$z_{t+1} = \begin{bmatrix} \cos\phi_{t+1} & -\sin\phi_{t+1} & 0 \\ \sin\phi_{t+1} & \cos\phi_{t+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{L_{t+1}} - x_{t+1} \\ y_{L_{t+1}} - y_{t+1} \\ \phi_{L_{t+1}} - \phi_{t+1} \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_\phi \end{bmatrix}$$

The measurement equation is nonlinear and must also be linearized!

Sensor Model for a Robot with a Perfect Map

Now, we have to compute the linearized sensor function. Once again, we make use of the indirect Kalman filter where the error in the reading must be estimated.

In order to linearize the system, the following small-angle assumptions are made:

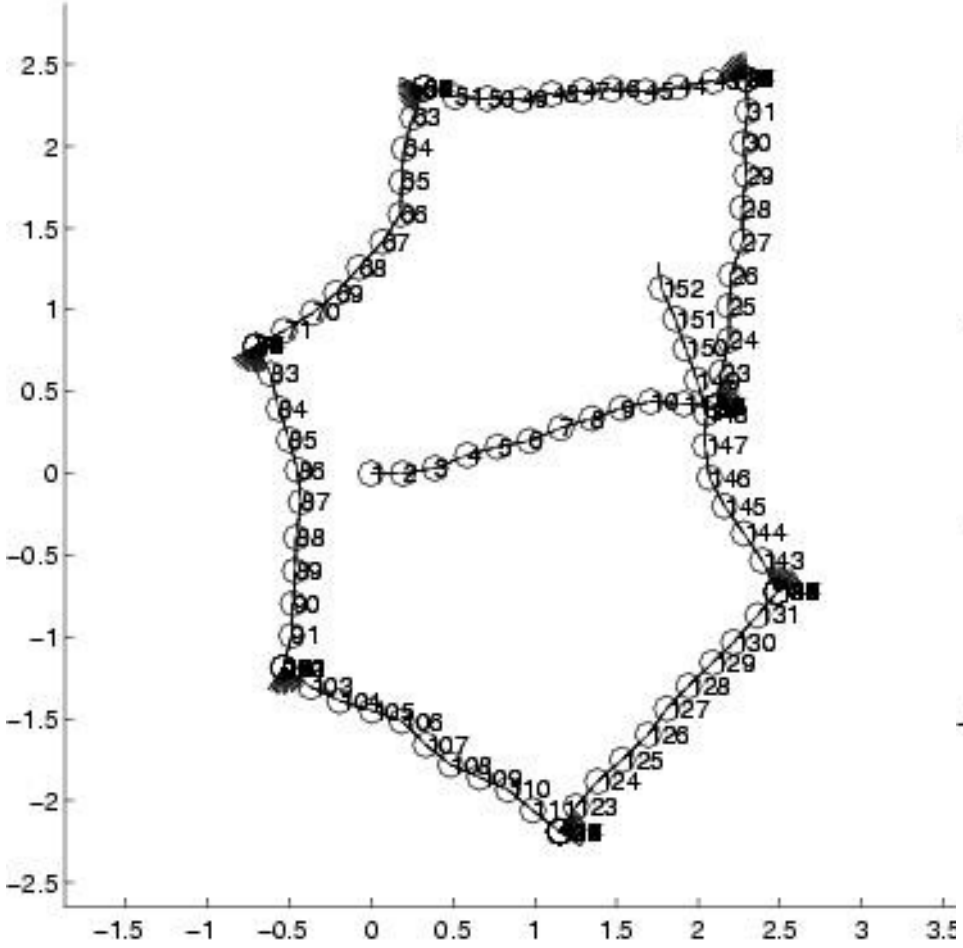
$$\cos\tilde{\phi} \cong 1$$

