Robotics 2
Target Tracking

Kai Arras, Cyrill Stachniss,
Maren Bennewitz, Wolfram Burgard

Slides by Kai Arras, Gian Diego Tipaldi, v.1.1, Jan 2012

Chapter Contents

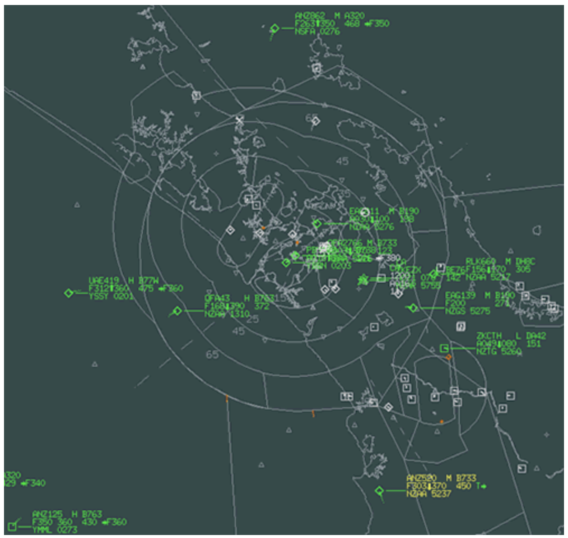
- **Target Tracking Overview**
  - Applications
  - Types of Problems and Errors
  - Algorithms
- **Linear Discrete Systems (LDS)**
- **Kalman Filter (KF)**
  - Overview
  - Error Propagation
  - Esimation Cycle
  - Limitations
- **Extended Kalman Filter (EKF)**
  - Overview
  - First-Order Error Propagation
- **Tracking Example**

Target Tracking Overview

"Tracking is the estimation of the state of a moving object based on remote measurements." [Bar-Shalom]

- **Detection** is knowing the presence of an object (possibly with some attribute information)
- **Tracking** is maintaining the **state** and **identity** of an object over time despite **detection errors** (false negatives, false alarms), **occlusions**, and the presence of **other objects**

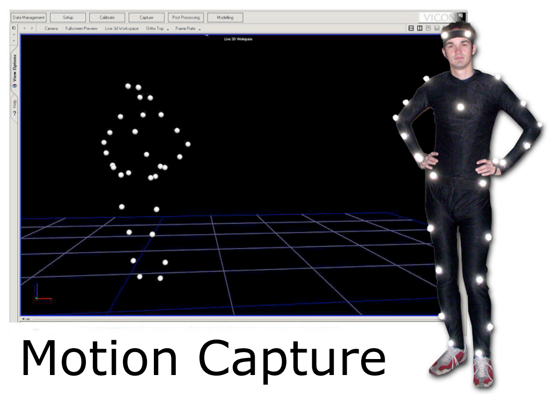
Target Tracking Applications



Air Traffic Control

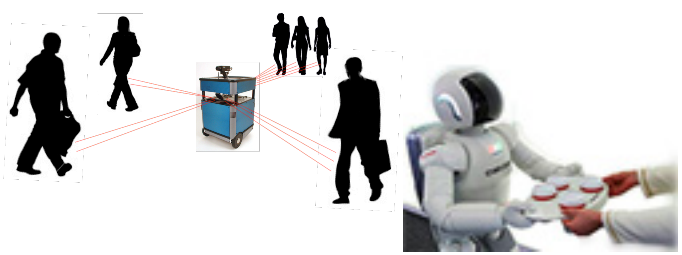
Fleet Management

Surveillance



Motion Capture

Military Applications



Robotics, HRI

Tracking: Problem Types

- **Track stage**
  - Track formation (initialization)
  - Track maintenance (continuation)
- **Number of sensors**
  - Single sensor
  - Multiple sensors
- **Sensor characteristics**
  - Detection probability (PD)
  - False alarm rate (PF)
- **Target behavior**
  - Nonmaneuvering
  - Maneuvering
- **Number of targets**
  - Single target
  - Multiple targets
- **Target size**
  - Point-like target
  - Extended target

Tracking: Error Types

1. Uncertainty in the **values** of measurements:
   Called "noise"

   ➔ Solution: **Filtering** (State estimation theory)

2. Uncertainty in the **origin** of measurements:
   measurement might originate from sources different from the target of interest. Reasons :
   - False alarms
   - Decoys and countermeasures
   - Multiple targets

   ➔ Solution: **Data Association**
   (Statistical decision theory)

Tracking: Problem Statement

- **Given**
  - Model of the **system dynamics** (process or plant model). Decribes the evolution of the state
  - Model of the **sensor** with which the target is observed
  - Probabilistic models of the **random factors** (noise sources) and the prior information
- **Wanted**
  - System state estimate such as **kinematic** (e.g. position), **feature** (e.g. target class) or **parameters** components
- **In a way that…**
  - Accuracy and/or reliability is higher than the raw measurements
  - Contains information not available in the measurements

Tracking Algorithms

- **Single non-maneuvering** target, **no** origin uncert.
  - Kalman filter (KF)/Extended Kalman filter (EKF)
- **Single maneuvering** target, **no** origin uncertainty
  - KF/EKF with variable process noise
  - Muliple model approaches (MM)
- **Single non-maneuvering** target, **origin** uncertainty
  - KF/EKF with Nearest/Strongest Neighbor Data Association
  - Probabilistic Data Association filter (PDAF)
- **Single maneuvering** target, **origin** uncertainty
  - Multiple model-PDAF

Tracking Algorithms

- **Multiple non-maneuvering targets**
  - Joint Probabilistic Data Association filter (JPDAF)
  - Multiple Hypothesis Tracker (MHT)
- **Multiple maneuvering targets**
  - MM-variants of MHT (e.g. IMMMHT)

- Other Bayesian filtering schemes such as **Particle filters** have also been sucessfully applied to the tracking problem

Linear Dynamic System (LDS)

- A continuous-time **Linear Dynamic System (LDS)** can be described by a state equation of the form

  $$\dot{x}(t) = A(t)x(t) + B(t)u(t) + \xi(t)$$

  Called process or plant model

- The system can be **observed remotely** through

  $$z(t) = H(t)x(t) + \epsilon(t)$$

  Called observation or measurement model

- This is the **State-Space Representation**, omni-present in physics, control or estimation theory

- Provides the **mathematical formulation** for our estimation task

Linear Dynamic System (LDS)

- Stochastic process governed by

$$\boxed{\dot{x}(t) = A(t)x(t) + B(t)u(t) + \xi(t)}$$

  - $x \in \mathbb{R}^{n_x}$ is the state vector
  - $u \in \mathbb{R}^{n_u}$ is the input vector
  - $\xi \in \mathbb{R}^{n_x}$ is the process noise
  - $A \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ is the system matrix
  - $B \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ is the input gain

- The system can be observed through

$$\boxed{z(t) = H(t)x(t) + \epsilon(t)}$$

  - $z \in \mathbb{R}^{n_z}$ is the measurement vector
  - $\epsilon \in \mathbb{R}^{n_z}$ is the measurement noise
  - $H \in \mathbb{R}^{n_z} \times \mathbb{R}^{n_x}$ is the measurement matrix

Discrete-Time LDS

- Continuous model are difficult to realize
  - Algorithms work at discrete time steps
  - Measurements are acquired with certain rates

- In practice, **discrete models** are employed

- Discrete-time LDS are governed by

$$\boxed{x(k+1) = F(k)x(k) + G(k)u(k) + \xi(k)}$$

  - $F \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ is the state transition matrix
  - $G \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ is the discrete-time input gain

- Same observation function of continuous models

Discrete-Time LDS

- Continuous model are difficult to realize
  - Algorithms work at discrete time steps
  - Measurements are acquired with certain rates

- In practice, **discrete models** are employed

- Discrete-time LDS are governed by

In target tracking, the input is unknown!

$$\boxed{x(k+1) = F(k)x(k) + \cancel{G(k)u(k)} + \xi(k)}$$

  - $F \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ is the state transition matrix
  - $G \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_u}$ is the discrete-time input gain

- Same observation function of continuous models

LDS Example – Throwing ball

- We want to throw a ball and **compute its trajectory**
- This can be **easily done with a LDS**
- No uncertainties, no tracking, just physics
- The ball's **state** shall be represented as

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$$

- We ignore winds but consider the **gravity force** g

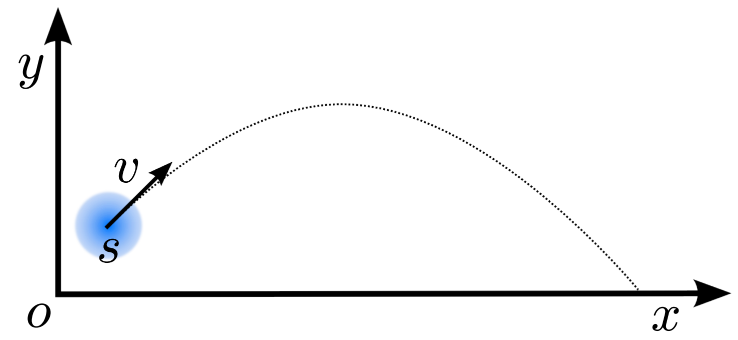
$$\mathbf{u} = -g$$

- No floor constraints
- We **observe** the ball with a noise-free position sensor

$$\mathbf{z} = \begin{bmatrix} x & y \end{bmatrix}^T$$

LDS Example – Throwing ball

- Throwing a ball from s with initial velocity v
- Consider only the gravity force, g, of the ball



- State vector
  $$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$$
- Initial state
  $$\mathbf{x}_0 = \begin{bmatrix} s_x & s_y & v_x & v_y \end{bmatrix}^T$$
- Input vector (scalar)
  $$\mathbf{u} = -g$$
- Measurement vector
  $$\mathbf{z} = \begin{bmatrix} x & y \end{bmatrix}^T$$

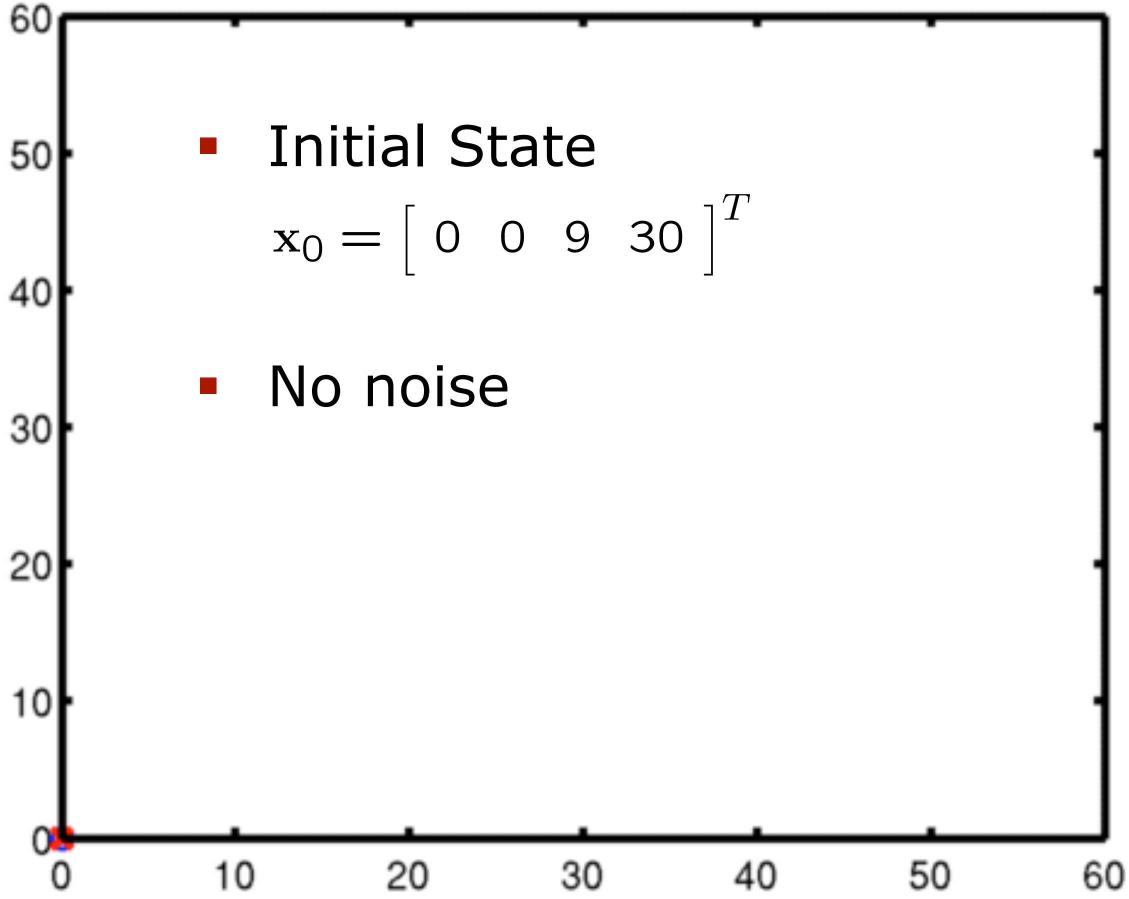
- Process matrices
  $$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
  $$G = \begin{bmatrix} 0 & \frac{T^2}{2} & 0 & T \end{bmatrix}^T$$
- Measurement matrix
  $$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

