

# Human-Oriented Robotics

## Temporal Reasoning

---

Part 2/3

Kai Arras

Social Robotics Lab, University of Freiburg

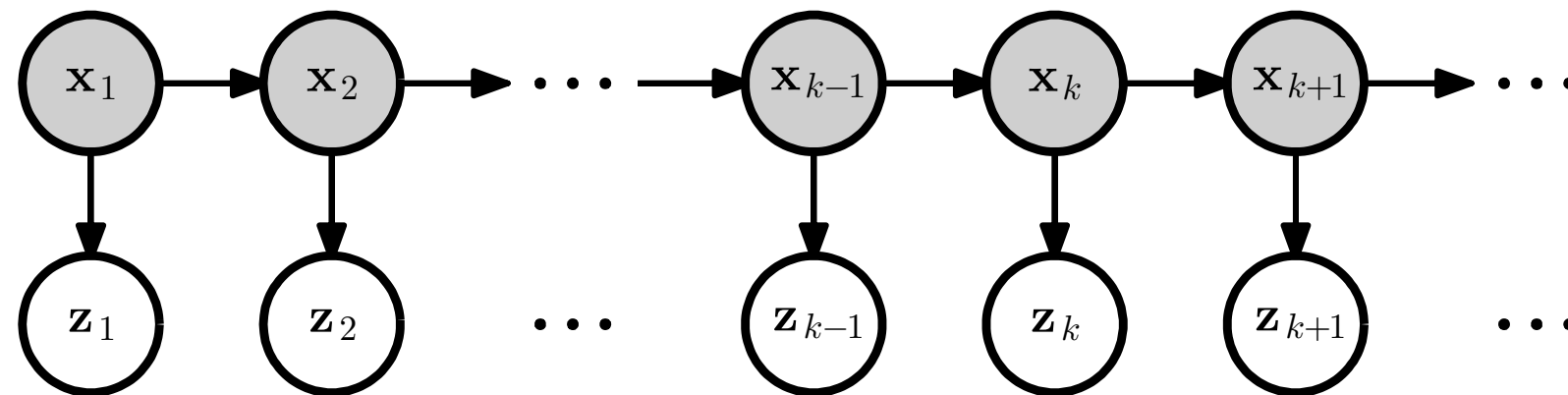
Winter term 2014/2015

## Contents

- Introduction
- Temporal Reasoning
- Hidden Markov Models
- **Linear Dynamical Systems**
- **Kalman Filter**
- **Extended Kalman Filter**
- Tracking and Data Association

## State Space Model

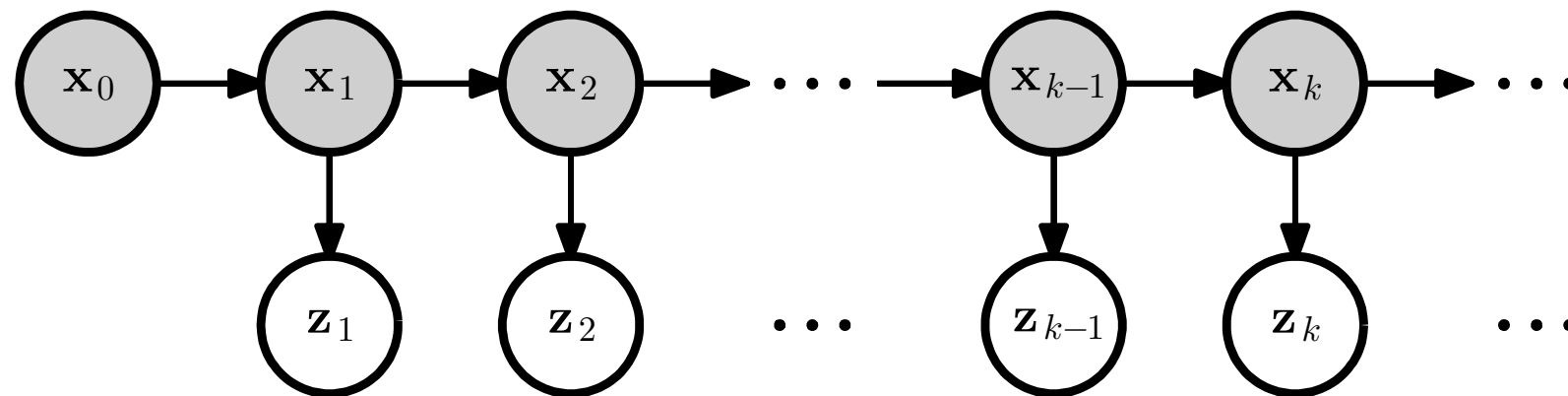
- We recall the **two models for sequential data** described by this graph



1. **Discrete case:** if the latent variables are discrete, we obtain a hidden Markov model (**HMM**)
2. **Continuous case:** If both the latent and the observed variables are continuous and Gaussian, we have a linear dynamical system (**LDS**)

## State Space Model

- We also recall the **three parameters** of a state space model



$$p(\mathbf{x}_{0:K}, \mathbf{z}_{1:K}) = p(\mathbf{x}_0) \prod_{k=1}^K p(\mathbf{x}_k | \mathbf{x}_{k-1}) p(\mathbf{z}_k | \mathbf{x}_k)$$

Prior


Transition  
model

Observation  
model

## State Space Model

- HMMs correspond to the state space model in which latent variables are **discrete**. However, the model describes a much broader class of probability distributions, all of which factorize according to the above equation
- We will now consider **Gaussian distributions**, the most important distribution for this purpose from a practical perspective
- In particular, we will consider the **linear-Gaussian state space model** where the latent variables  $\mathbf{x}$  and the observations  $\mathbf{z}$  are **multivariate Gaussians** whose means are **linear functions** of their parents in the graph

Multivariate Gaussians


$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}_{\mathbf{x}_k}(F_k \mathbf{x}_{k-1}, P'_k)$$

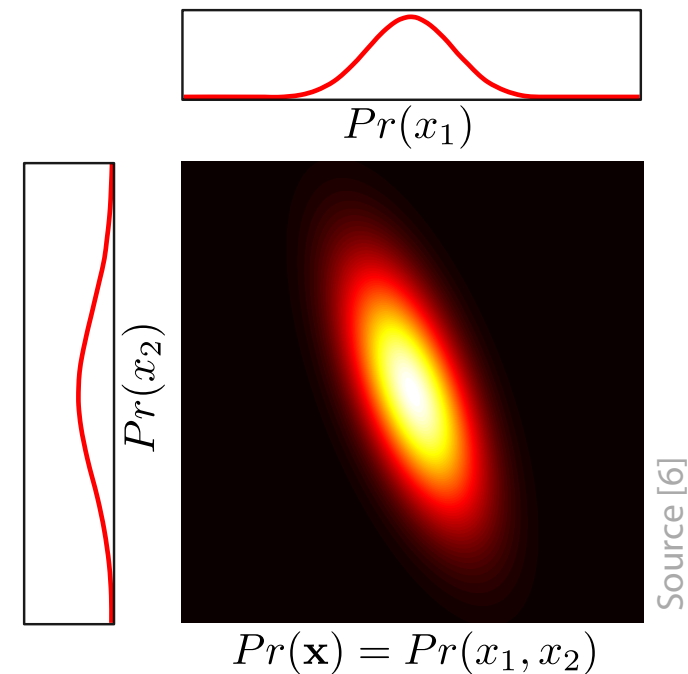
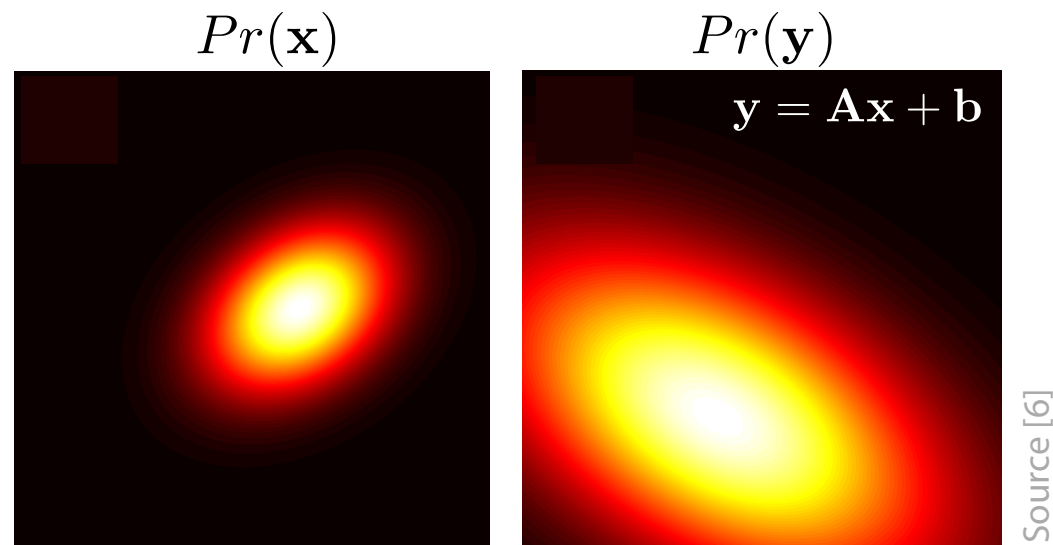
$$p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}_{\mathbf{z}_k}(H_k \mathbf{x}_k, R'_k)$$