$$\sin\tilde{\phi} \cong \tilde{\phi}$$

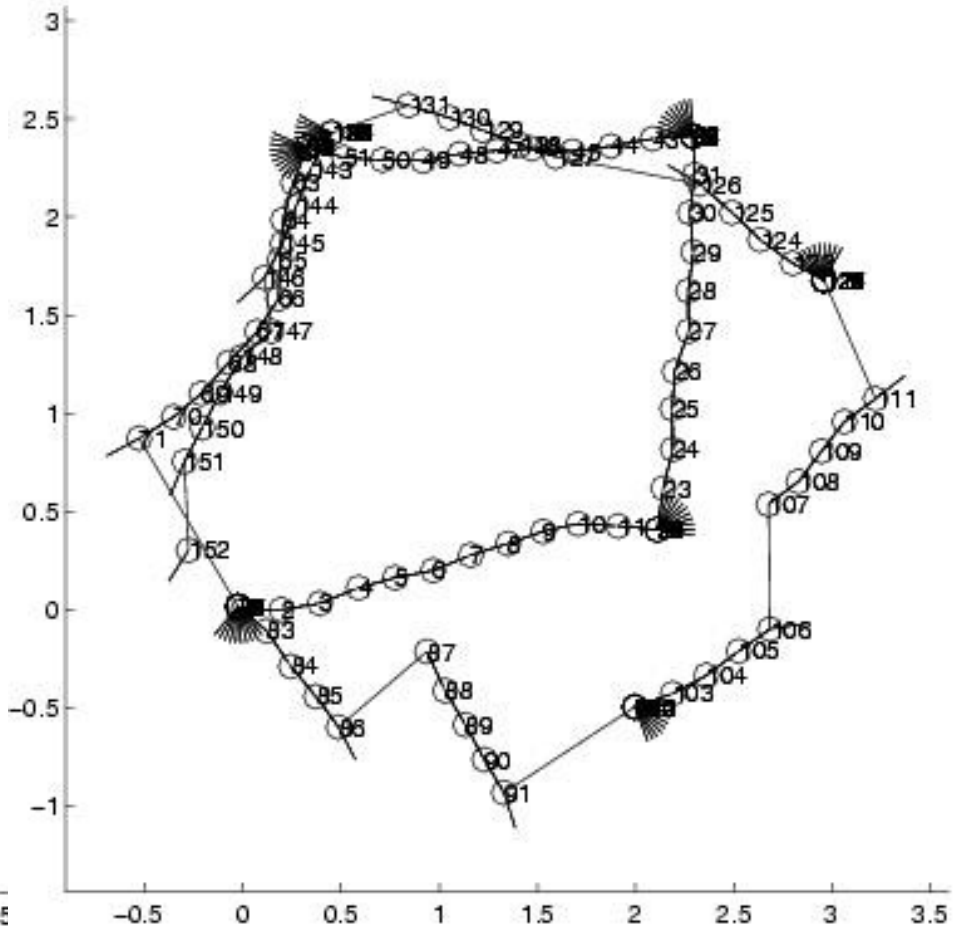
The final expression for the error in the sensor reading is:

$$\begin{bmatrix} \tilde{x}_{L_{t+1}} \\ \tilde{y}_{L_{t+1}} \\ \tilde{\phi}_{L_{t+1}} \end{bmatrix} = \begin{bmatrix} -\cos\hat{\phi}_{t+1} & -\sin\hat{\phi}_{t+1} & -\sin\hat{\phi}_{t+1}(x_L - \hat{x}_{t+1}) + \cos\hat{\phi}_t(y_L - \hat{y}_{t+1}) \\ \sin\hat{\phi}_{t+1} & -\cos\hat{\phi}_{t+1} & -\cos\hat{\phi}_{t+1}(x_L - \hat{x}_{t+1}) - \sin\hat{\phi}_t(y_L - \hat{y}_{t+1}) \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} \tilde{x}_{t+1} \\ \tilde{y}_{t+1} \\ \tilde{\phi}_{t+1} \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \\ n_\phi \end{bmatrix}$$