LDS Example – Throwing ball

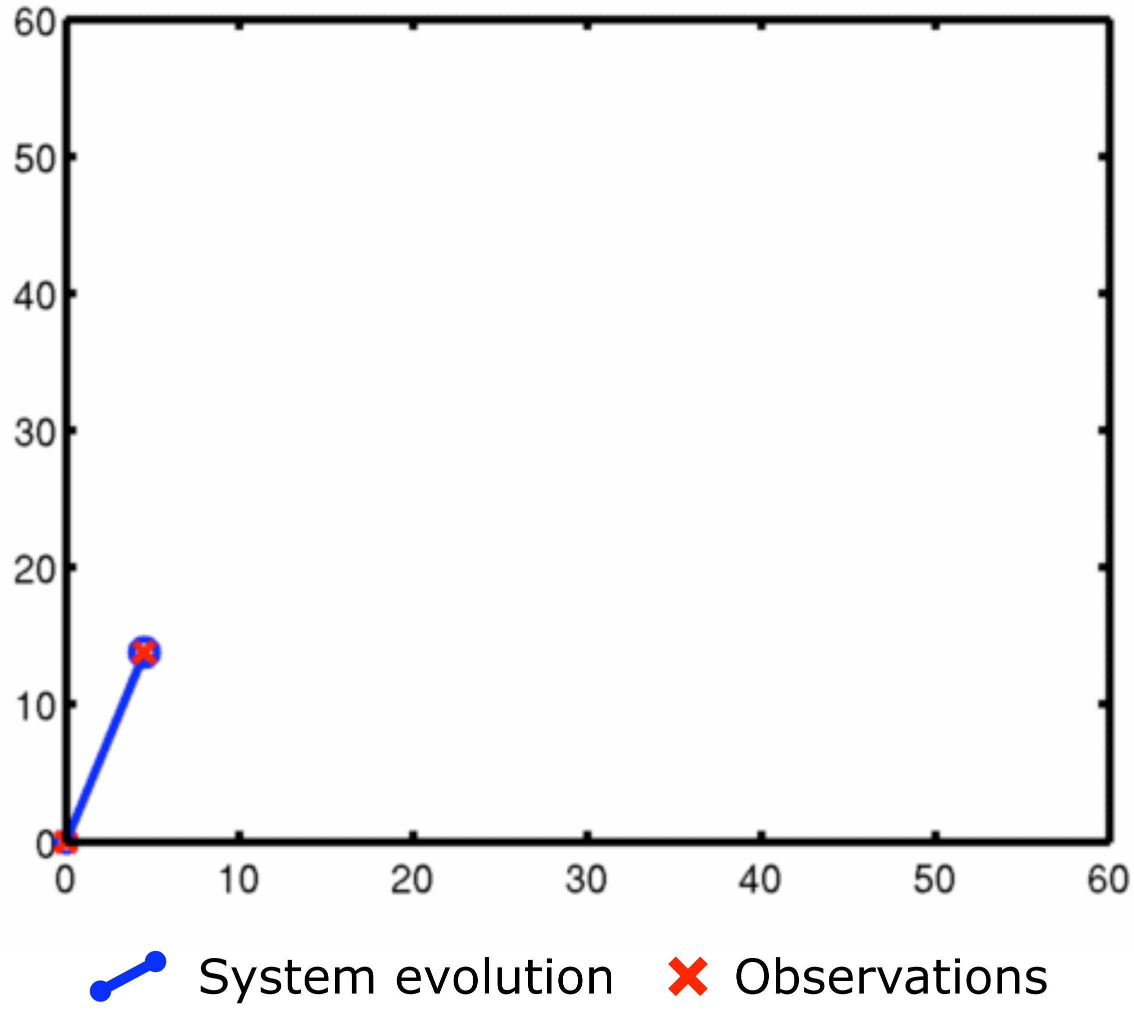
- Initial State

  $$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 9 & 30 \end{bmatrix}^T$$

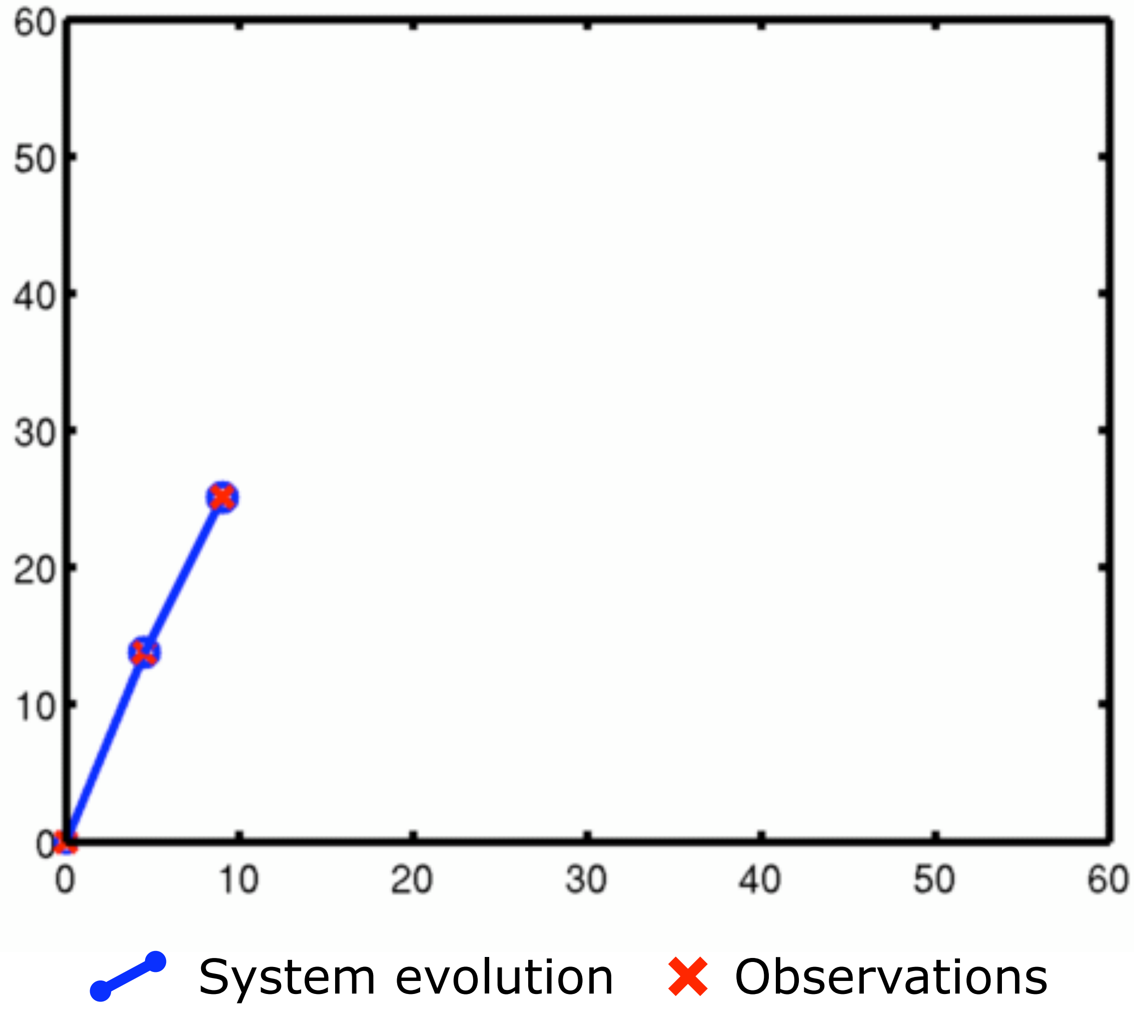
- No noise



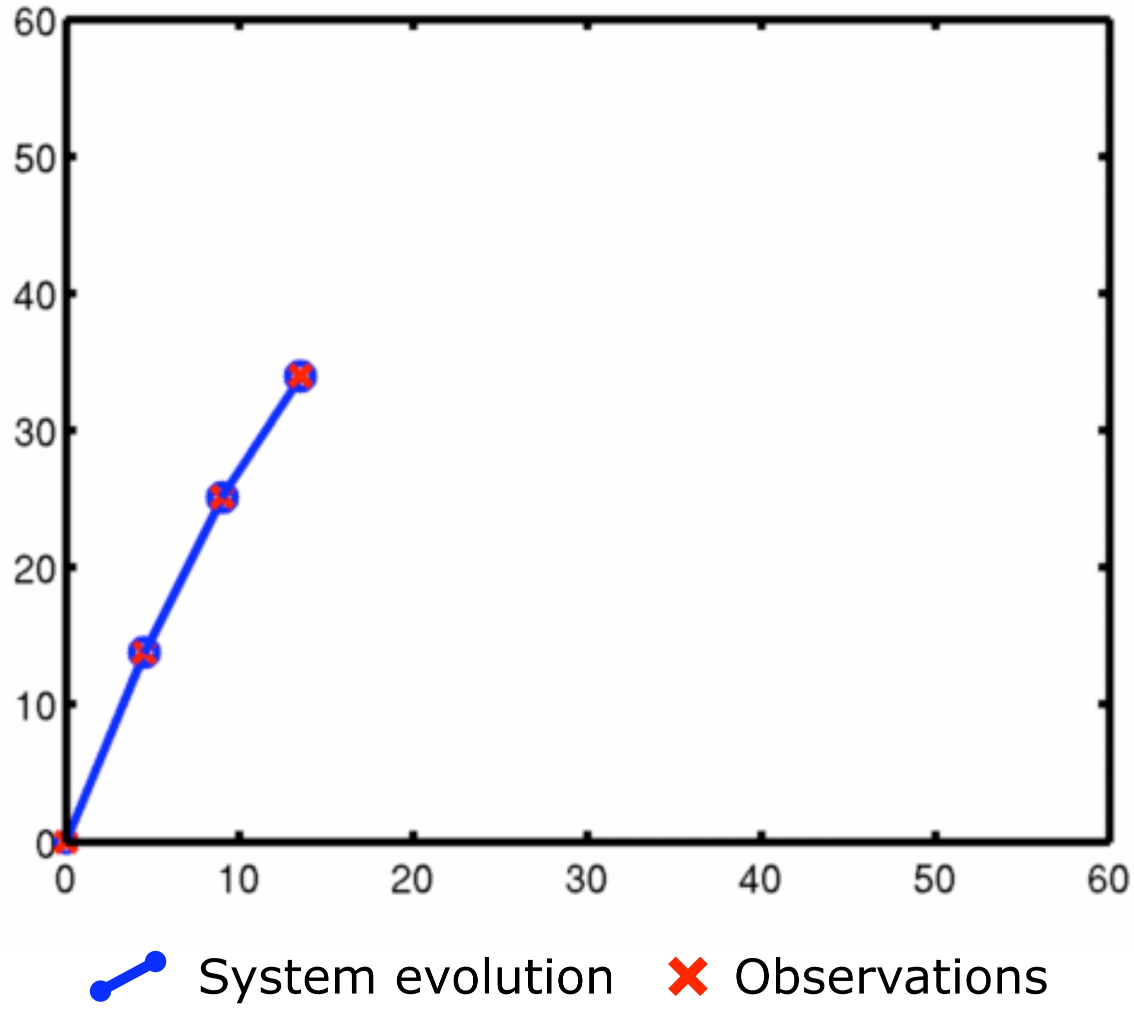
LDS Example – Throwing ball



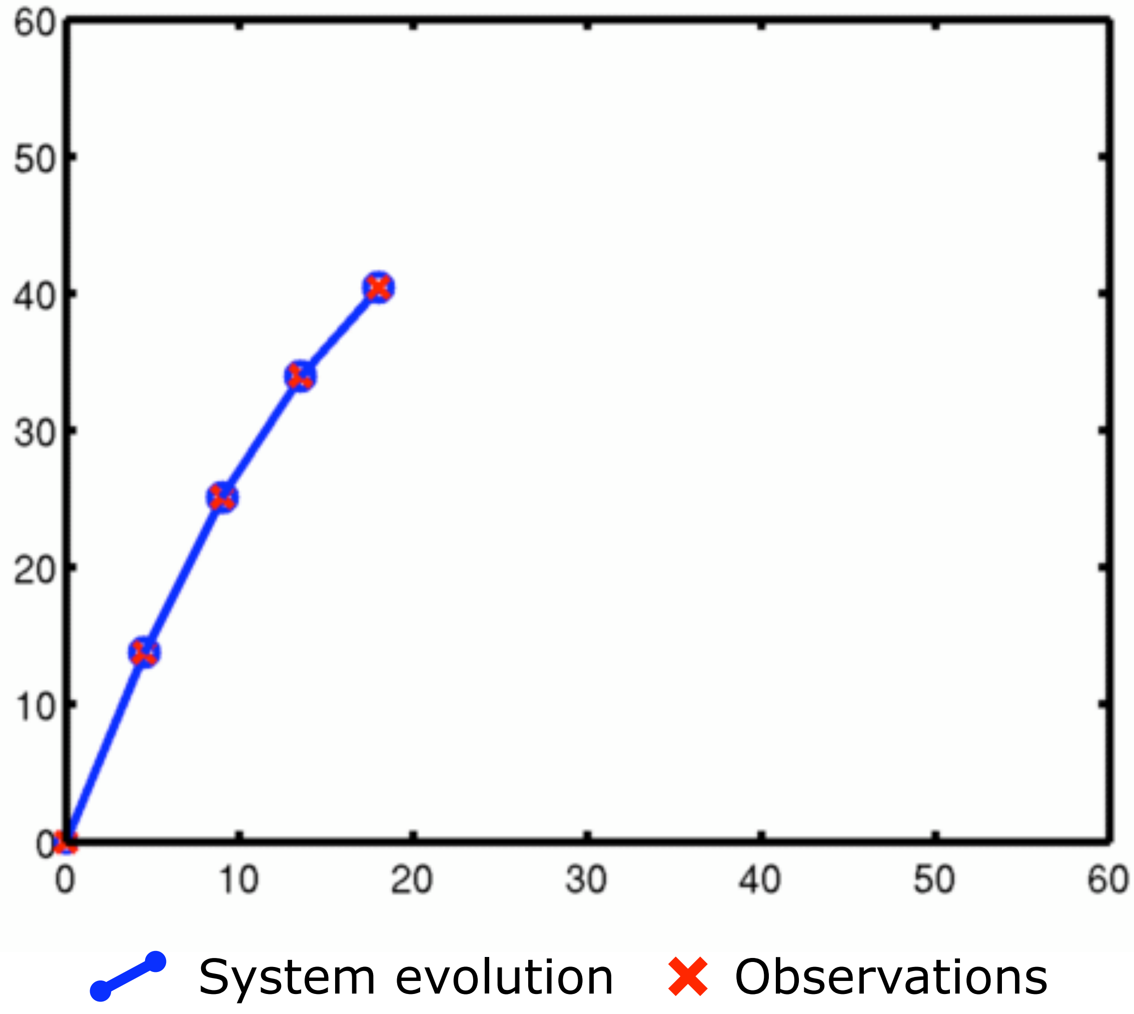
LDS Example – Throwing ball



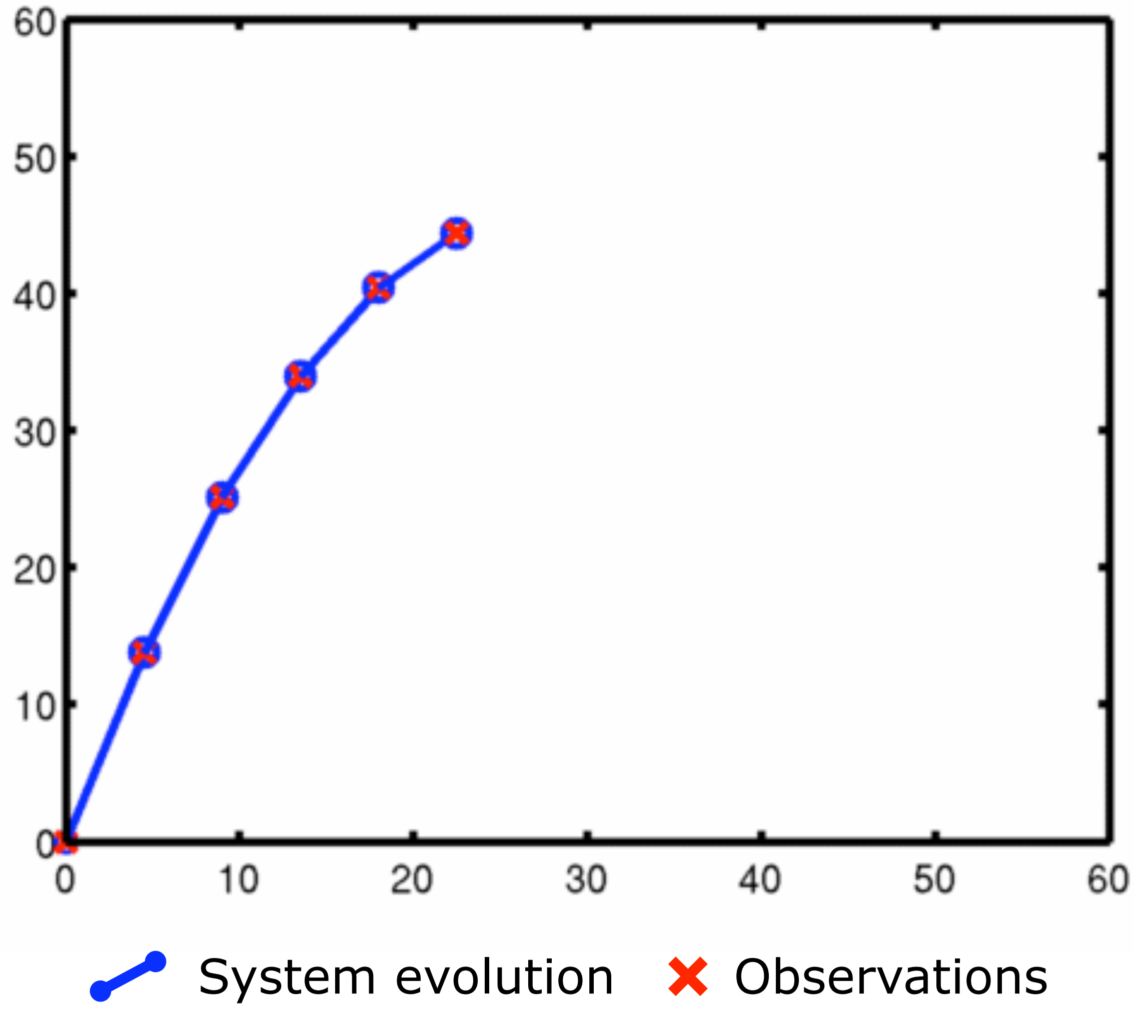
LDS Example – Throwing ball



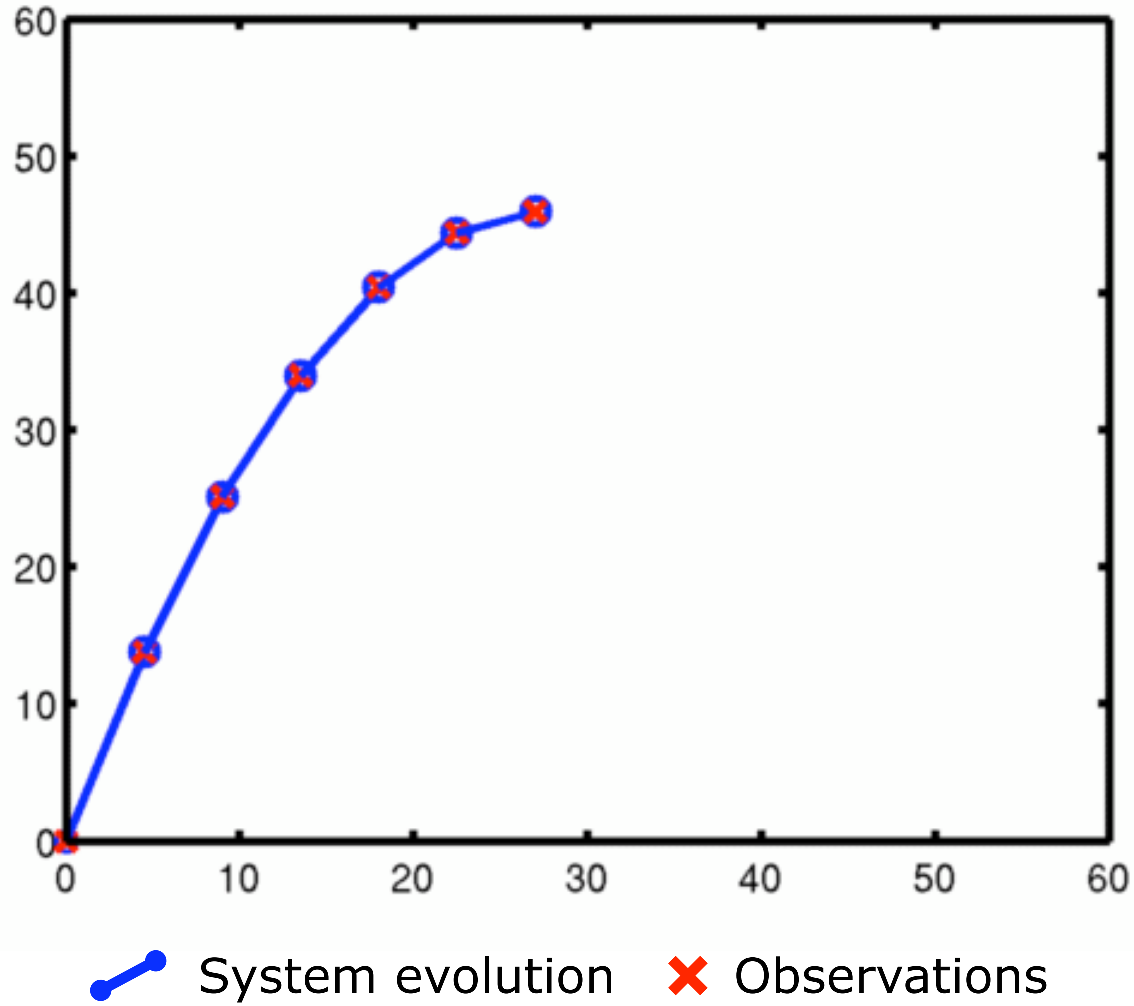
LDS Example – Throwing ball



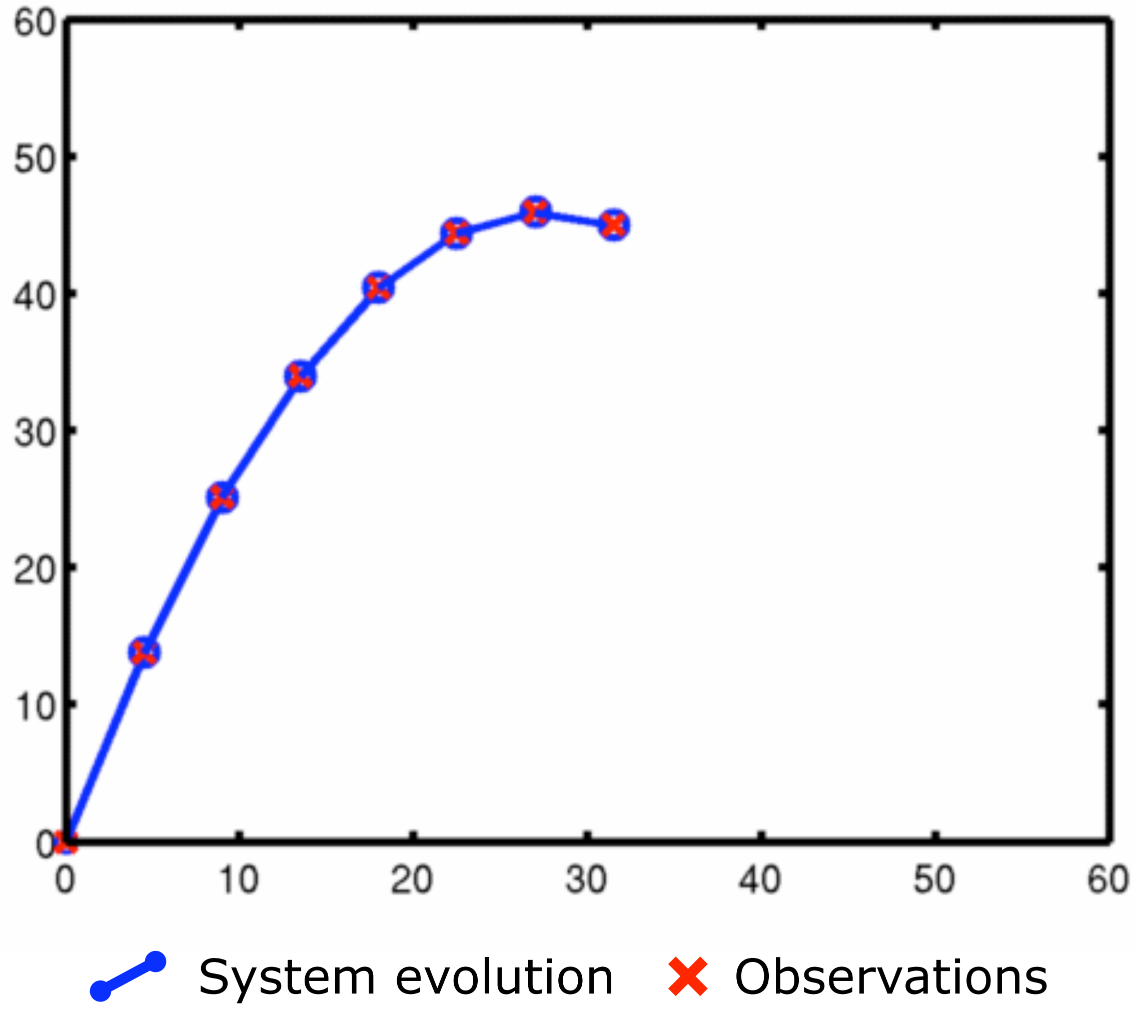
LDS Example – Throwing ball



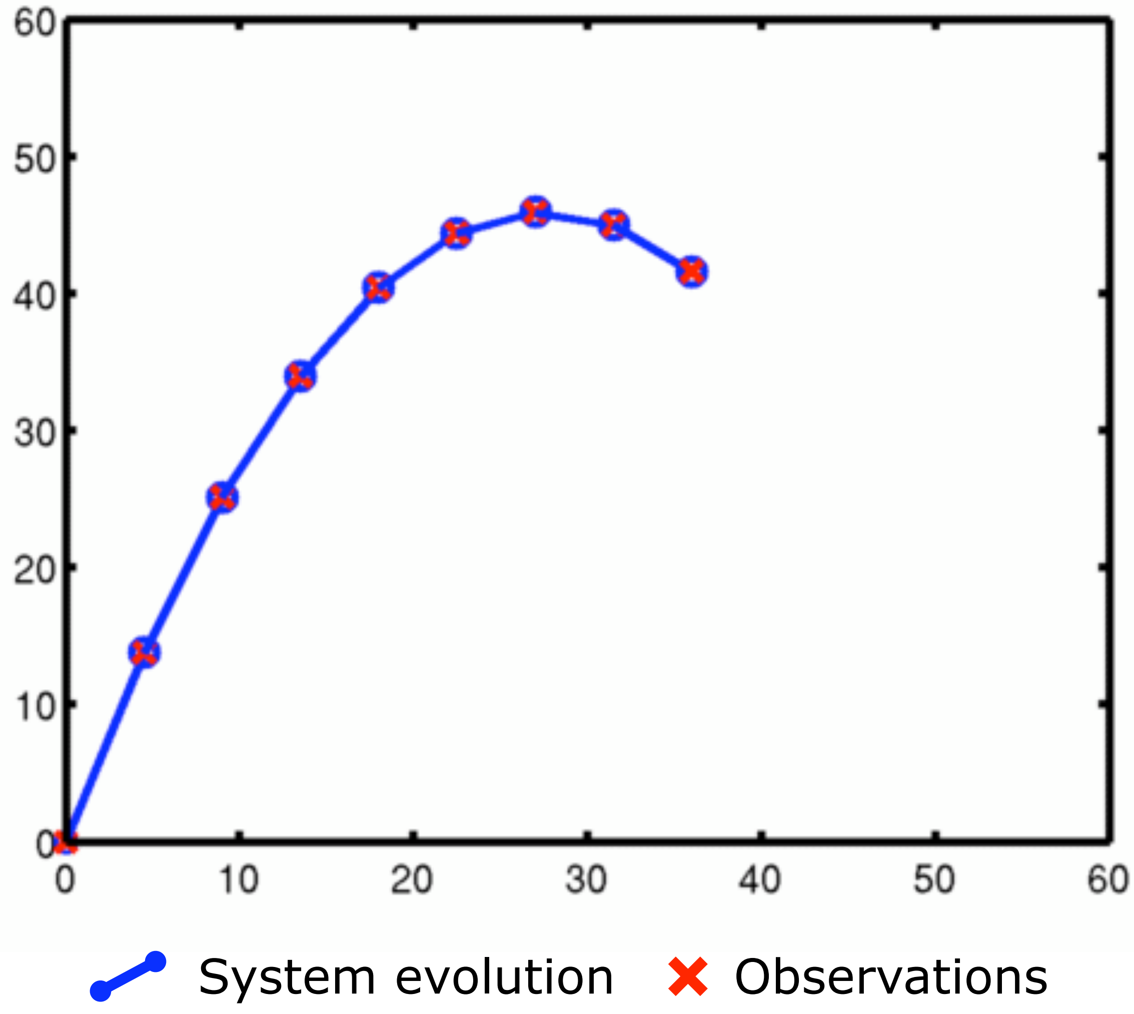
LDS Example – Throwing ball



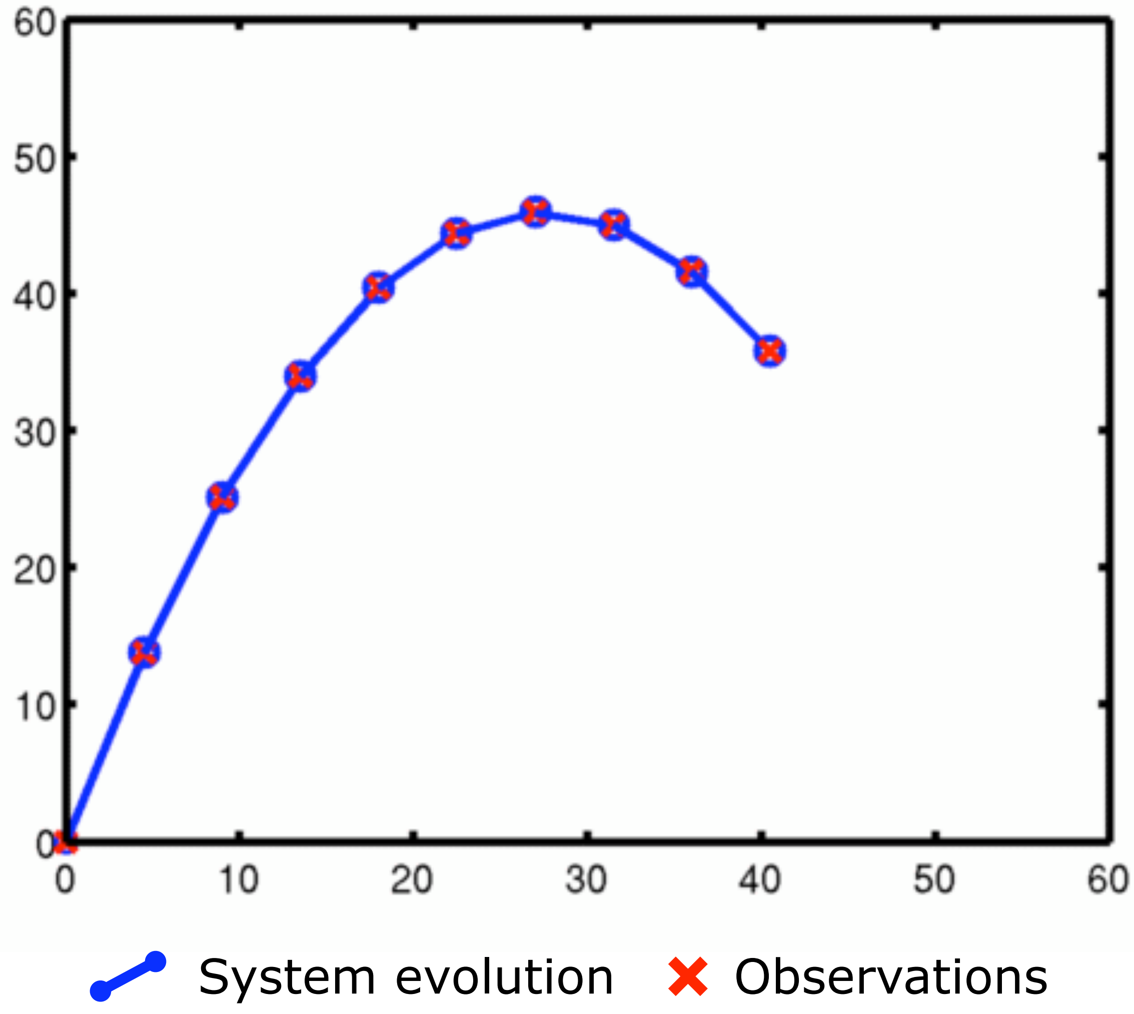
LDS Example – Throwing ball



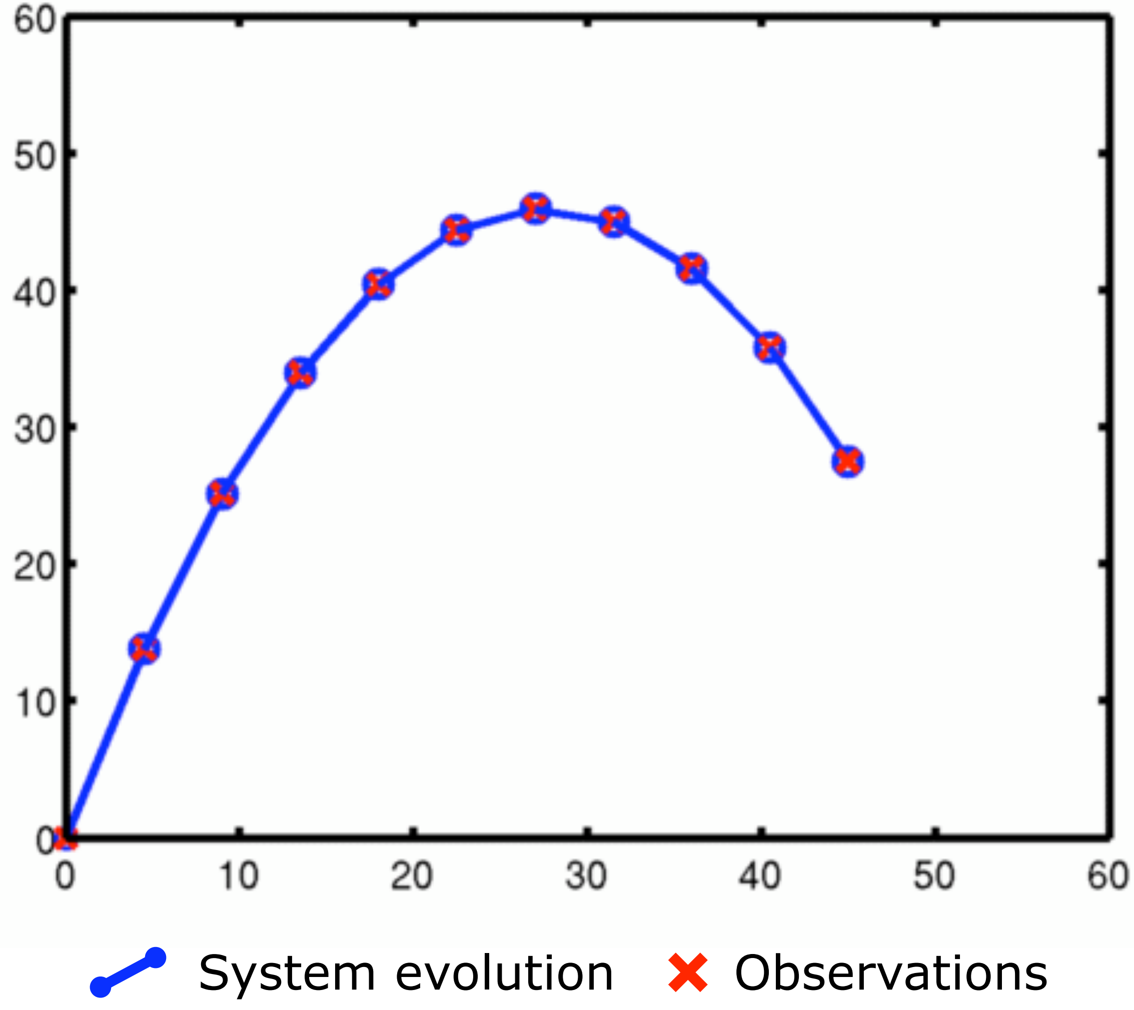
LDS Example – Throwing ball



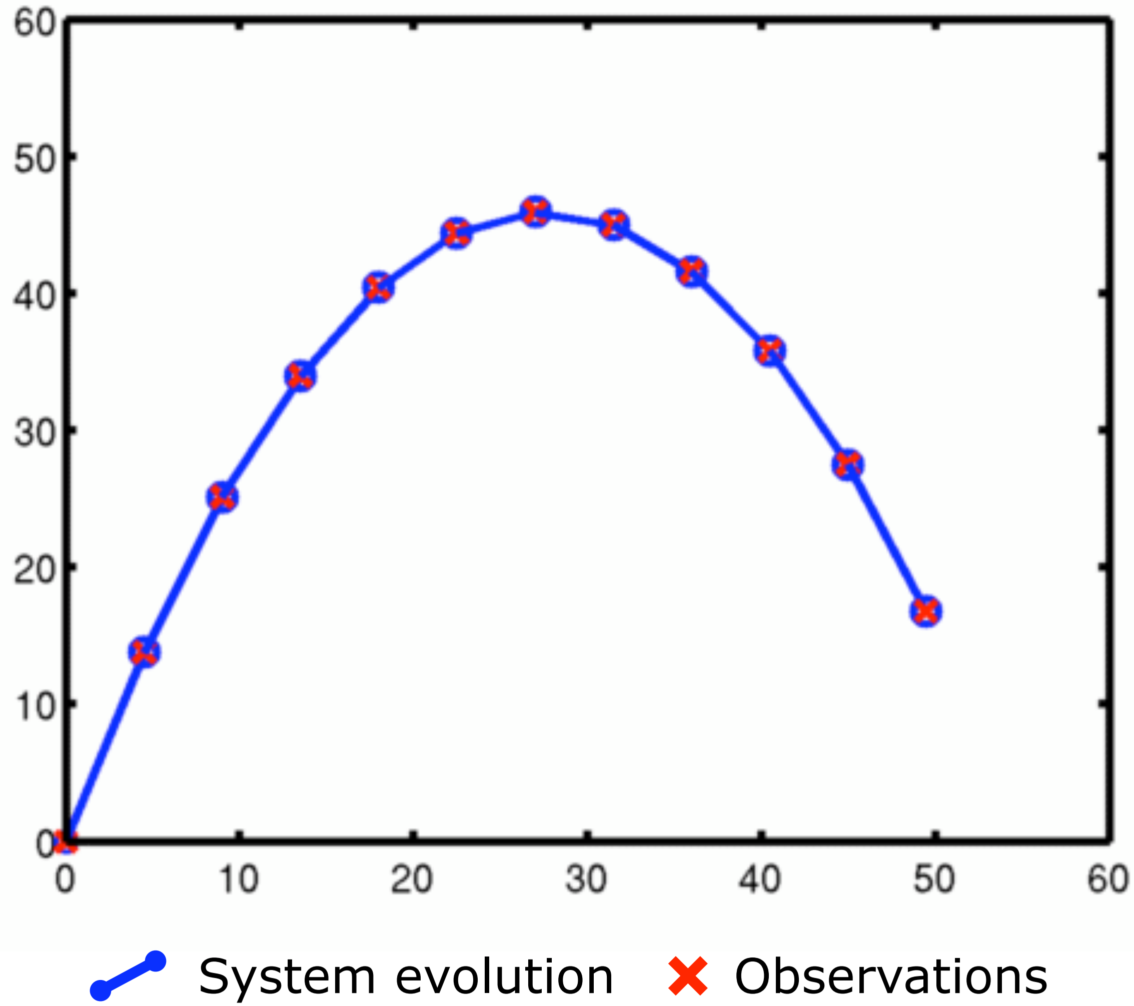
LDS Example – Throwing ball



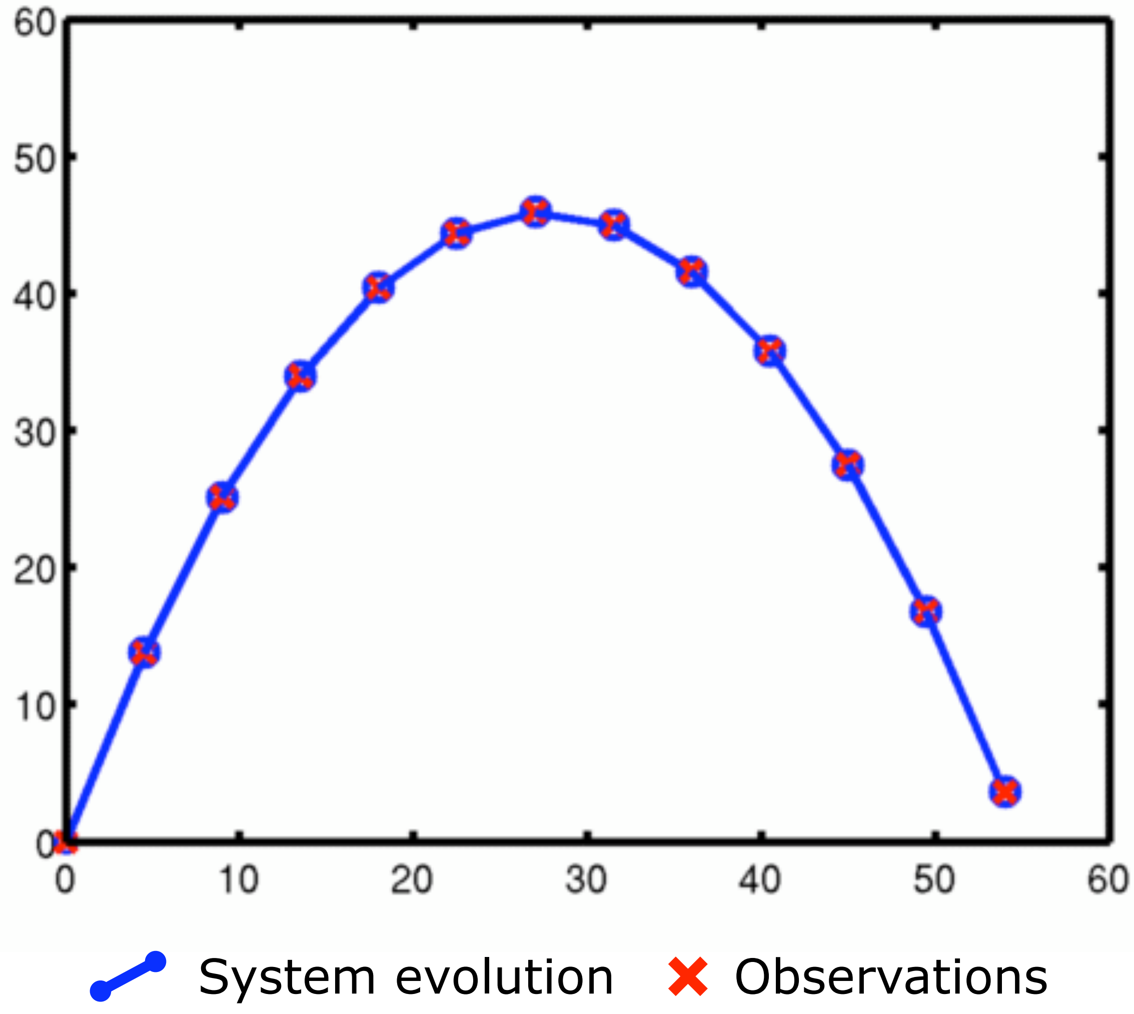
LDS Example – Throwing ball



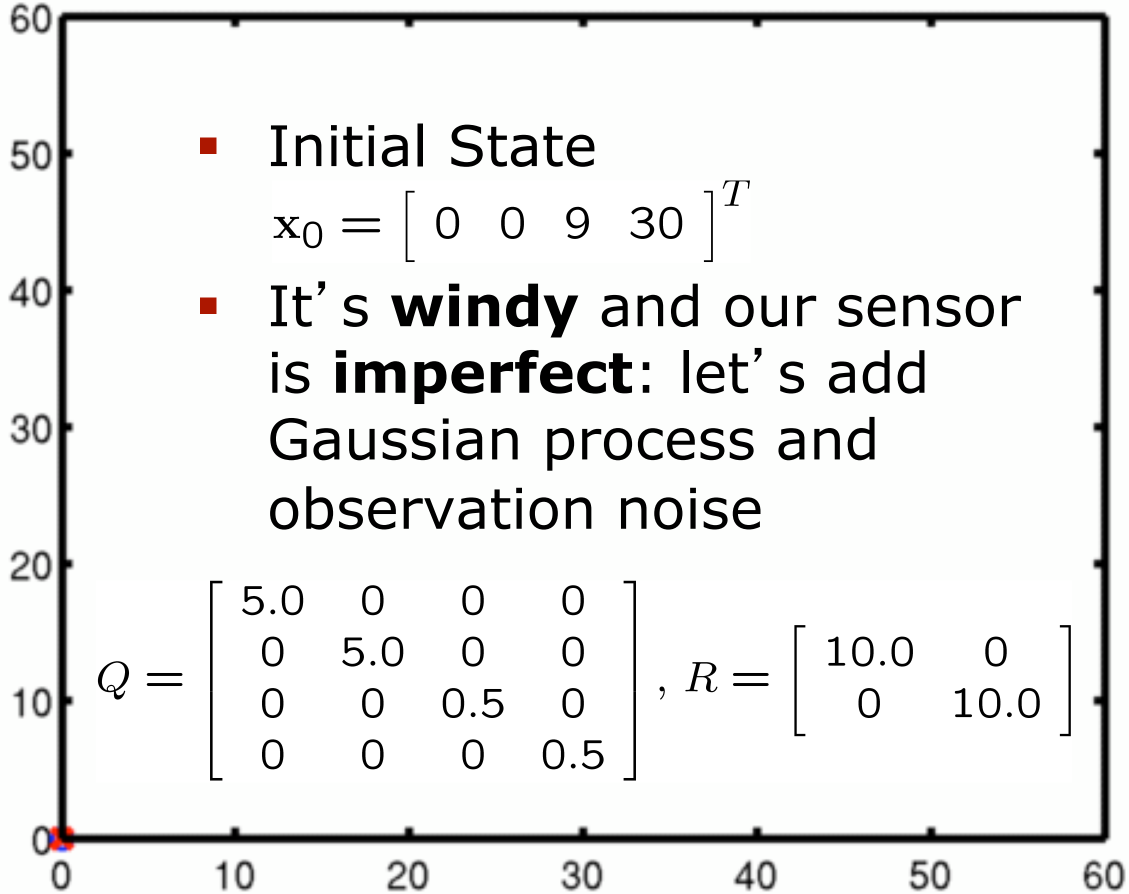
LDS Example – Throwing ball



LDS Example – Throwing ball



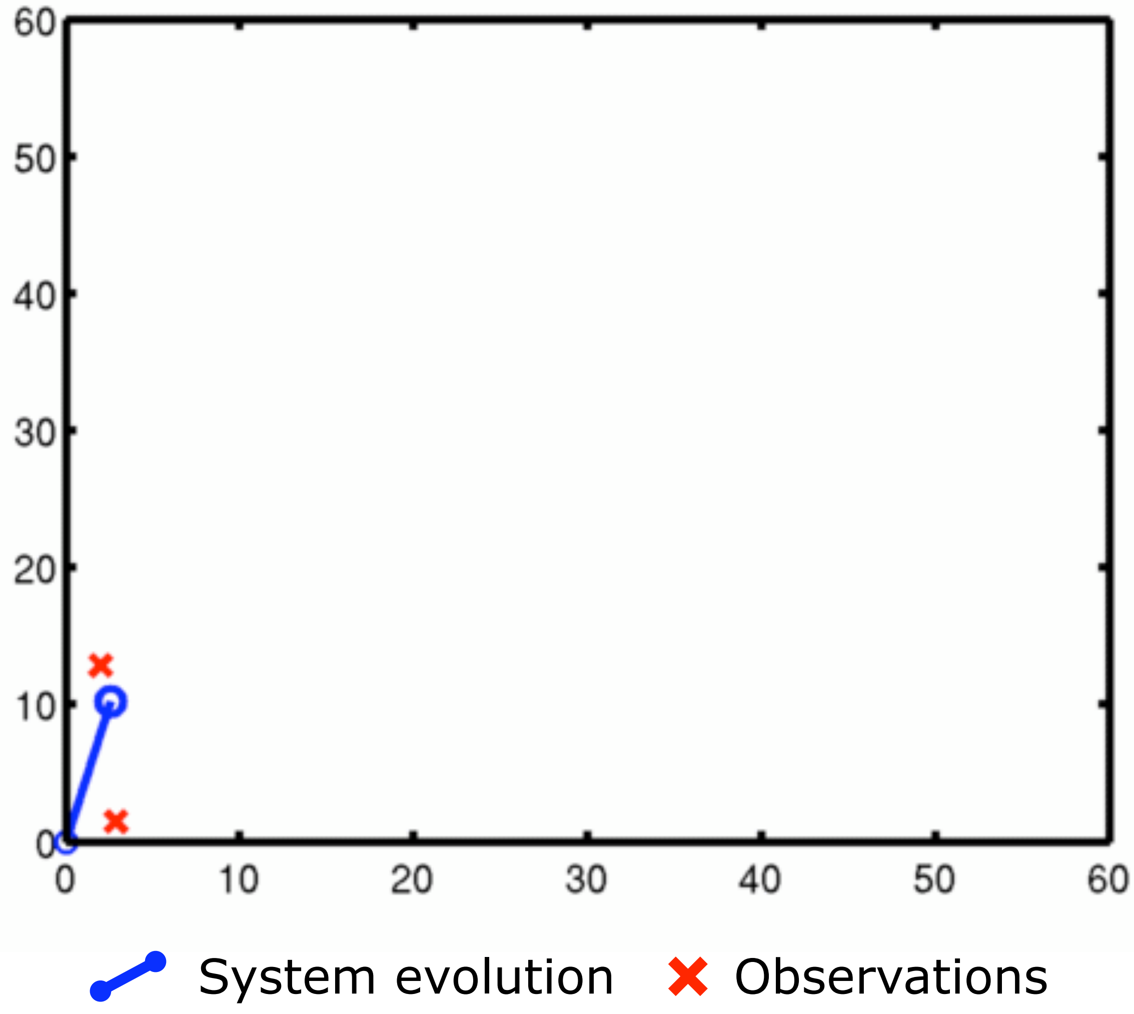
LDS Example – Throwing ball



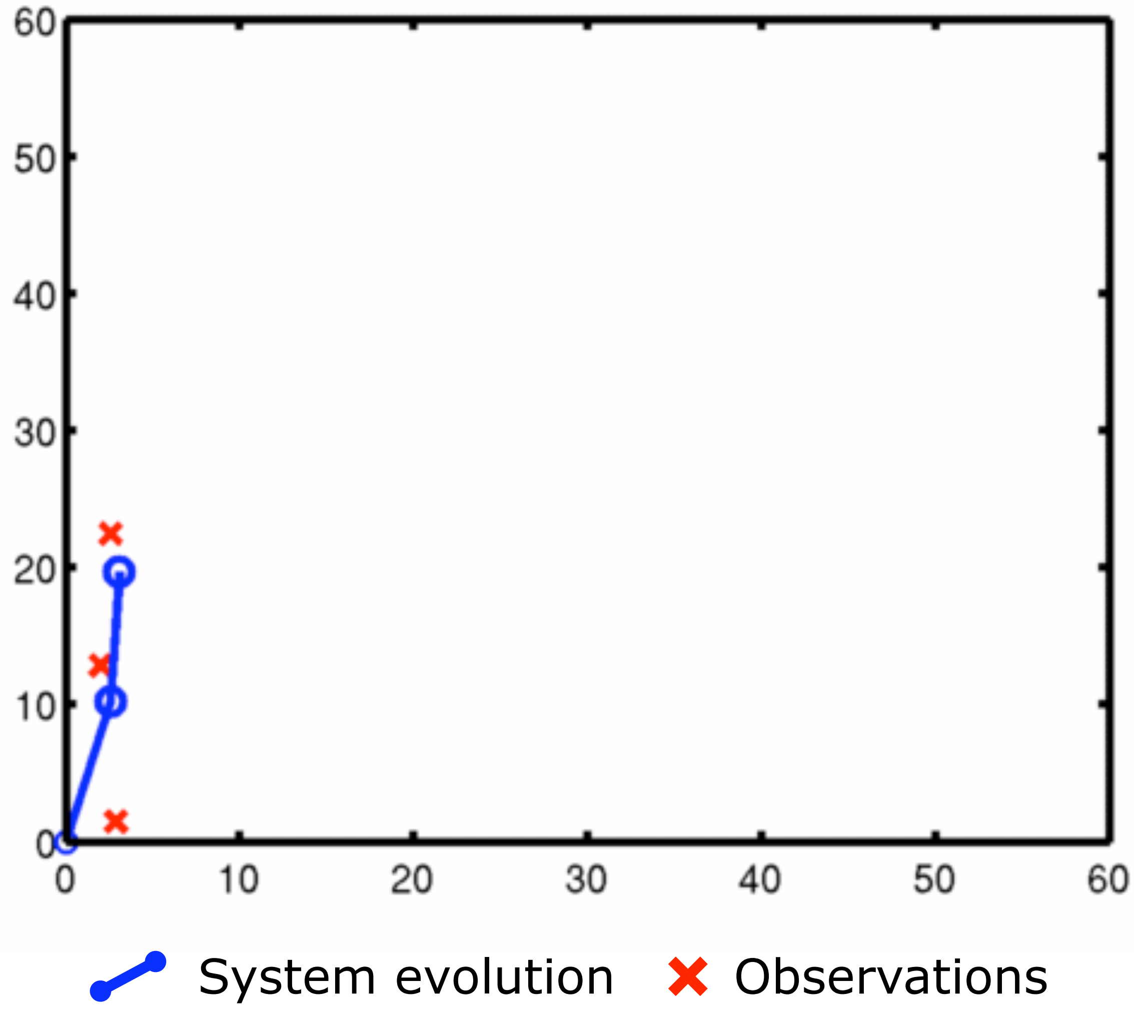
- Initial State
  $$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 & 9 & 30 \end{bmatrix}^T$$
- It's **windy** and our sensor is **imperfect**: let's add Gaussian process and observation noise

$$Q = \begin{bmatrix} 5.0 & 0 & 0 & 0 \\ 0 & 5.0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}, \; R = \begin{bmatrix} 10.0 & 0 \\ 0 & 10.0 \end{bmatrix}$$