Linear functions



## State Space Model

- What's so special about the linear-Gaussian assumption?
  - 1) **Gaussian stays Gaussian** under linear transformations  
(proven later in this course)
  - 2) Given a Gaussian joint distribution, all derived **marginal distributions** are **Gaussian** as well

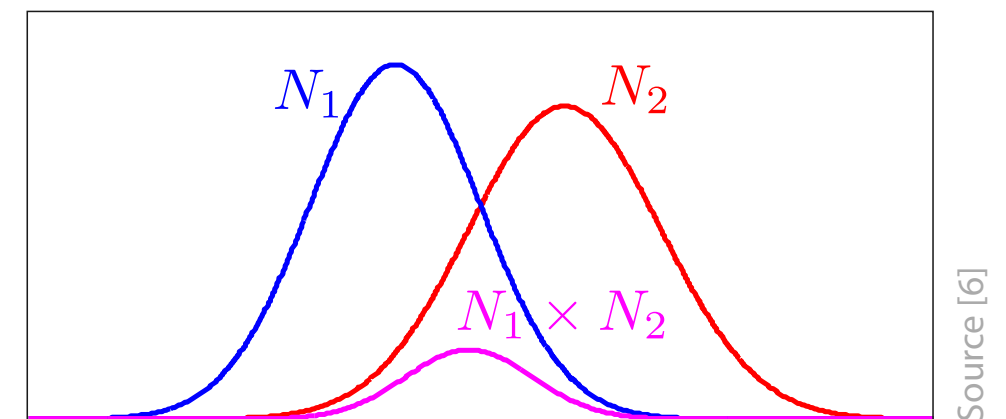
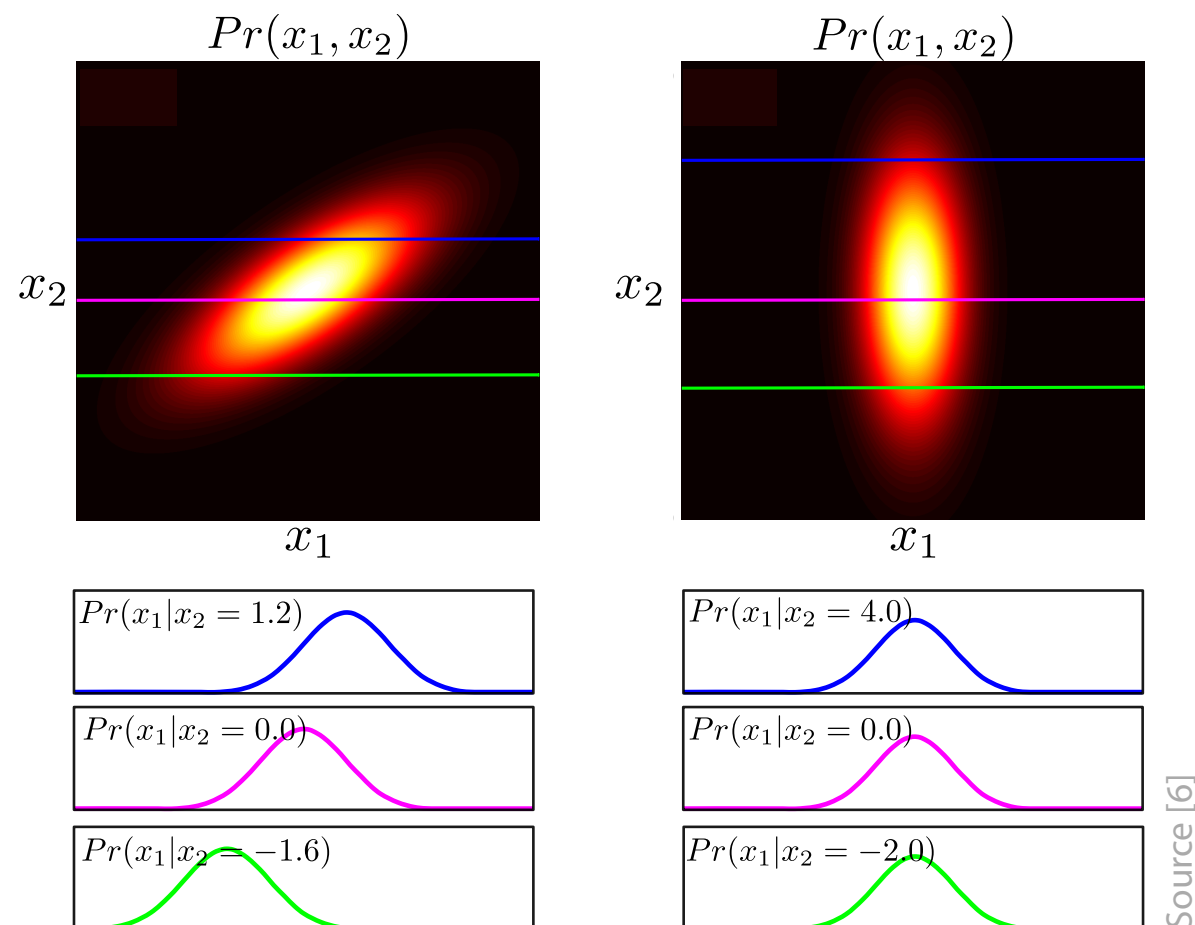


## State Space Model

- What's so special about the linear-Gaussian assumption?

3) Given a Gaussian joint distribution, all derived **conditional distributions** are **Gaussian** as well

4) The **product** of two Gaussian distributions is also a **Gaussian** distribution



Source [6]

## Linear Dynamical Systems

- These properties ensure that
  - we can **always deal with Gaussian distributions**
  - “the linear-Gaussian family remains closed”
  - Inference processes do not become more complex along the chain with more incoming observations
- A temporal model under the linear-Gaussian assumption is called **linear dynamical system (LDS)**
- Let us first consider the **representation** of a linear dynamical system



## LDS Representation

- The **transition model** of an LDS is

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}_{\mathbf{x}_k}(F_k \mathbf{x}_{k-1}, P'_k)$$

which implies the following linear model that describes how the world or system evolves (also called plant or process model)

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + \mathbf{v}_k$$

- Matrix  $F \in \mathbb{R}^{n_x \times n_x}$  is the **state transition matrix**
- Vector  $\mathbf{v} \in \mathbb{R}^{n_x \times 1}$  is the zero-mean Gaussian **process noise** with

$$\mathbf{v} \sim \mathcal{N}_{\mathbf{v}}(0, Q) \quad \text{i.e.} \quad \mathbb{E}[\mathbf{v}] = 0 \quad \text{Var}[\mathbf{v}] = Q$$

- Matrix  $Q \in \mathbb{R}^{n_x \times n_x}$  is the **process noise covariance**

## LDS Representation

- The **observation model** of an LDS is

$$p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}_{\mathbf{z}_k}(H_k \mathbf{x}_k, R'_k)$$

which implies the following linear relationship between states and observations through which the system can be observed remotely

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{w}_k$$

- Matrix  $H \in \mathbb{R}^{n_z \times n_x}$  is the **observation matrix** (note its dimension)
- Vector  $\mathbf{w} \in \mathbb{R}^{n_z \times 1}$  is the zero-mean Gaussian **observation noise** with

$$\mathbf{w} \sim \mathcal{N}_{\mathbf{w}}(0, R) \quad \text{i.e.} \quad \mathbb{E}[\mathbf{w}] = 0 \quad \text{Var}[\mathbf{w}] = R$$

- Matrix  $R \in \mathbb{R}^{n_z \times n_z}$  is the **observation noise covariance**