Updating the State Vector

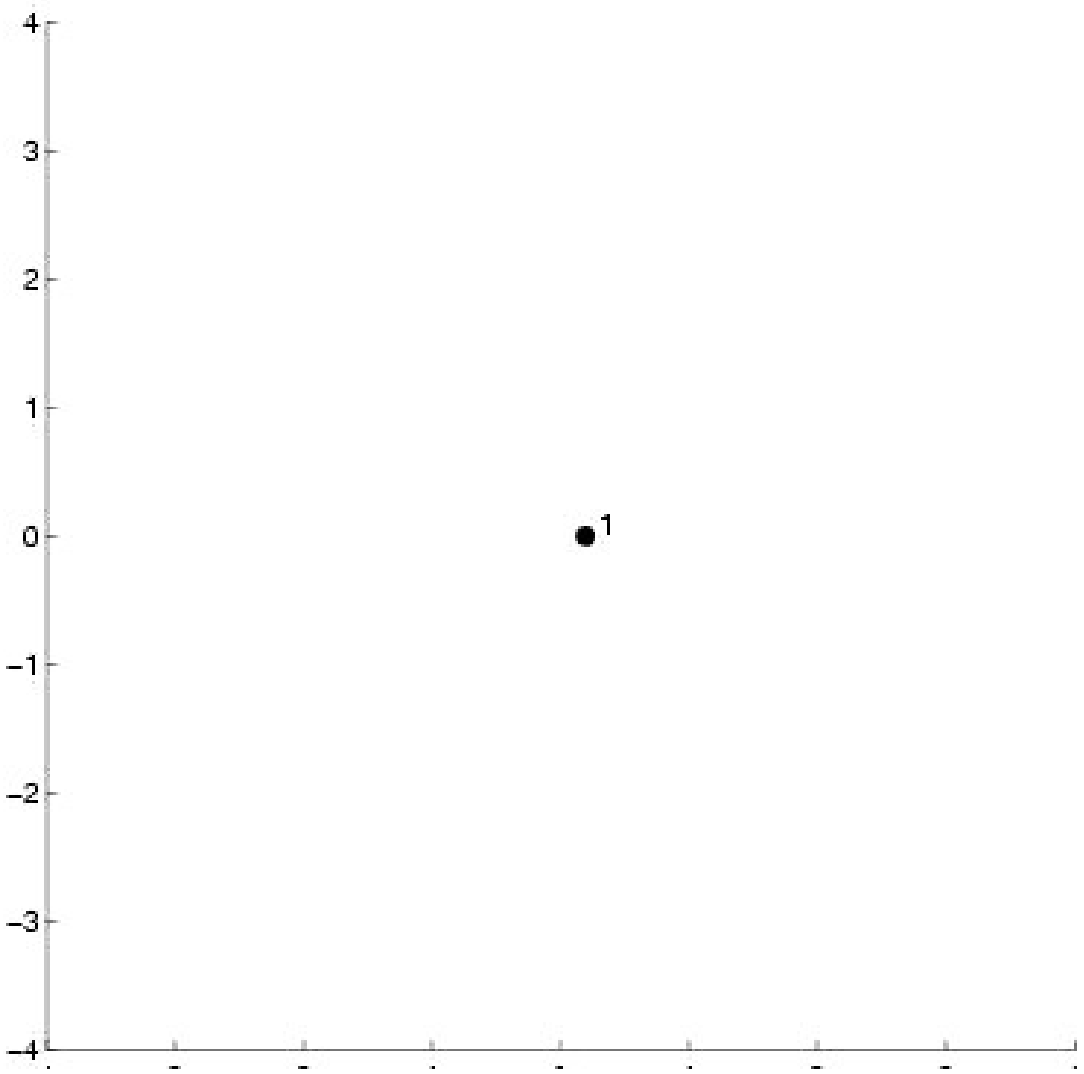


Propagation only

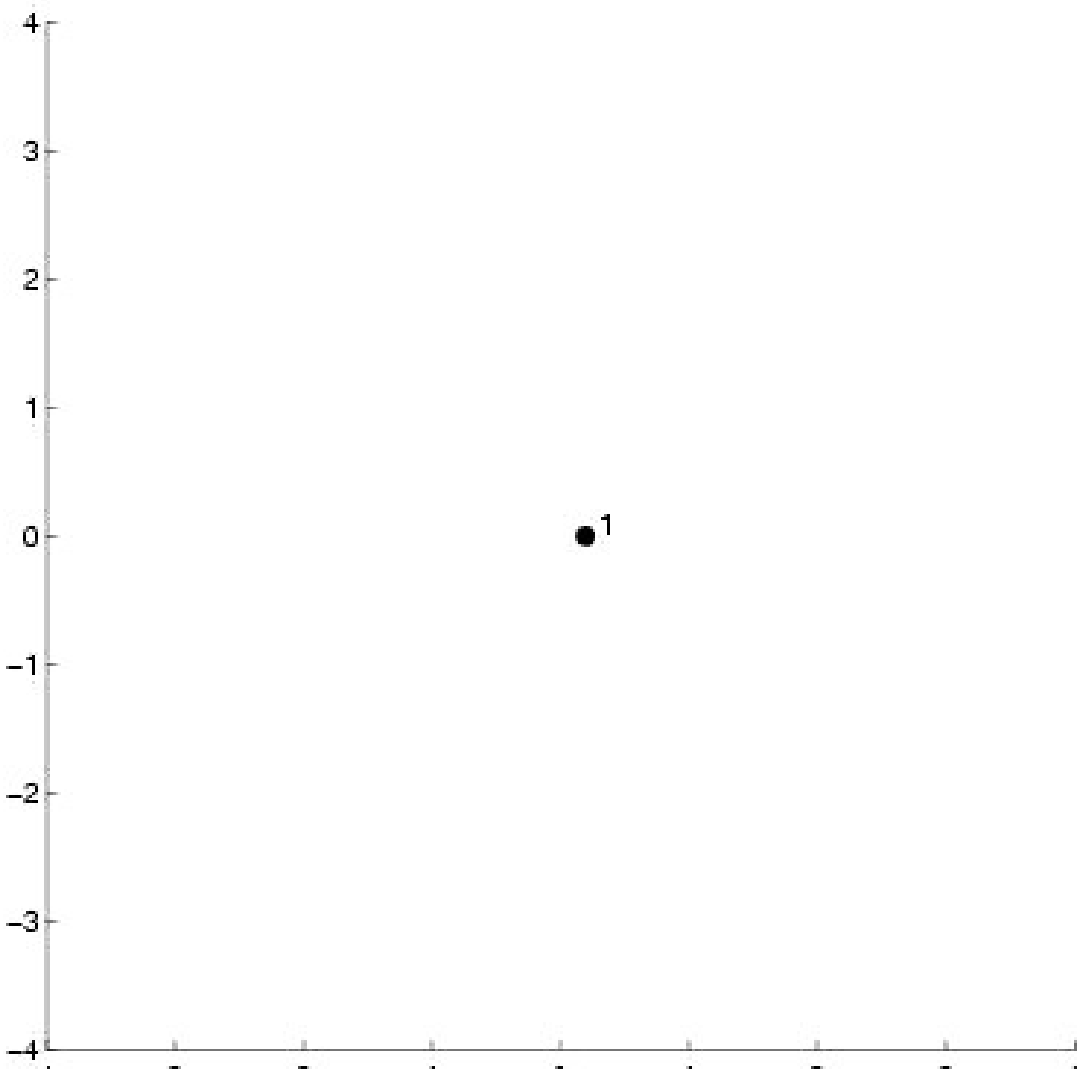


Propagation and update

Simulated Results Pose Uncertainty with no Updates



Simulated Results Pose Uncertainty with Updates



Extended Kalman Filter for SLAM

- State vector
  - Expanded to contain entries for all landmarks positions: $X = \begin{bmatrix} X_R^T & X_{L_1}^T & \dots & X_{L_n}^T \end{bmatrix}^T$
  - State vector can be grown as new landmarks are discovered
  - Covariance matrix is also expanded

$$\begin{bmatrix} P_{RR} & P_{RL_1} & \cdots & P_{RL_N} \\ P_{L_1R} & P_{L_1L_1} & \cdots & P_{L_1L_N} \\ \vdots & \vdots & \ddots & \vdots \\ P_{L_NR} & P_{L_NL_1} & \cdots & P_{L_NL_N} \end{bmatrix}$$