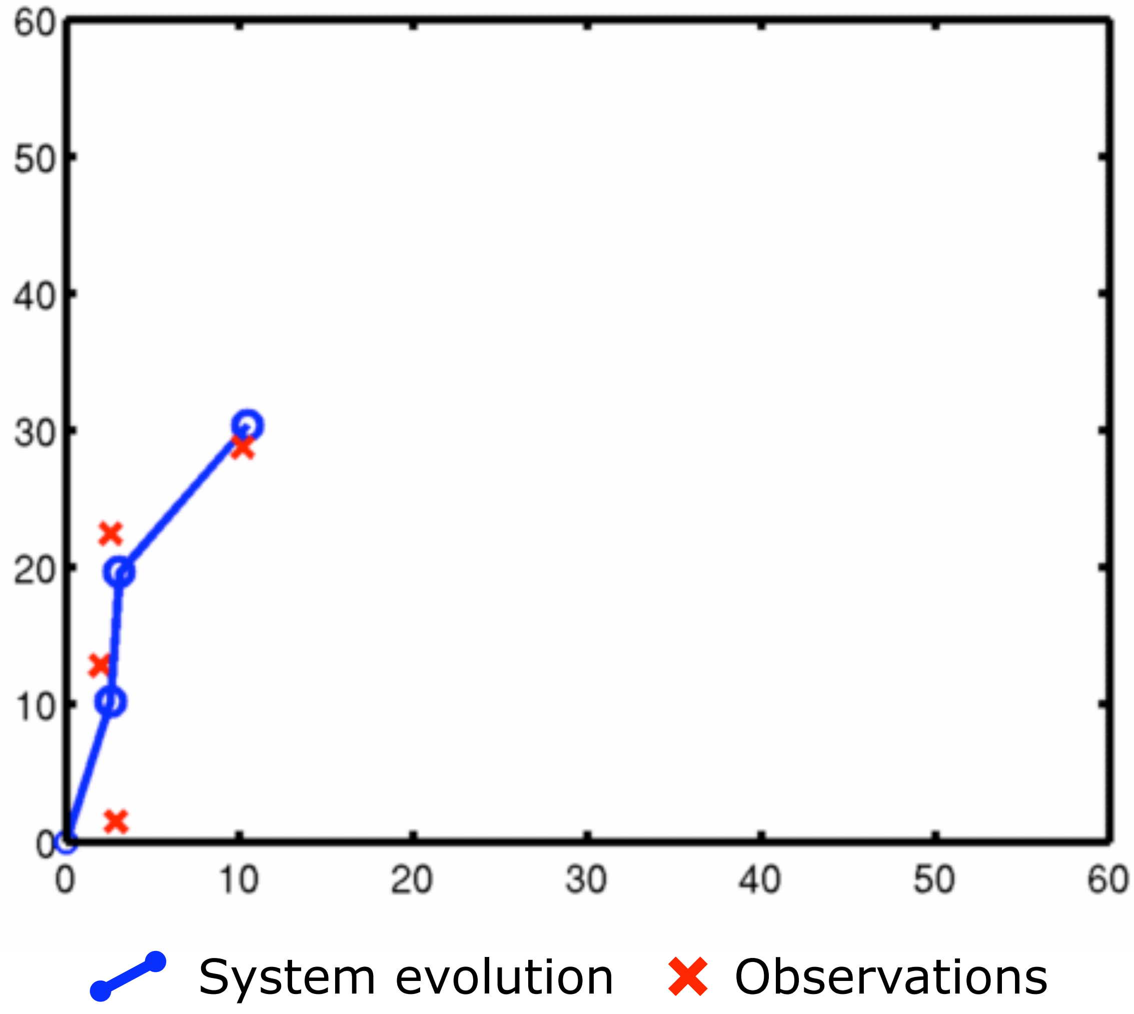
LDS Example – Throwing ball



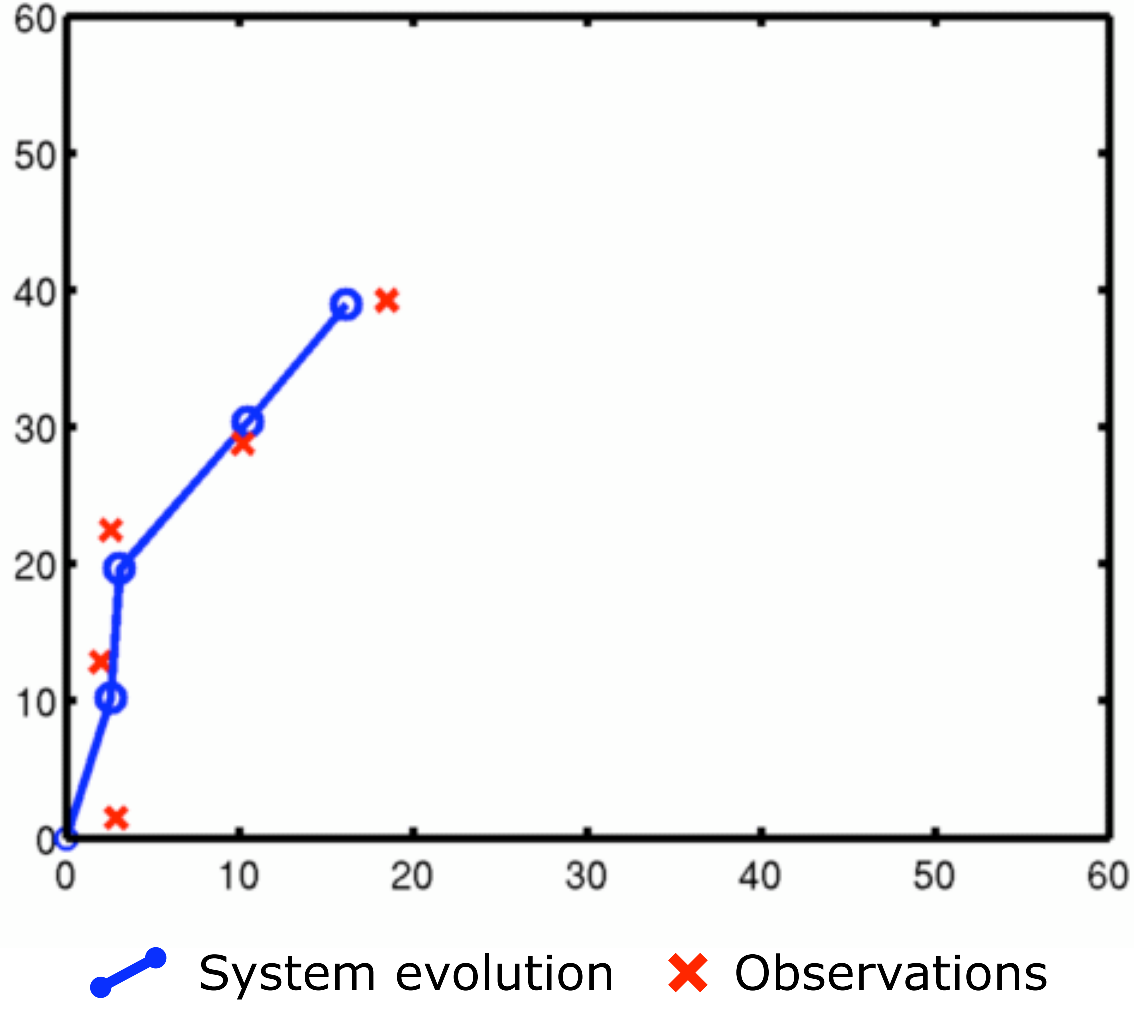
LDS Example – Throwing ball



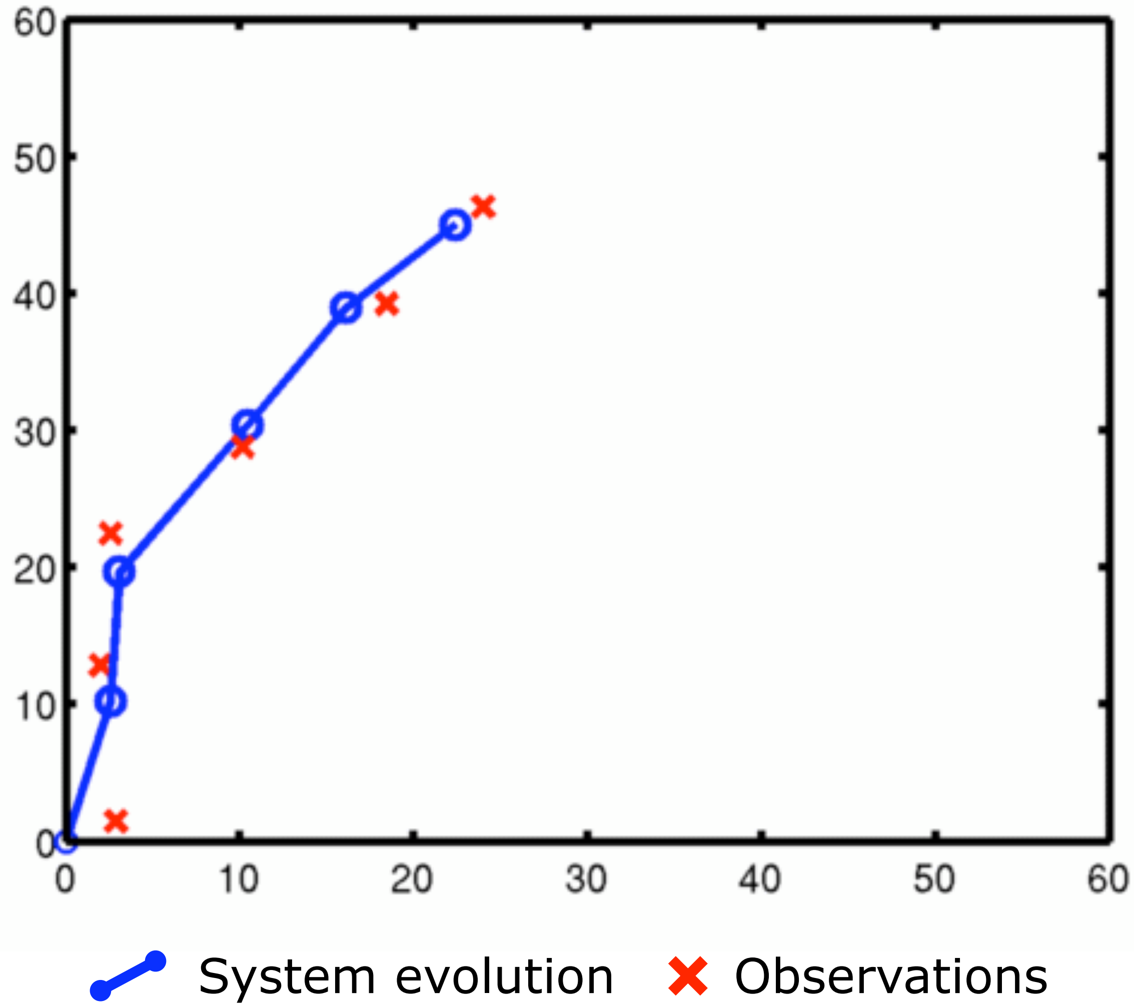
LDS Example – Throwing ball



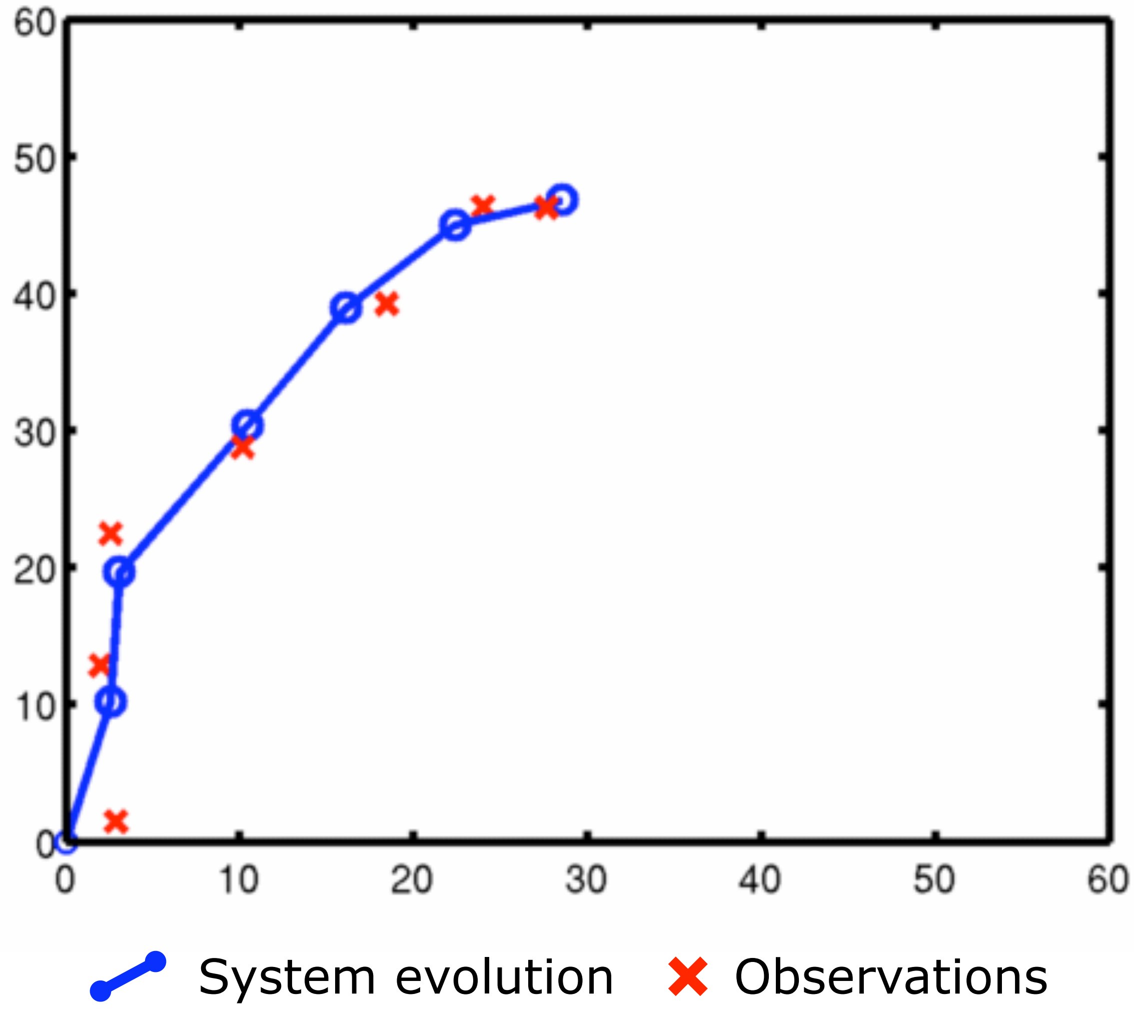
LDS Example – Throwing ball



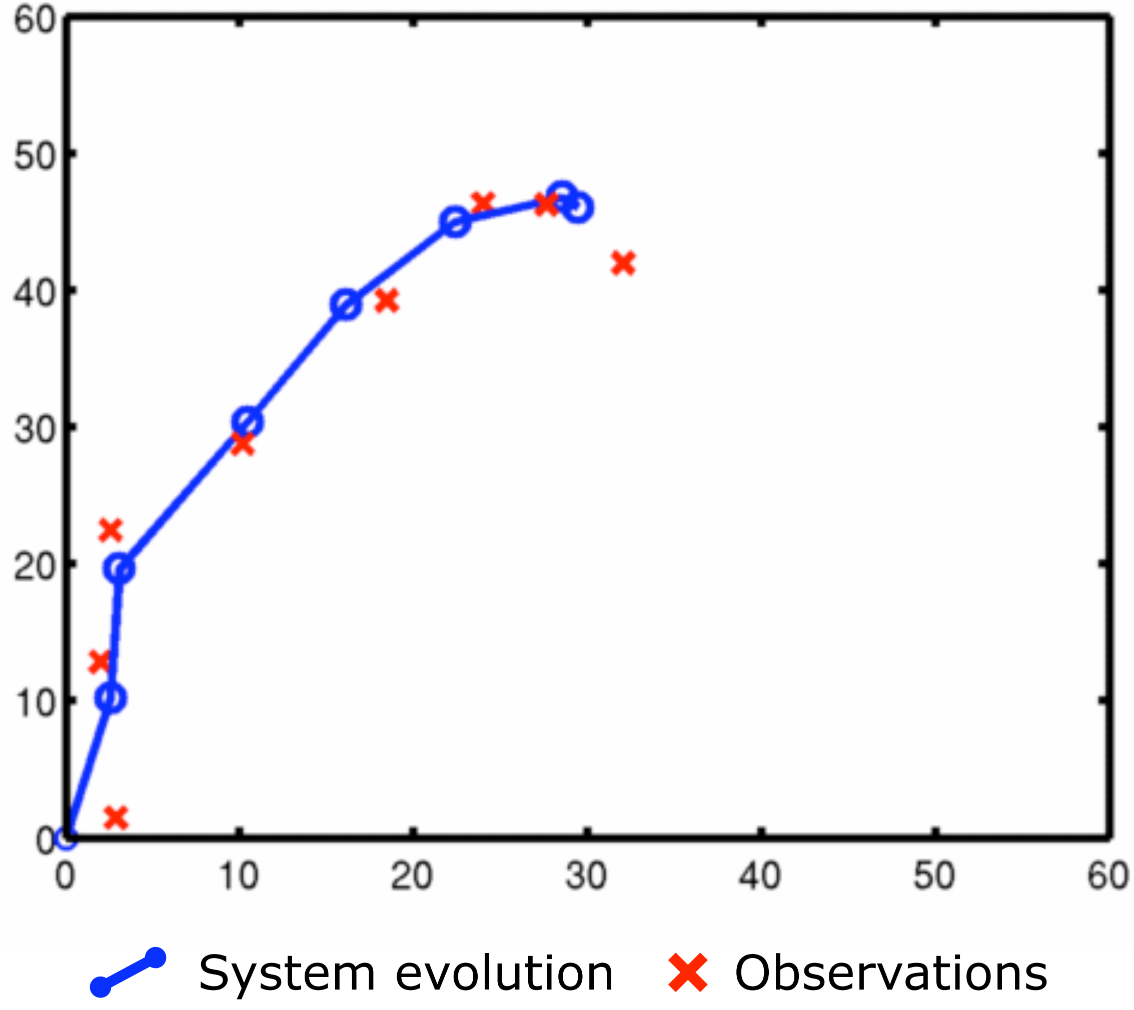
LDS Example – Throwing ball



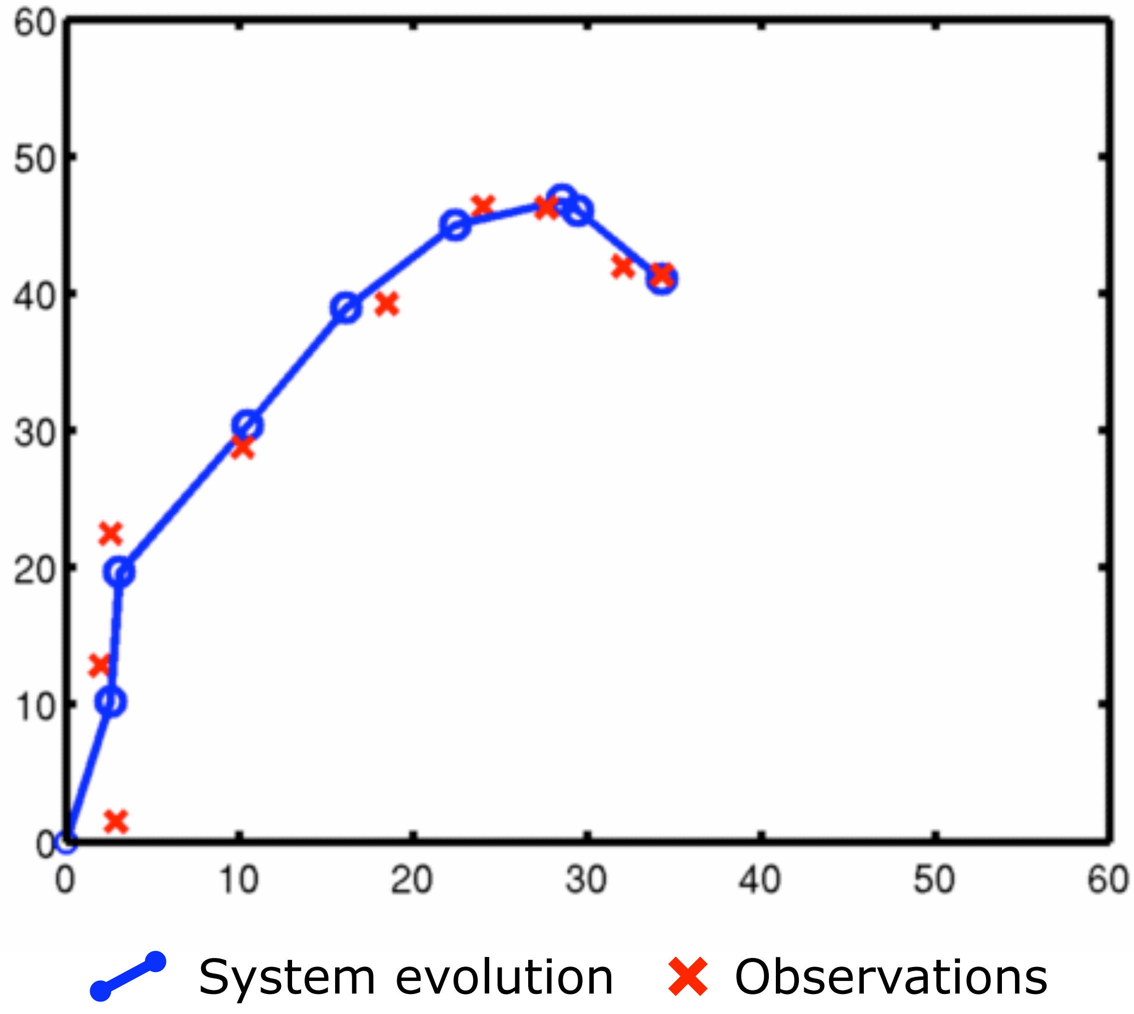
LDS Example – Throwing ball



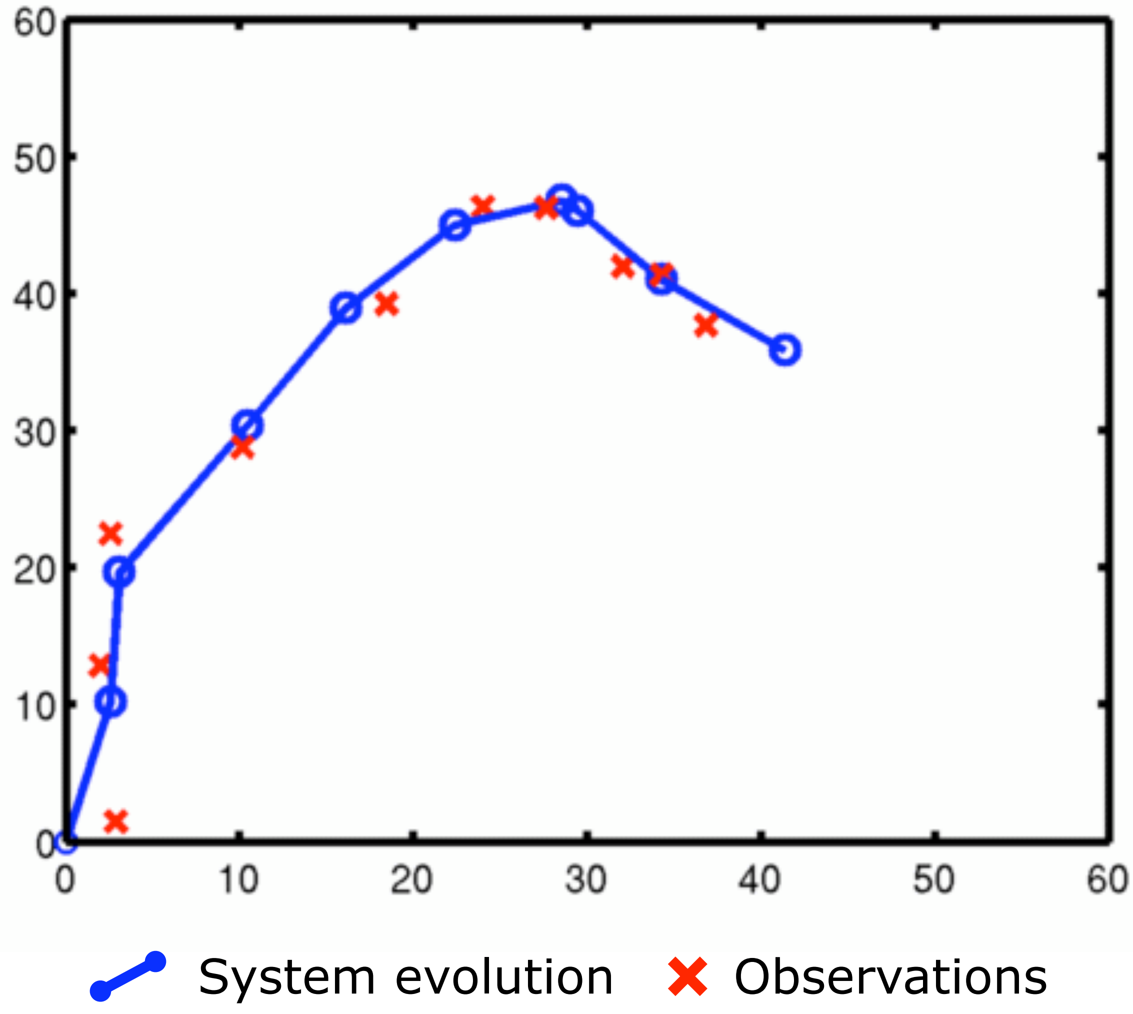
LDS Example – Throwing ball



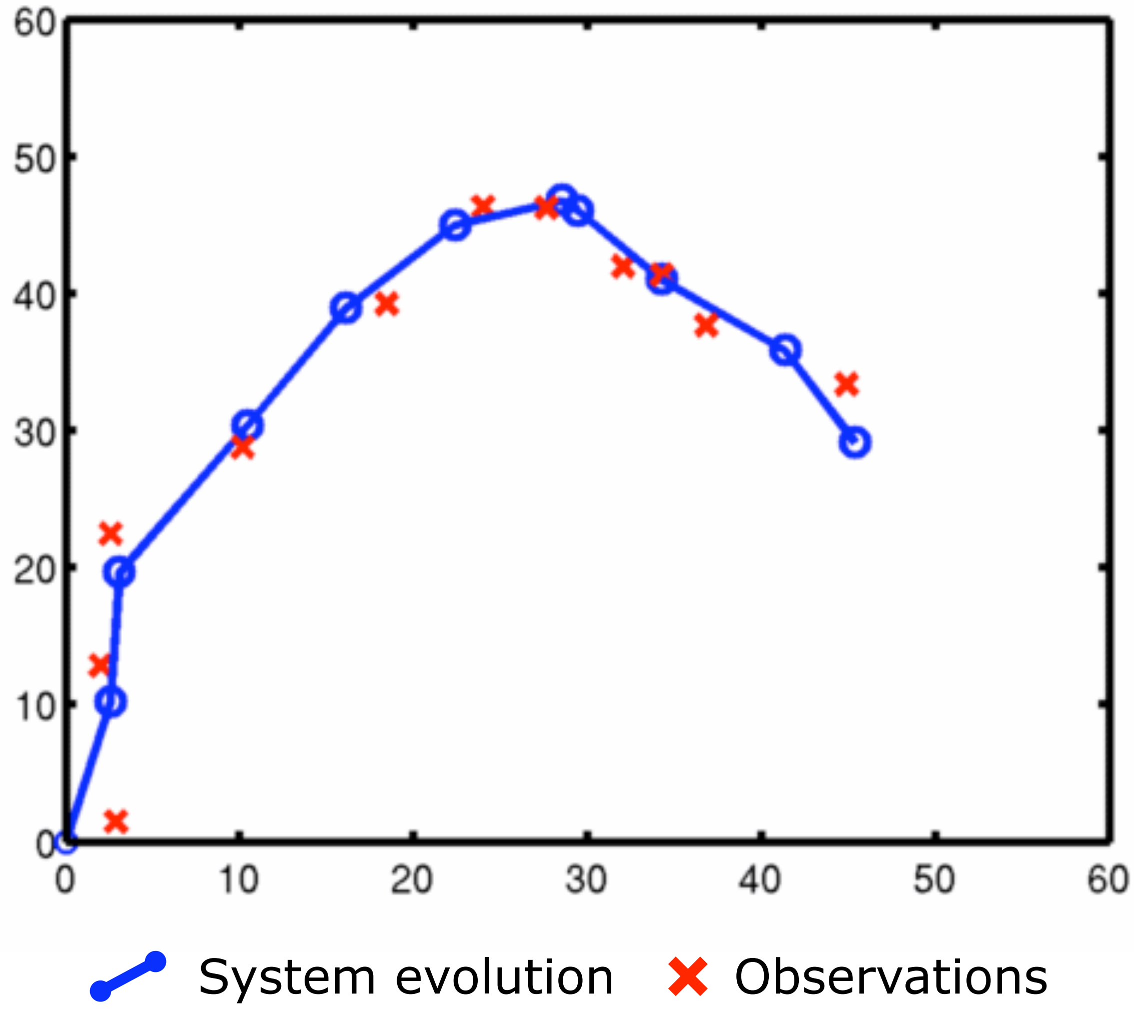
LDS Example – Throwing ball



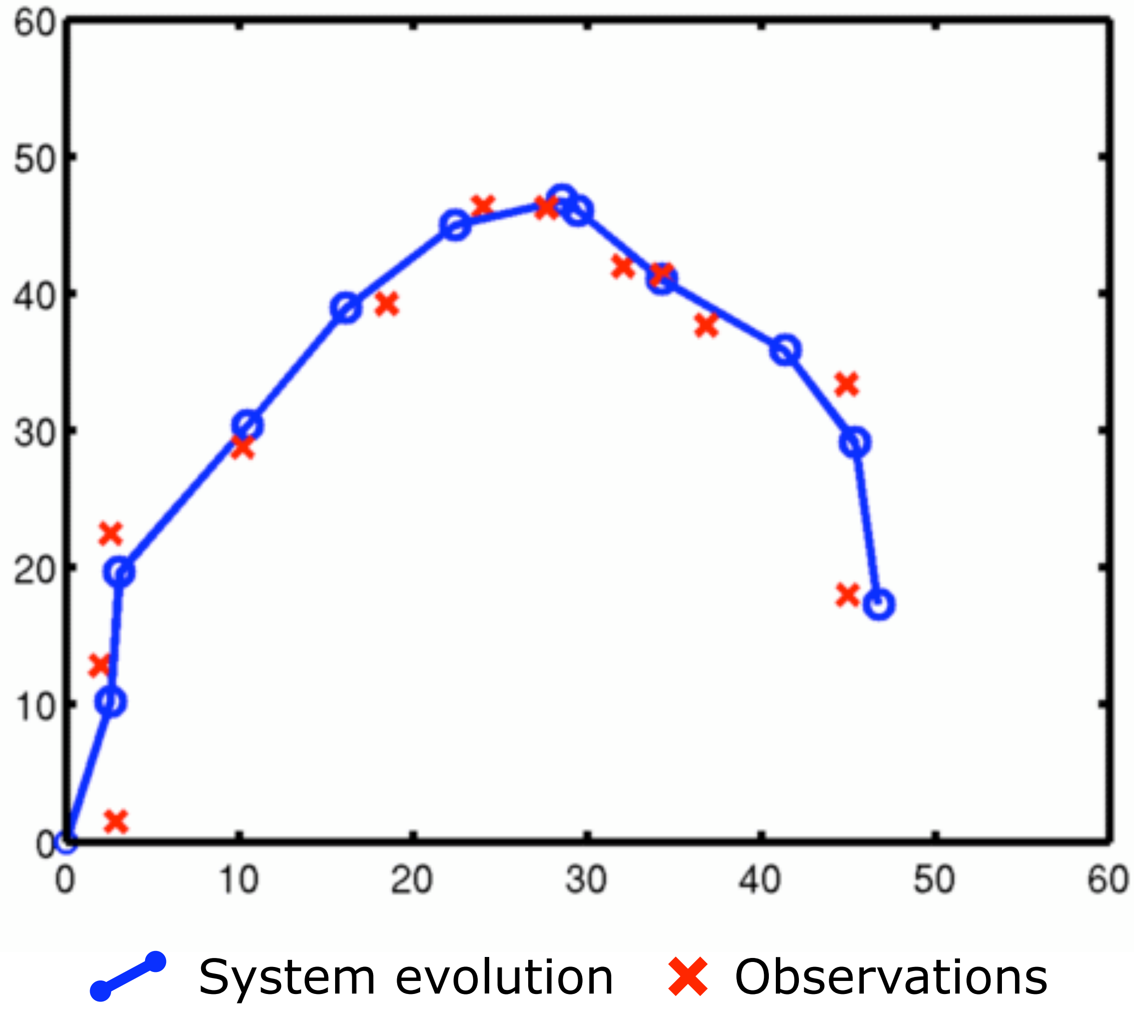
LDS Example – Throwing ball



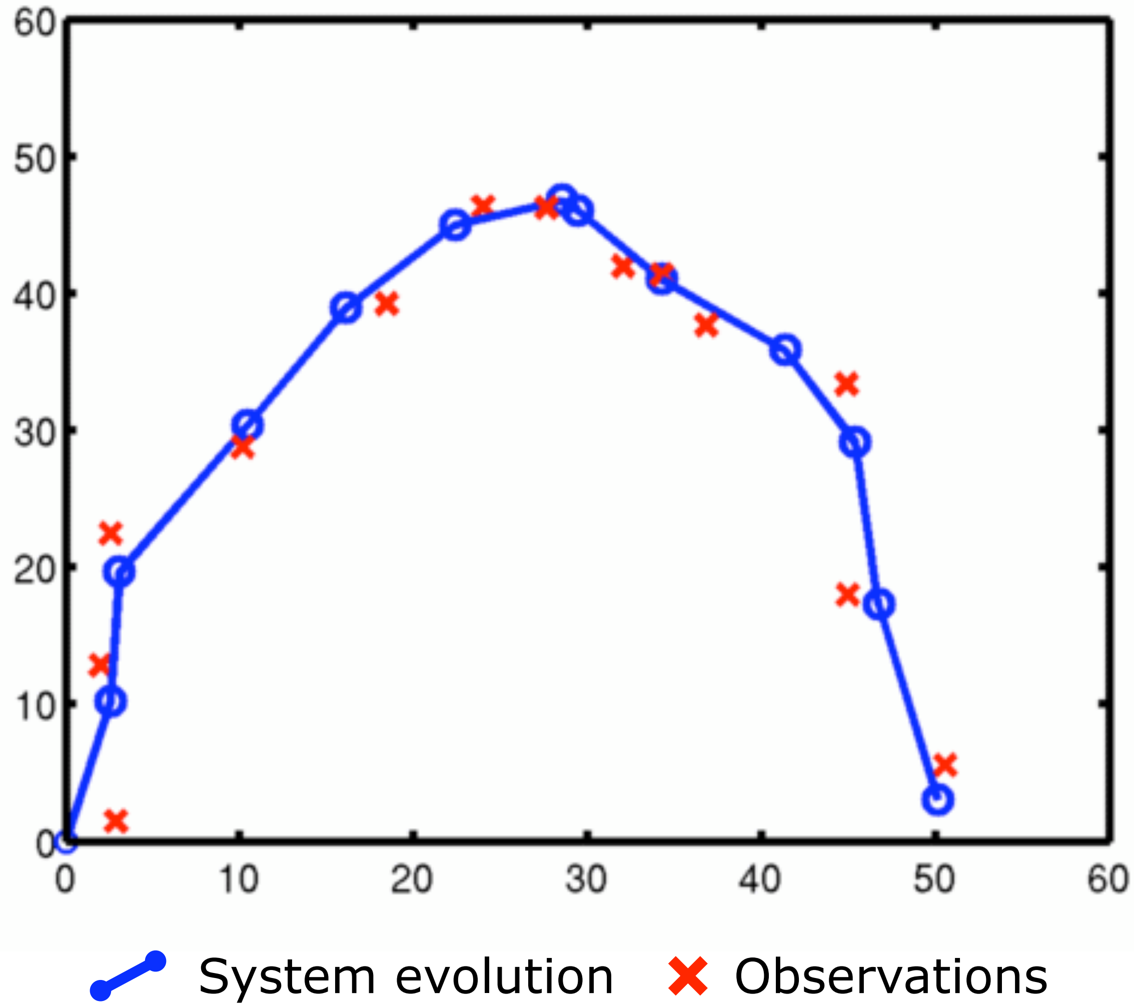
LDS Example – Throwing ball



LDS Example – Throwing ball



LDS Example – Throwing ball



Kalman Filter (KF)

- The Kalman filter is the **workhorse of tracking**
- Under linear Gaussian assumptions, the KF is the **optimal** minimum mean squared error (MMSE) estimator. It is still the **optimal linear** MMSE estimator if these conditions are not met
- An "**optimal**" estimator is an algorithm that processes observations to yield a state estimate that **minimizes** a certain error criterion (e.g. RMS, MSE)
- It is basically a (recursive) **weighted sum** of the prediction and observation. The weights are given by the **process** and the **measurement covariances**
- See literature for detailed tutorials

Kalman Filter (KF)

- Consider a discrete time LDS with **dynamic model**

$$x(k+1) = F(k)x(k) + \xi(k)$$

  where $\xi(k)$ is the process noise (no input assumed)

$$\xi(k) \sim \mathcal{N}(0, Q(k))$$

- The **measurement model** is

$$z(k) = H(k)x(k) + \epsilon(k)$$

  where $\epsilon(k)$ is the measurement noise

$$\epsilon(k) \sim \mathcal{N}(0, R(k))$$

- The **initial state** is generally unknown and modeled as a Gaussian random variable

$$\hat{x}(0|0) = x_0 \quad \textbf{State estimate}$$

$$\hat{P}(0|0) = P_0 \quad \textbf{Covariance estimate}$$

Kalman Filter

- State Prediction

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k)$$

$$\hat{P}(k+1|k) = F(k)\hat{P}(k|k)F^T(k) + Q(k)$$

- Measurement Prediction

$$\hat{z}(k) = H(k)\hat{x}(k+1|k)$$

$$\hat{S}(k) = H(k)\hat{P}(k+1|k)H^T(k) + R(k)$$

- Update

$$K(k) = \hat{P}(k+1|k)H^T(k)\hat{S}(k)^{-1}$$

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K(k)\nu(k)$$

$$\hat{P}(k+1|k+1) = (I - K(k)H(k))\hat{P}(k+1|k)$$

EKF: Error Propagation

- **Error Propagation** is everywhere in Kalman filtering
  - From the uncertain previous state to the next state over the system dynamics

$$\hat{P}(k+1|k) = \boxed{F(k)\,\hat{P}(k|k)\,F^T(k)} + E(k)\,U(k)\,E(k)^T + G(k)\,A(k)\,G(k)^T$$

  - From the uncertain inputs to the state over the input gain relationship

$$\hat{P}(k+1|k) = F(k)\,\hat{P}(k|k)\,F^T(k) + \boxed{E(k)\,U(k)\,E(k)^T} + G(k)\,A(k)\,G(k)^T$$

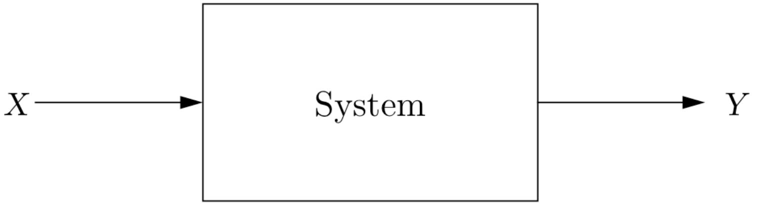
  - From the uncertain predicted state to the predicted measurements over the measurement model

$$\hat{S}(k+1) = \boxed{H(k)\,\hat{P}(k+1|k)\,H^T(k)} + R(k)$$

Error Propagation Law

Given

- A linear system $Y = F_X \cdot X$
  X, Y assumed to be Gaussian
- Input covariance matrix C_X
- System matrix F_X



the **Error Propagation Law**

$$\boxed{C_Y = F_X C_X F_X^T}$$

computes the **output covariance matrix** C_Y

Error Propagation Law

- **Derivation in Matrix Notation**

Blackboard...

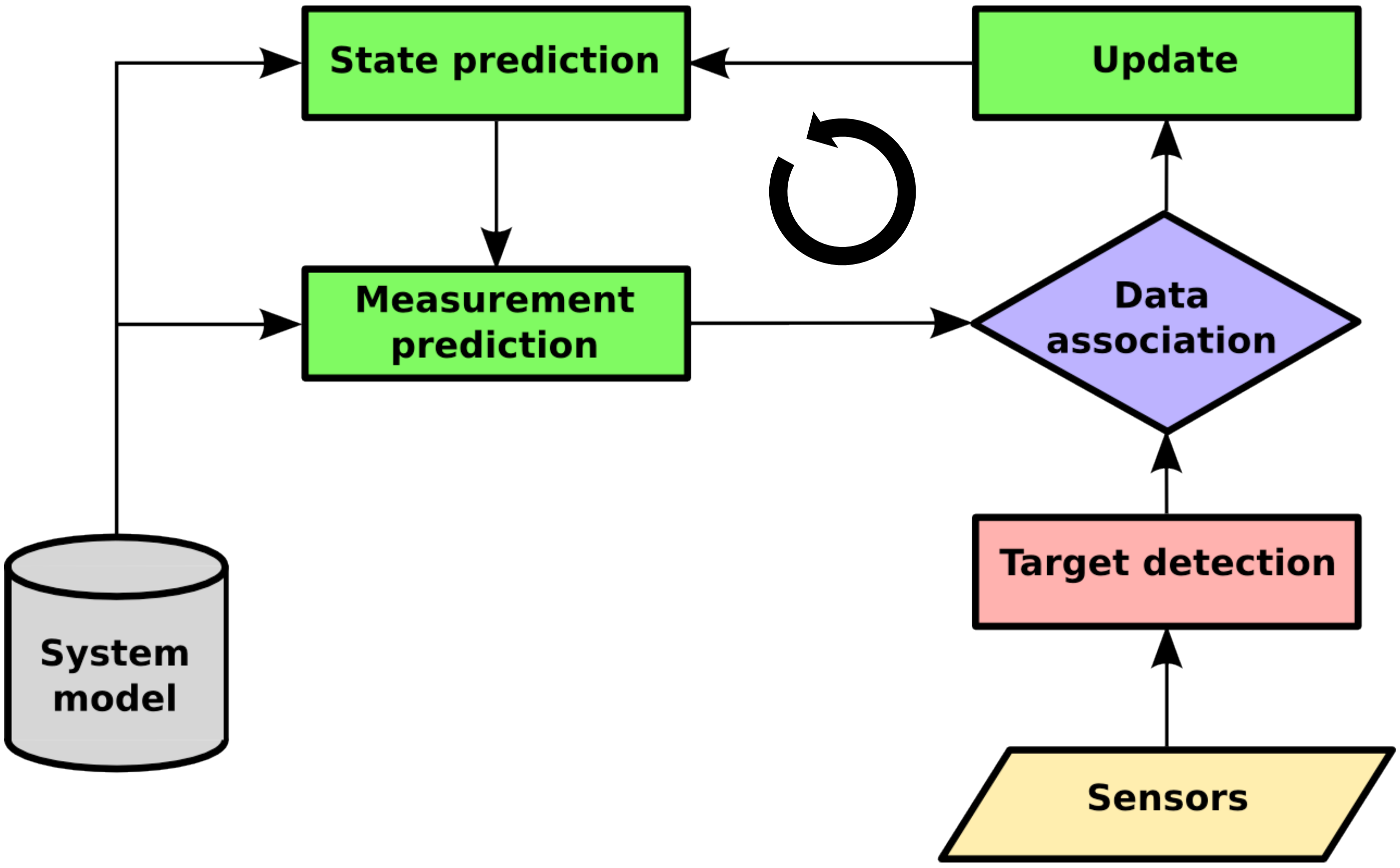
Error Propagation Law

- **Derivation in Matrix Notation**

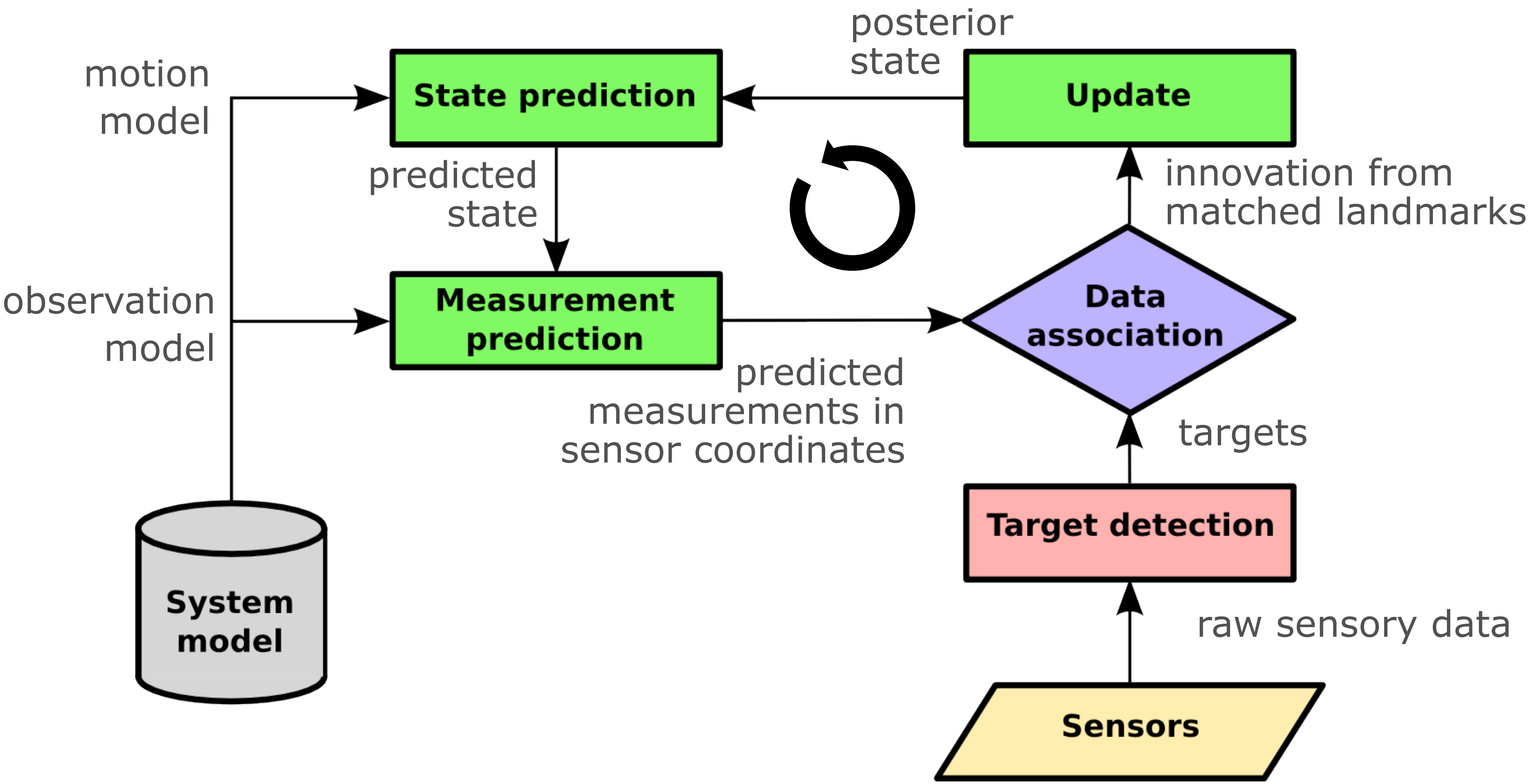
$$\begin{aligned}
\mu_x &= E(x) \\
&= E(Au + b) \\
&= AE(u) + b \\
&= A\mu_u + b
\end{aligned}$$

$$\begin{aligned}
\Sigma_x &= E((x - E(x))(x - E(x))^T) \\
&= E((Au + b - AE(u) - b)(Au + b - AE(u) - b)^T) \\
&= E((A(u - E(u)))(A(u - E(u)))^T) \\
&= E((A(u - E(u)))((u - E(u))^T A^T)) \\
&= AE((u - E(u))(u - E(u))^T)A^T \\
&= A\Sigma_u A^T
\end{aligned}$$

Kalman Filter Cycle



Kalman Filter Cycle



KF Cycle 1/4: State prediction

- **State prediction**

$$\widehat{x}(k+1|k) = F(k)\widehat{x}(k|k)$$
$$\widehat{P}(k+1|k) = F(k)\widehat{P}(k|k)F^T(k) + Q(k)$$

- In target tracking, **no a priori knowledge** of the dynamic equation is generally available
- Instead, different **motion models** (MM) are used
  - Brownian MM
  - Constant velocity MM
  - Constant acceleration MM
  - Constant turn MM
  - Specialized models (problem-related, e.g. ship models)

Motion Models: Brownian

No-motion assumption

- Useful to describe stop-and-go motion behavior

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$$

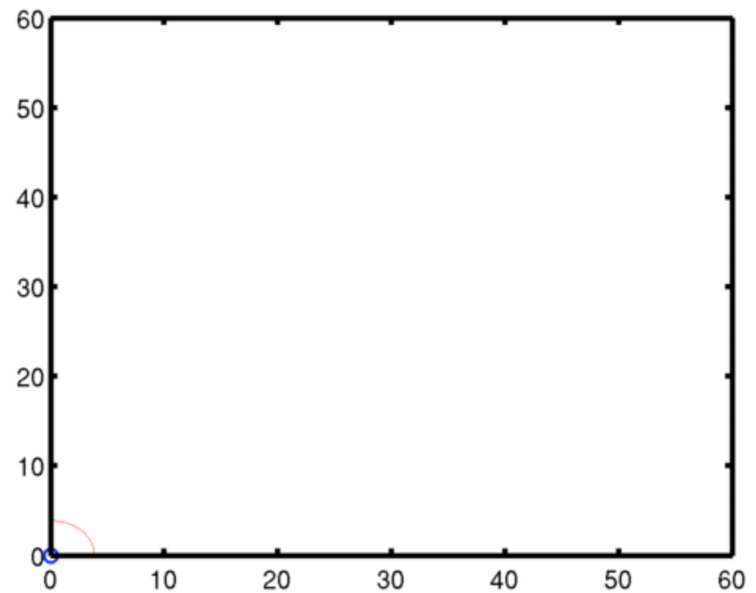
- Initial state

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ball example



Motion Models: Brownian

No-motion assumption

- Useful to describe stop-and-go motion behavior

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$$

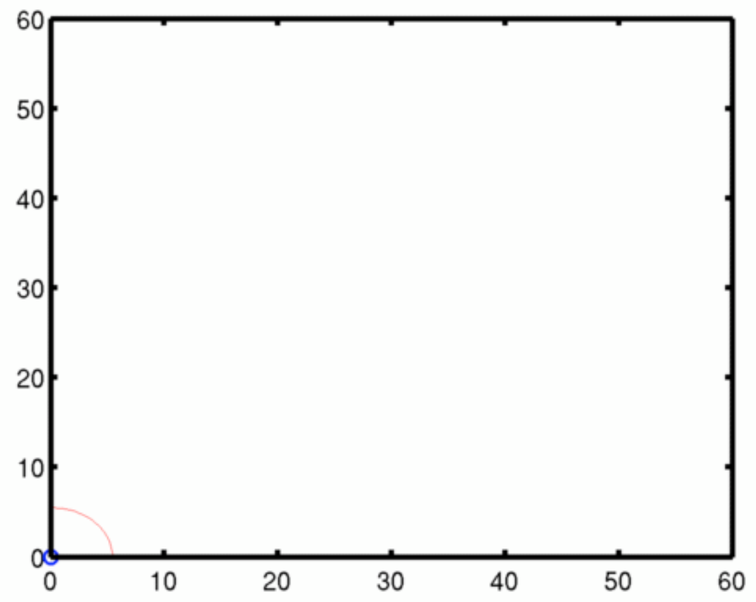
- Initial state

$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Uncertainty grows**

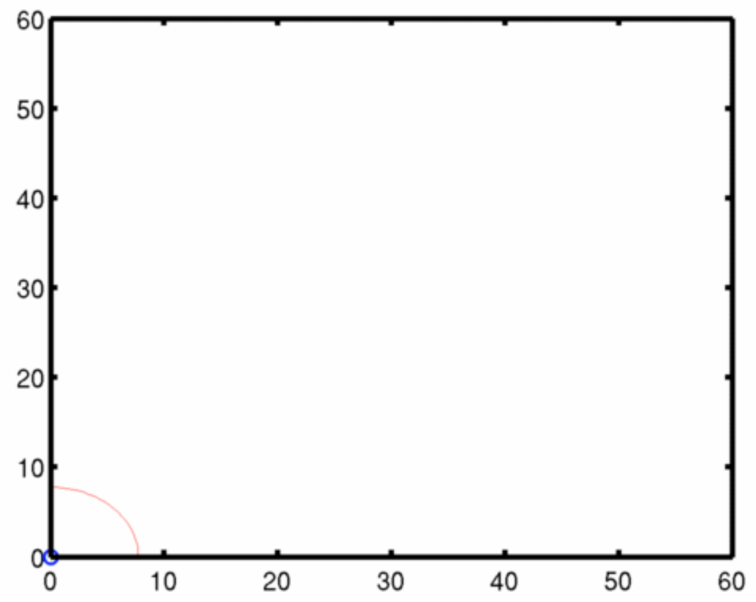
Motion Models: Brownian

No-motion assumption

- Useful to describe stop-and-go motion behavior

- State representation
$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$$
- Initial state
$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$
- Transition matrix
$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Uncertainty grows**