## LDS Representation

- The **prior distribution** for time index 0 is also Gaussian

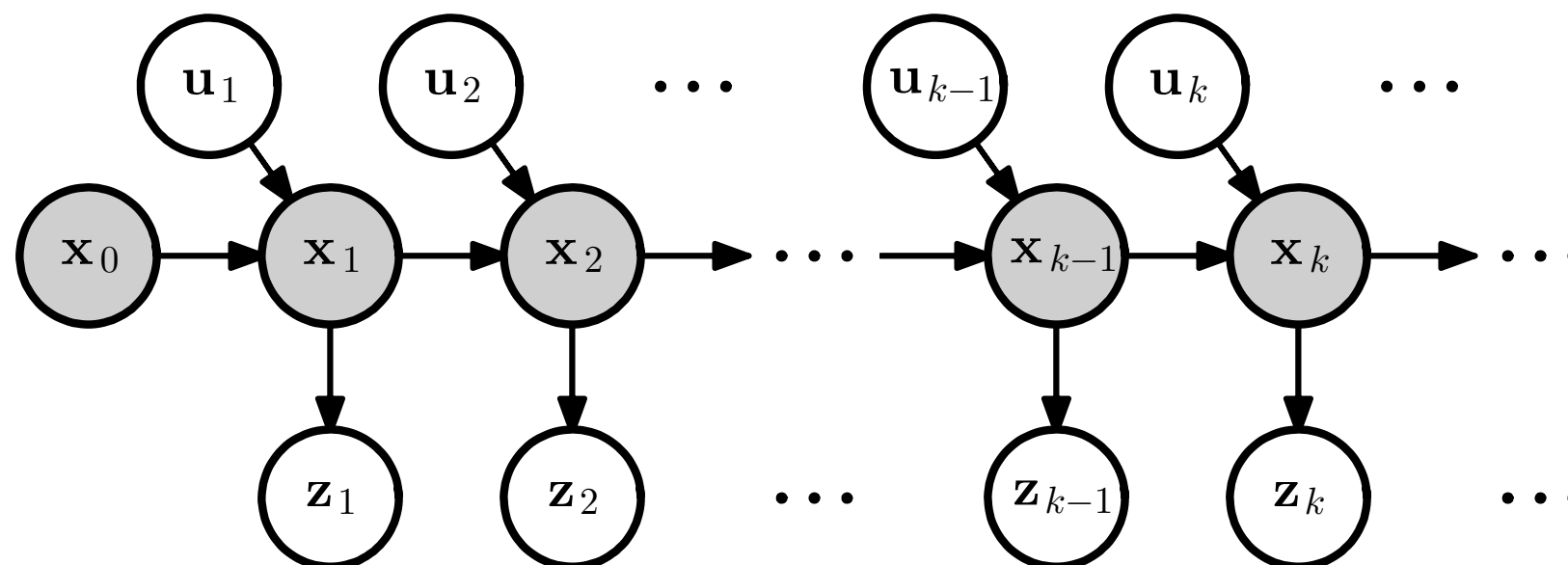
$$p(\mathbf{x}_0) = \mathcal{N}_{\mathbf{x}_0}(\boldsymbol{\mu}_0, P_0)$$

with parameters  $\boldsymbol{\mu}_0 \in \mathbb{R}^{n_x \times 1}$ ,  $P_0 \in \mathbb{R}^{n_x \times n_x}$

- So far, we have used time-varying matrices and noise sources  $F_k, \mathbf{v}_k, H_k, \mathbf{w}_k$ . For notation simplicity, we will assume the models to be **time-invariant** and the noise variables to be **stationary**, i.e.  $F, \mathbf{v}, H, \mathbf{w}$
- Finally, an important assumption of LDS is that all noise variables are **mutually uncorrelated**

## LDS Representation

- In general, LDS can have an **external control input**  $u_k$
- Allows to describe a flexible class of dynamical systems that both **evolve on their own** and are **controlled by an external influence**
- The corresponding graphical model



## LDS Representation

- The transition model **with control input**

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + G_k \mathbf{u}_k + \mathbf{v}_k$$

postulates a linear relationship between states and controls

- Vector  $\mathbf{u} \in \mathbb{R}^{n_u \times 1}$  is the **control input**
- Matrix  $G \in \mathbb{R}^{n_x \times n_u}$  is the **input gain matrix**

## LDS Representation Summary

- The system is governed by

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + G_k \mathbf{u}_k + \mathbf{v}_k$$

- $\mathbf{x} \in \mathbb{R}^{n_x \times 1}$ : **state vector**
- $\mathbf{u} \in \mathbb{R}^{n_u \times 1}$ : **control input**
- $\mathbf{v} \in \mathbb{R}^{n_x \times 1}$ : **process noise**
- $F \in \mathbb{R}^{n_x \times n_x}$ : **transition matrix**
- $G \in \mathbb{R}^{n_x \times n_u}$ : **input gain matrix**
- $Q \in \mathbb{R}^{n_x \times n_x}$ : **process noise cov.**

- The system can be observed by

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{w}_k$$

- $\mathbf{z} \in \mathbb{R}^{n_z \times 1}$ : **observation vector**
- $\mathbf{w} \in \mathbb{R}^{n_z \times 1}$ : **observation noise**
- $H \in \mathbb{R}^{n_z \times n_x}$ : **observation matrix**
- $R \in \mathbb{R}^{n_z \times n_z}$ : **observation noise cov.**

## LDS Example: Throwing a Ball

- We want to throw a ball and **compute its trajectory**. This can be easily done with an LDS
- The LDS describes the physics of the process. No uncertainties/covariances, no tracking
- The ball's **state** is represented as

$$\mathbf{x} = (x \quad y \quad \dot{x} \quad \dot{y})^T$$

- We have the **gravity force**  $g$  as external influence

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

- We assume windless conditions and ignore floor constraints
- We **observe** the ball with a **noise-free position sensor**

$$\mathbf{z} = (x \quad y)^T$$



## LDS Example: Throwing a Ball

- The **physics** of the process

$$x_k = x_{k-1} + \dot{x}_{k-1} \Delta t$$

$$y_k = y_{k-1} + \dot{y}_{k-1} \Delta t - \frac{\Delta t^2}{2} g$$

$$\dot{x}_k = \dot{x}_{k-1}$$

$$\dot{y}_k = \dot{y}_{k-1} - \Delta t g$$

- This can be written in matrix form as

$$\mathbf{x}_k = F \mathbf{x}_{k-1} + G \mathbf{u}$$

$$F = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$G = \begin{pmatrix} 0 \\ \Delta t^2/2 \\ 0 \\ \Delta t \end{pmatrix}$$





## LDS Example: Throwing a Ball

- The **physics** of the process

$$x_k = x_{k-1} + \dot{x}_{k-1} \Delta t$$

$$y_k = y_{k-1} + \dot{y}_{k-1} \Delta t - \frac{\Delta t^2}{2} g$$

$$\dot{x}_k = \dot{x}_{k-1}$$

$$\dot{y}_k = \dot{y}_{k-1} - \Delta t g$$



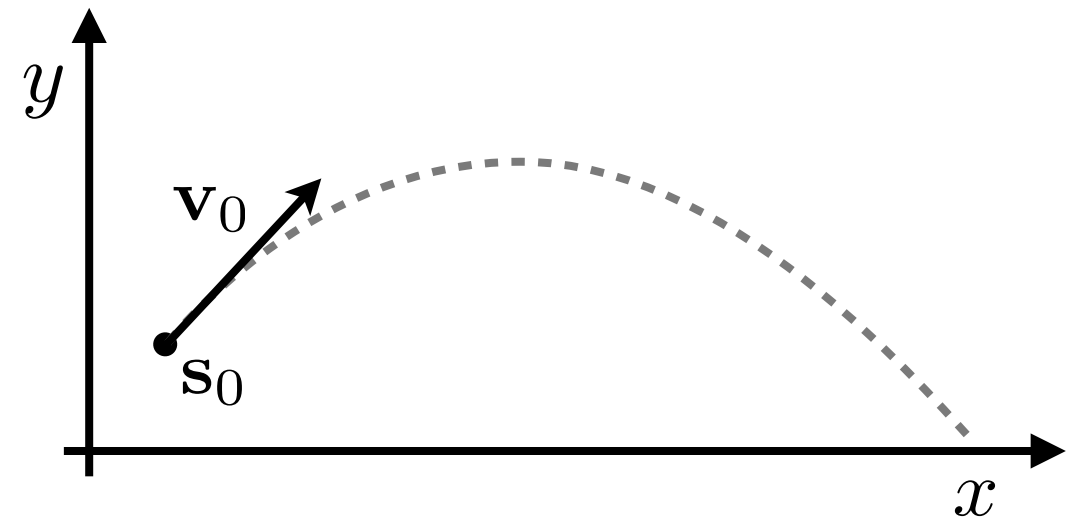
- This can be written in matrix form as

$$\mathbf{x}_k = F \mathbf{x}_{k-1} + G \mathbf{u}$$

$$\begin{pmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{pmatrix} + \begin{pmatrix} 0 \\ \Delta t^2/2 \\ 0 \\ \Delta t \end{pmatrix} \cdot -g$$