Extended Kalman Filter for SLAM

- Kinematic equations for landmark propagation

$$\hat{x}_{t+1} = \hat{x}_t + (V_t + w_{V_t})\delta t \cos\hat{\phi}_t$$

$$\hat{y}_{t+1} = \hat{y}_t + (V_t + w_{V_t})\delta t \sin\hat{\phi}_t$$

$$\hat{\phi}_{t+1} = \hat{\phi}_t + (\omega_t + w_{\omega_t})\delta t$$

$$\hat{x}_{L_i t+1} = \hat{x}_{L_i t}$$

$$\hat{y}_{L_i t+1} = \hat{y}_{L_i t}$$

$$\hat{\phi}_{L_i t+1} = \hat{\phi}_{L_i t}$$

Extended Kalman Filter for SLAM

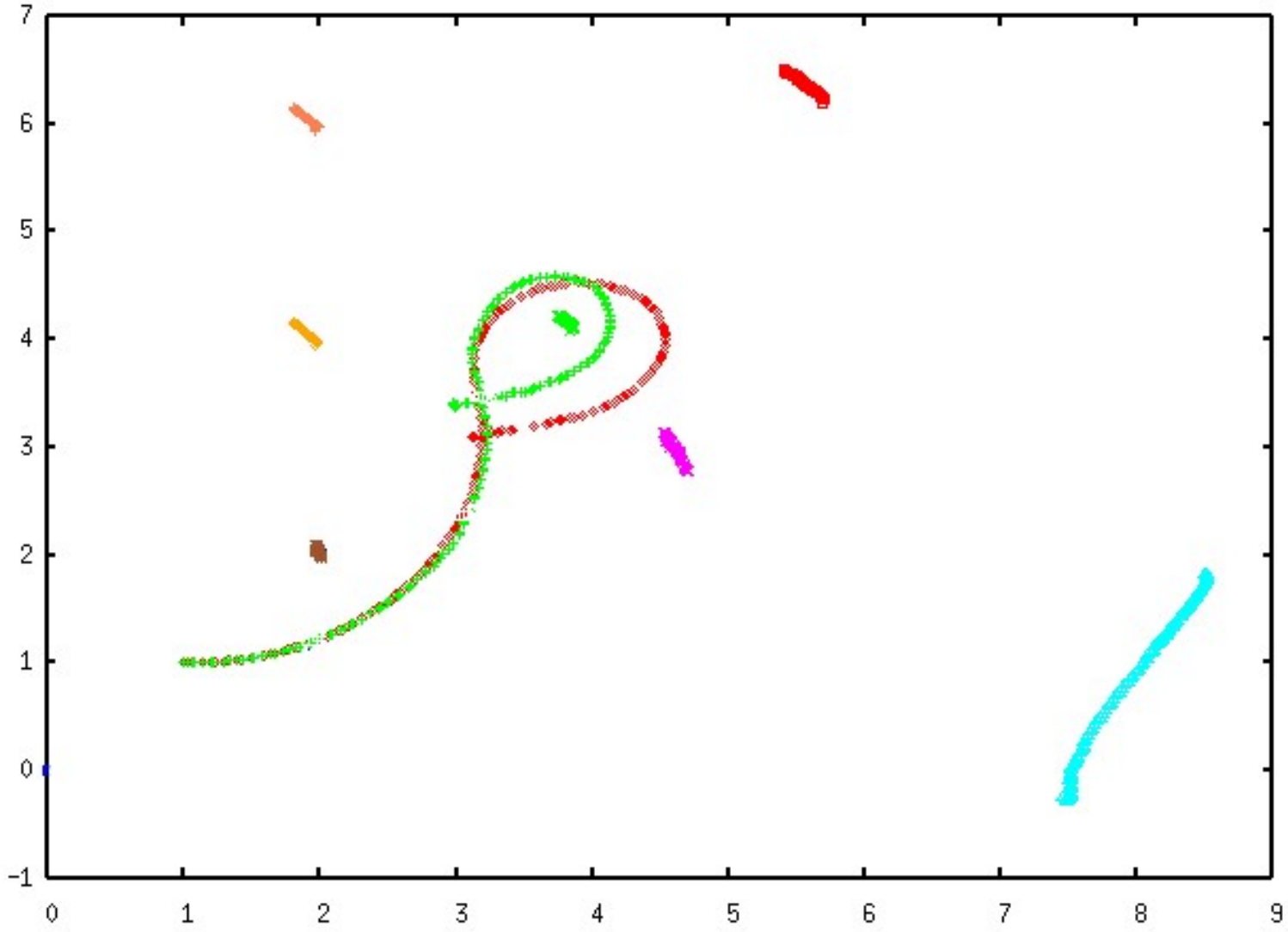
- Sensor equations for update:

$$\tilde{X} = \begin{bmatrix} \tilde{X}_R^T & \tilde{X}_{L_1}^T & \cdots & \tilde{X}_{L_i}^T & \cdots & \tilde{X}_{L_n}^T \end{bmatrix}$$

$$H = \begin{bmatrix} H_R & 0 & \cdots & 0 & H_{L_i} & 0 & \cdots & 0 \end{bmatrix}$$