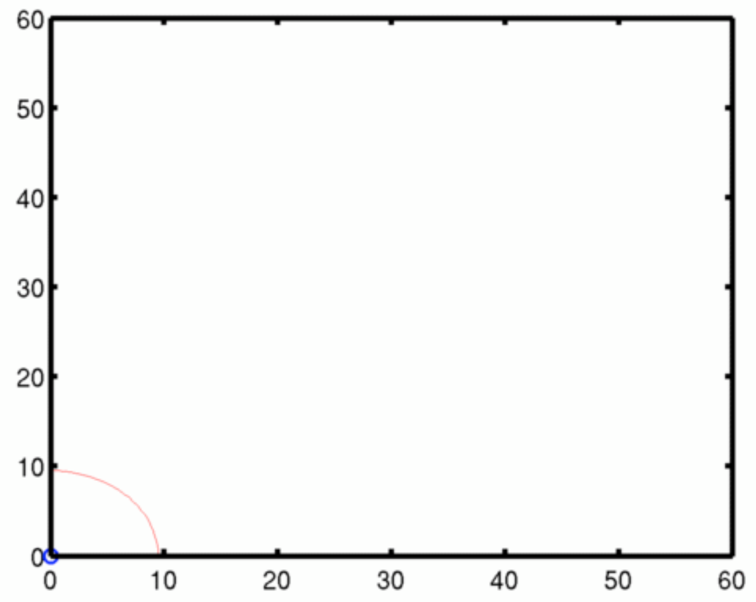
Motion Models: Brownian

No-motion assumption

- Useful to describe stop-and-go motion behavior

- State representation
$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$$
- Initial state
$$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$
- Transition matrix
$$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Uncertainty grows**

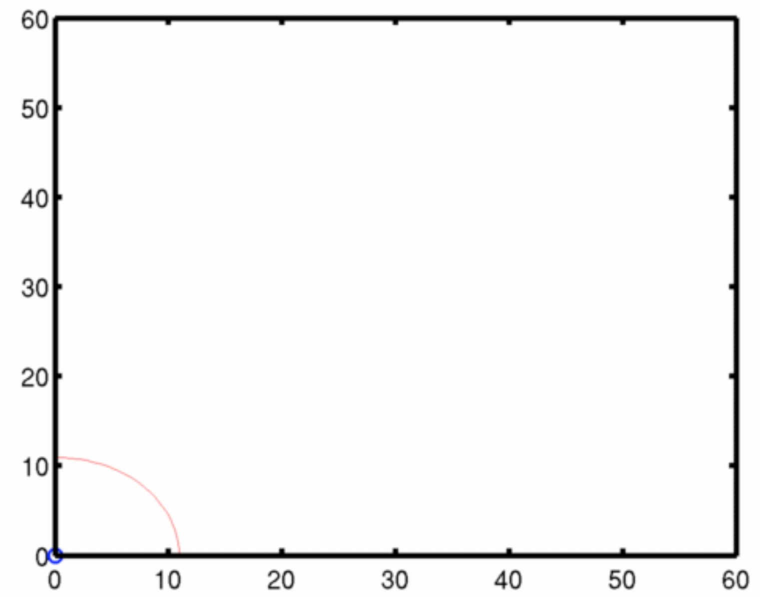
Motion Models: Brownian

No-motion assumption

- Useful to describe stop-and-go motion behavior

- State representation
  $$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^T$$
- Initial state
  $$\mathbf{x}_0 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$
- Transition matrix
  $$F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

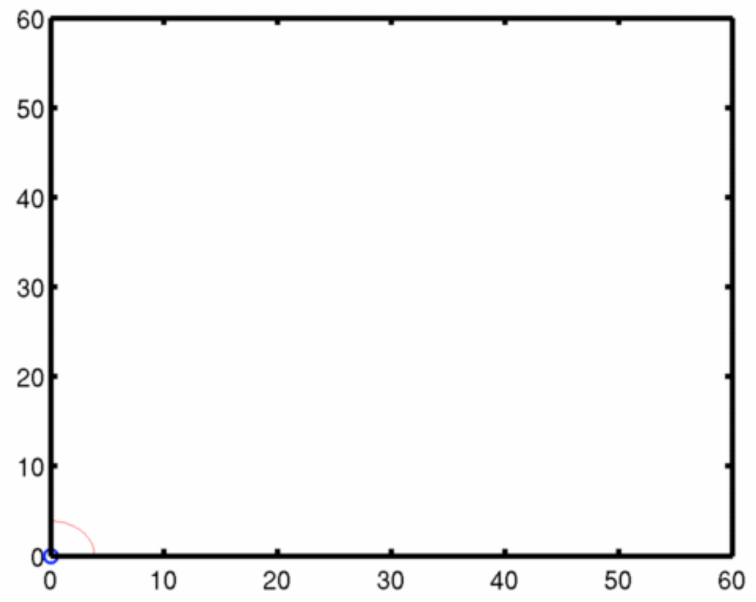
- **Uncertainty grows**

MMs: Constant Velocity

Constant target velocity assumption

- Useful to model smooth target motion

Ball example



- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$$

- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MMs: Constant Velocity

Constant target velocity assumption

- Useful to model smooth target motion

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$$

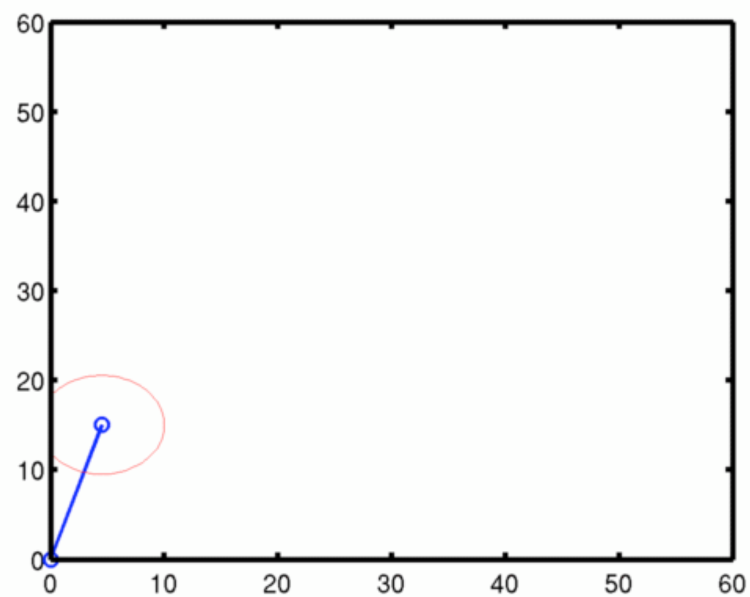
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Linear target motion**
- **Uncertainty grows**

MMs: Constant Velocity

Constant target velocity assumption

- Useful to model smooth target motion

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$$

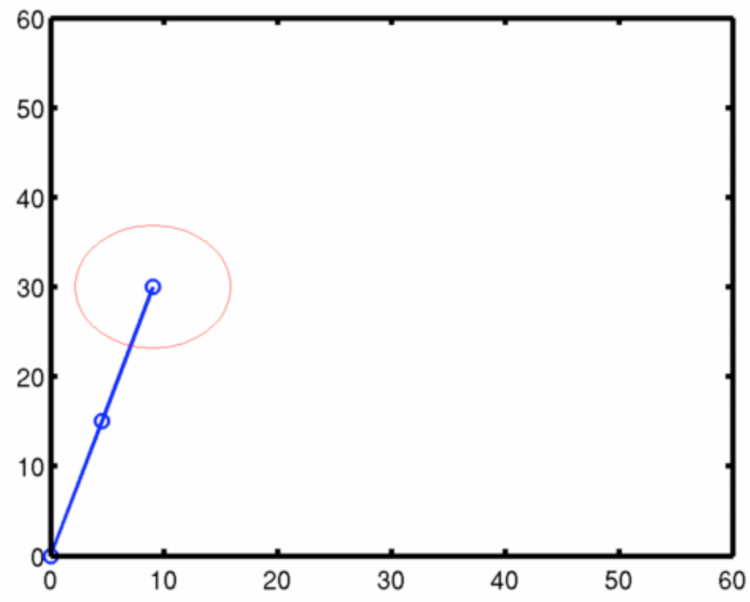
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Linear target motion**
- **Uncertainty grows**

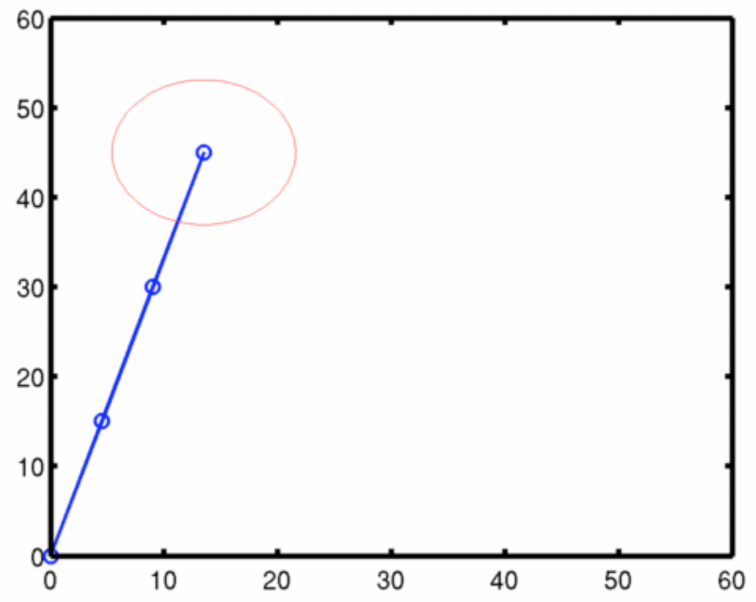
MMs: Constant Velocity

Constant target velocity assumption

- Useful to model smooth target motion

- State representation
$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$$
- Initial state
$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 \end{bmatrix}^T$$
- Transition matrix
$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:
- **Linear target motion**
- **Uncertainty grows**

MMs: Constant Velocity

Constant target velocity assumption

- Useful to model smooth target motion

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} \end{bmatrix}^T$$

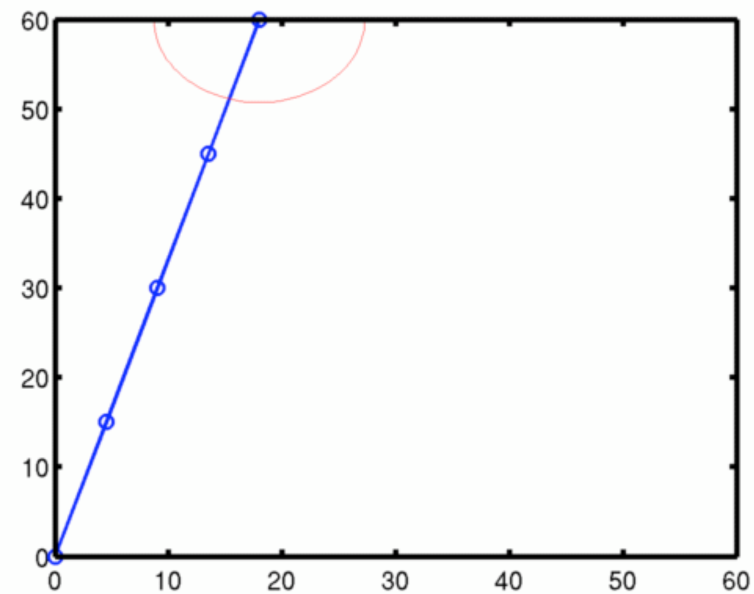
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

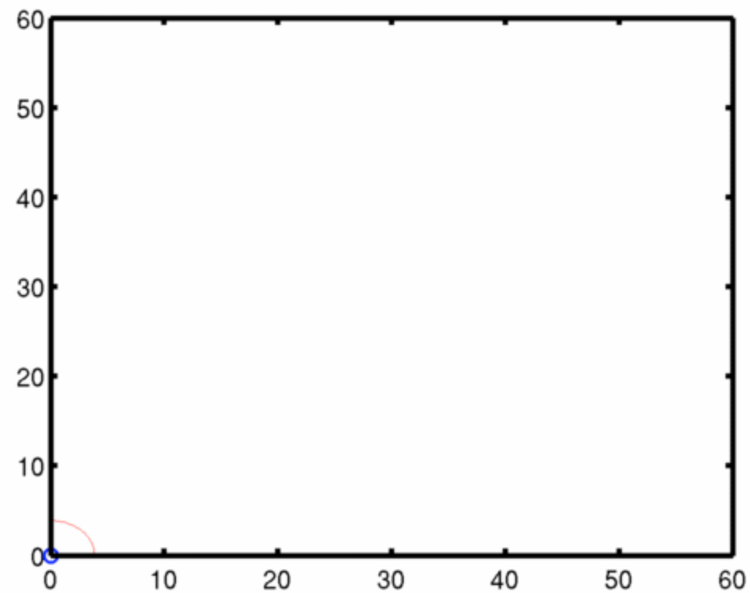
- **Linear target motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

Ball example



- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 & 0 & -g \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

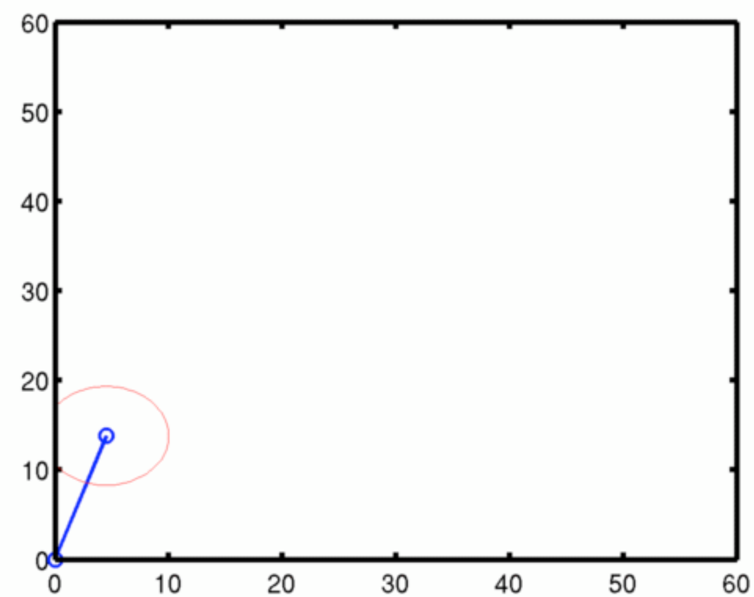
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 & 0 & -g \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Non-linear motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

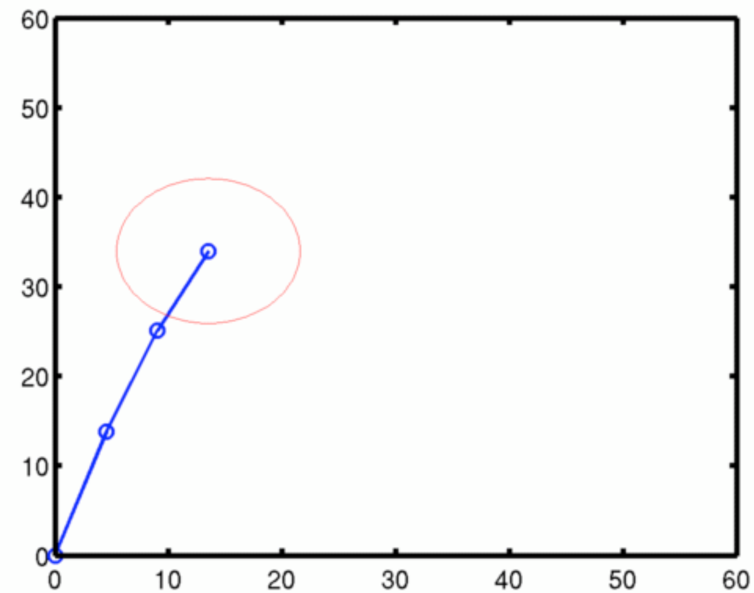
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 & 0 & -g \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Non-linear motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

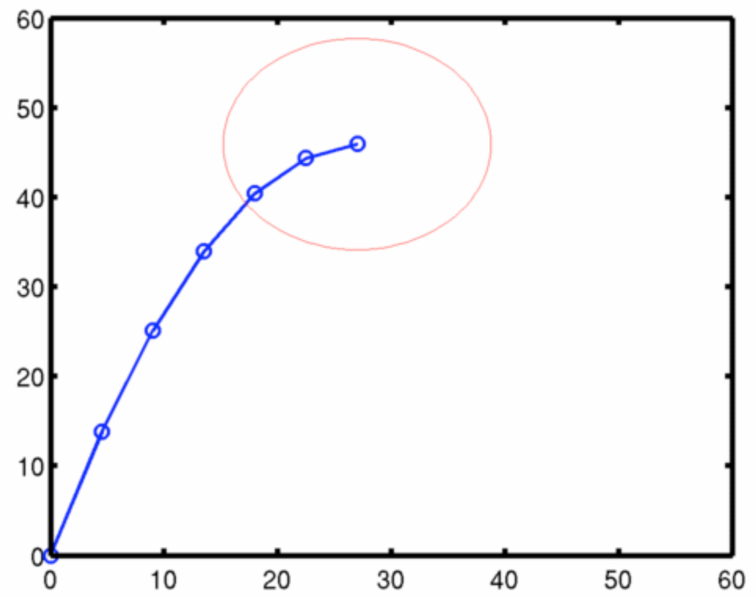
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 & 0 & -g \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Non-linear motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

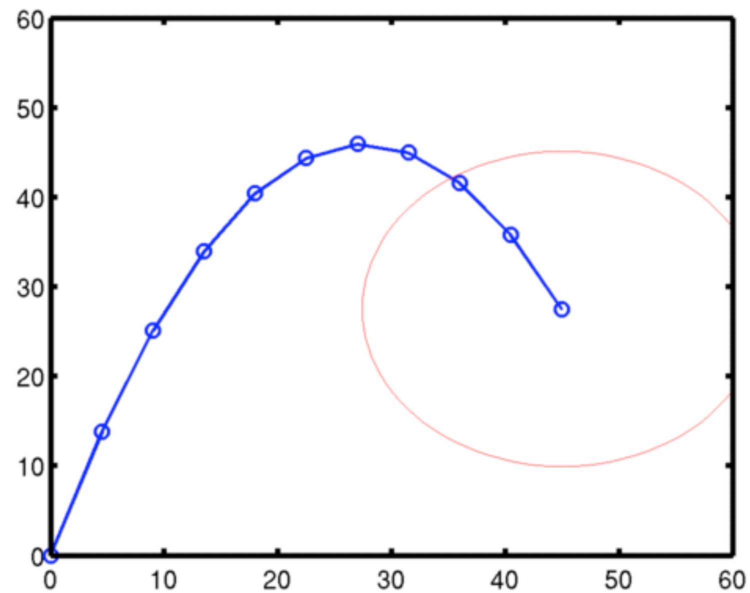
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 9 & 30 & 0 & -g \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Non-linear motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

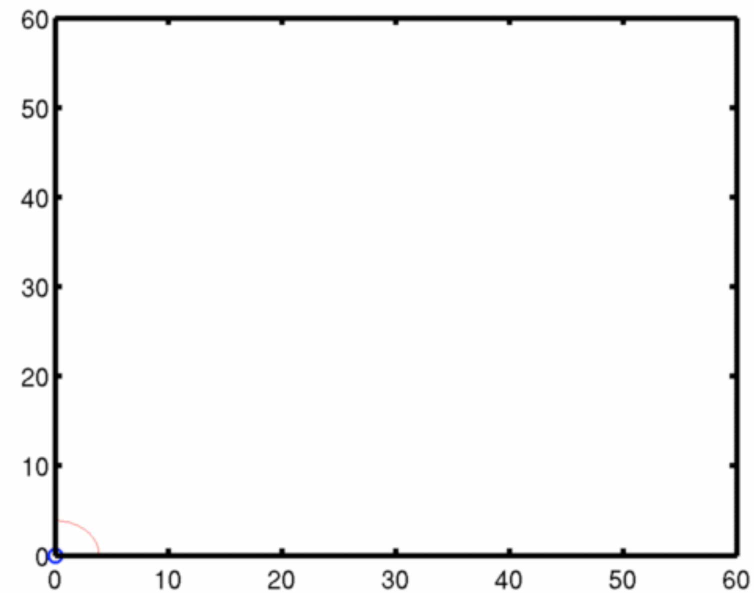
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 30 & 30 & -20 & -12 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

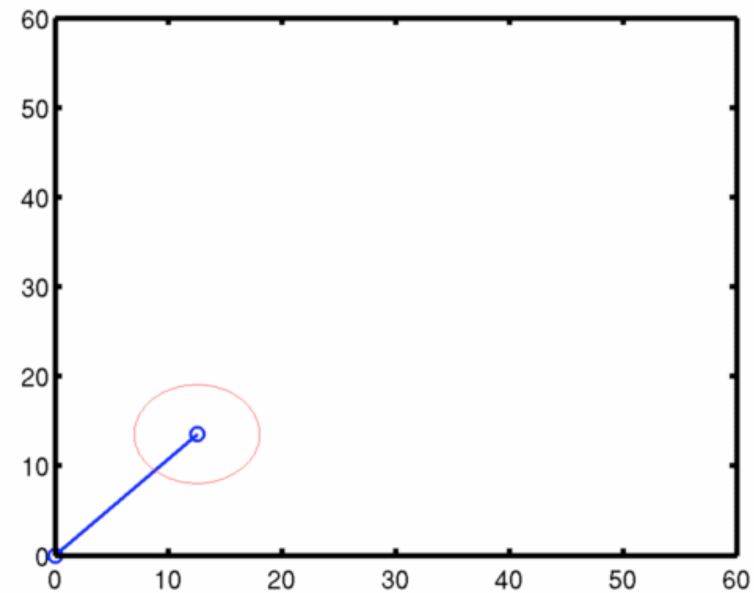
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 30 & 30 & -20 & -12 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Non-linear motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes
- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

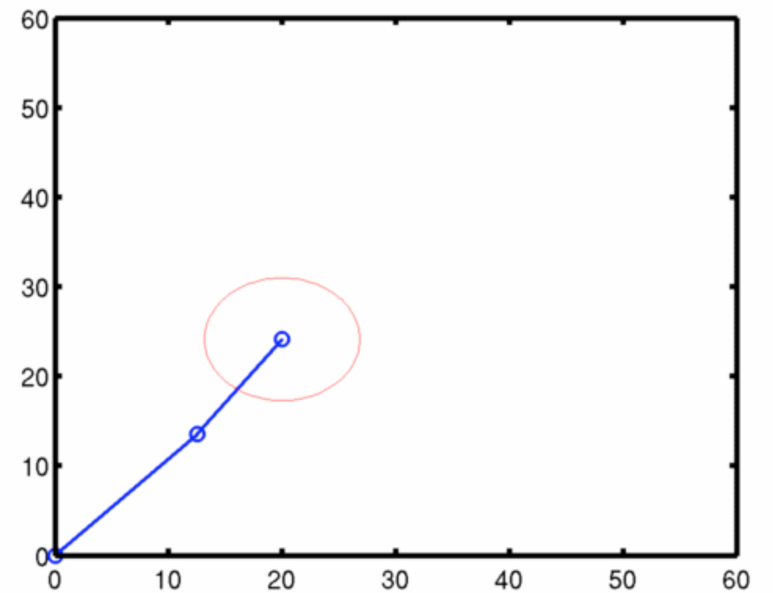
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 30 & 30 & -20 & -12 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

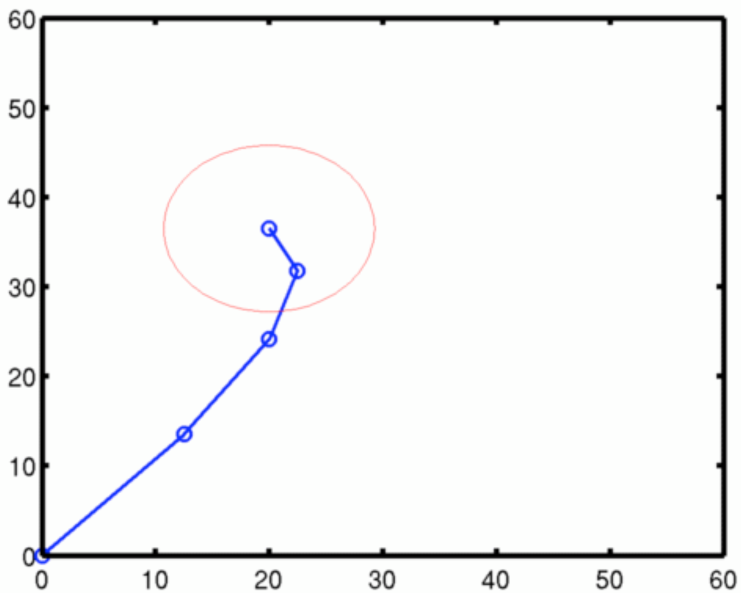
- **Non-linear motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

Ball example



- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 30 & 30 & -20 & -12 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

State prediction:

- **Non-linear motion**
- **Uncertainty grows**

MMs: Constant Acceleration

Constant target acceleration assumed

- Useful to model target motion that is smooth in position and velocity changes

- State representation

$$\mathbf{x} = \begin{bmatrix} x & y & \dot{x} & \dot{y} & \ddot{x} & \ddot{y} \end{bmatrix}^T$$

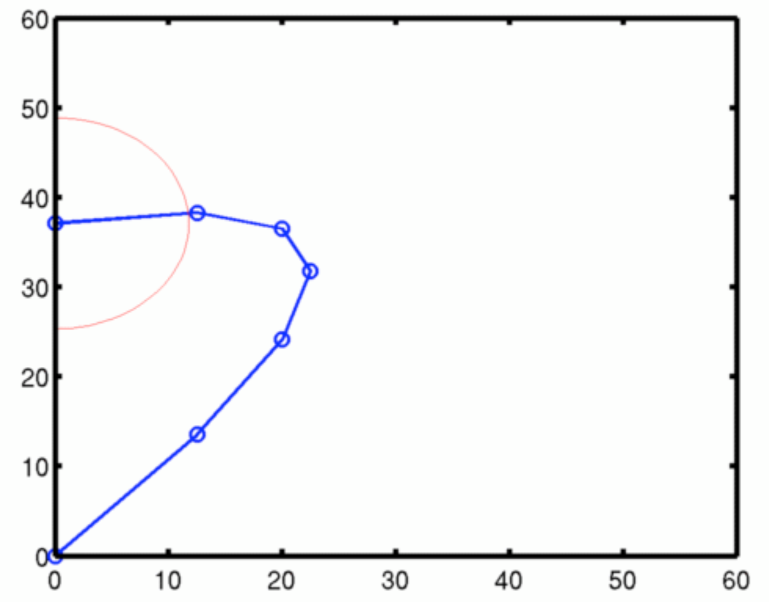
- Initial state

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & 30 & 30 & -20 & -12 \end{bmatrix}^T$$

- Transition matrix

$$F = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ball example



State prediction:

- **Non-linear motion**
- **Uncertainty grows**

KF Cycle 2/4: Meas. Prediction

- **Measurement prediction**

$$\hat{z}(k) = H(k)\hat{x}(k+1|k)$$

$$\hat{S}(k) = H(k)\hat{P}(k+1|k)H^T(k) + R(k)$$

- **Observation**
  Typically, only the target **position** is observed. The measurement matrix is then

$$\mathbf{z} = \begin{bmatrix} x & y \end{bmatrix}^T \qquad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

  Note: One can also observe
  - Velocity (Doppler radar)
  - Acceleration (accelerometers)

KF Cycle 3/4: Data Association

- Once measurements are predicted and observed, we have to **associate them with each other**
- This is resolving the **origin uncertainty** of observations
- Data association is typically done in the **sensor reference frame**
- Data association can be a **hard problem** and many advanced techniques exist

  ⇨ More on this later in this course

KF Cycle 3/4: Data Association

Step 1: Compute the pairing difference and its associated uncertainty

- The difference between predicted measurement and observation is called **innovation**

$$\nu_{ij}(k) \;=\; z_i(k) - \widehat{z}_j(k)$$

- The associated covariance estimate is called the **innovation covariance**

$$\widehat{S}_{ij}(k) \;=\; H(k)\widehat{P}_j(k+1|k)H^T(k) + R_i(k)$$

- The prediction-observation pair is often called **pairing**

KF Cycle 3/4: Data Association

Step 2: Check if the pairing is statistically compatible

- Compute the **Mahalanobis distance**

$$d_{ij}^2 \;=\; \nu_{ij}(k)^T \widehat{S}_{ij}(k)^{-1} \nu_{ij}(k)$$

- Compare it against the proper threshold from an cumulative χ^2 (“**chi square**”) **distribution**

$$d_{ij} \;\leq\; \chi^2_{n,\alpha}$$

Significance level

Degrees of freedom

Compatibility on level α is finally given if this is true

KF Cycle 3/4: Data Association

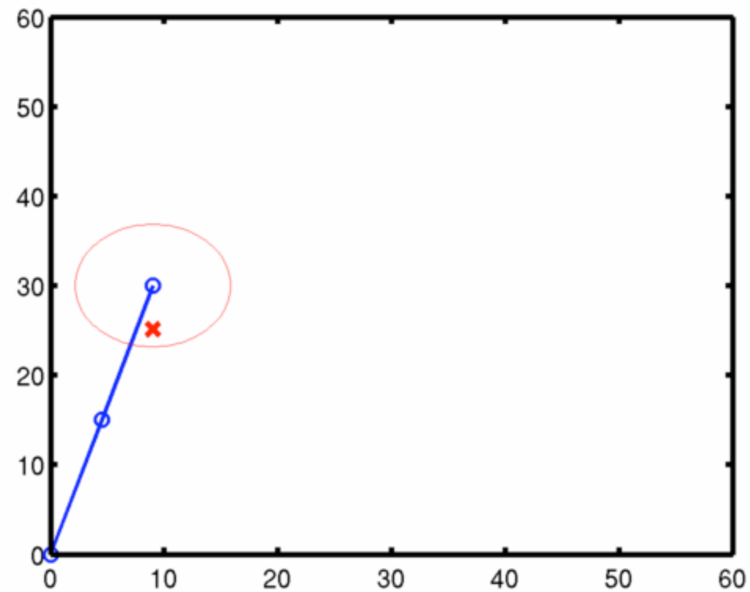
- Constant velocity model
- Process noise

$$Q = \begin{bmatrix} 5.0 & 0 & 0 & 0 \\ 0 & 5.0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

- Measurement noise

$$R = \begin{bmatrix} 10.0 & 0 \\ 0 & 10.0 \end{bmatrix}$$

- No false alarm



➔ **No problem**

KF Cycle 3/4: Data Association

- Constant velocity model
- Process noise

$$Q = \begin{bmatrix} 5.0 & 0 & 0 & 0 \\ 0 & 5.0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

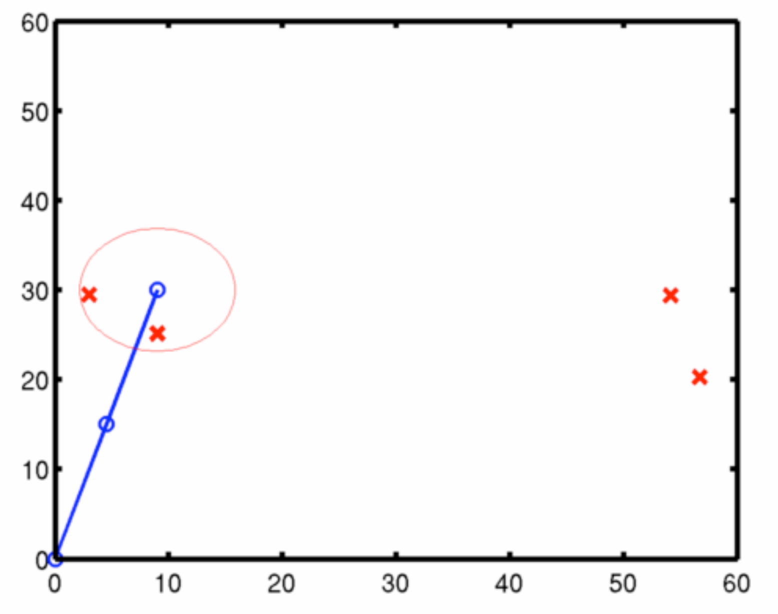
- Measurement noise

$$R = \begin{bmatrix} 10.0 & 0 \\ 0 & 10.0 \end{bmatrix}$$

- Uniform false alarm

$$x \sim \mathcal{U}(0,60), \quad y \sim \mathcal{U}(0,60)$$

- False alarm rate = 3



➔ **Ambiguity**: several observations in the validation gate

KF Cycle 3/4: Data Association

- Constant velocity model
- Process noise

$$Q = \begin{bmatrix} 5.0 & 0 & 0 & 0 \\ 0 & 5.0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

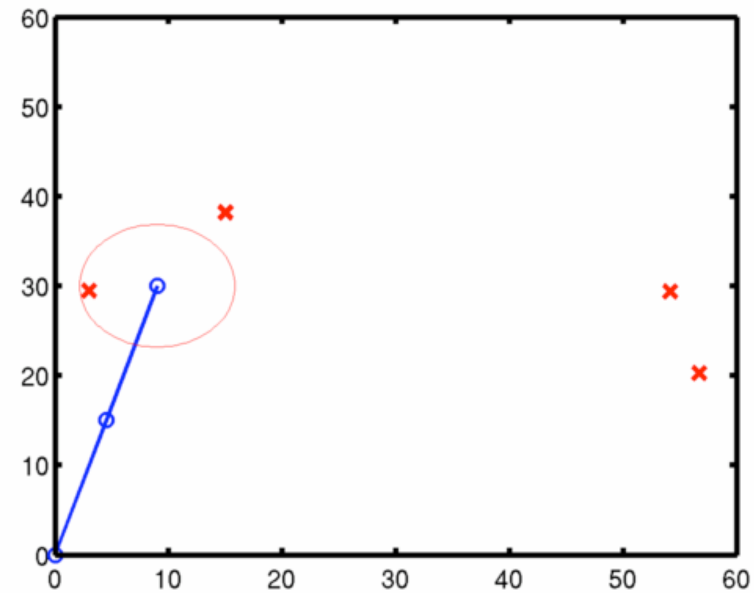
- Measurement noise

$$R = \begin{bmatrix} 50.0 & 0 \\ 0 & 50.0 \end{bmatrix}$$

- Uniform false alarm

$$x \sim \mathcal{U}(0, 60), \quad y \sim \mathcal{U}(0, 60)$$

- False alarm rate = 3



→ **Wrong association** as closest observation is false alarm

KF Cycle 4/4: Update

- Computation of the **Kalman gain**

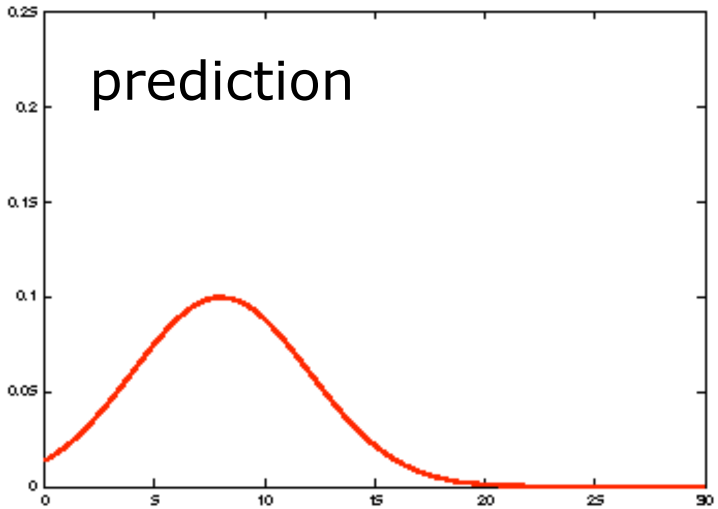
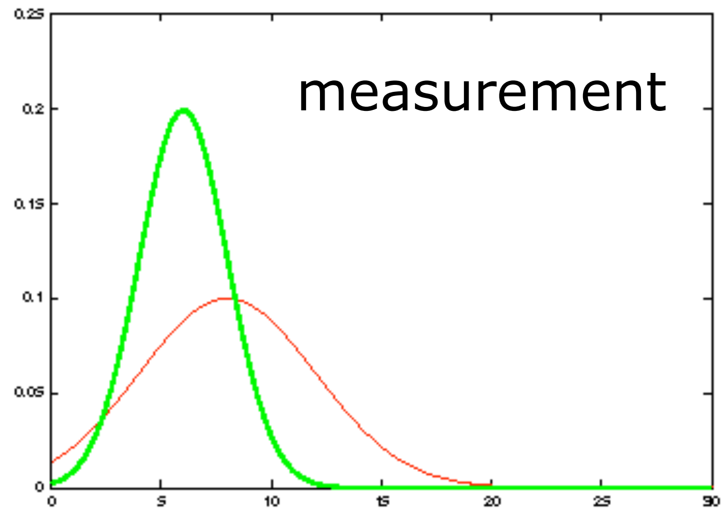
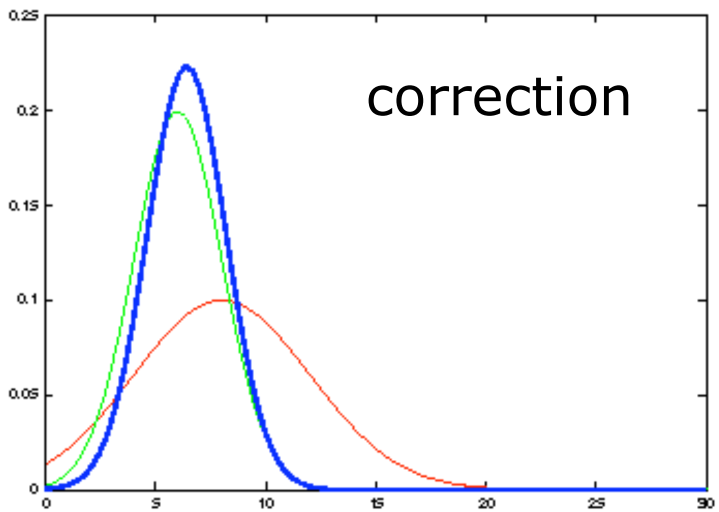
$$K(k) = \widehat{P}(k+1|k)H^T(k)\widehat{S}(k)^{-1}$$

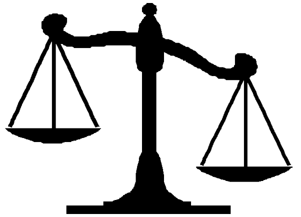
- State and state covariance **update**

$$\widehat{x}(k+1|k+1) = \widehat{x}(k+1|k) + K(k)\nu(k)$$

$$\widehat{P}(k+1|k+1) = (I - K(k)H(k))\widehat{P}(k+1|k)$$

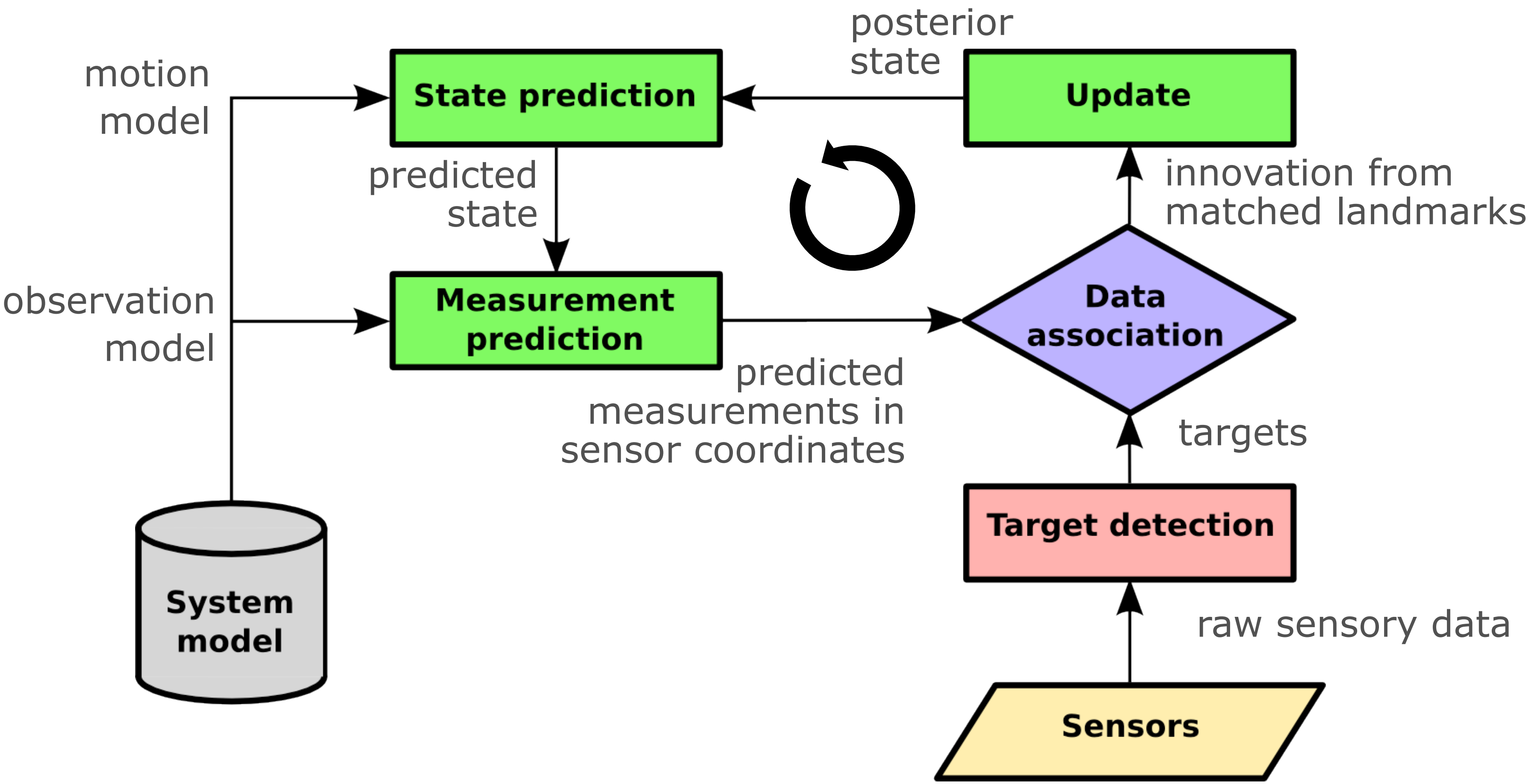
KF Cycle 4/4: Update



It's a weighted mean!