## LDS Example: Throwing a Ball

- Throwing a ball from  $s_0 = (x_0, y_0)$  with initial velocity  $v_0 = (\dot{x}_0, \dot{y}_0)$
- Initial state:  $\mathbf{x}_0 = (x_0 \quad y_0 \quad \dot{x}_0 \quad \dot{y}_0)^T$
- Input vector (scalar):  $\mathbf{u} = -g$
- Observation:  $\mathbf{z} = (x \quad y)^T$
- Fixed time step:  $\Delta t$



- Process matrices

$$F = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad G = \begin{pmatrix} 0 \\ \Delta t^2/2 \\ 0 \\ \Delta t \end{pmatrix}$$

- Observation matrix

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

## LDS Example: Throwing a Ball

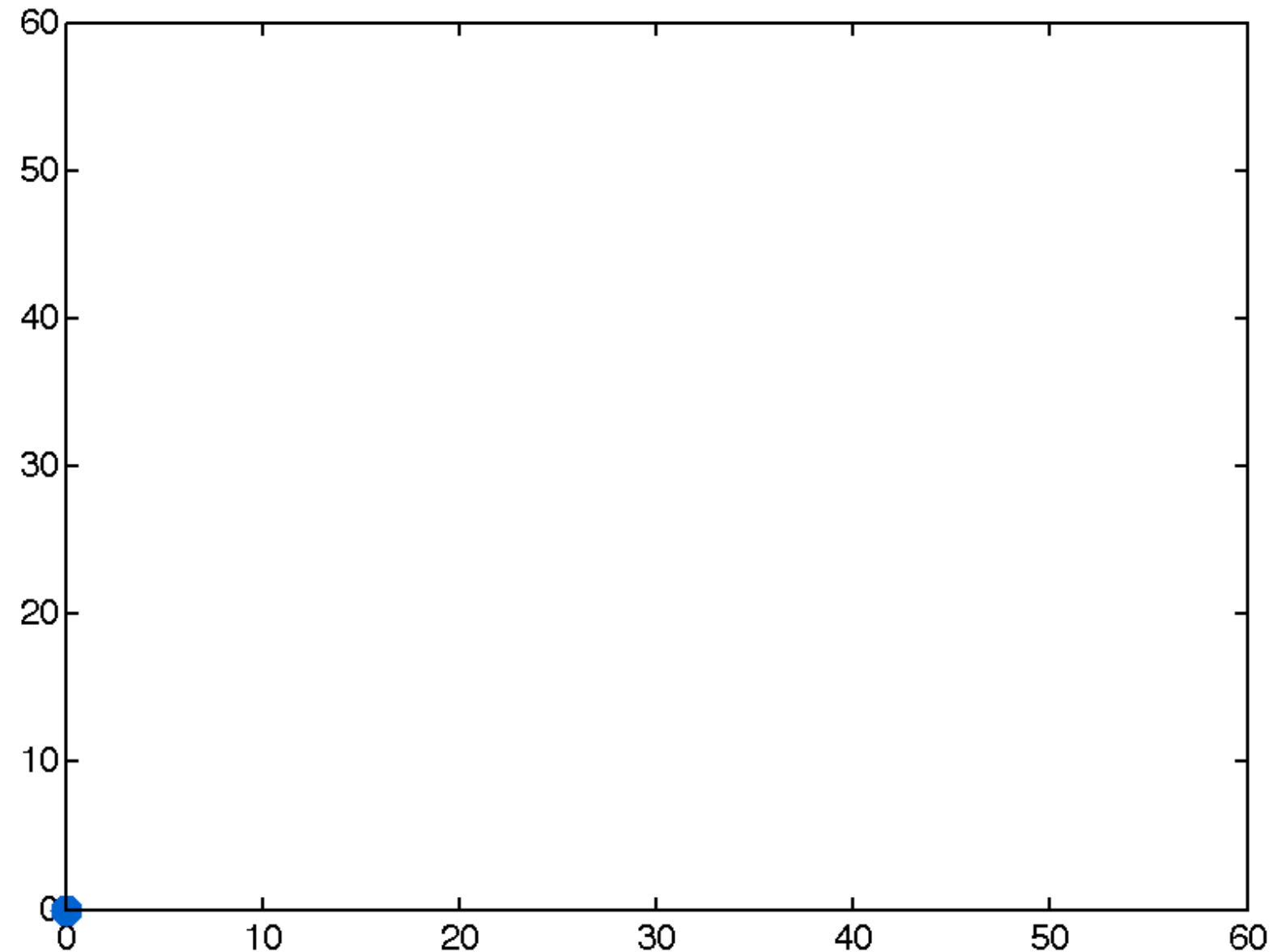
- Initial state