- Very powerful because covariance update records *shared information* between landmarks and robot positions

EKF for SLAM



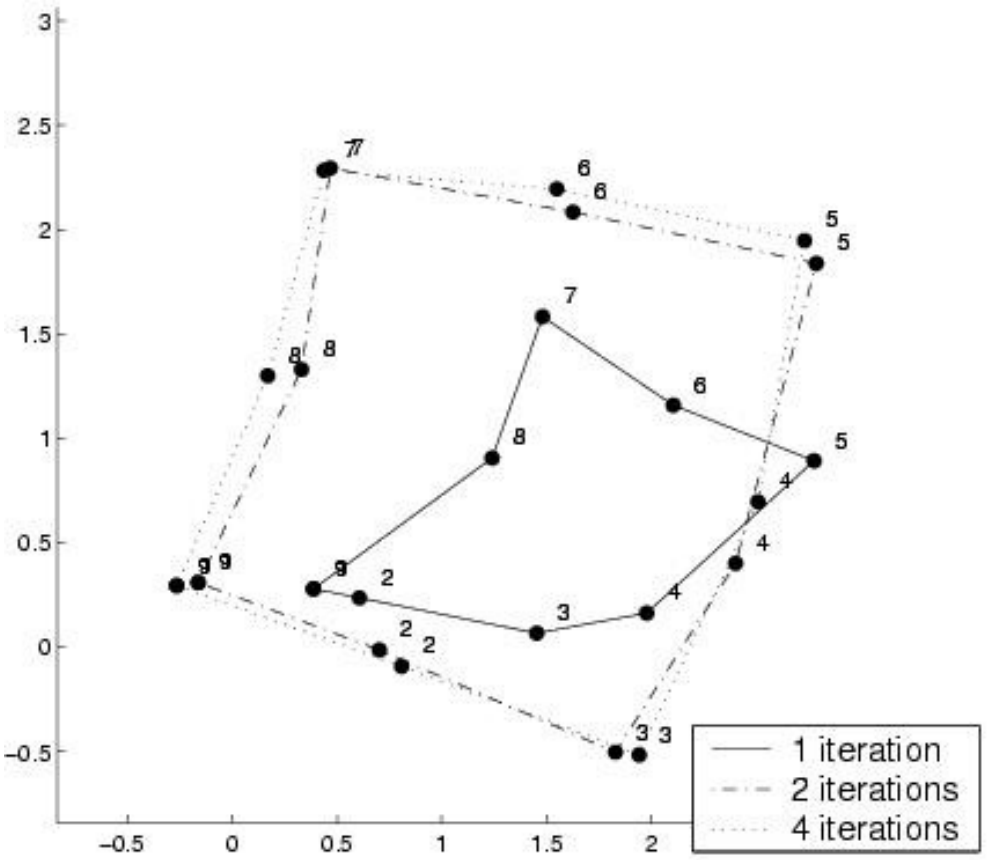
Enhancements to EKF

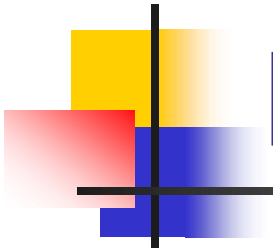
- Iterated Extended Kalman Filter

State and covariance update

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1}$$
$$S_{t+1} = H_{t+1} P_{k+1/k} H_{t+1}^T + R_{t+1}$$
$$K_{t+1} = P_{k+1/k} H_{t+1}^T S_{t+1}^{-1}$$
$$\hat{X}_{k+1/k+1} = \hat{X}_{k+1/k} + K_{t+1} r_{t+1}$$
$$P_{k+1/k+1} = P_{k+1/k} - K_{t+1} S_{t+1} K_{t+1}^T$$

Iterate state update equation until convergence





Enhancements to the EKF

- Multiple hypothesis tracking
  - Multiple Kalman filters are used to track the data
  - Multi-Gaussian approach allows for representation of arbitrary probability densities
  - Consistent hypothesis are tracked while highly inconsistent hypotheses are dropped
  - Similar in spirit to particle filter, but orders of magnitude fewer filters are tracked as compared to the particle filter