Kalman Filter Cycle



Kalman Filter: Limitations

- **Non-linear motion** models and/or **non-linear measurement** models
  - Extended Kalman filter
- **Unknown inputs** into the dynamic process model (values and modes)
  - Enlarged process noise (simple but there are implications)
  - **Multiple model** approaches (accounts for mode changes)
- **Uncertain origin** of measurements
  - Data Association
- How **many** targets are there?
  - Track formation and deletion techniques
  - Multiple Hypothesis Tracker (MHT)

Extended Kalman Filter

- The **Extended Kalman filter** deals with **non-linear process** and **non-linear measurement models**

- Consider a discrete time LDS with **dynamic model**

  $$x(k+1) = f(k, x(k)) + \xi(k)$$

  where $\xi(k)$ is the process noise (no input assumed)

- The **measurement model** is

  $$z(k) = h(k, x(k)) + \epsilon(k)$$

  where $\epsilon(k)$ is the measurement noise

- The same KF-assumptions for the initial state

Kalman Filter

- State Prediction

$$\widehat{x}(k+1|k) = F(k)\widehat{x}(k|k)$$
$$\widehat{P}(k+1|k) = F(k)\widehat{P}(k|k)F^T(k) + Q(k)$$

- Measurement Prediction

$$\widehat{z}(k) = H(k)\widehat{x}(k+1|k)$$
$$\widehat{S}(k) = H(k)\widehat{P}(k+1|k)H^T(k) + R(k)$$

- Update

$$K(k) = \widehat{P}(k+1|k)H^T(k)\widehat{S}(k)^{-1}$$
$$\widehat{x}(k+1|k+1) = \widehat{x}(k+1|k) + K(k)\nu(k)$$
$$\widehat{P}(k+1|k+1) = (I - K(k)H(k))\widehat{P}(k+1|k)$$

Extended Kalman Filter

- State Prediction

$$\hat{x}(k+1|k) = f(k, \hat{x}(k|k))$$

Jacobian

$$\widehat{P}(k+1|k) = F(k)\widehat{P}(k|k)F^T(k) + Q(k)$$

- Measurement Prediction

$$\hat{z}(k) = h(k, \hat{x}(k+1|k))$$

Jacobian

$$\widehat{S}(k) = H(k)\widehat{P}(k+1|k)H^T(k) + R(k)$$

- Update

$$K(k) = \widehat{P}(k+1|k)H^T(k)\widehat{S}(k)^{-1}$$

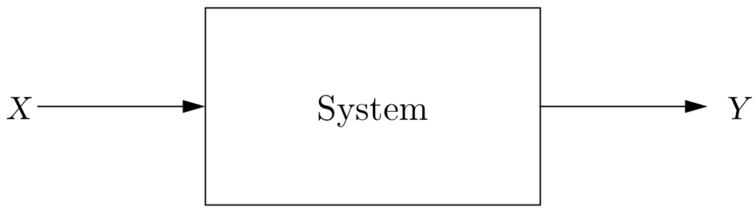
$$\widehat{x}(k+1|k+1) = \widehat{x}(k+1|k) + K(k)\nu(k)$$

$$\widehat{P}(k+1|k+1) = (I - K(k)H(k))\widehat{P}(k+1|k)$$

First-Order Error Propagation

Given

- A non-linear system $Y = f(X)$
  X, Y assumed to be Gaussian
- Input covariance matrix C_X
- Jacobian matrix F_X



the **Error Propagation Law**

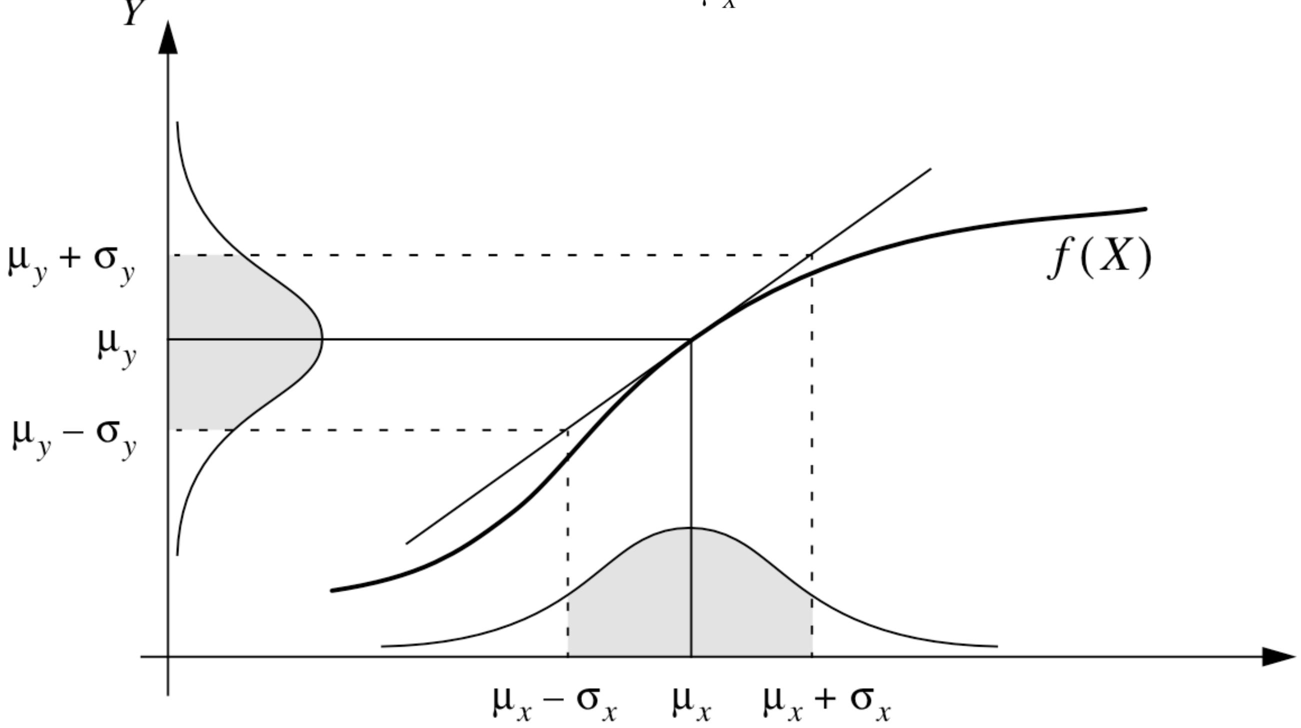
$$\boxed{C_Y = F_X C_X F_X^T}$$

computes the **output covariance matrix** C_Y

First-Order Error Propagation

- Approximating $f(X)$ by a **first-order** Taylor series expansion about the point $X = \mu_X$

$$Y \approx f(\mu_X) + \left.\frac{\partial f}{\partial X}\right|_{X = \mu_X} (X - \mu_X)$$



Jacobian Matrix

- It's a **non-square matrix** $n \times m$ in general
- Suppose you have a vector-valued function $f(\mathbf{x}) = \left[\begin{array}{c} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{array} \right]$
- Let the **gradient operator** be the vector of (first-order) partial derivatives

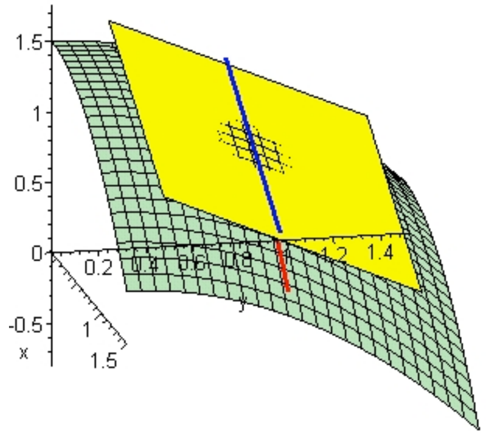
$$\nabla_{\mathbf{x}} = \left[\begin{array}{cccc} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \cdots & \frac{\partial}{\partial x_n} \end{array} \right]^T$$

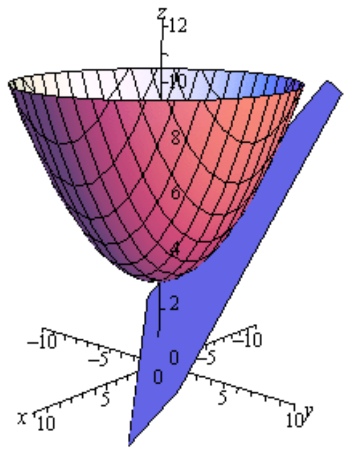
- Then, the **Jacobian matrix** is defined as

$$\mathbf{F}_{\mathbf{x}} = \left[\begin{array}{c} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{array} \right] \cdot \left[\begin{array}{ccc} \frac{\partial}{\partial x_1} & \cdots & \frac{\partial}{\partial x_n} \end{array} \right] = \left[\begin{array}{ccc} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \end{array} \right]$$

Jacobian Matrix

- It's the orientation of the tangent plane to the vector-valued function at a given point





- Generalizes the gradient of a scalar valued function

Track Management

Formation

- When to create a new track?
- What is the initial state?

Heuristics:

- Greedy initialization
  - Every observation not associated is a new track
  - Initialize only position
- Lazy initialization
  - Accumulate several unassociated observations
  - Initialize position & velocity

Occlusion/deletion

- When to delete a track?
- Is it just occluded?

Heuristics:

- Greedy deletion
  - Delete if no observation can be associated
  - No occlusion handling
- Lazy deletion
  - Delete if no observation can be associated for several time steps
  - Implicit occlusion handling

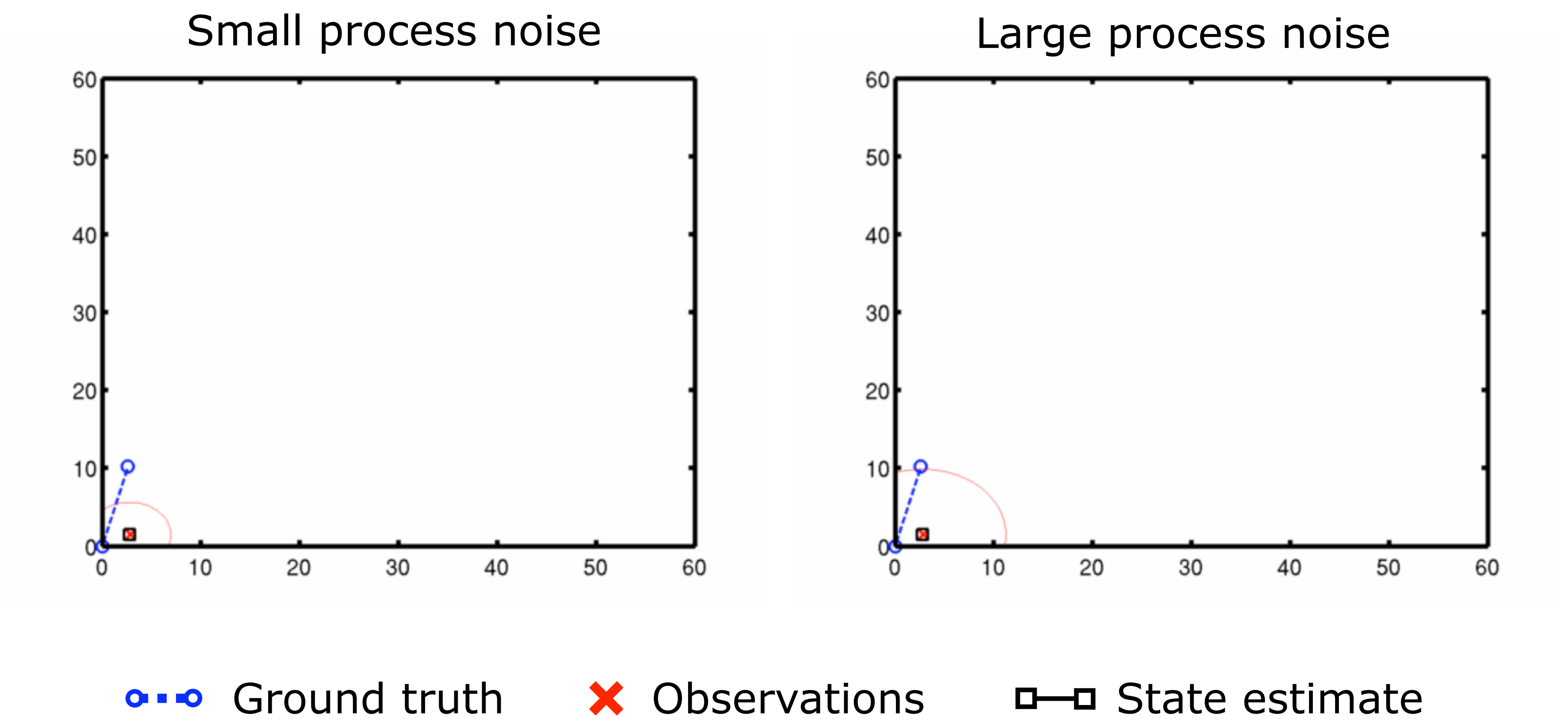
Example: Tracking the Ball

- Unlike the previous experiment in which we had a model of the ball's trajectory and just observed it, **we now want to track the ball**

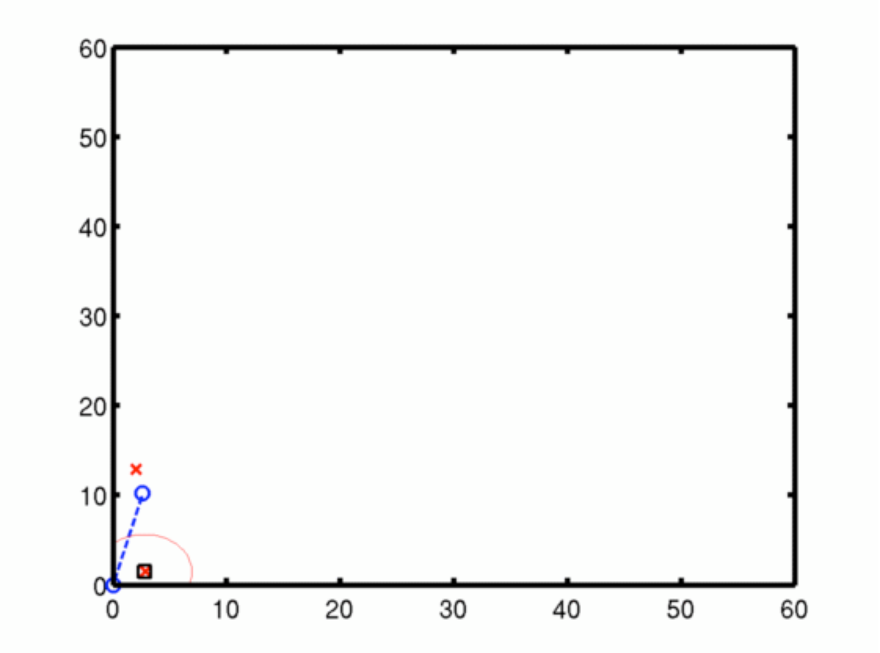
- Comparison: small versus large process noise Q and the effect of the three different motion models

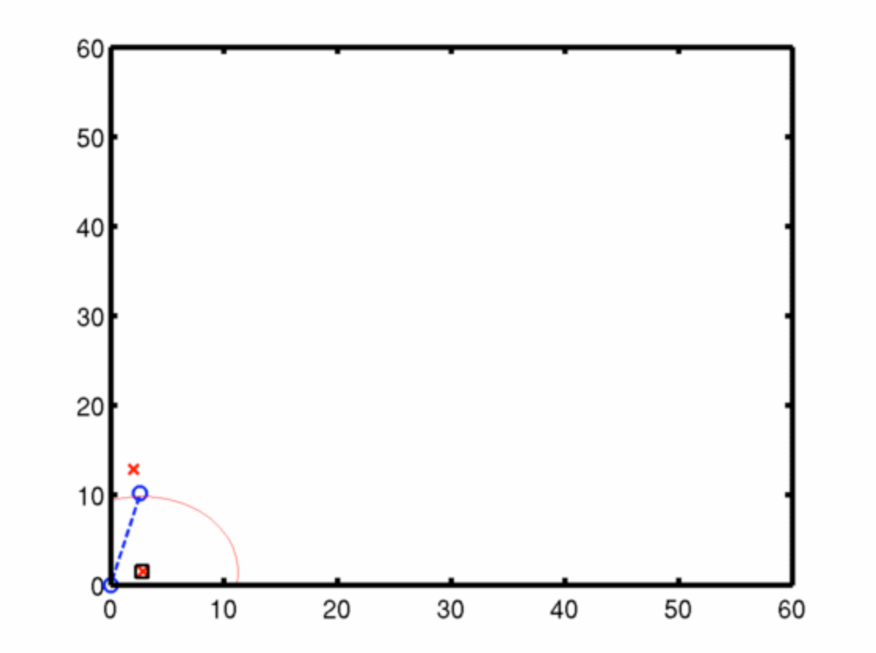
- For simplicity, we perform **no gaiting** (i.e. no Mahalanobis test) but accept the pairing every time

Ball Tracking: Brownian



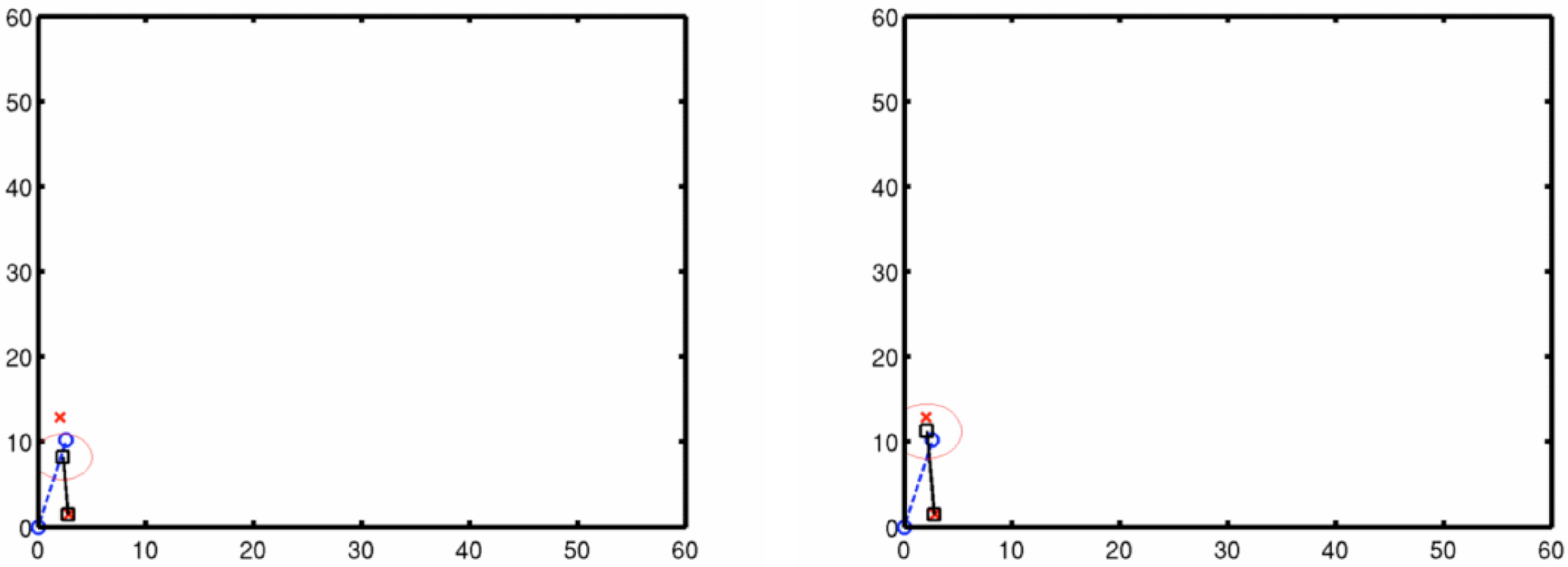
Ball Tracking: Brownian





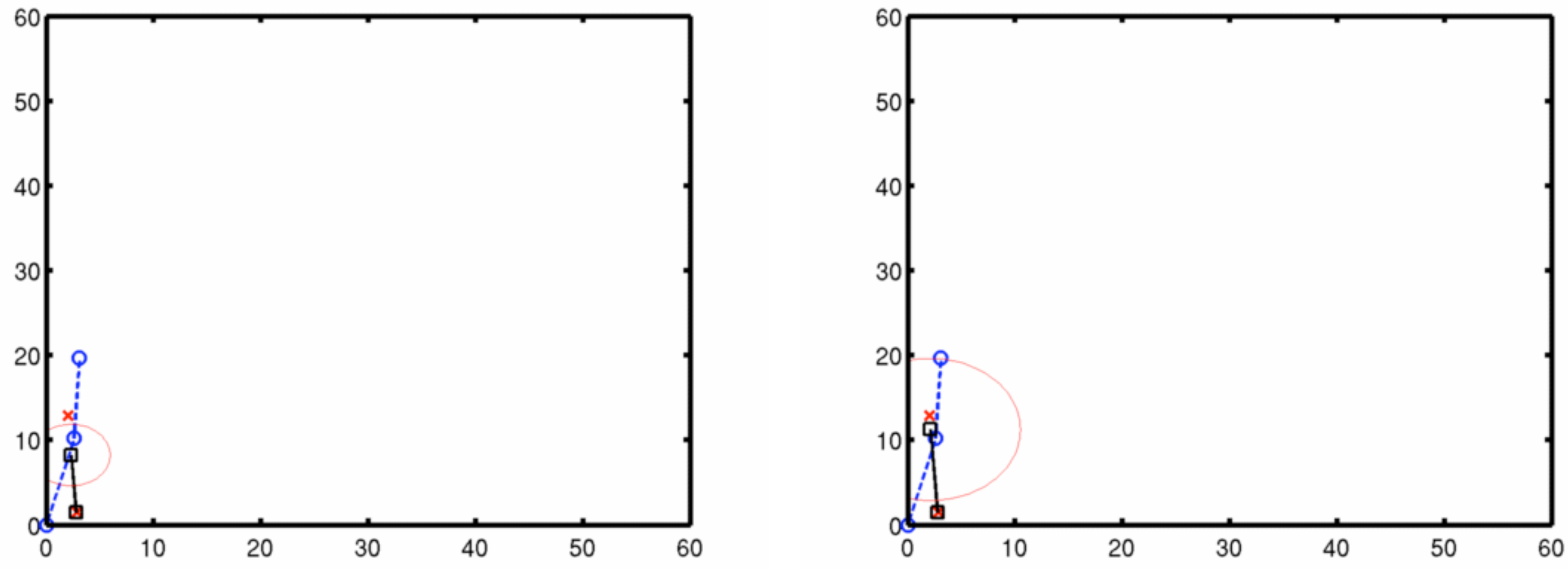
Ground truth   Observations   State estimate

Ball Tracking: Brownian

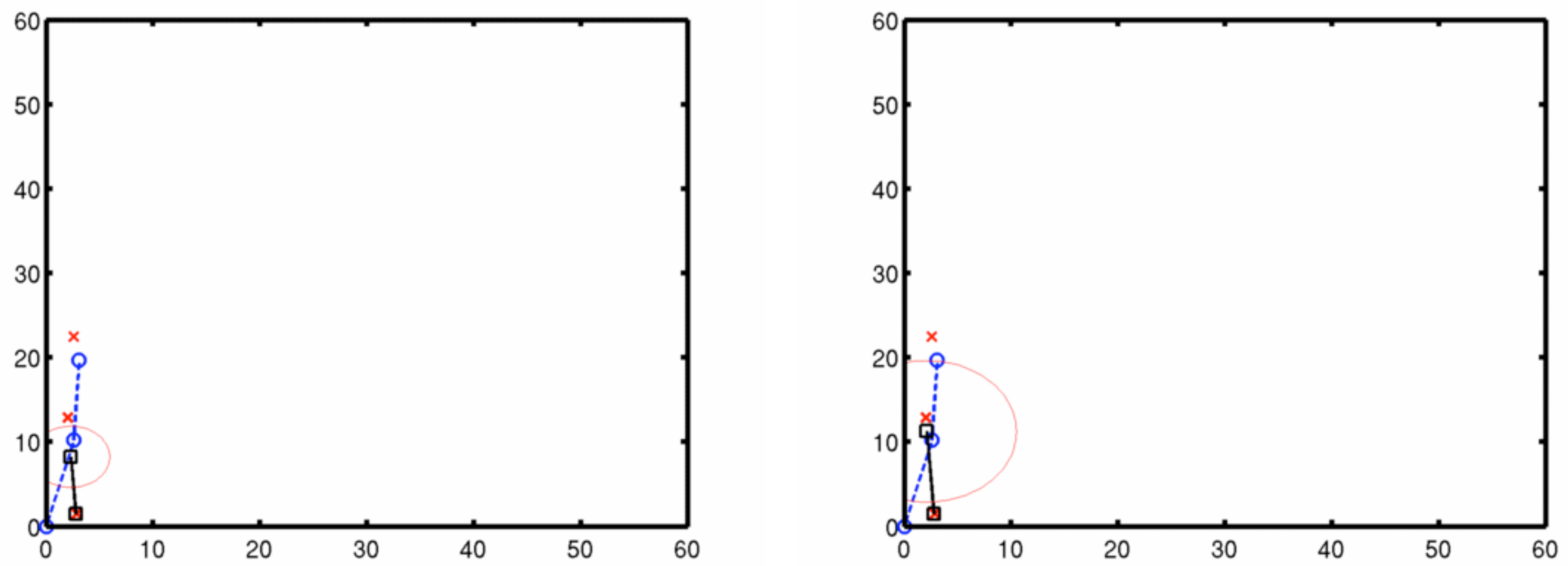


Ground truth   Observations   State estimate

Ball Tracking: Brownian

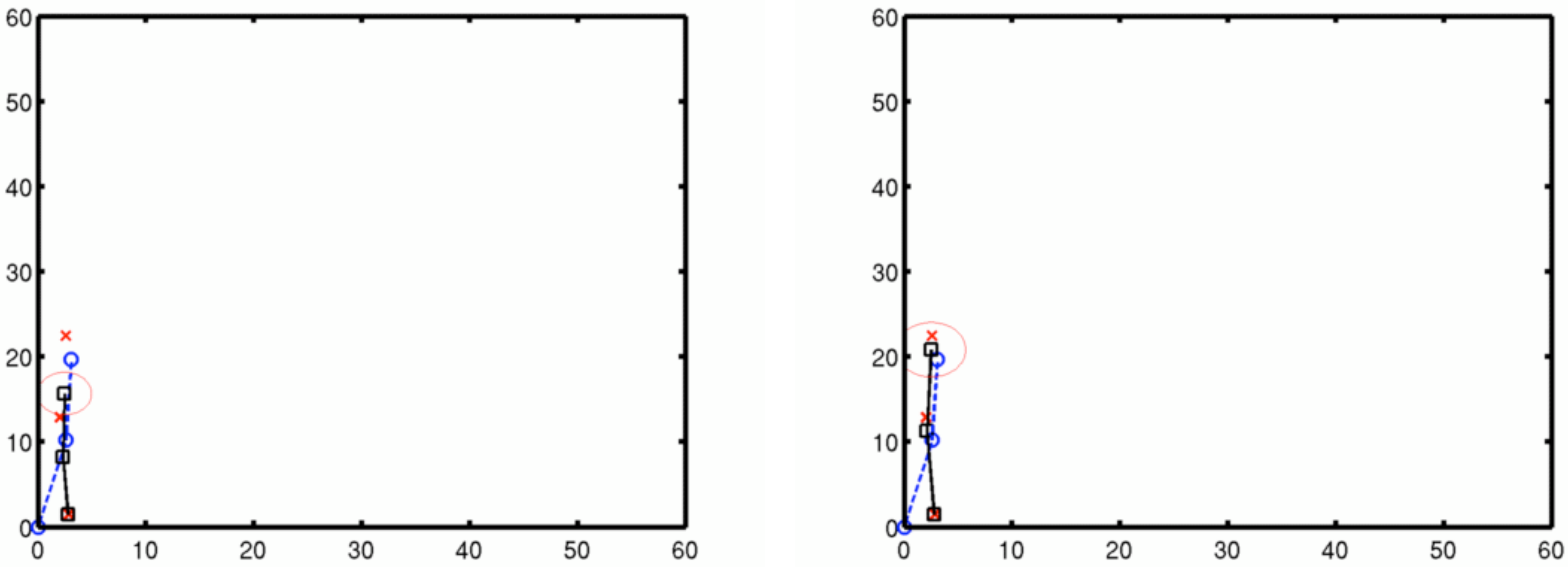


Ball Tracking: Brownian



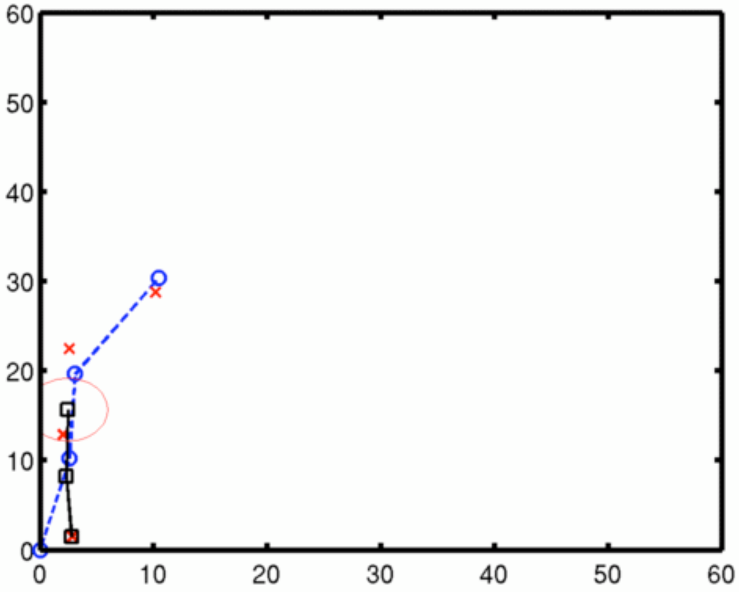
Ground truth   Observations   State estimate