$$\mathbf{x}_0 = \begin{pmatrix} 0 \\ 0 \\ 9 \\ 30 \end{pmatrix}$$

- Time step

$$\Delta t = 0.5$$

- No observations



 System evolution

## LDS Example: Throwing a Ball

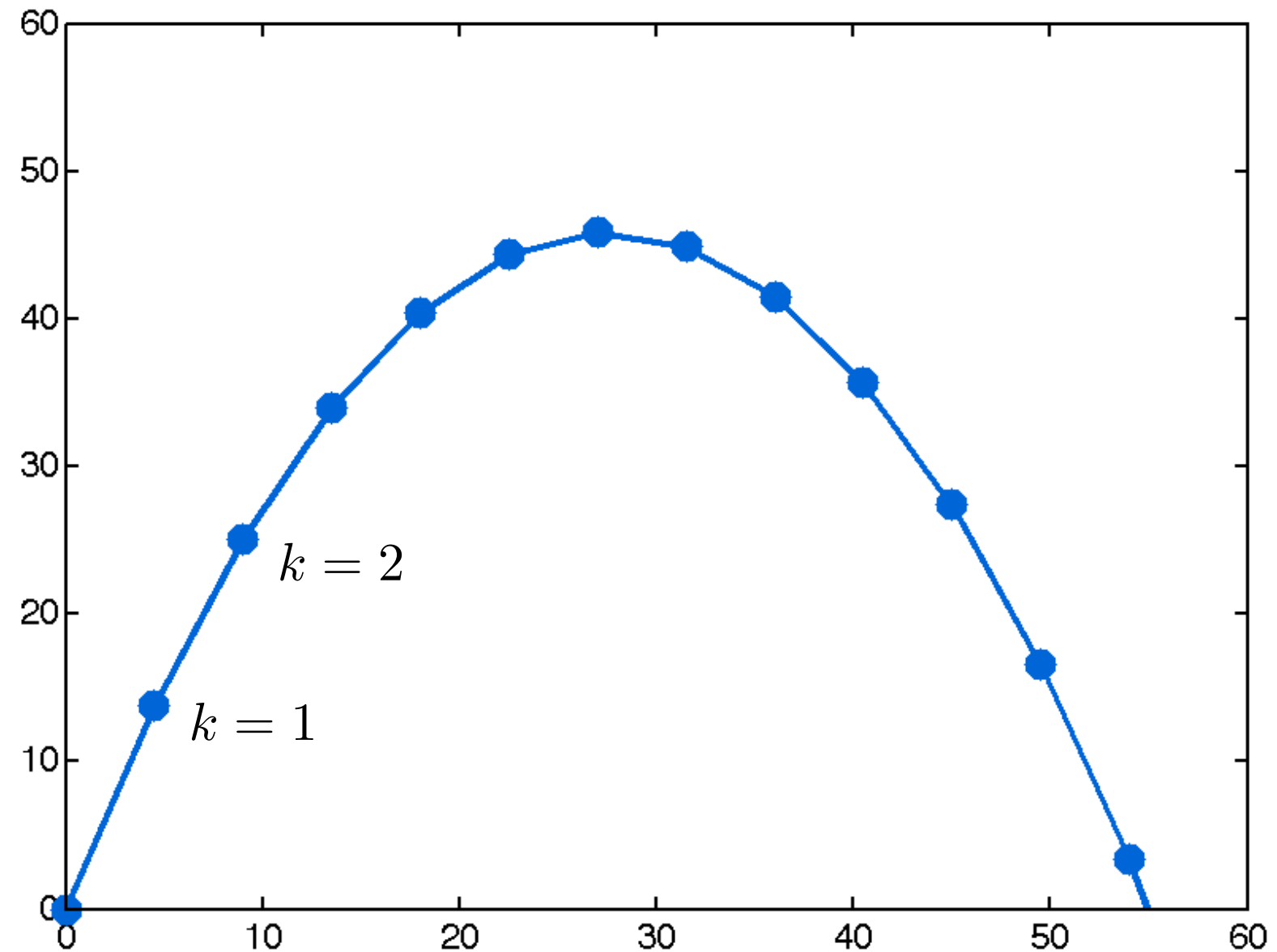
- Initial state

$$\mathbf{x}_0 = \begin{pmatrix} 0 \\ 0 \\ 9 \\ 30 \end{pmatrix}$$

- Time step

$$\Delta t = 0.5$$

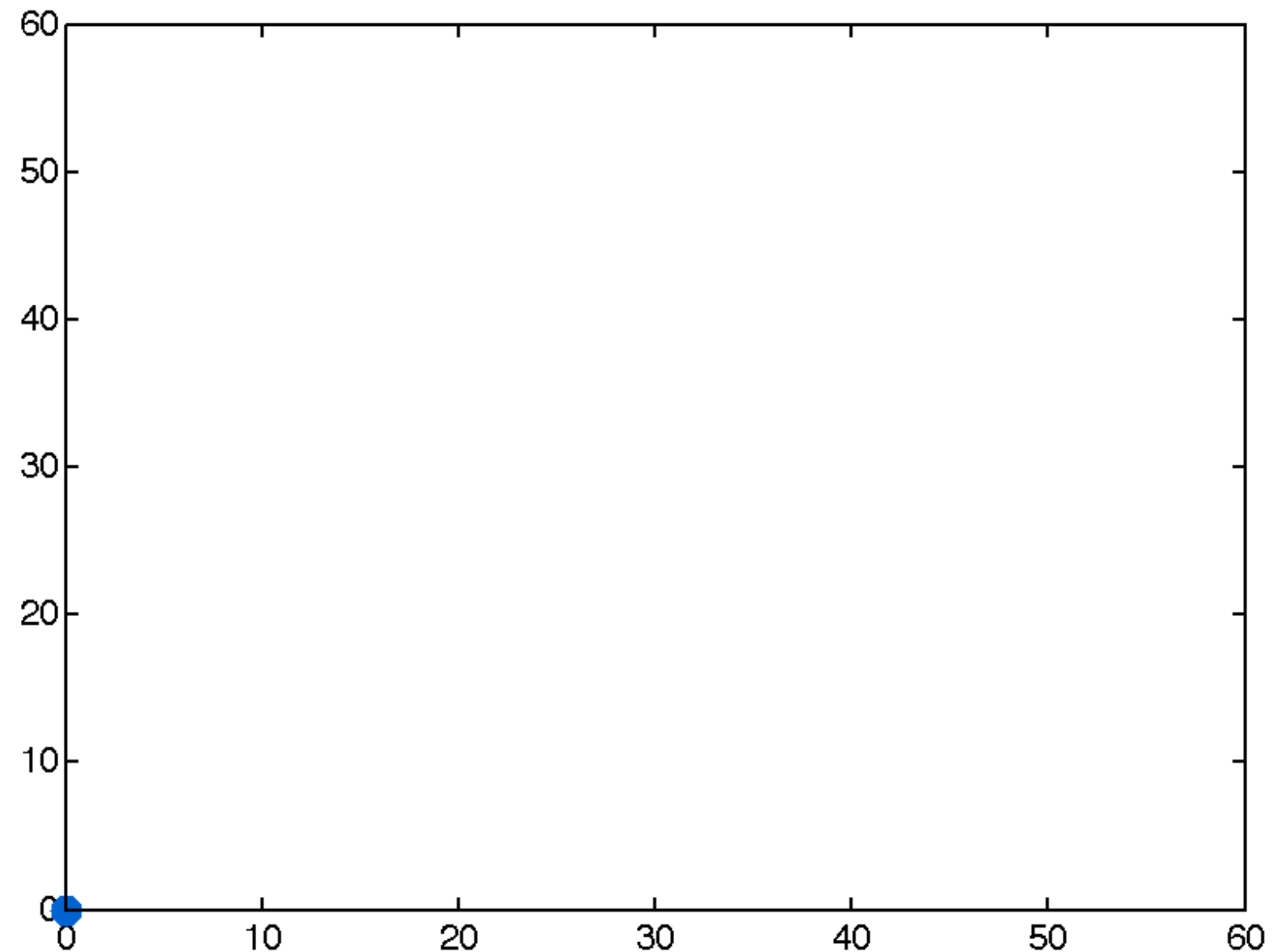
- No observations



 System evolution

## LDS Example: Throwing a Ball

- Observations with **noise-free** sensor



System evolution

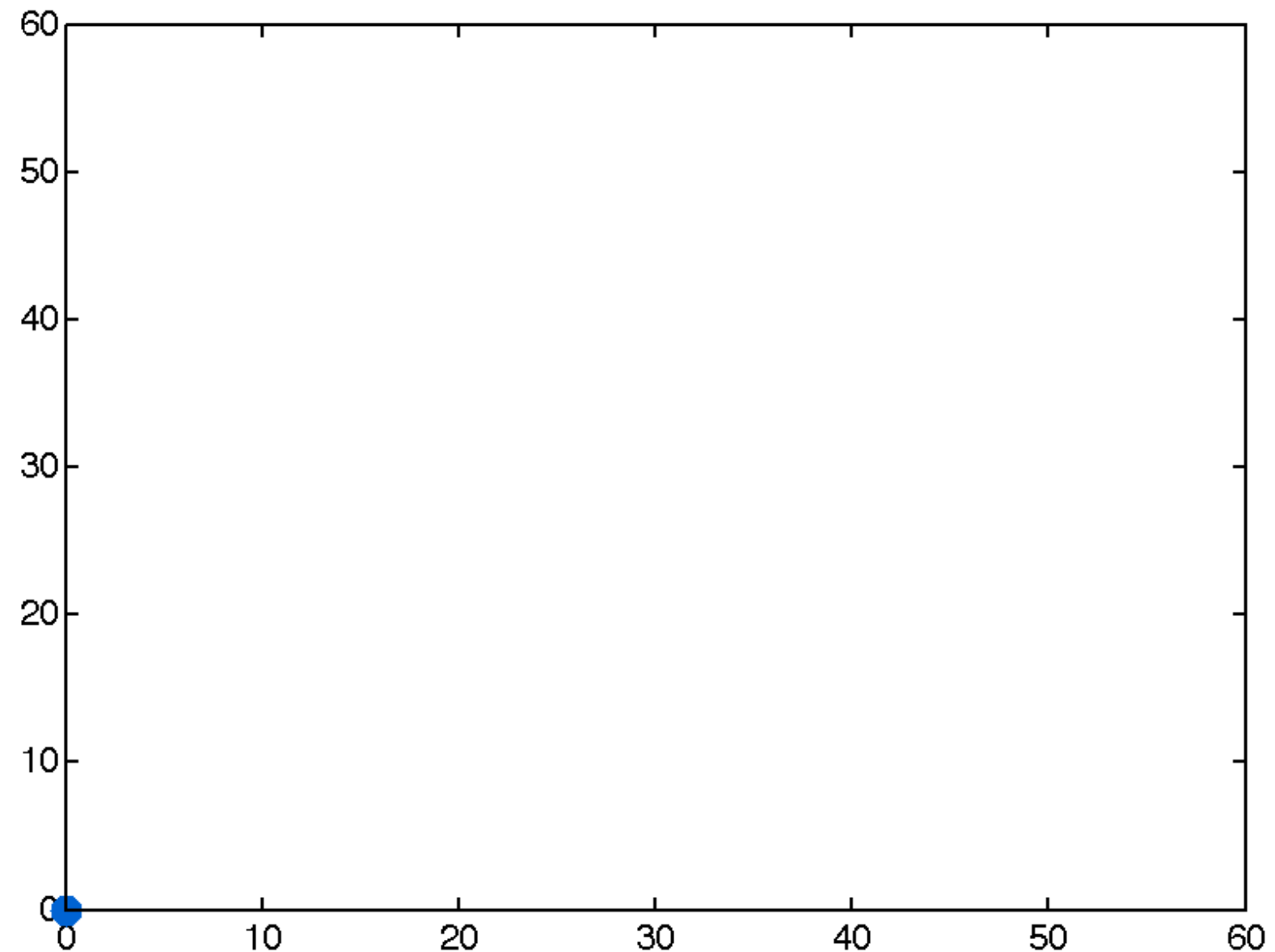


Observations

## LDS Example: Throwing a Ball

- Observations with **noisy** sensor