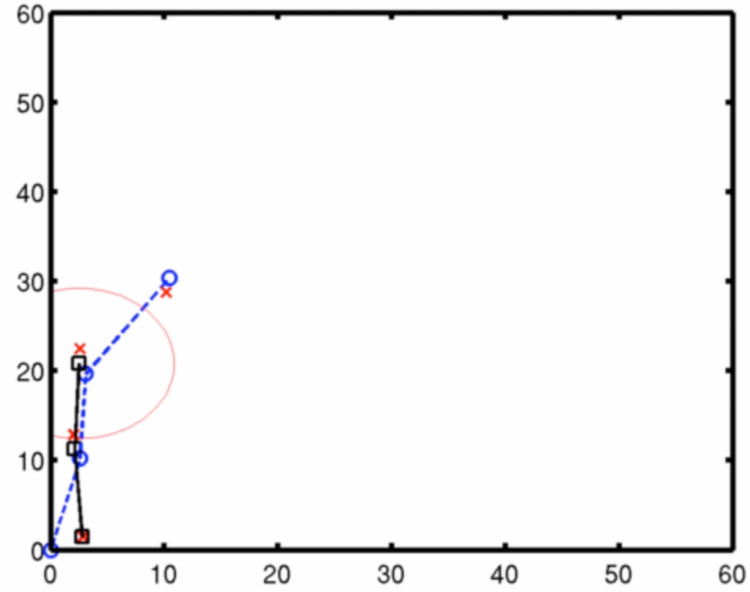
Ball Tracking: Brownian



Ground truth   Observations   State estimate

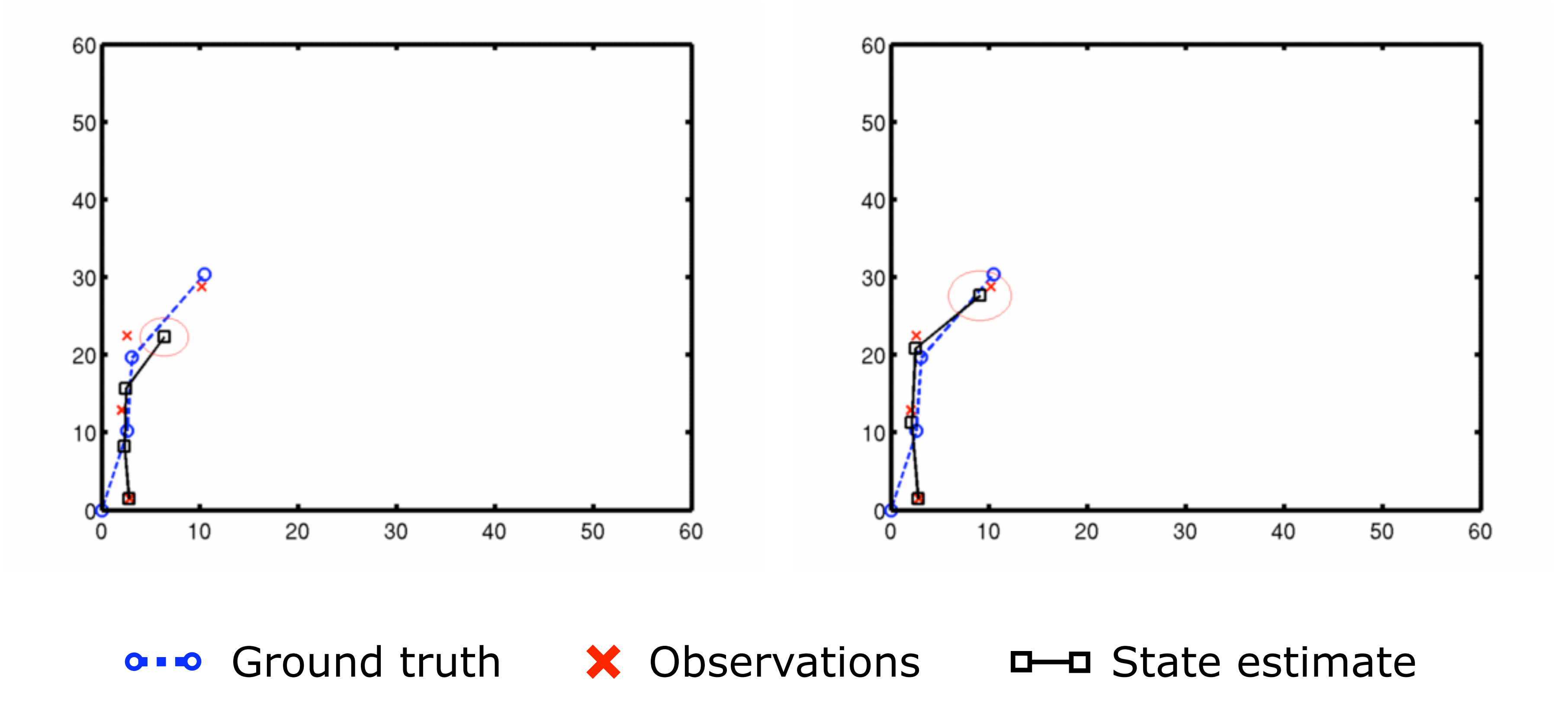
Ball Tracking: Brownian



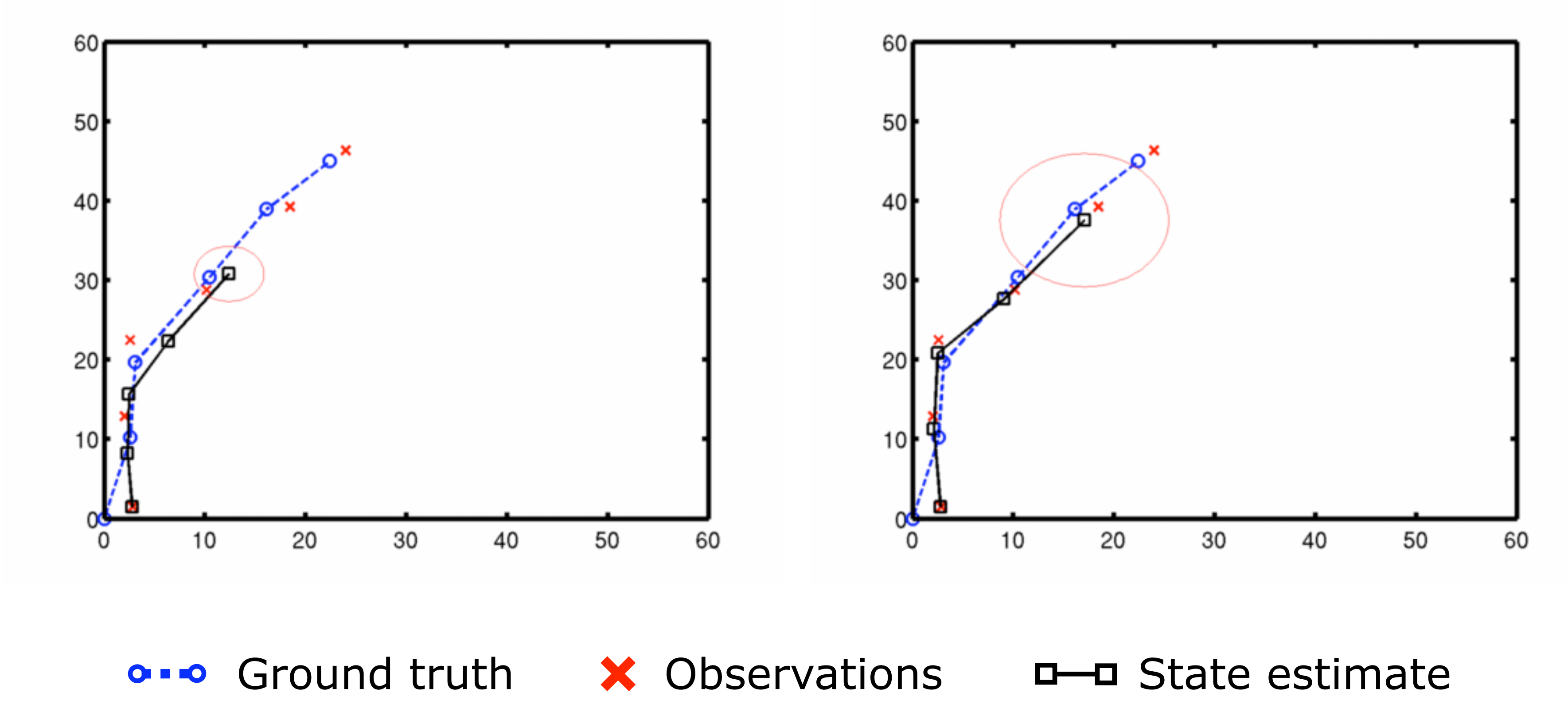


Ground truth   Observations   State estimate

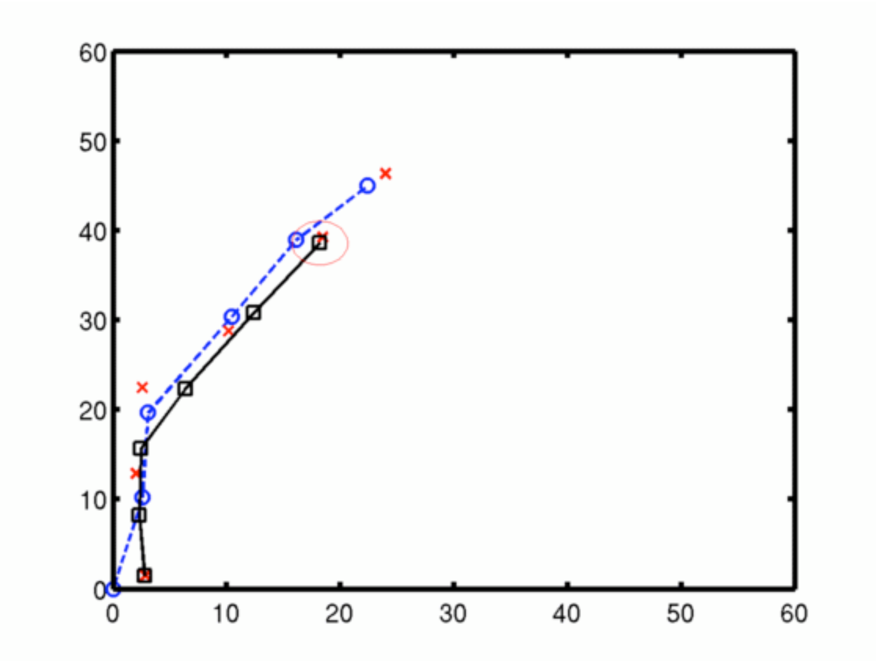
Ball Tracking: Brownian

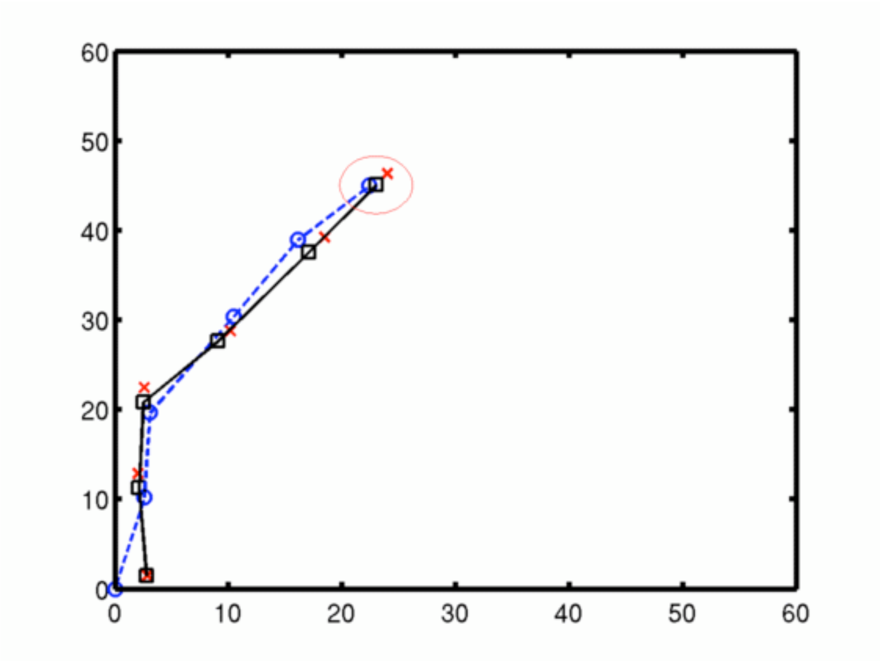


Ball Tracking: Brownian

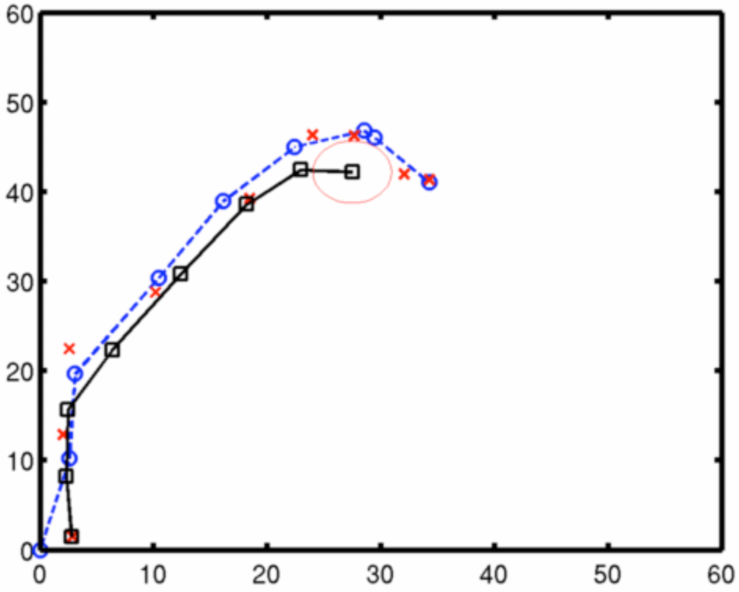


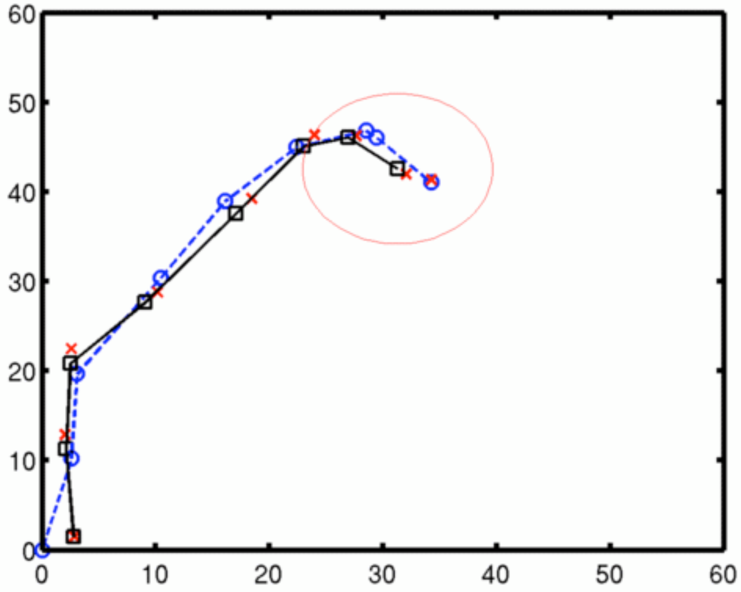
Ball Tracking: Brownian





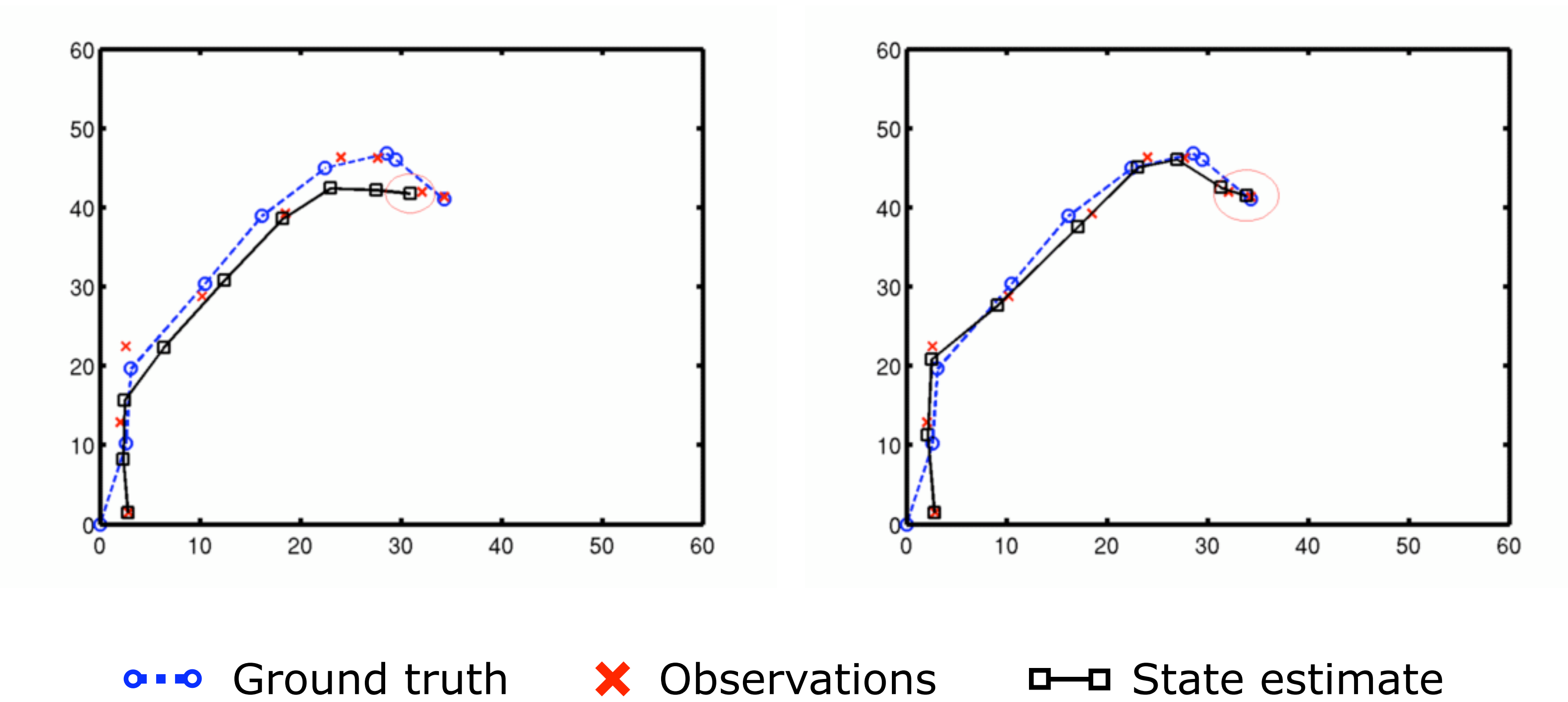
Ball Tracking: Brownian



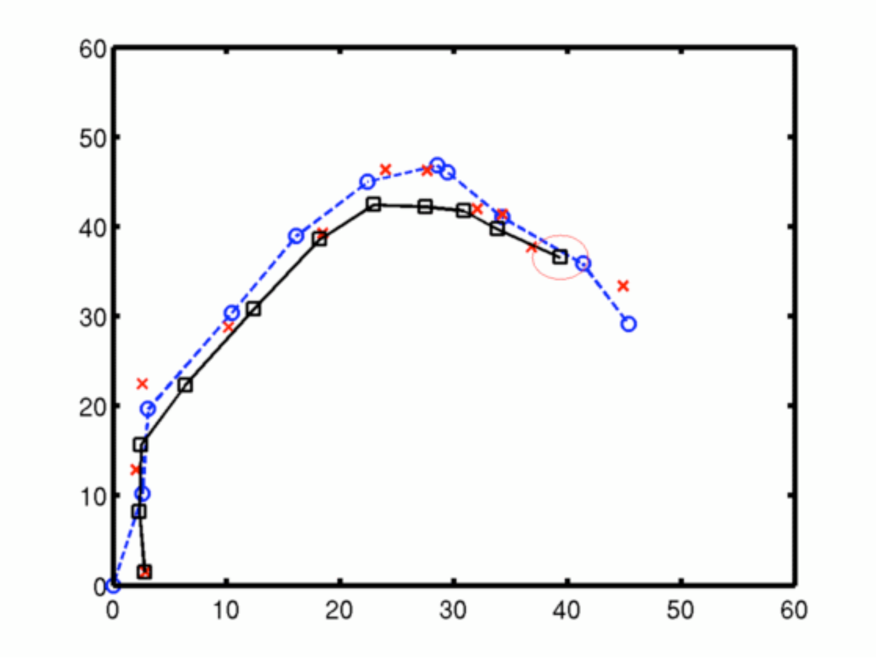


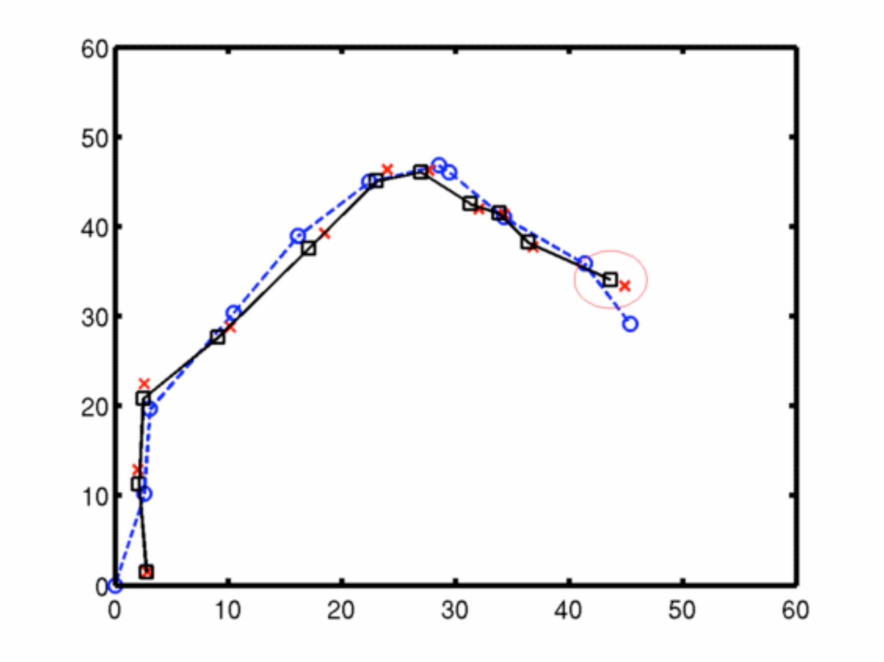
Ground truth Observations State estimate

Ball Tracking: Brownian



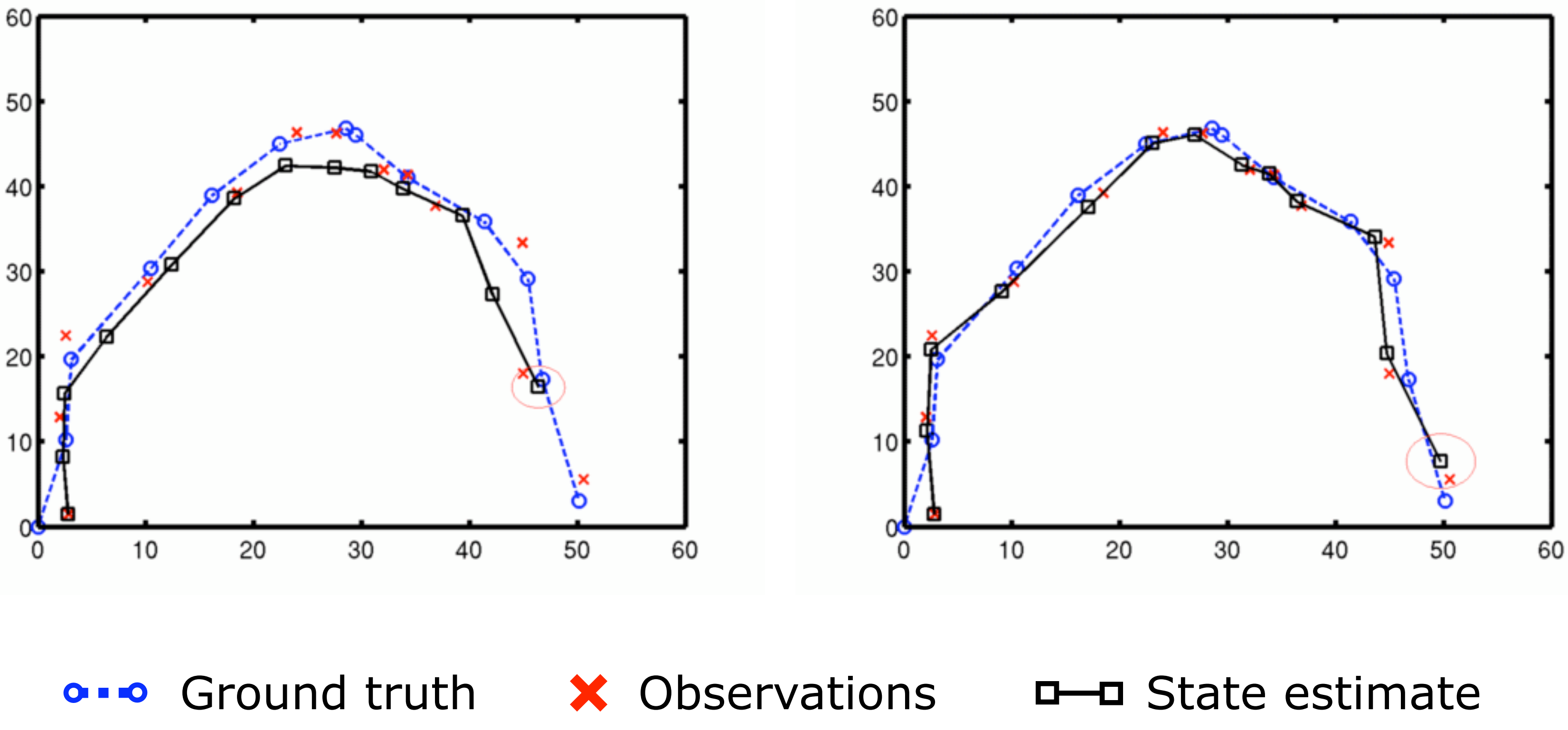
Ball Tracking: Brownian



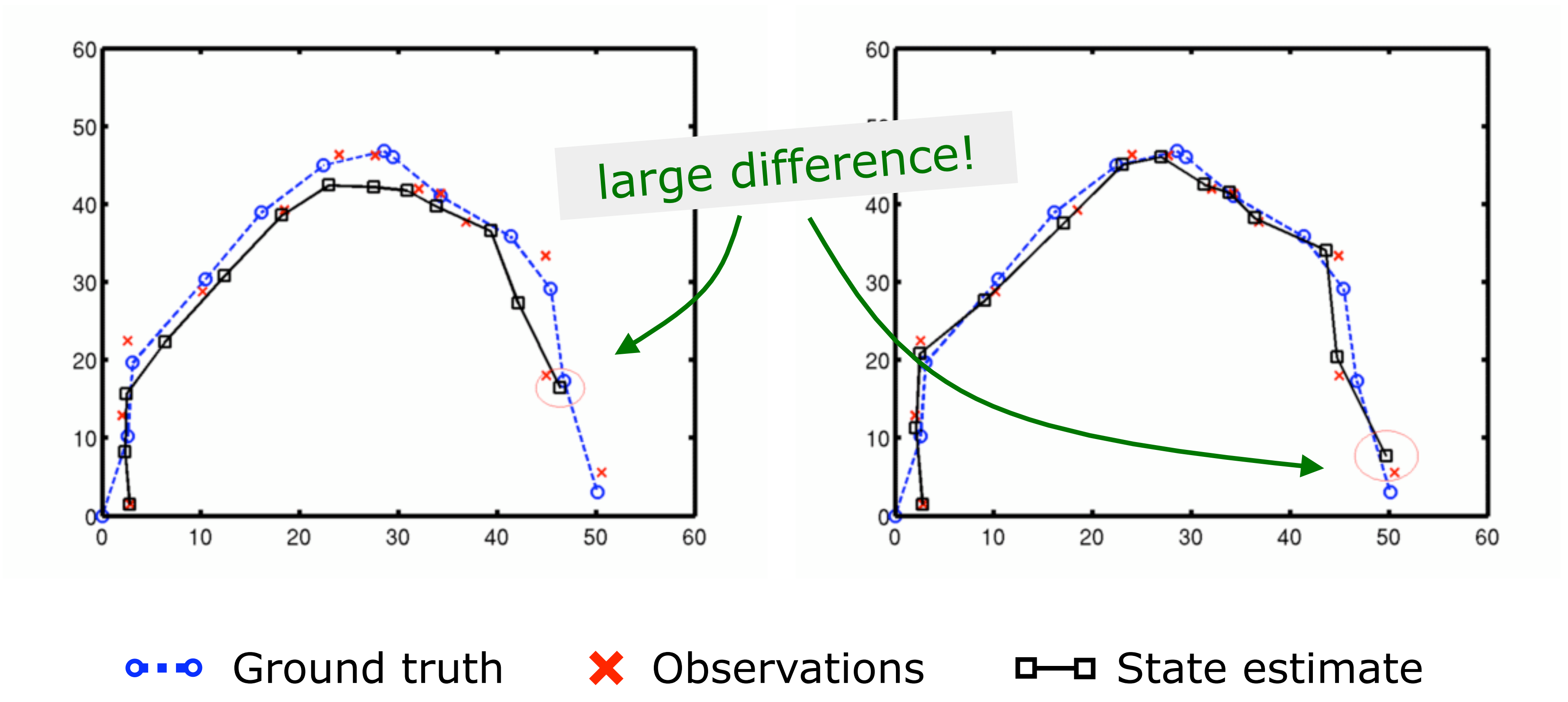


Ground truth Observations State estimate

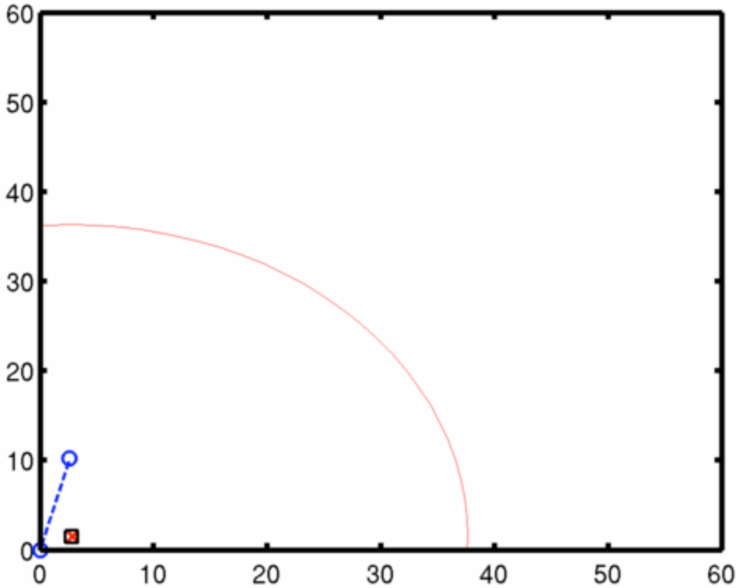
Ball Tracking: Brownian

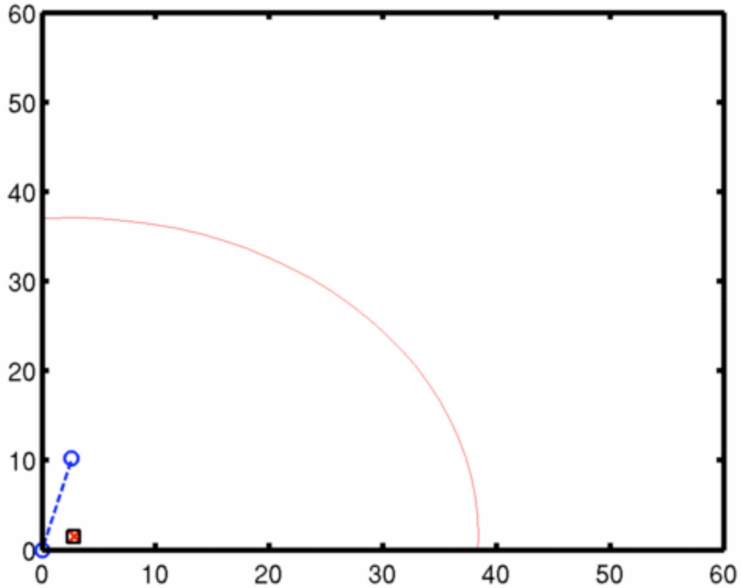


Ball Tracking: Brownian



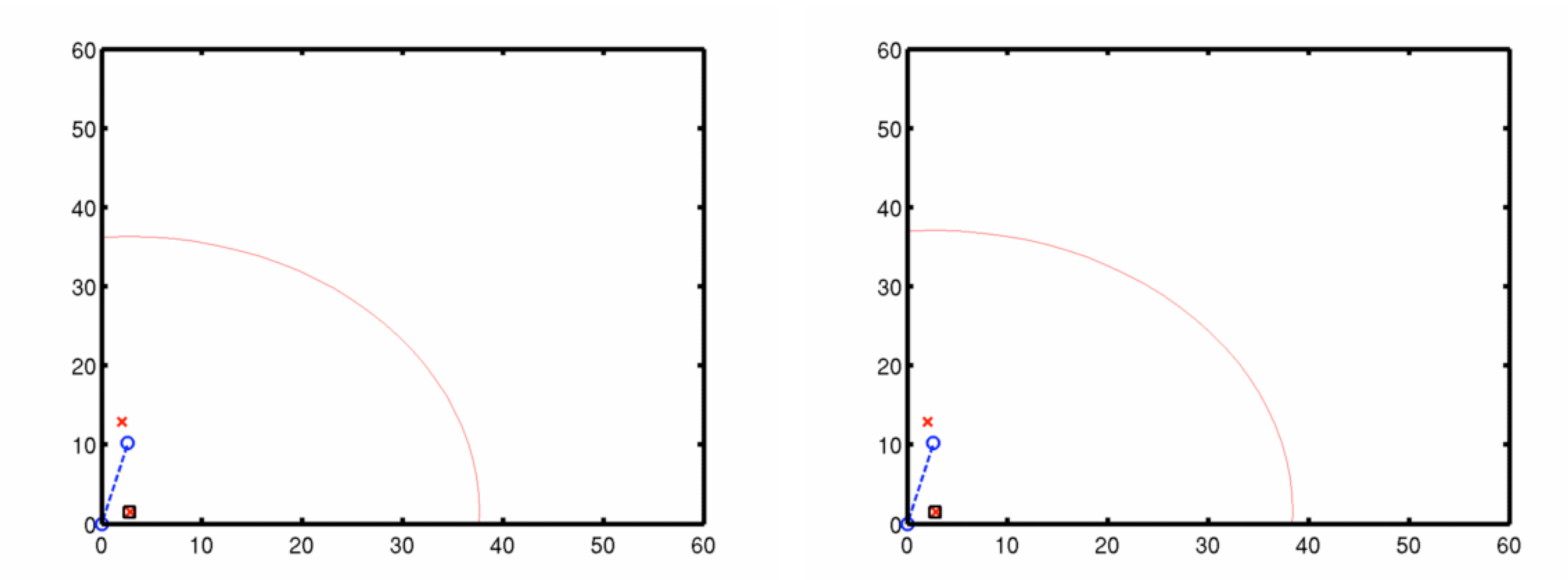
Ball Tracking: Constant Velocity





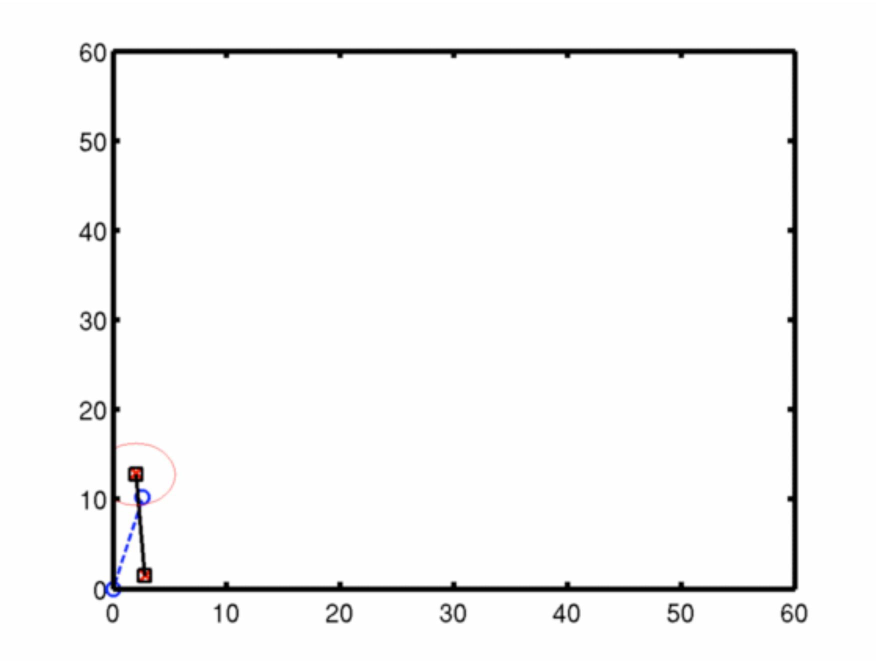
Ground truth   Observations   State estimate

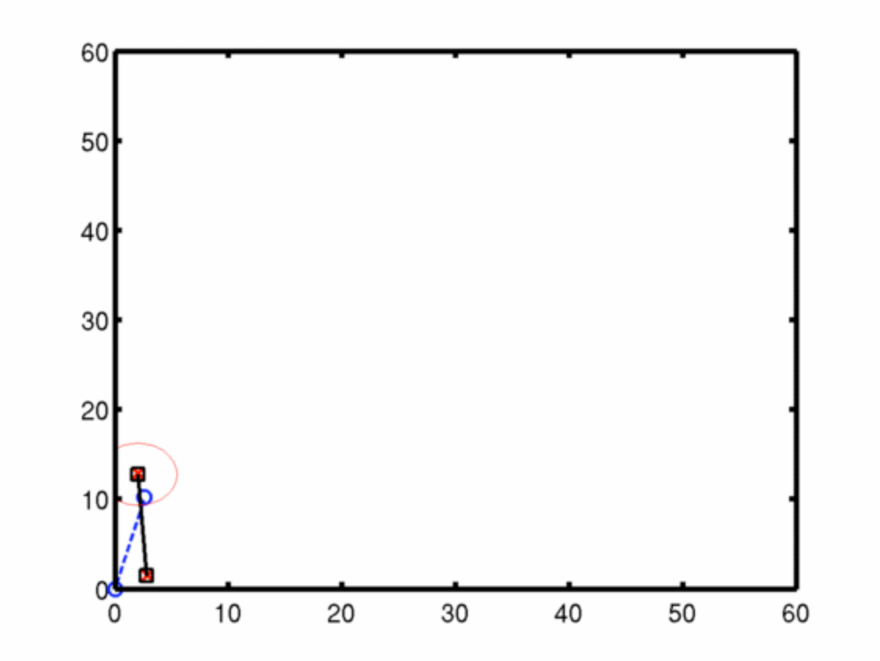
Ball Tracking: Constant Velocity



Ground truth Observations State estimate

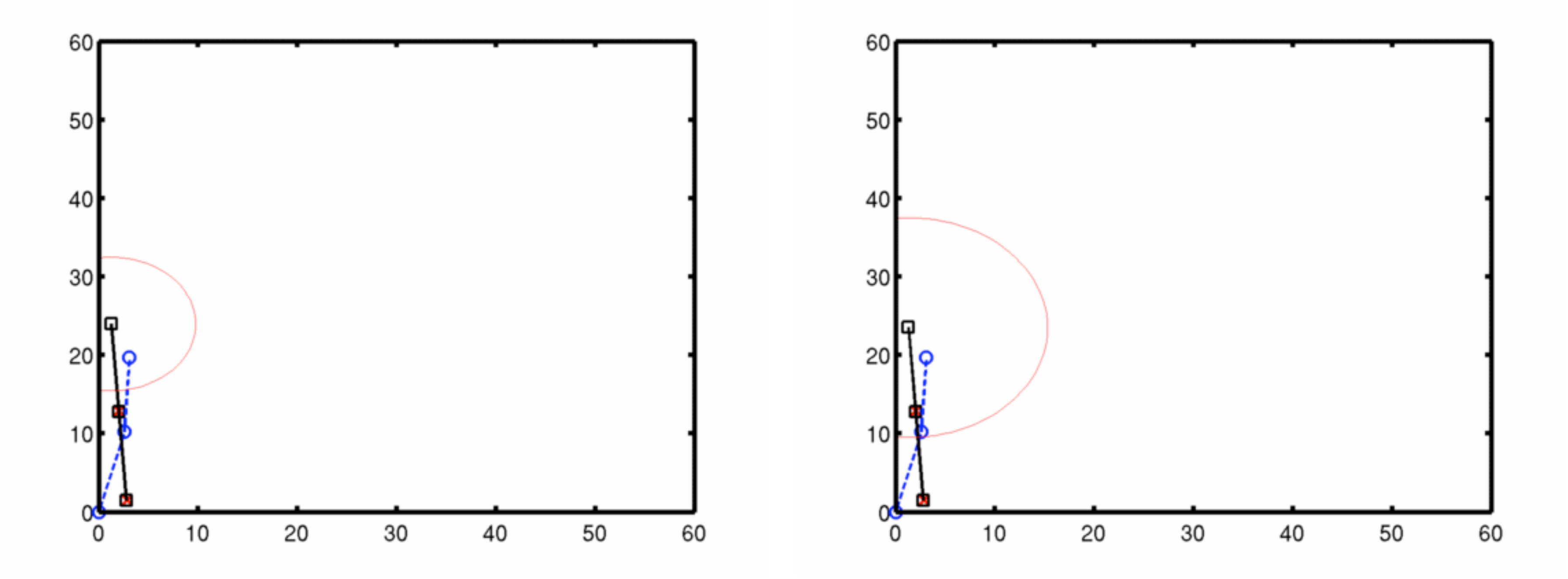
Ball Tracking: Constant Velocity





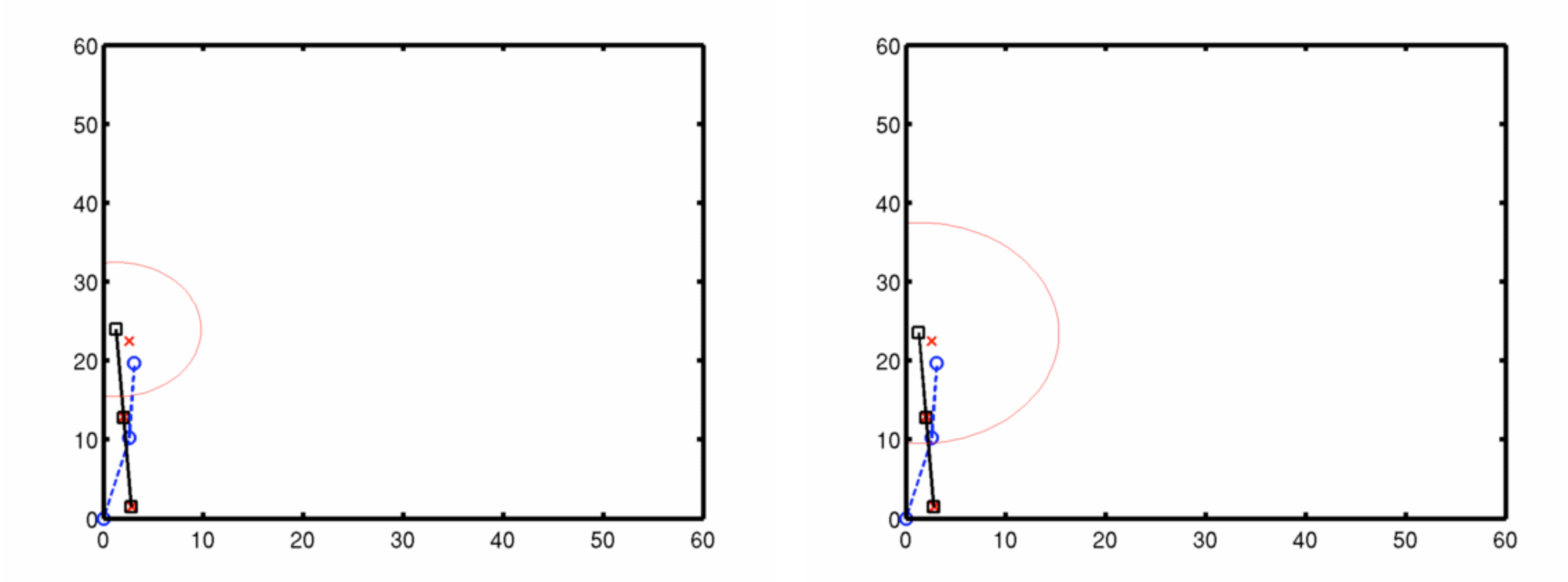
Ground truth Observations State estimate

Ball Tracking: Constant Velocity



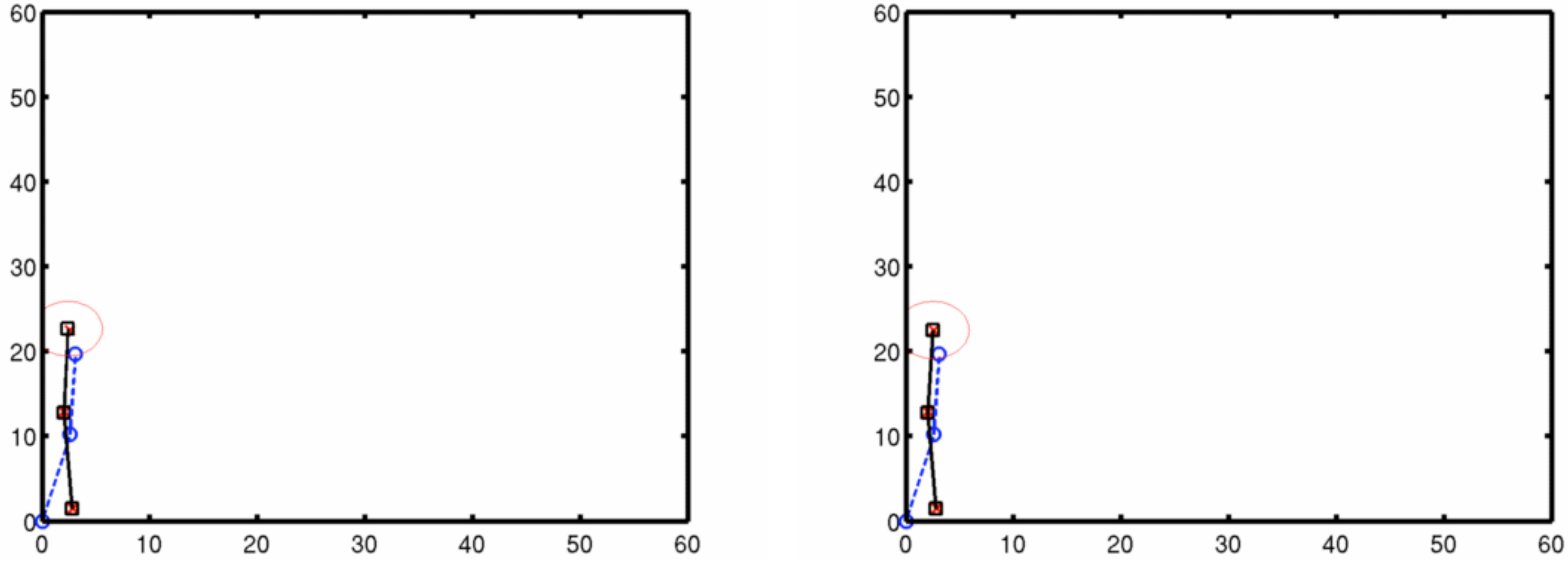
Ground truth   Observations   State estimate

Ball Tracking: Constant Velocity



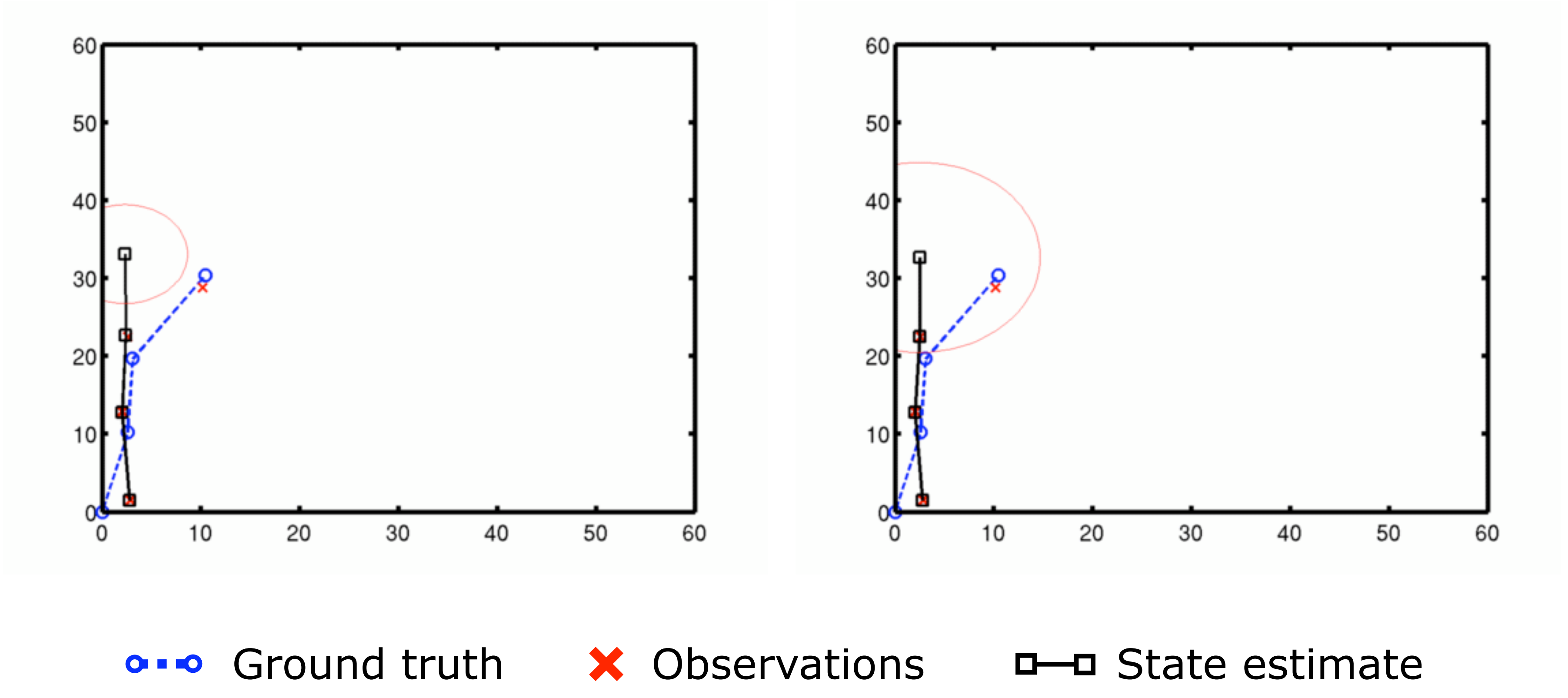
Ground truth   Observations   State estimate

Ball Tracking: Constant Velocity

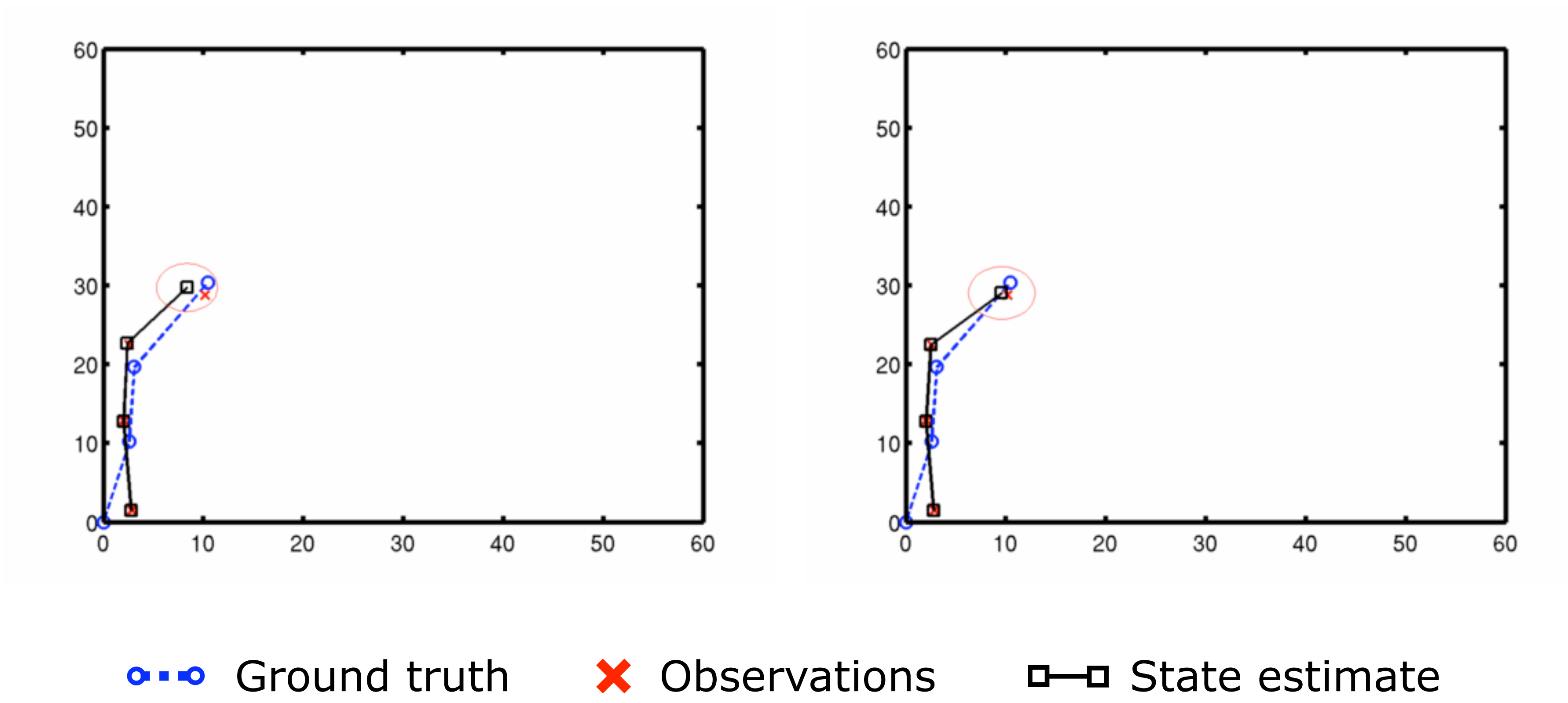


Ground truth Observations State estimate

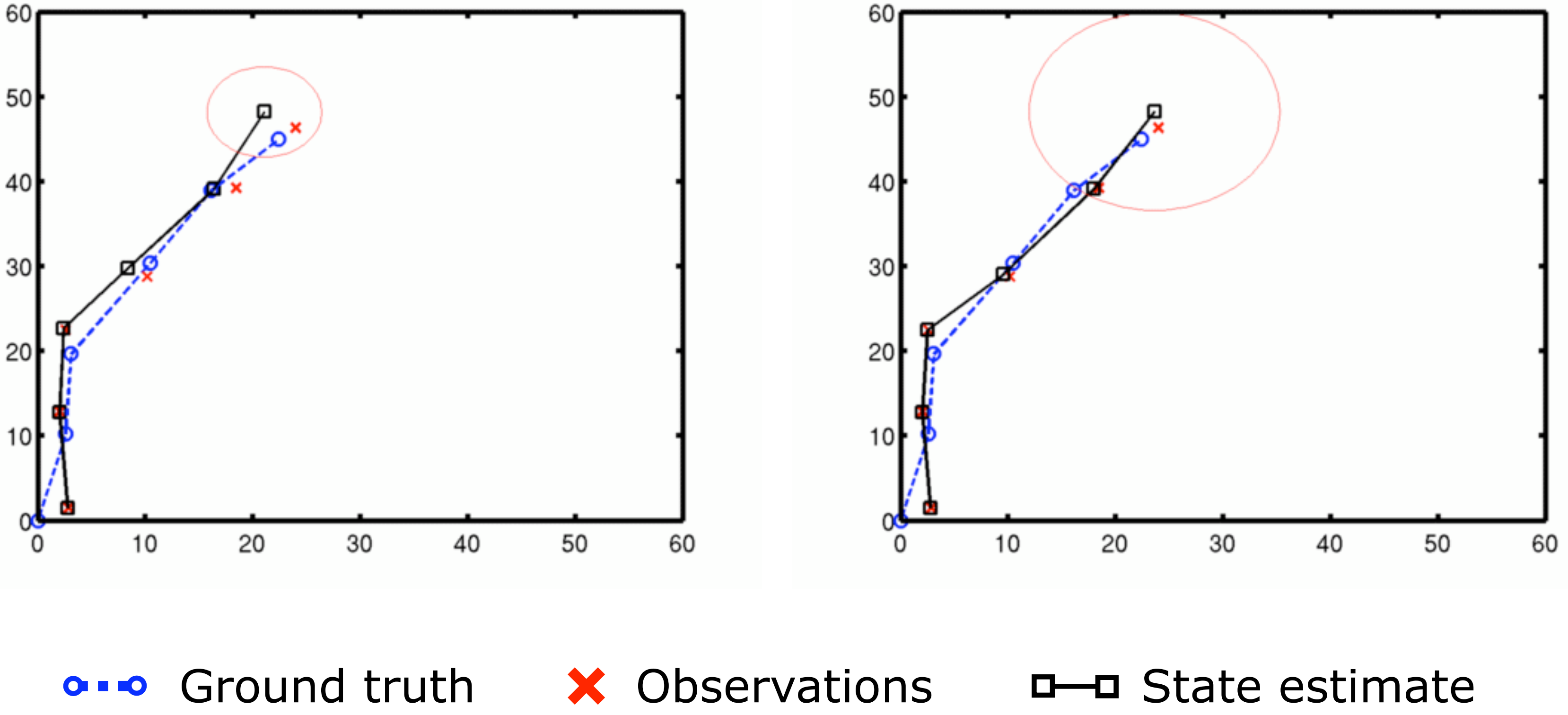
Ball Tracking: Constant Velocity



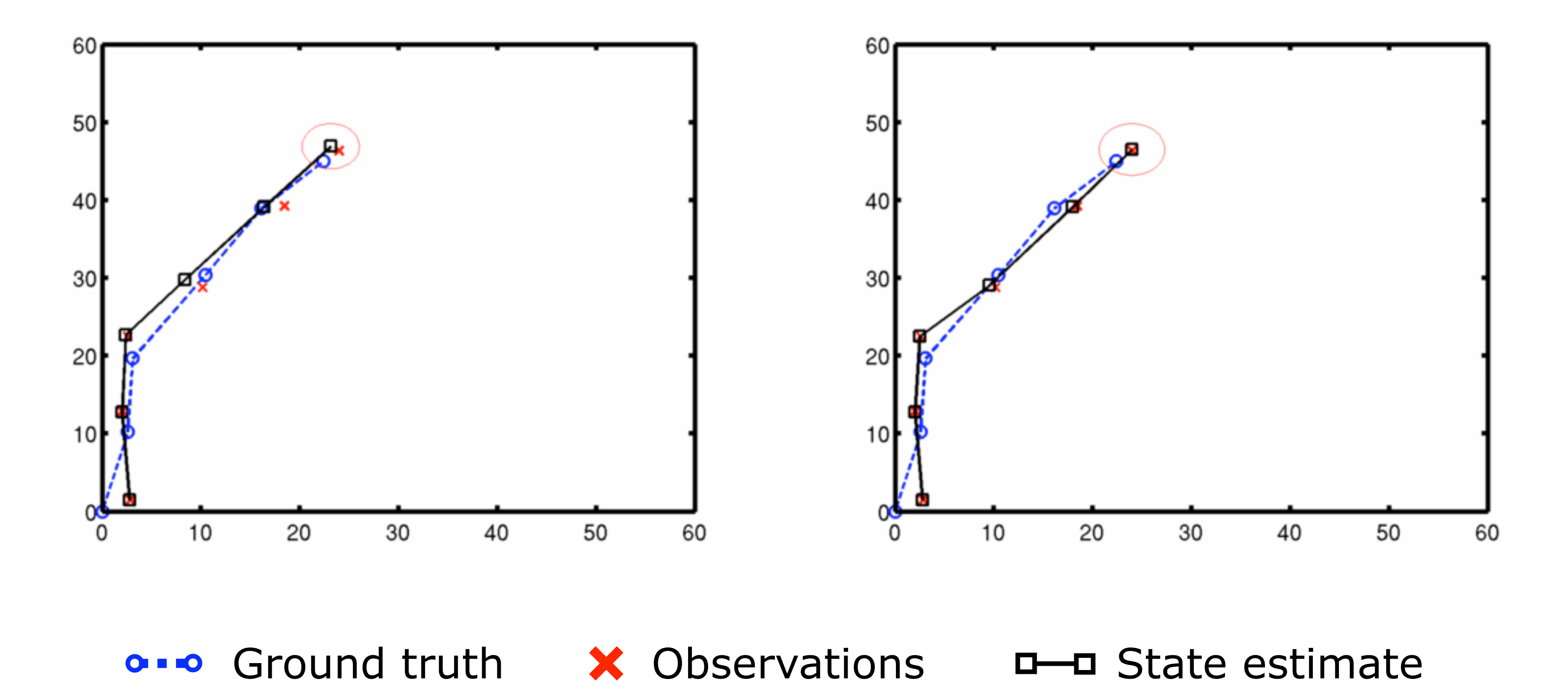
Ball Tracking: Constant Velocity



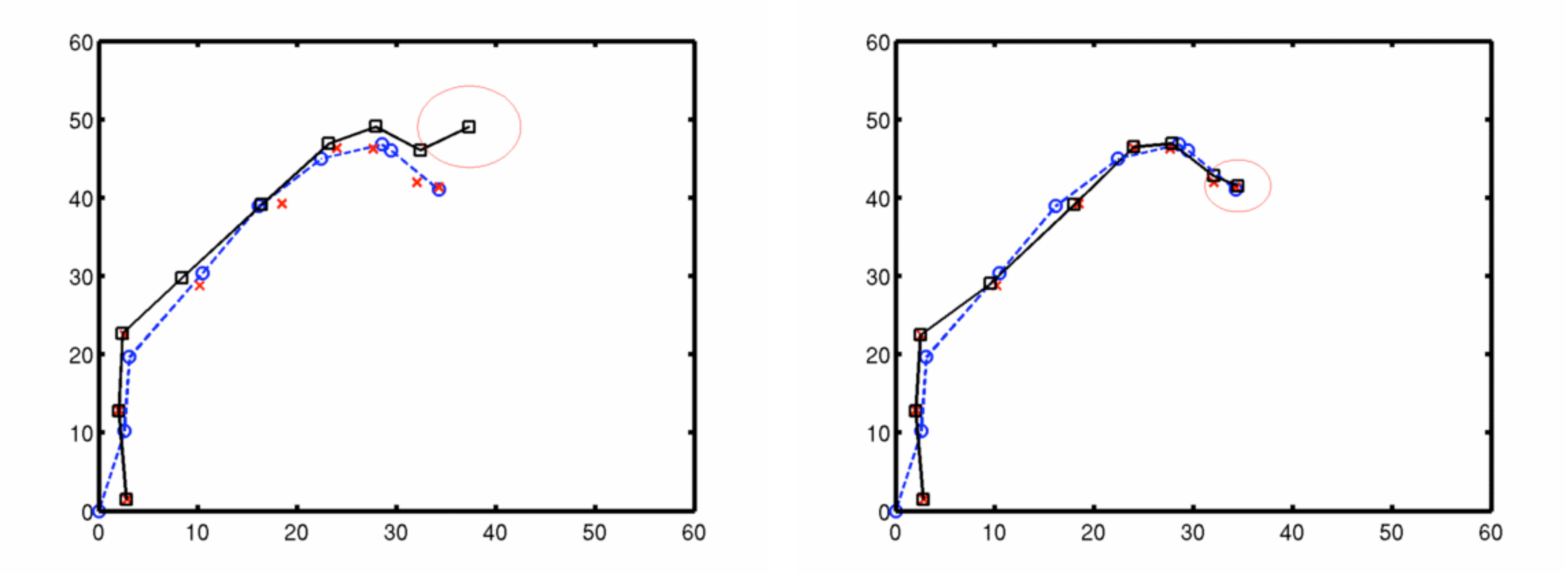
Ball Tracking: Constant Velocity



Ball Tracking: Constant Velocity

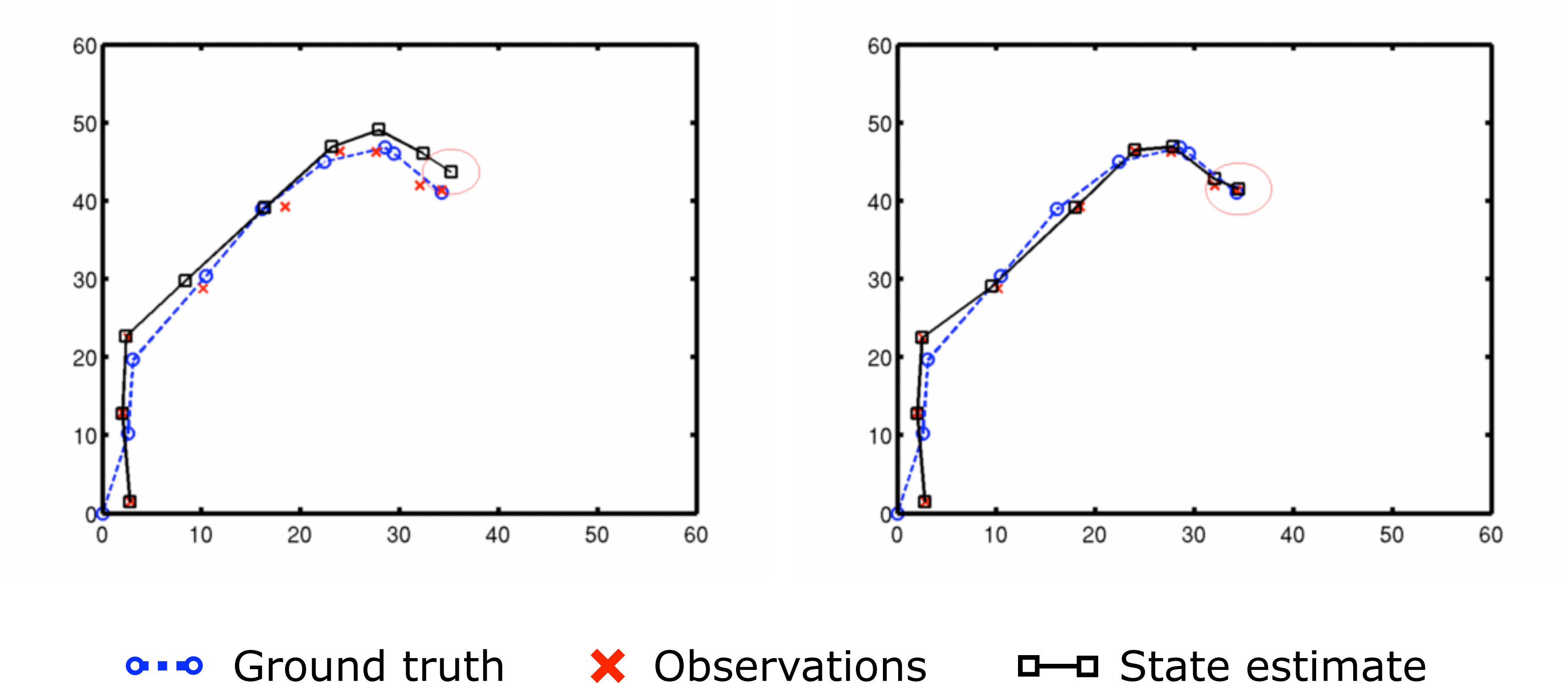


Ball Tracking: Constant Velocity

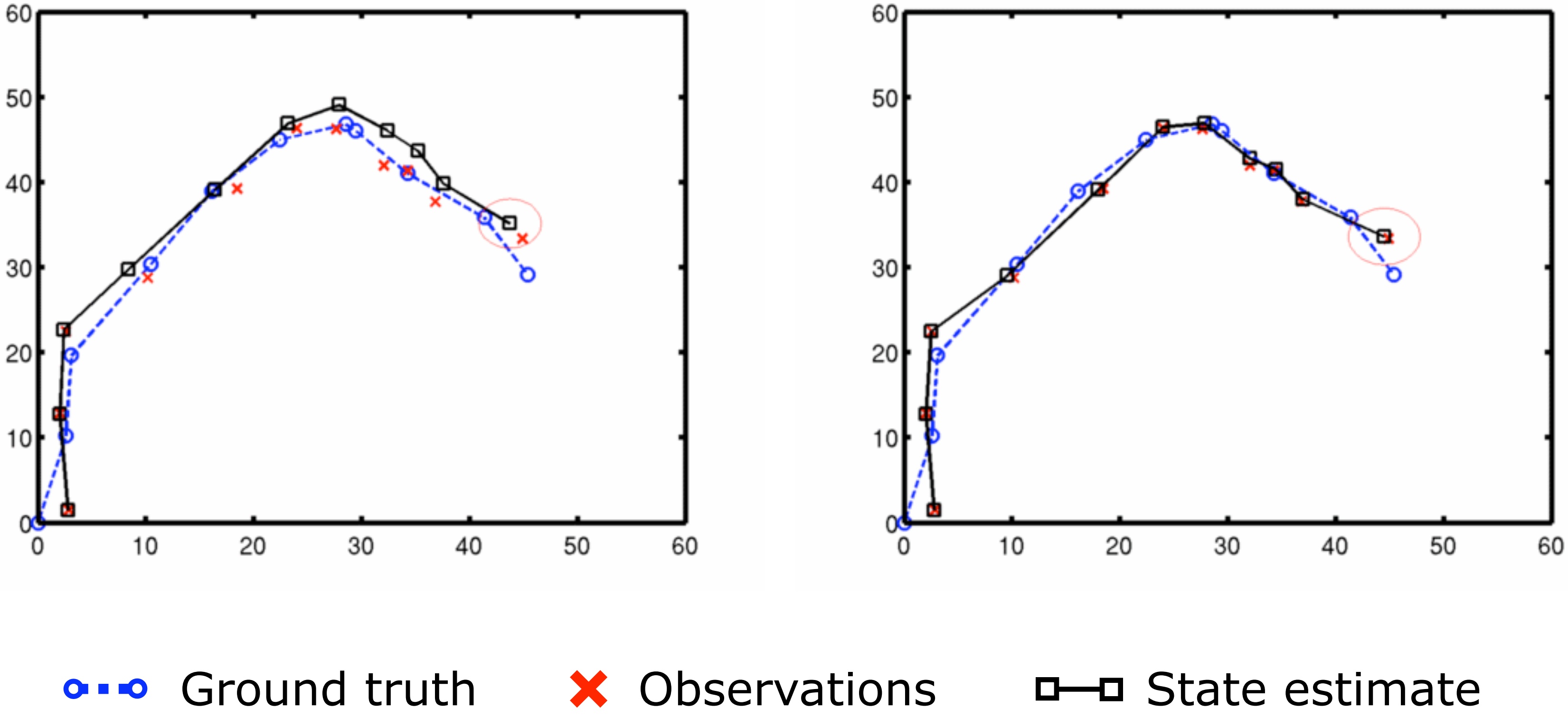


Ground truth   Observations   State estimate

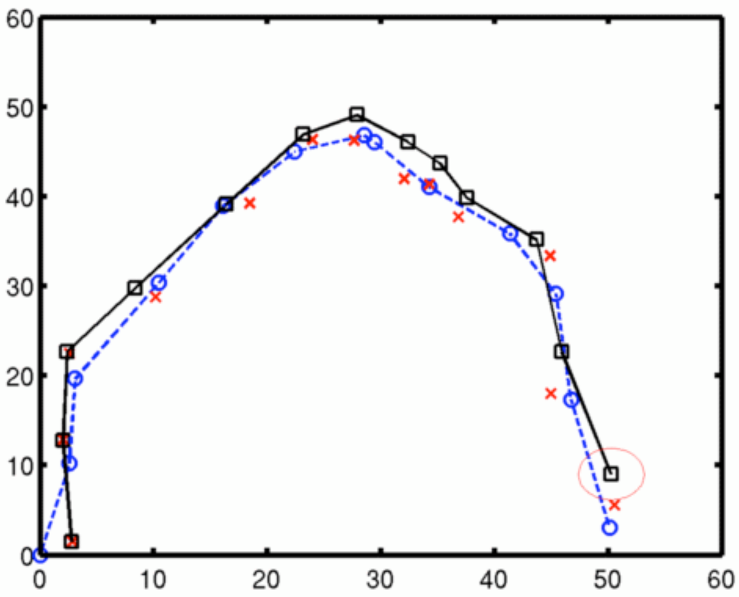
Ball Tracking: Constant Velocity

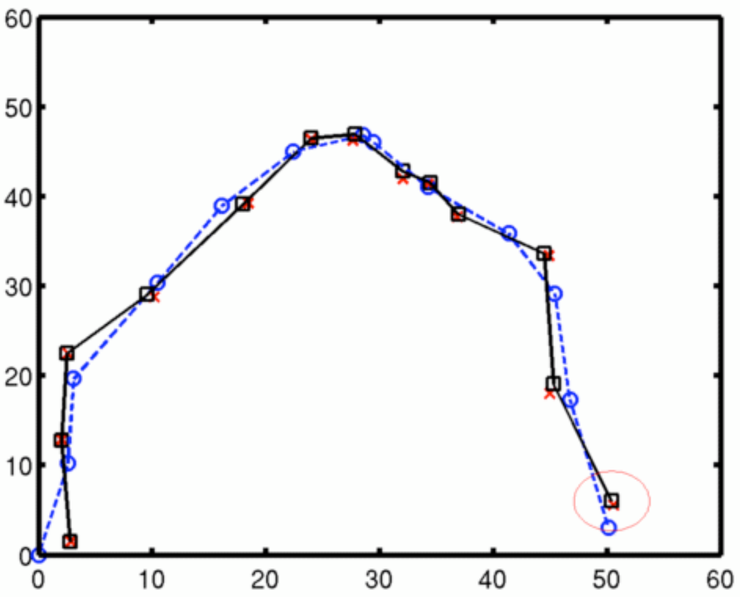


Ball Tracking: Constant Velocity



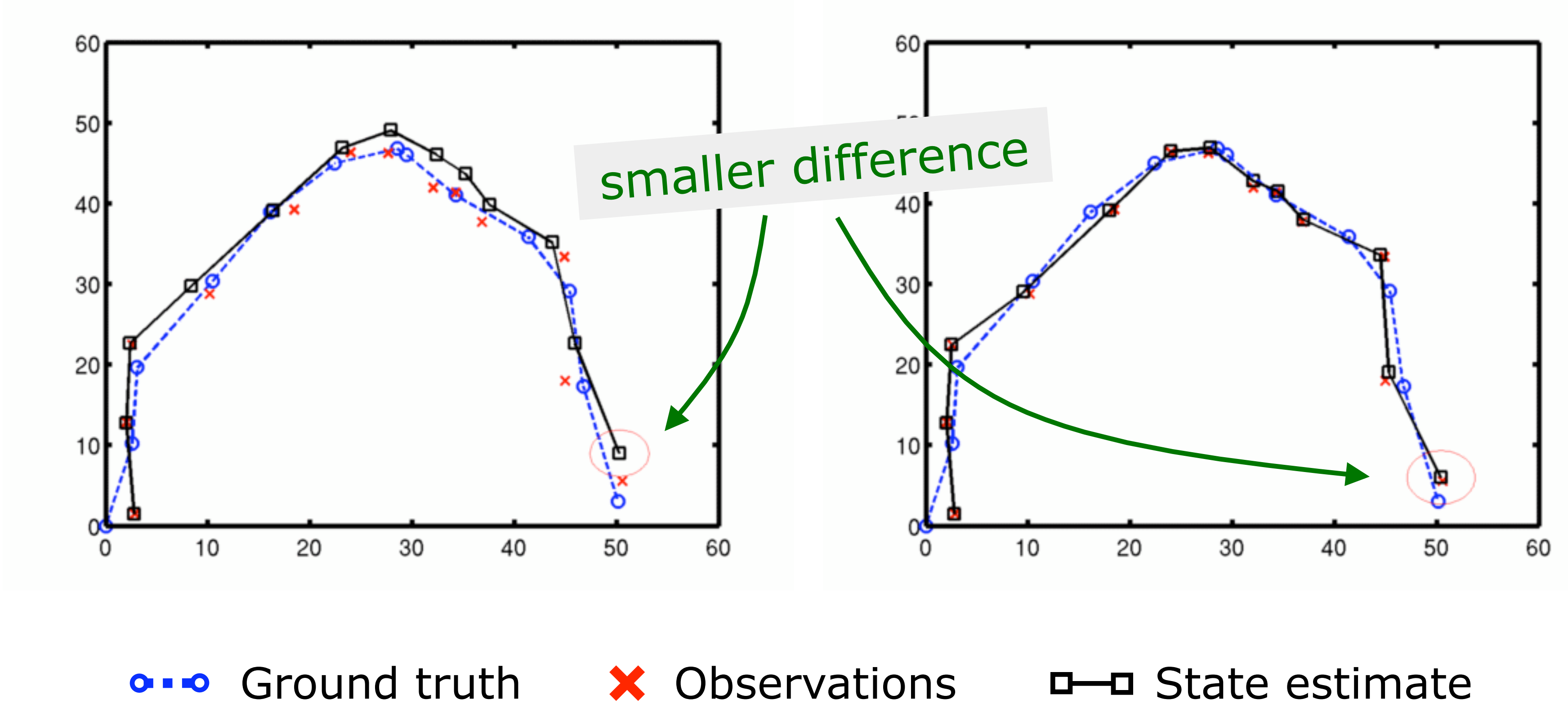
Ball Tracking: Constant Velocity



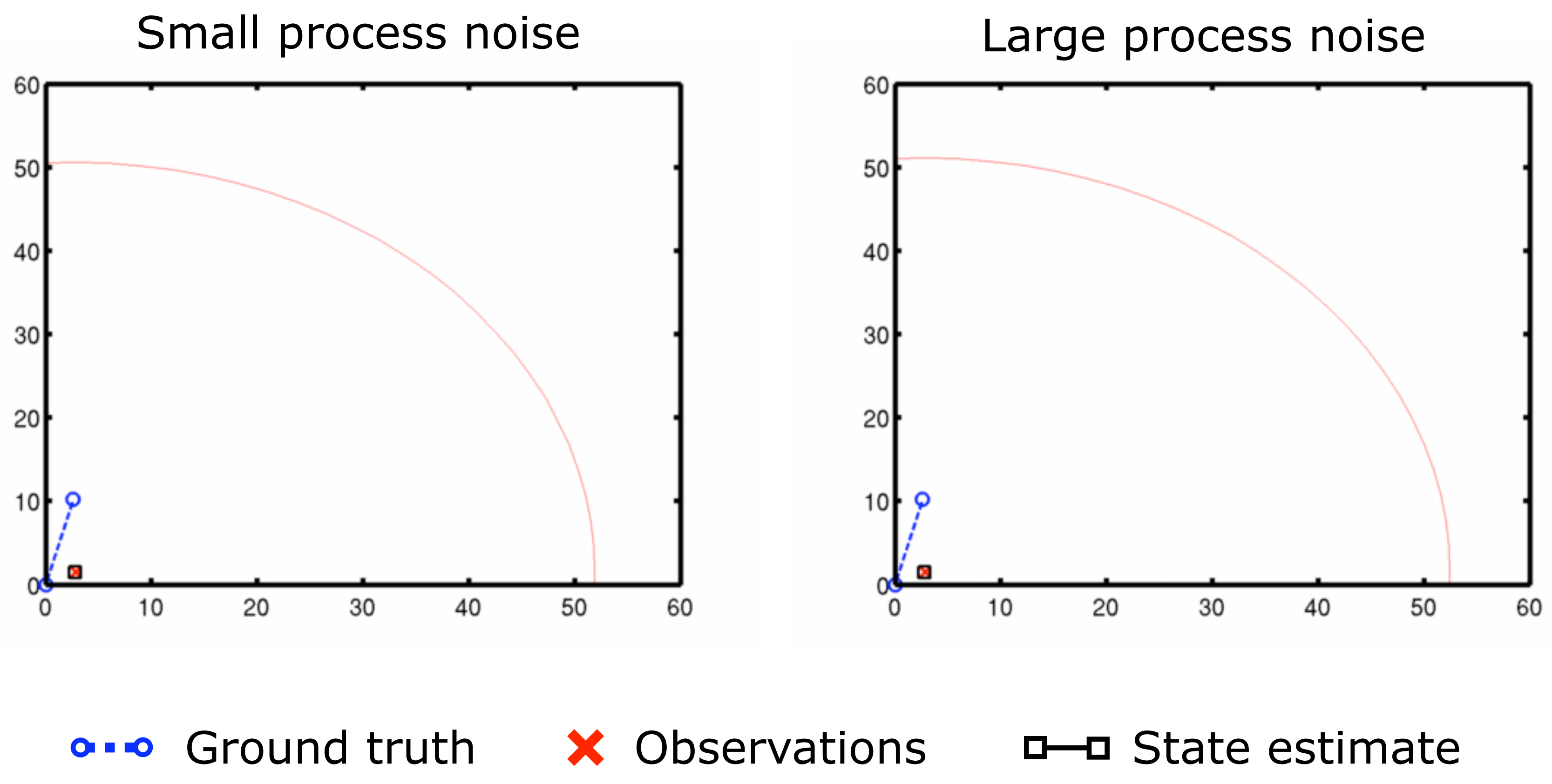


Ground truth Observations State estimate

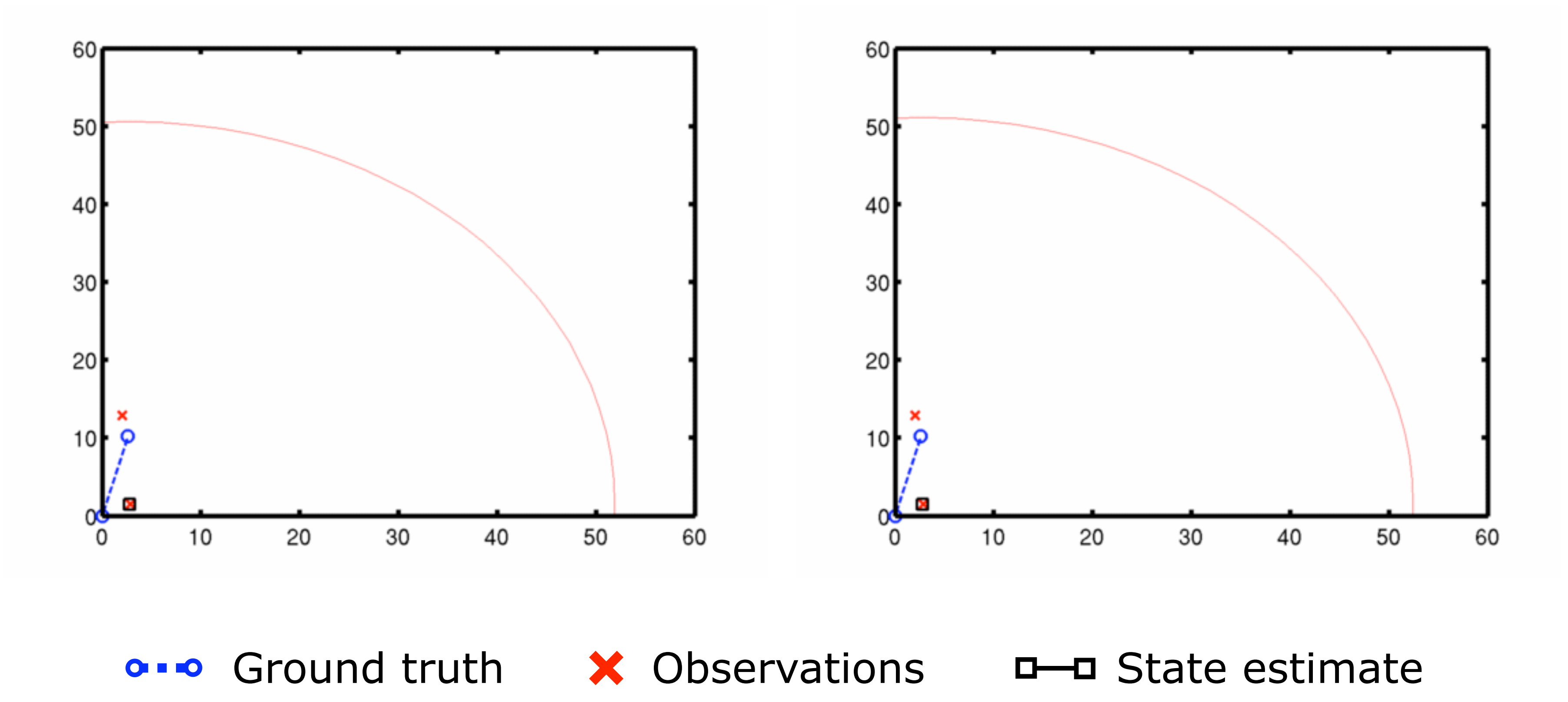
Ball Tracking: Constant Velocity



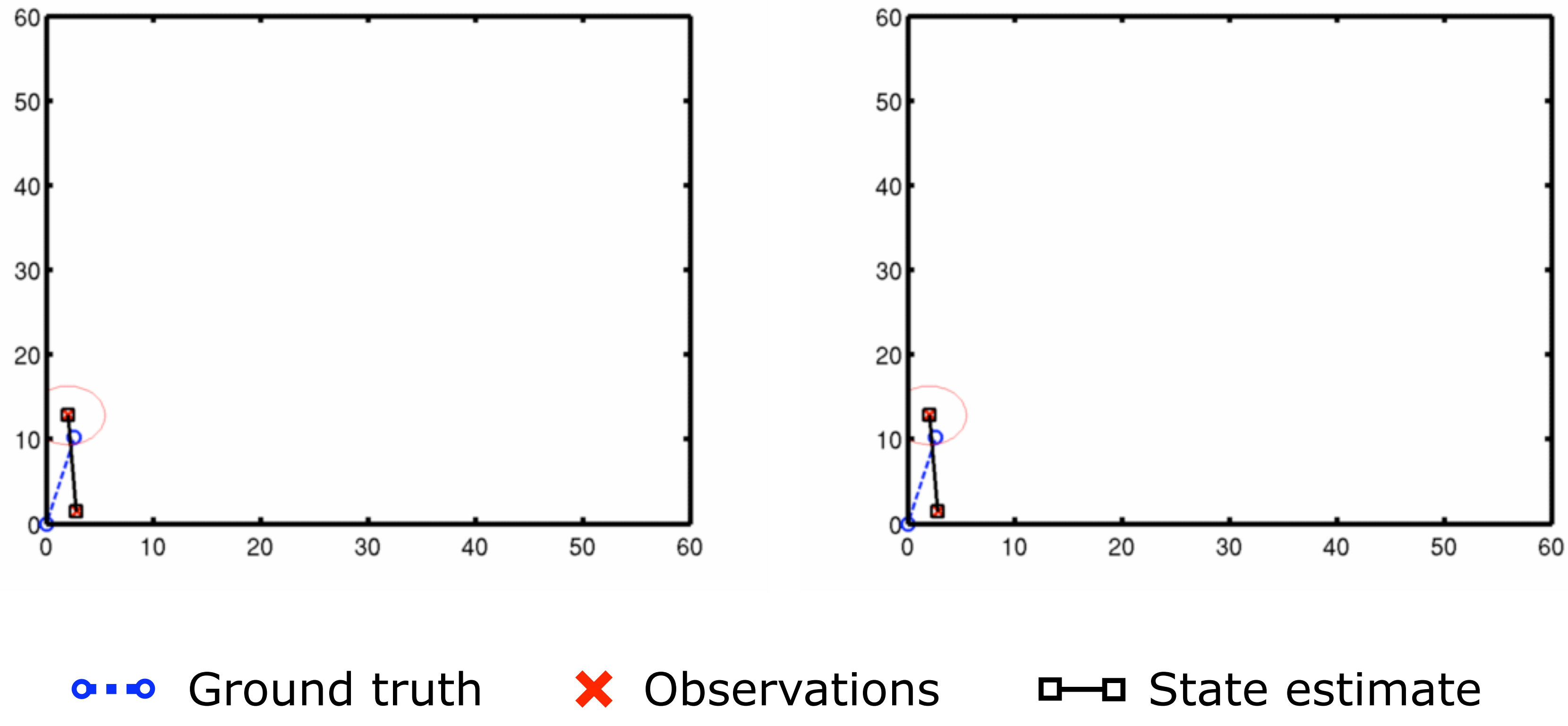
Ball Tracking: Const. Acceleration



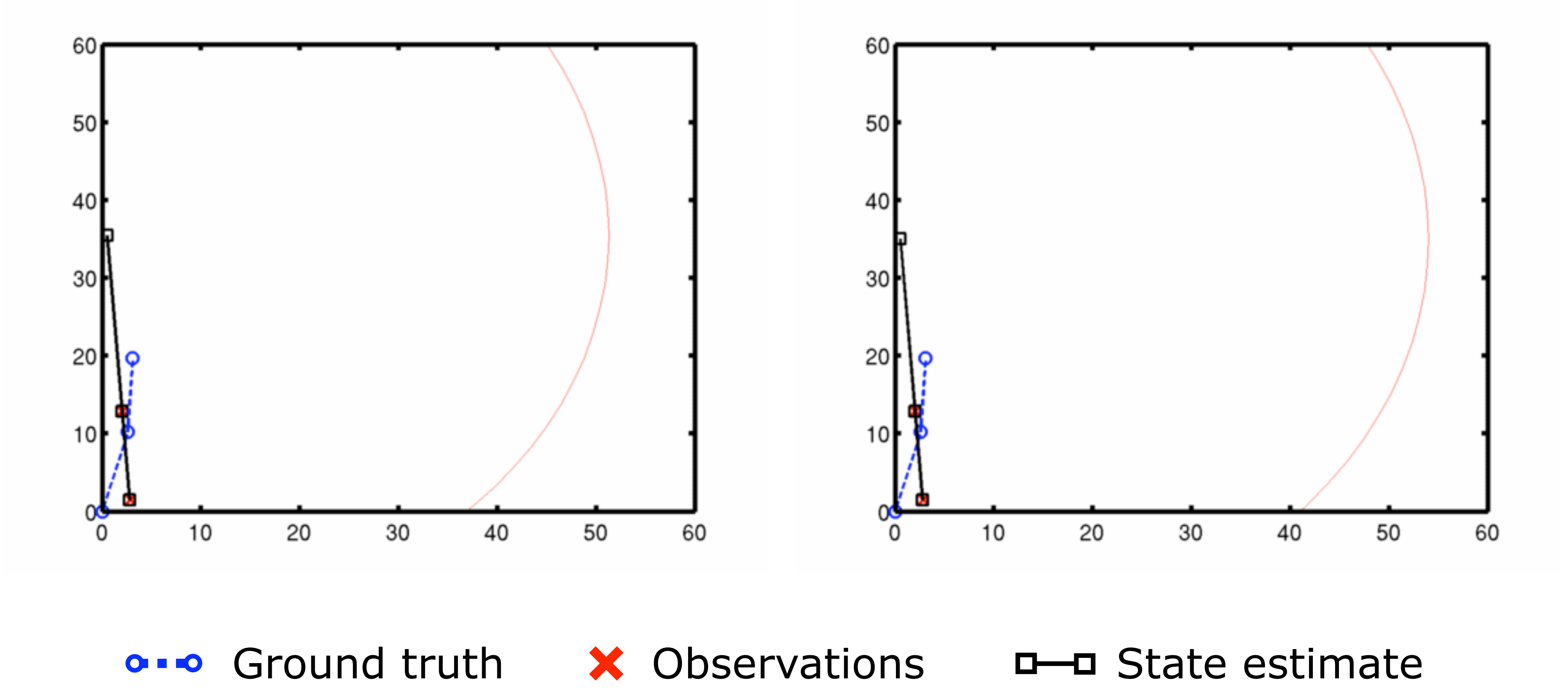
Ball Tracking: Const. Acceleration



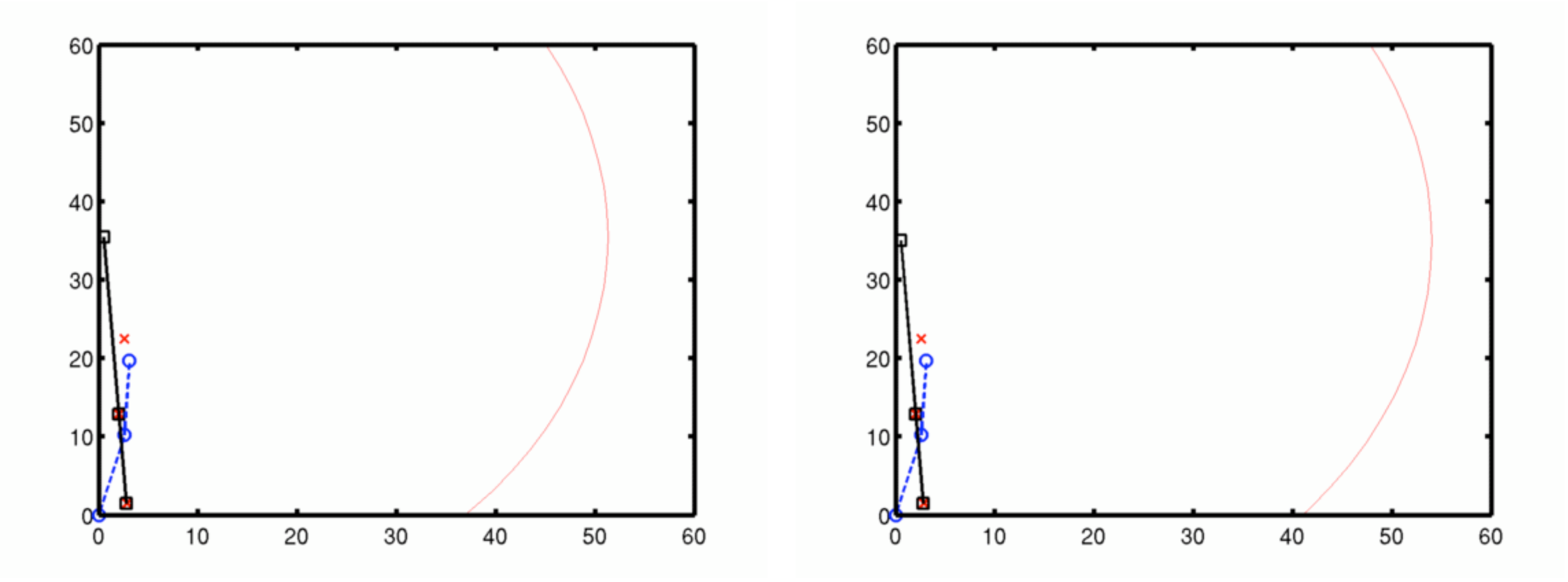
Ball Tracking: Const. Acceleration



Ball Tracking: Const. Acceleration

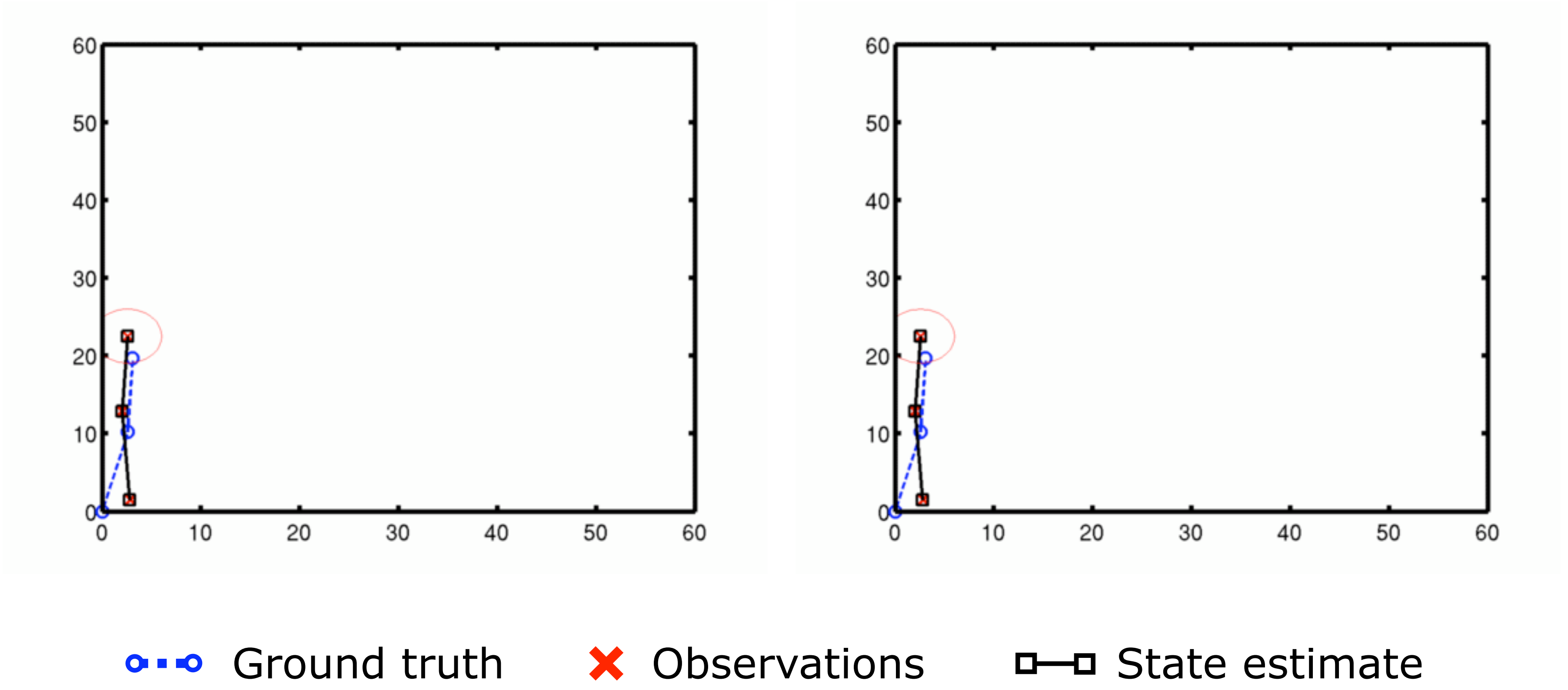


Ball Tracking: Const. Acceleration

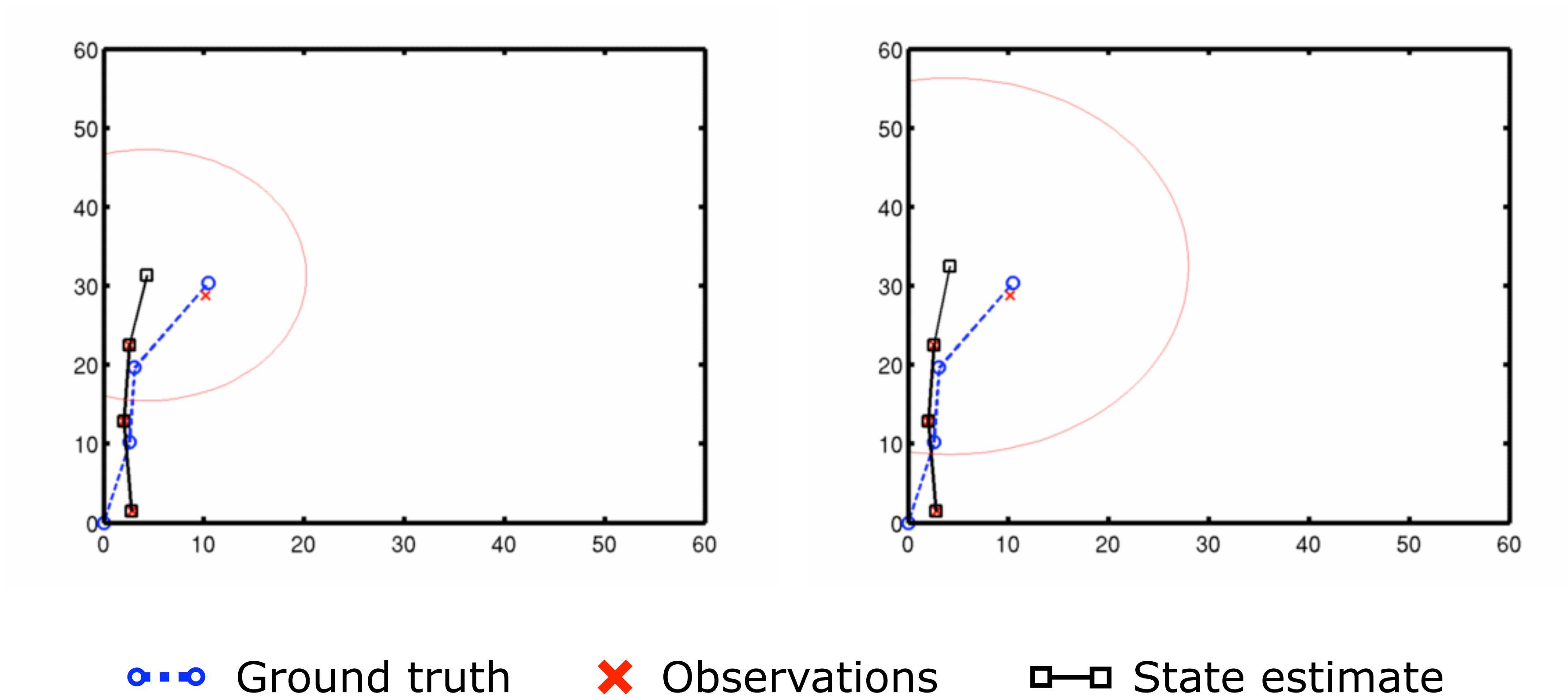


Ground truth   Observations   State estimate

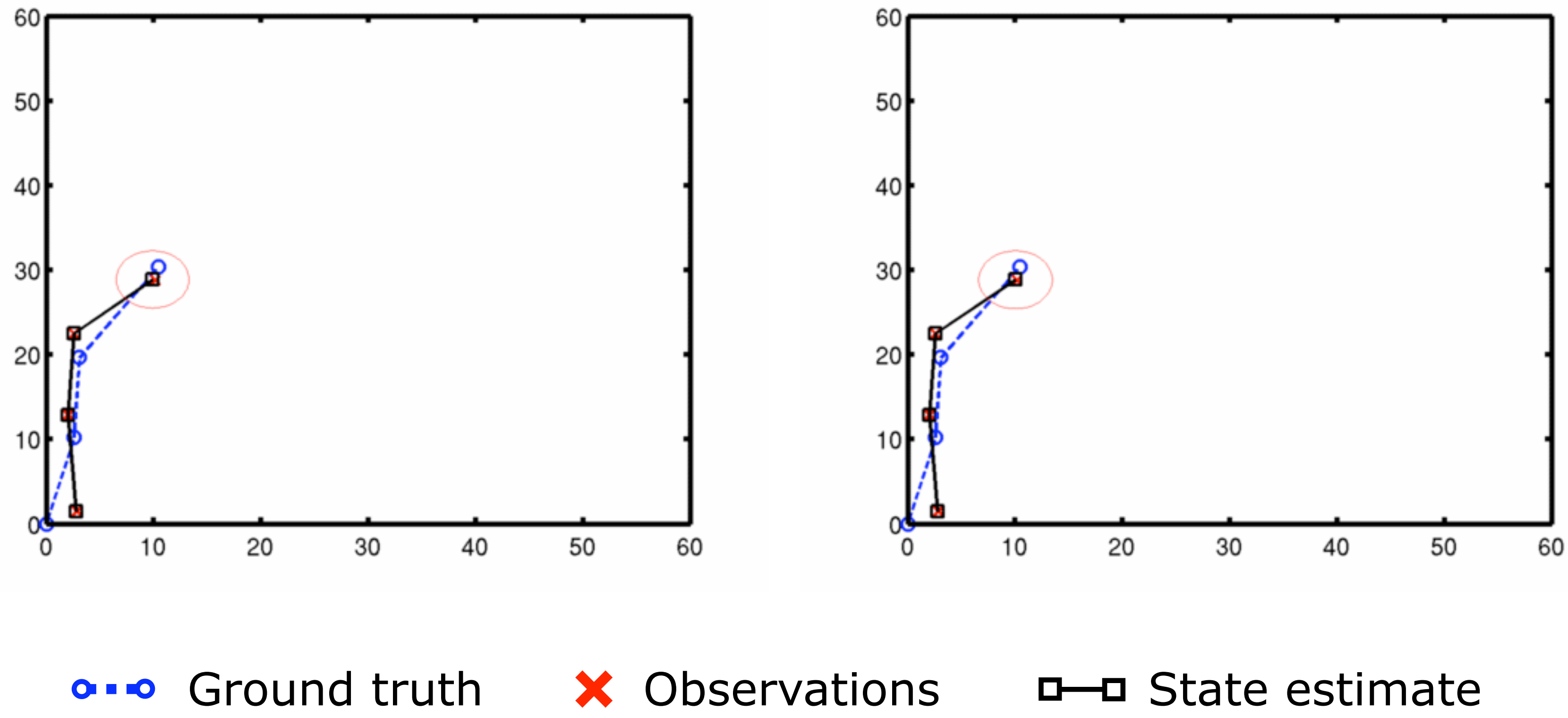
Ball Tracking: Const. Acceleration



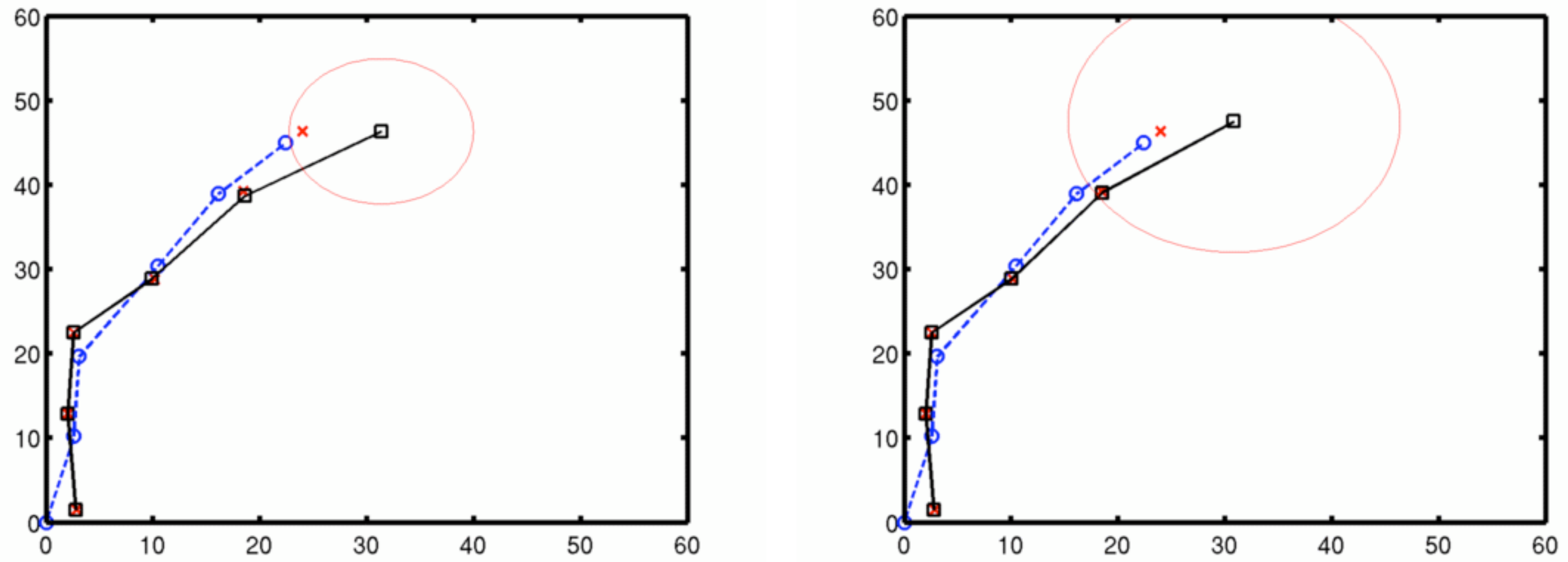
Ball Tracking: Const. Acceleration



Ball Tracking: Const. Acceleration

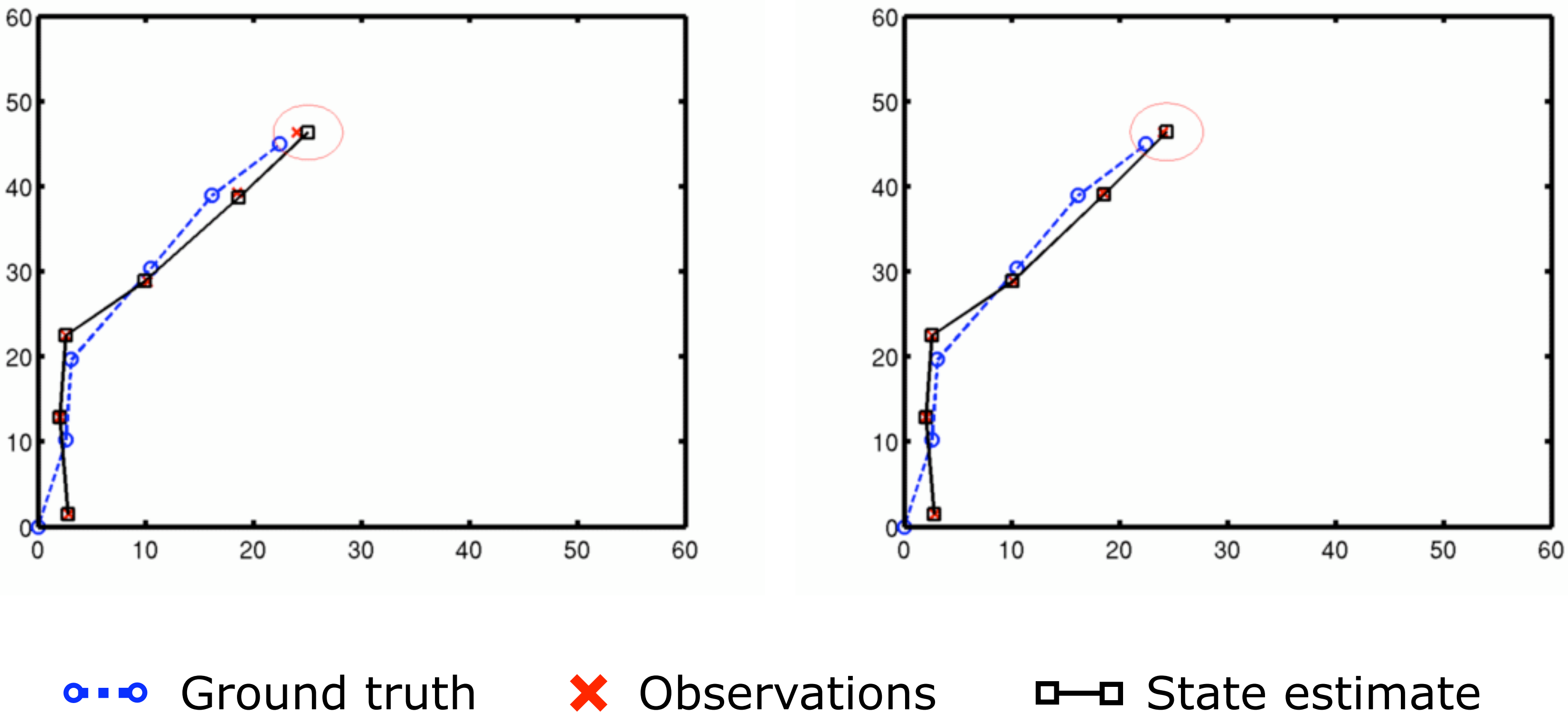


Ball Tracking: Const. Acceleration