$$R = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$



System evolution



Observations

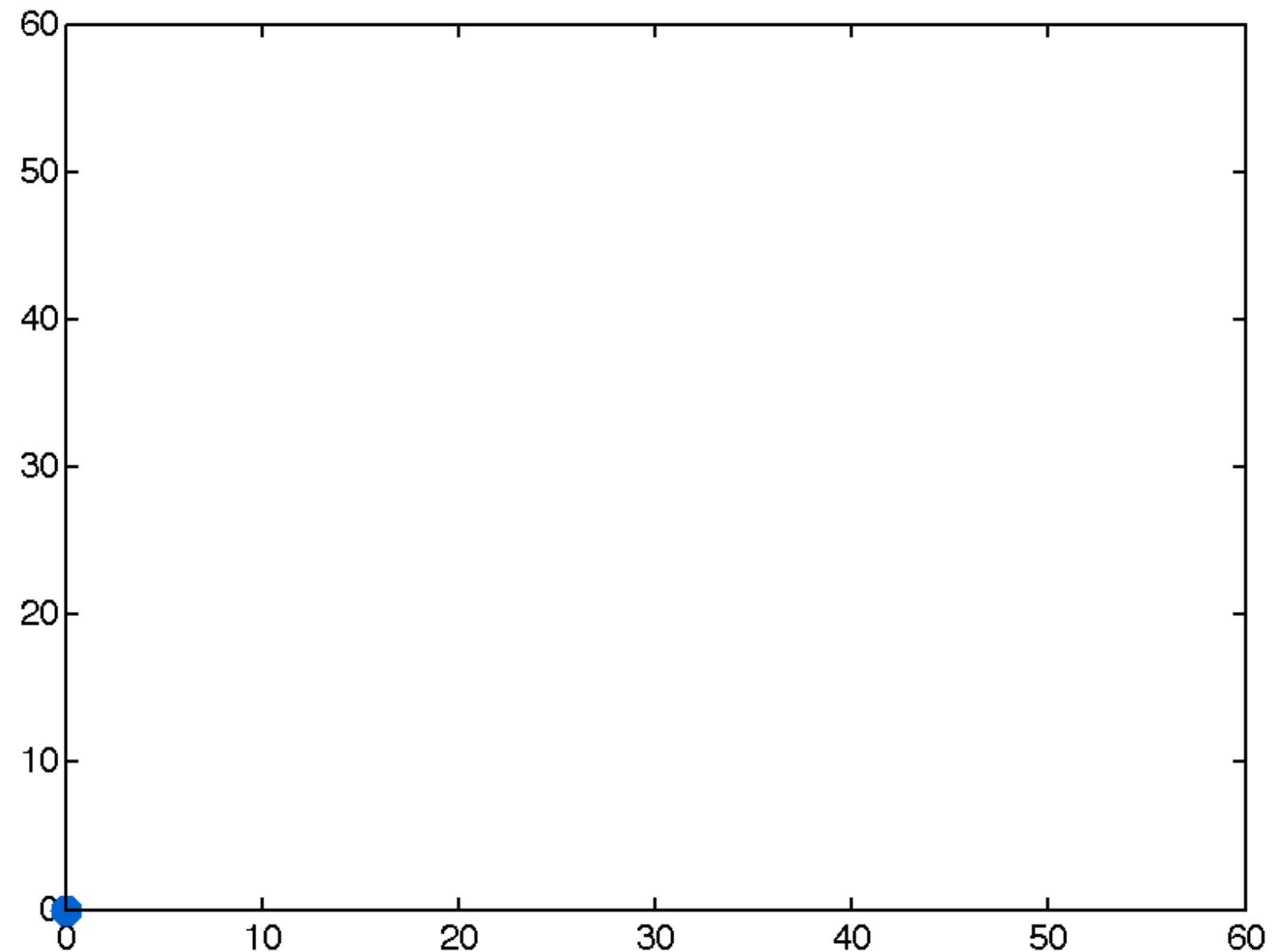
## LDS Example: Throwing a Ball

- Observations with **noisy sensor**

$$R = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

- It's windy!**  
System dynamics with **noise**

$$Q = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{pmatrix}$$



System evolution

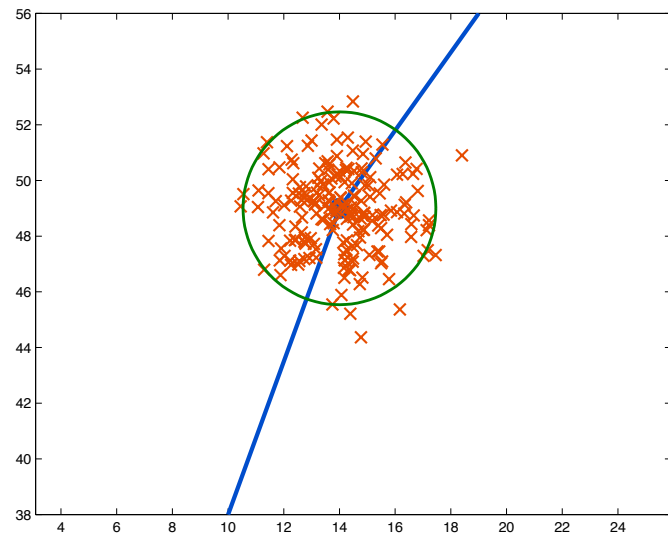


Observations

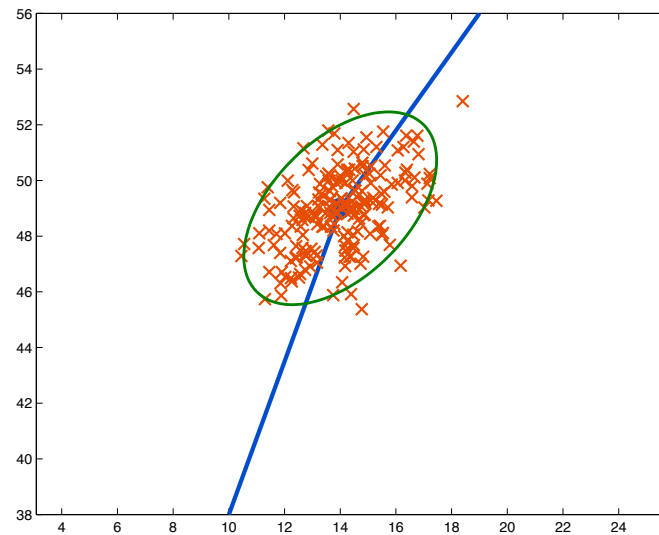
## LDS Example: Throwing a Ball

- Visualizing different **observation noise matrices**
- Remember,  $R$  is a covariance matrix, it's **symmetric** and **positive semi-definite**

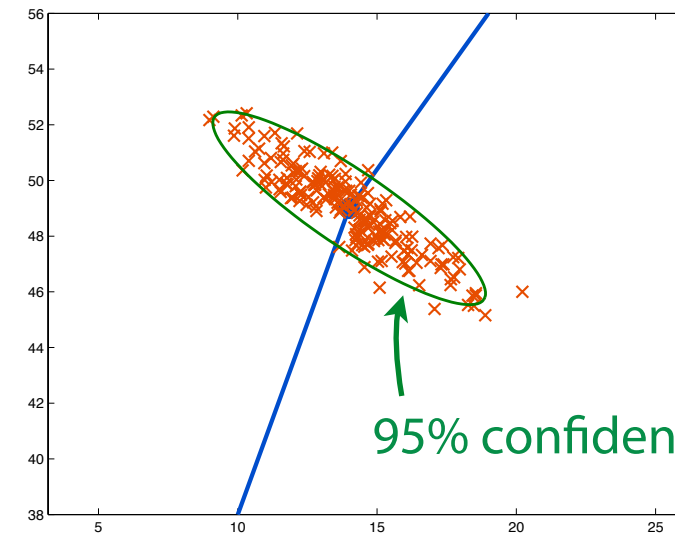
$$R = \begin{pmatrix} \sigma_{r_1}^2 & \sigma_{r_1 r_2} \\ \sigma_{r_2 r_1} & \sigma_{r_2}^2 \end{pmatrix}$$



$$R = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$



$$R = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$



$$R = \begin{pmatrix} 4 & -2.5 \\ -2.5 & 2 \end{pmatrix}$$



## Inference

- The four **inference tasks** for Hidden Markov Models (HMM):
  - Filtering
  - Smoothing
  - Prediction
  - Most likely sequence
- Do the same tasks exist for linear dynamical systems? **Yes!**
- Easiest task: **most likely sequence**. It turns out that due to the linear-Gaussian assumption, the most likely sequence, solved by the Viterbi algorithm for HMMs, is equal to the **sequence of individually most probable latent variable values** (statement without proof)
- Next, let us consider **filtering**

## Inference: Filtering

- For HMMs we have derived the **recursive Bayes filter**, a general sequential state estimation scheme

$$p(\mathbf{x}_t \mid \mathbf{z}_t) = \eta p(\mathbf{z}_t \mid \mathbf{x}_t) \sum_{\mathbf{x}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1})$$

- This finding holds for **linear dynamical systems**, too. In the continuous case, the sum becomes an integral

$$p(\mathbf{x}_t \mid \mathbf{z}_t) = \underbrace{\eta p(\mathbf{z}_t \mid \mathbf{x}_t)}_{\text{update}} \underbrace{\int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1}) \delta \mathbf{x}_{t-1}}_{\text{one-step prediction}}$$

- Since HMM and LDS rely on the same general state space model we can expect strong similarities in their inference algorithms

## Inference: Filtering

- If we substitute the **Gaussian transition and observation models**

$$p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}_{\mathbf{x}_k}(F_k \mathbf{x}_{k-1}, P'_k)$$

$$p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}_{\mathbf{z}_k}(H_k \mathbf{x}_k, R'_k)$$

into the Bayes filter equation

$$p(\mathbf{x}_t \mid \mathbf{z}_t) = \eta p(\mathbf{z}_t \mid \mathbf{x}_t) \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{z}_{t-1}) \delta \mathbf{x}_{t-1}$$

evaluate the integral, use some key results from linear algebra, marginalize some Gaussian terms, and perform a couple of more transformations, then we obtain the **following important result**:

## Kalman Filter

- The Kalman filter equations

$$\begin{aligned}\mathbf{x}_k &= F \mathbf{x}_{k-1} + K_k (\mathbf{z}_k - H F \mathbf{x}_{k-1}) \\ P_k &= (\mathbf{I} - K_k H) P'_k\end{aligned}$$

where

$$P'_k = F P_{k-1} F^T + Q$$

and  $K_k$  is defined to be the **Kalman gain matrix**

$$K_k = P'_k H^T (H P'_k H^T + R)^{-1}$$

- Let us first try to **interpret this result**. There is an update equation for the mean and an update equation for the associated covariance

## Kalman Filter

- We can view the update equation for the mean as follows

prediction + scaled observation error

error between predicted and actual observation

observation prediction

state prediction

$$\mathbf{x}_k = F \mathbf{x}_{k-1} + K_k (\mathbf{z}_k - H F \mathbf{x}_{k-1})$$

$$P_k = (\mathbf{I} - K_k H) P'_k$$

- Let us introduce the commonly used **notation** for time indices  $(k|k)$ ,  $(k+1|k)$ , and  $(k+1|k+1)$ . It will help us to better structure the equations

## Kalman Filter

- We define
  - $\mathbf{x}(k|k), P(k|k)$  to be the state and state covariance at time  $k$  given all observations until  $k$  (the cycle's "**prior**")
  - $\mathbf{x}(k+1|k), P(k+1|k)$  to be the state and state covariance at time  $k+1$  given all observations until  $k$  (the "**prediction**")
  - $\mathbf{x}(k+1|k+1), P(k+1|k+1)$  to be the state and state covariance at time  $k+1$  given all observations until  $k+1$  (the cycle's "**posterior**")
- Let us restructure the equations to make the filter's prediction-update scheme more explicit and distinguish between **state prediction**, **measurement/observation prediction**, and **update**

## Kalman Filter

- State prediction

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

transition model

$$P(k+1|k) = F P(k|k) F^T + Q$$

- Measurement prediction

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

observation model

$$\nu(k+1) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1)$$

innovation

$$S(k+1) = H P(k+1|k) H^T + R$$

innovation covariance

- Update

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

Kalman gain

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

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

## Kalman Filter

- We have some understanding of the **update equations of the means**: a one-step state prediction using the transition model, a measurement prediction using the observation model and an update that adds a scaled observation error to the state prediction
- Can we also gain some insight into the **covariance update** expressions?
- We recognize the recurring pattern  $A \cdot B \cdot A^T$ , for example in

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

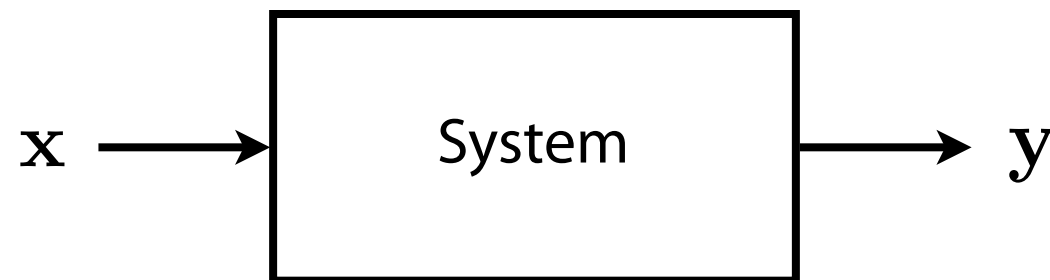
$$S(k+1) = \boxed{H P(k+1|k) H^T} + R$$

- This is the **error propagation law**. It computes the output covariance when an uncertain input is transformed by some (non-) linear function



## Kalman Filter

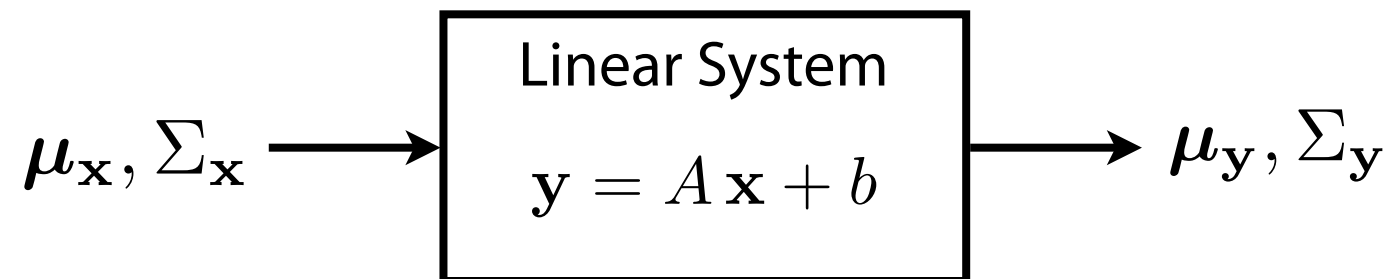
- **Error propagation** (a.k.a. propagation of uncertainty) is the problem of finding the distribution of a function of random variables
- It considers how the uncertainty, associated to a variable  $\mathbf{x}$  for example, “propagates” through a system or function  $\mathbf{y} = f(\mathbf{x})$



- Often we have a computational model of the system (the output as a function of the input and the system parameters) and we know something about the distribution of the input variables
- **Several methods** exist to determine the distribution of the output. Most popular: first-order approximations, Monte Carlo, unscented transform

## Kalman Filter

- Here, we consider **linear** functions and **Gaussian** random variables
- Then, error propagation has a **closed form** and is exact
- Let  $\mathbf{x} \sim \mathcal{N}_{\mathbf{x}}[\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{x}}]$  be the input variable with input covariance  $\boldsymbol{\Sigma}_{\mathbf{x}}$ ,  $\mathbf{y} \sim \mathcal{N}_{\mathbf{y}}[\boldsymbol{\mu}_{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}}]$  the output variable with output covariance  $\boldsymbol{\Sigma}_{\mathbf{y}}$  and  $\mathbf{y} = A \mathbf{x} + b$  the linear transform



- Then, the problem is to find  $\boldsymbol{\mu}_{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{y}}$

## Kalman Filter

- Mean

$$\begin{aligned}\mu_y &= E[y] \\ &= E[A \mathbf{x} + b] \\ &= A E[\mathbf{x}] + b \\ &= A \mu_x + b\end{aligned}$$

### Rules for $E[x]$ and $\text{Var}[x]$

---

$$E[a] = a$$

$$E[a \cdot x] = a E[x]$$

$$E[a \cdot x + b] = a E[x] + b$$

$$E[x + y] = E[x] + E[y]$$

$$\text{Var}[a \cdot x + b] = a^2 \cdot \text{Var}[x]$$

$$\text{Var}[x + y] = \text{Var}[x] + \text{Var}[y]$$

if  $x, y$  are indep.