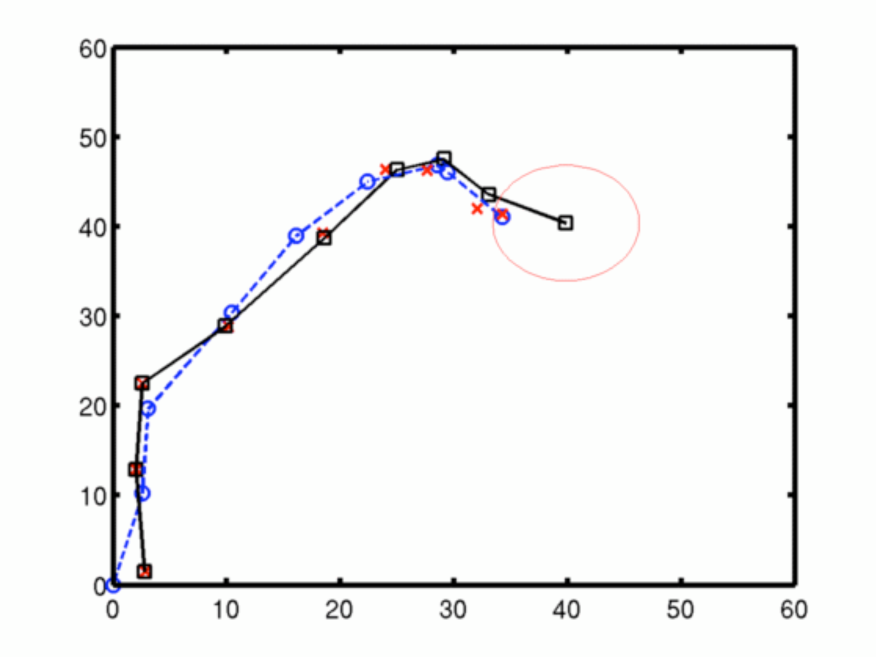


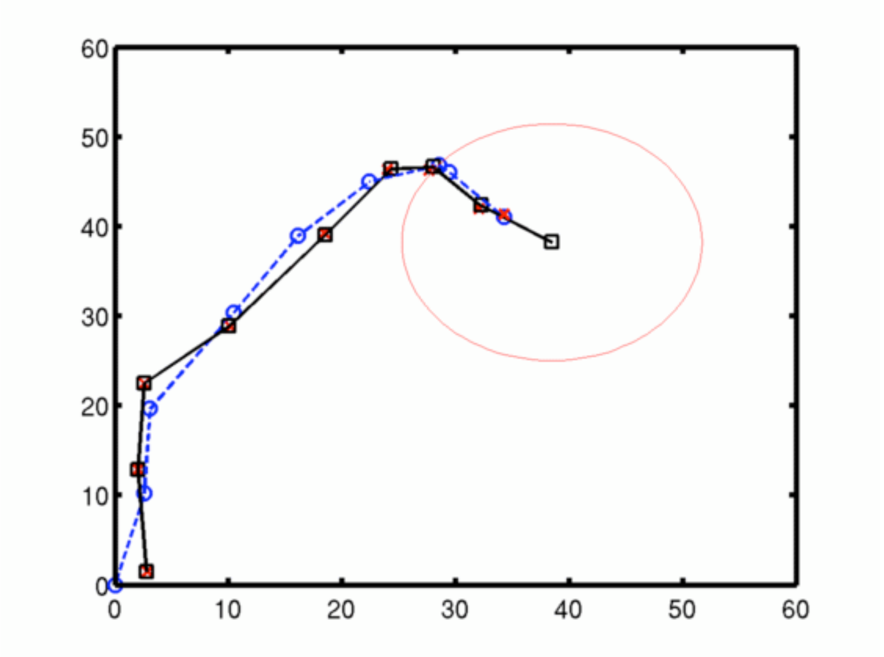
Ground truth   Observations   State estimate

Ball Tracking: Const. Acceleration



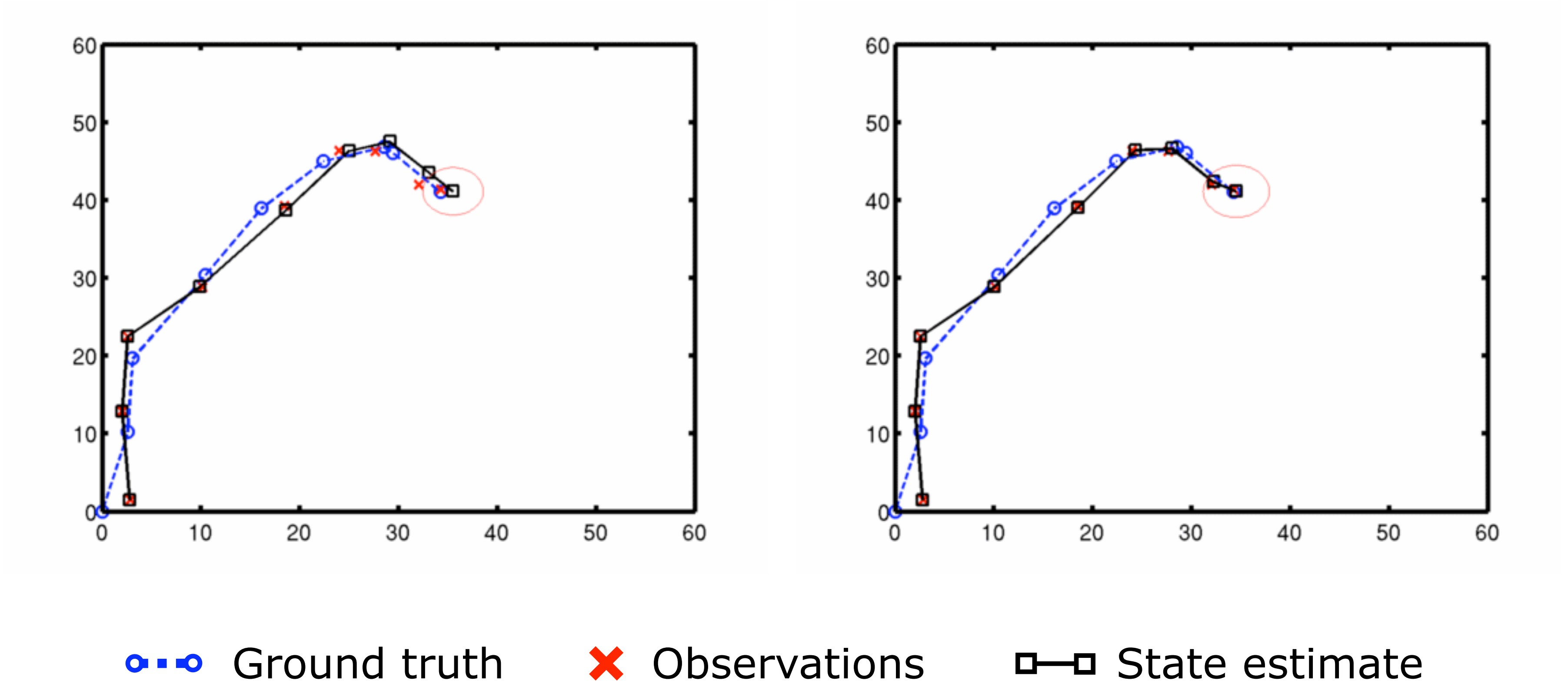
Ball Tracking: Const. Acceleration



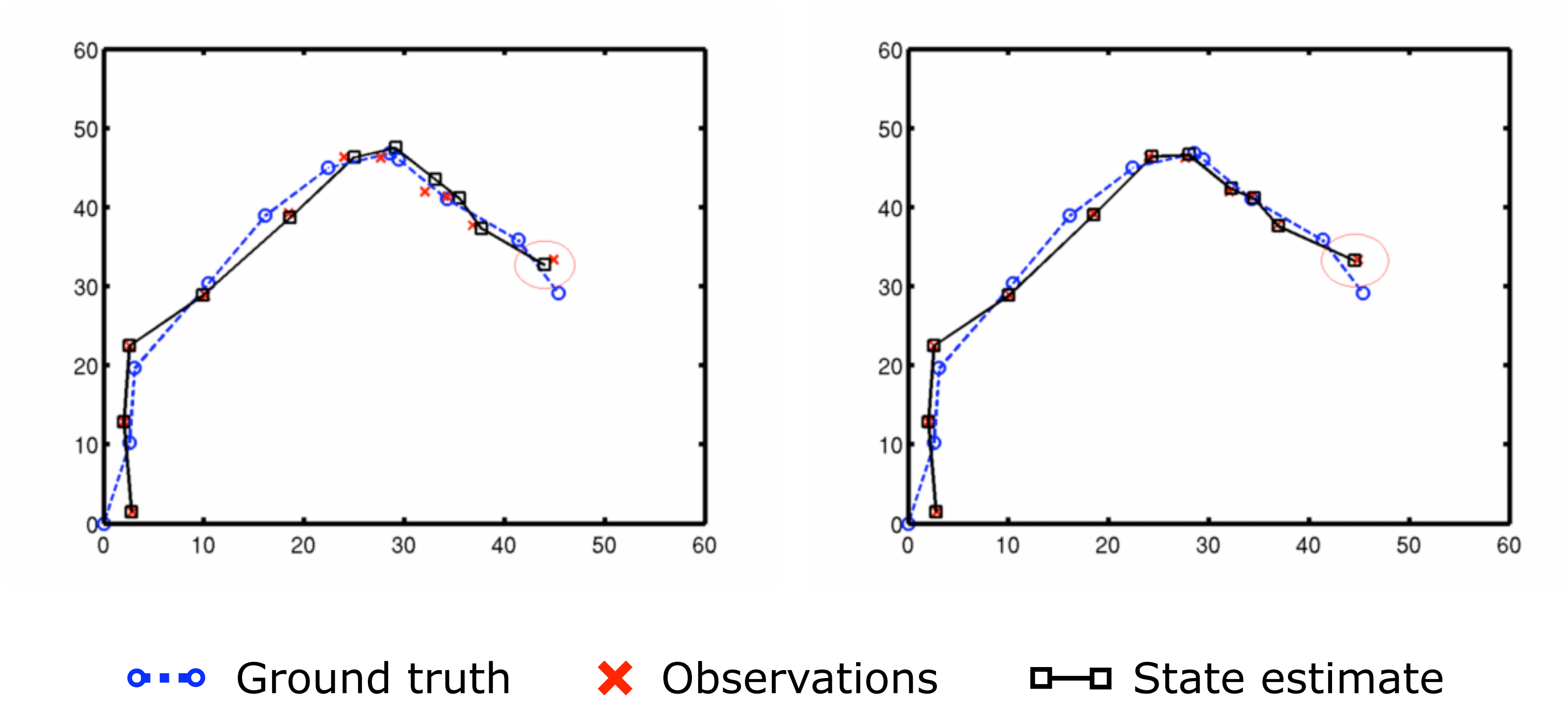


Ground truth Observations State estimate

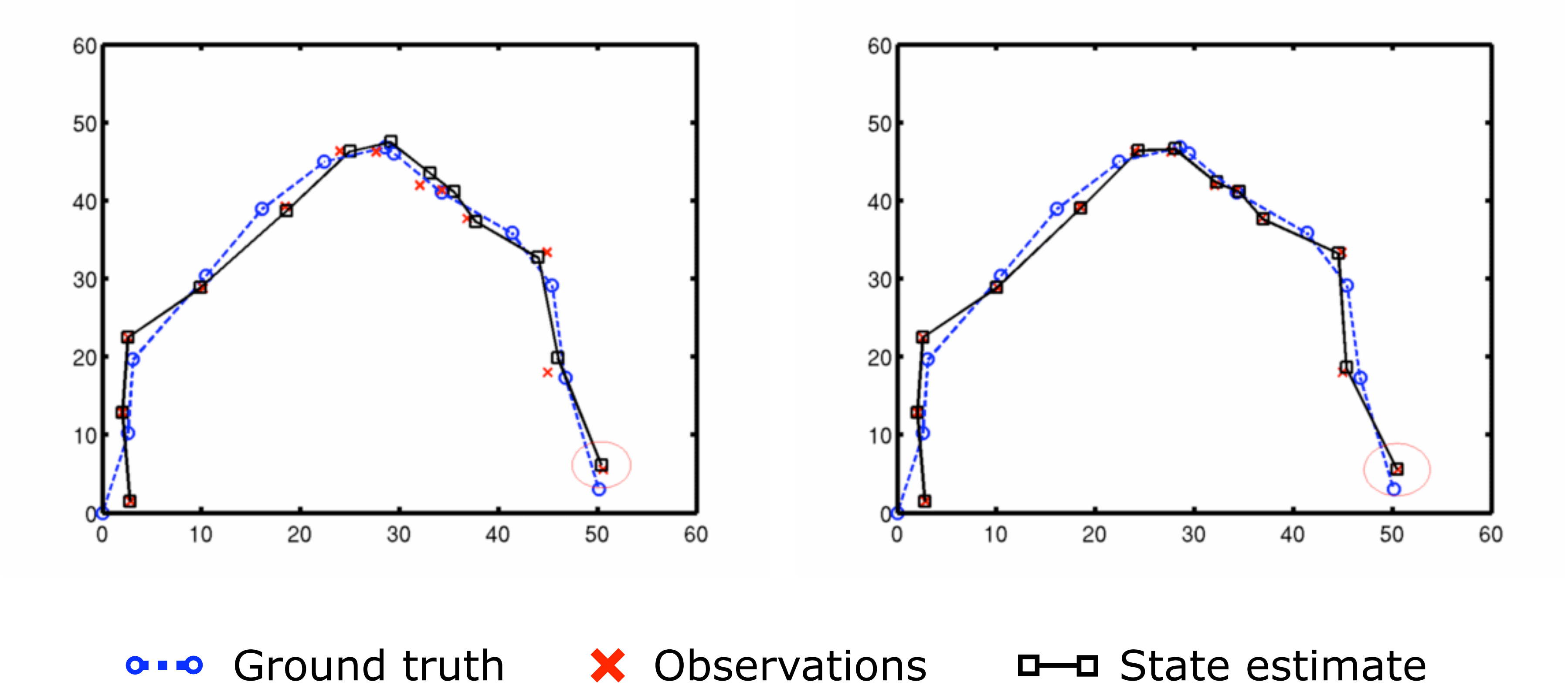
Ball Tracking: Const. Acceleration



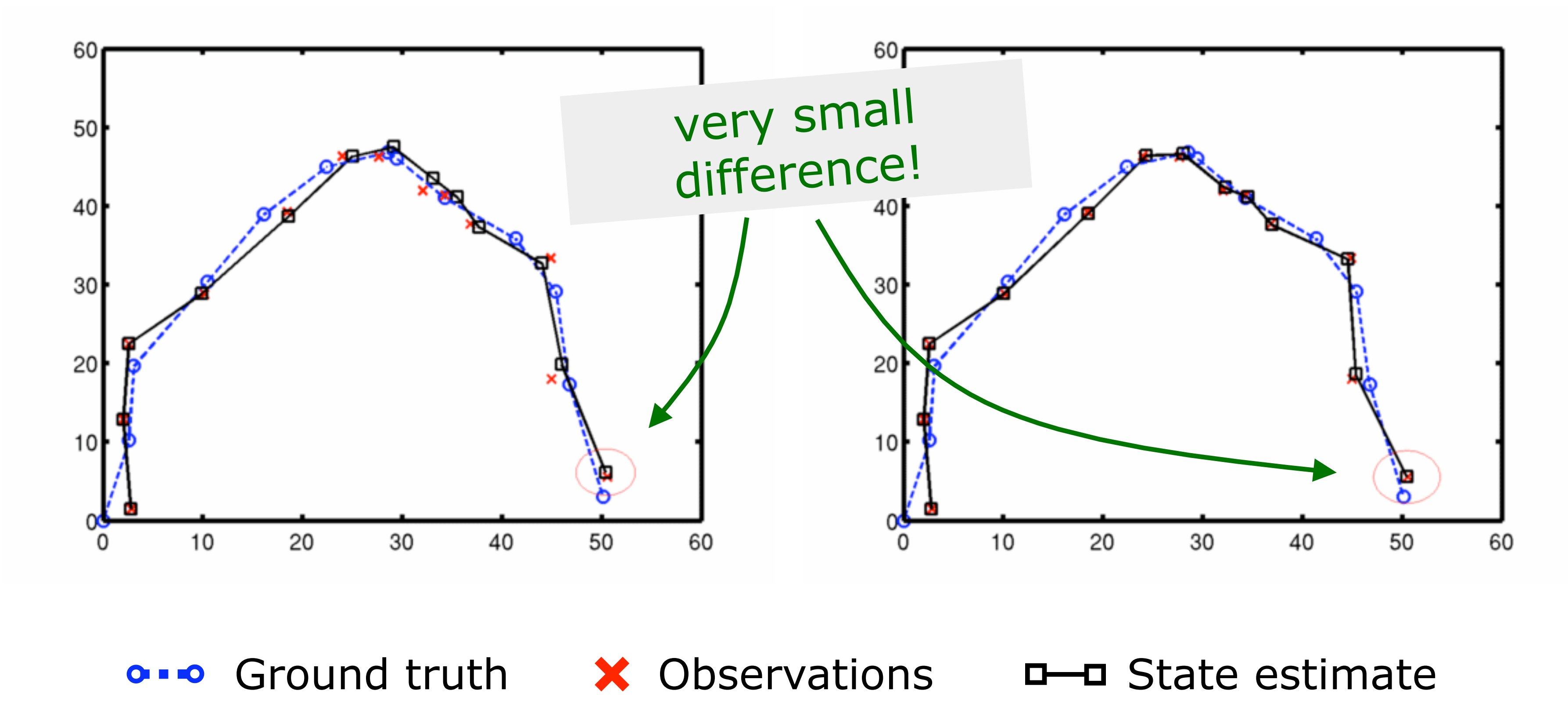
Ball Tracking: Const. Acceleration



Ball Tracking: Const. Acceleration



Ball Tracking: Const. Acceleration



Summary

- Tracking is maintaining the **state** and **identity** of a moving object over time despite **detection errors** (false negatives, false alarms), **occlusions**, and the presence of **other objects**
- **Linear Dynamic Systems** (a.k.a. the state-space representation) provide the mathematical framework for estimation
- For **tracking**, there is **no control input** u in the process model. Therefore good motion models are key
- The **Kalman filter** is a recurse Bayes filter that follows the typical **predict-update cycle**

Summary

- The **Extended Kalman filter** (EKF) is for cases of non-linear process or measurement models. It computes the Jacobians, **first-order linearizations** of the models, and has the same expressions than the KF
- A large process noise covariance can partly **compensate a poor motion model** for maneuvering targets
- But: large process noise covariances cause the validation gates to be large which in turn increases the **level of ambiguity for data association**. This is potentially problematic in case of multiple targets