## Kalman Filter

- Mean

$$\begin{aligned}\mu_y &= E[y] \\ &= E[A \mathbf{x} + b] \\ &= A E[\mathbf{x}] + b \\ &= A \mu_x + b\end{aligned}$$

- Covariance

$$\begin{aligned}\Sigma_y &= E[(\mathbf{y} - E[\mathbf{y}]) (\mathbf{y} - E[\mathbf{y}])^T] \\ &= E[(A \mathbf{x} + b - A E[\mathbf{x}] - b) (A \mathbf{x} + b - A E[\mathbf{x}] - b)^T] \\ &= E[(A (\mathbf{x} - E[\mathbf{x}])) (A (\mathbf{x} - E[\mathbf{x}]))^T] \\ &= E[(A (\mathbf{x} - E[\mathbf{x}])) ((\mathbf{x} - E[\mathbf{x}])^T A^T)] \\ &= A E[(\mathbf{x} - E[\mathbf{x}]) (\mathbf{x} - E[\mathbf{x}])^T] A^T \\ &= A \Sigma_x A^T\end{aligned}$$

### Rules for $E[x]$ and $\text{Var}[x]$

---

$$E[a] = a$$

$$E[a \cdot x] = a E[x]$$

$$E[a \cdot x + b] = a E[x] + b$$

$$E[x + y] = E[x] + E[y]$$

$$\text{Var}[a \cdot x + b] = a^2 \cdot \text{Var}[x]$$

$$\text{Var}[x + y] = \text{Var}[x] + \text{Var}[y]$$

if  $x, y$  are indep.

## Kalman Filter

- Summarizing: transforming a **Gaussian** random variable by a **linear function** results again in a **Gaussian** random variable
- Its parameters are

$$\mathbf{y} \sim \mathcal{N}_{\mathbf{y}}[A \boldsymbol{\mu}_{\mathbf{x}} + b, A \Sigma_{\mathbf{x}} A^T]$$

- The relationship for the output covariance matrix

$$\Sigma_y = A \Sigma_x A^T$$

is often called **error propagation law**

- Let us return to the Kalman filter and apply our finding

## Kalman Filter

- State prediction

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

$$P(k+1|k) = F P(k|k) F^T + Q$$

transition model

- Measurement prediction

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

$$\nu(k+1) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1)$$

$$S(k+1) = H P(k+1|k) H^T + R$$

obs. model

innovation

innovation covariance

- Update

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

Kalman gain

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

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

## Kalman Filter

- State prediction

$$\begin{aligned}\mathbf{x}(k+1|k) &= F \mathbf{x}(k|k) \\ P(k+1|k) &= \boxed{F P(k|k) F^T} + Q\end{aligned}$$

Propagation of the uncertainty  
of the previous state through  
the transition model

transition model

- Measurement prediction

$$\begin{aligned}\hat{\mathbf{z}}(k+1) &= H \mathbf{x}(k+1|k) \\ \nu(k+1) &= \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1) \\ S(k+1) &= \boxed{H P(k+1|k) H^T} + R\end{aligned}$$

Propagation of the  
uncertainty of the  
predicted state through  
the observation model

obs. model

innovation

innovation covariance

- Update

$$\begin{aligned}K(k+1) &= P(k+1|k) H^T S(k+1)^{-1} \\ \mathbf{x}(k+1|k+1) &= \mathbf{x}(k+1|k) + K(k+1) \nu(k+1) \\ P(k+1|k+1) &= (\mathbf{I} - K(k+1) H) P(k+1|k)\end{aligned}$$

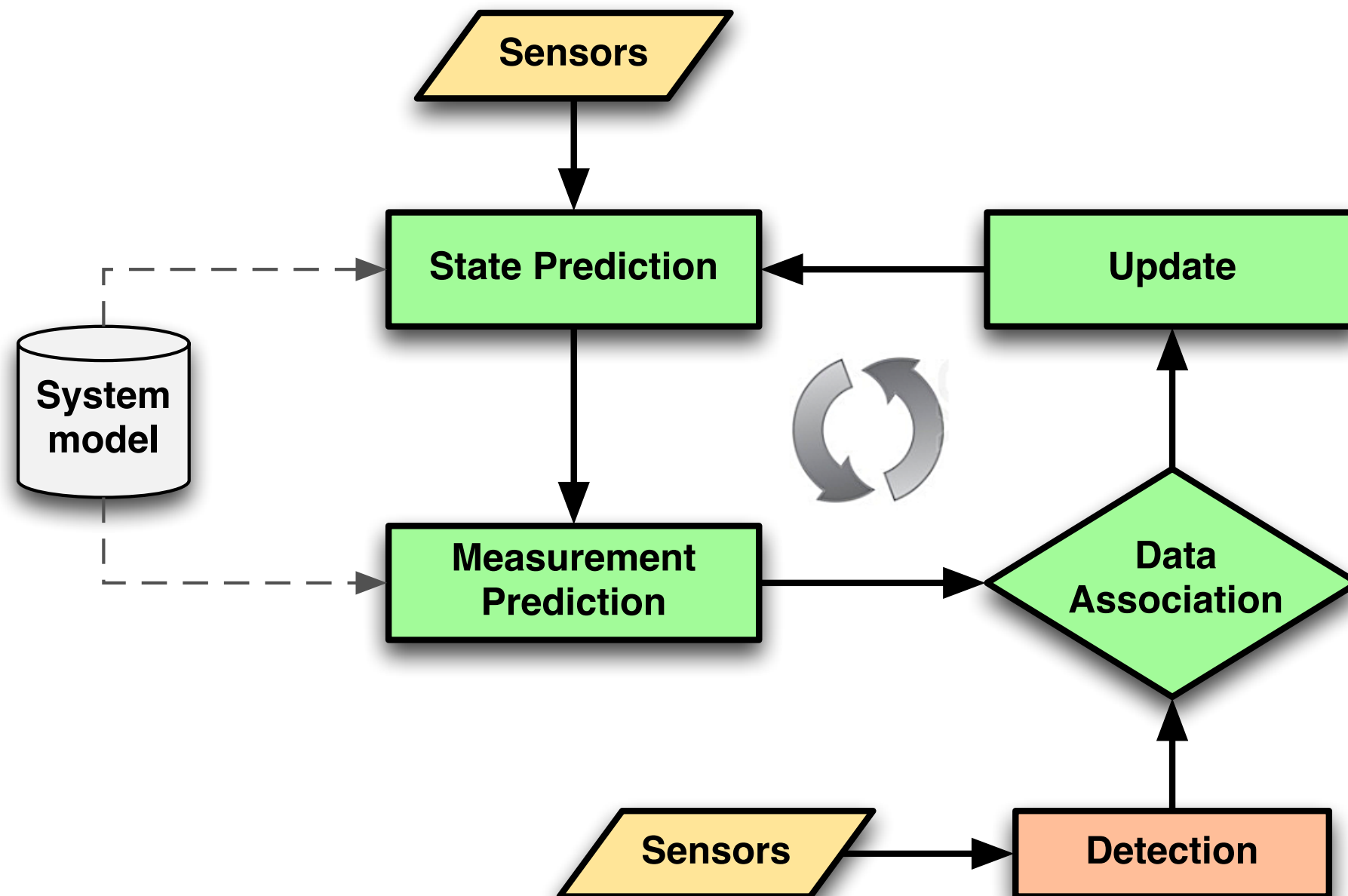
Kalman gain

## Kalman Filter

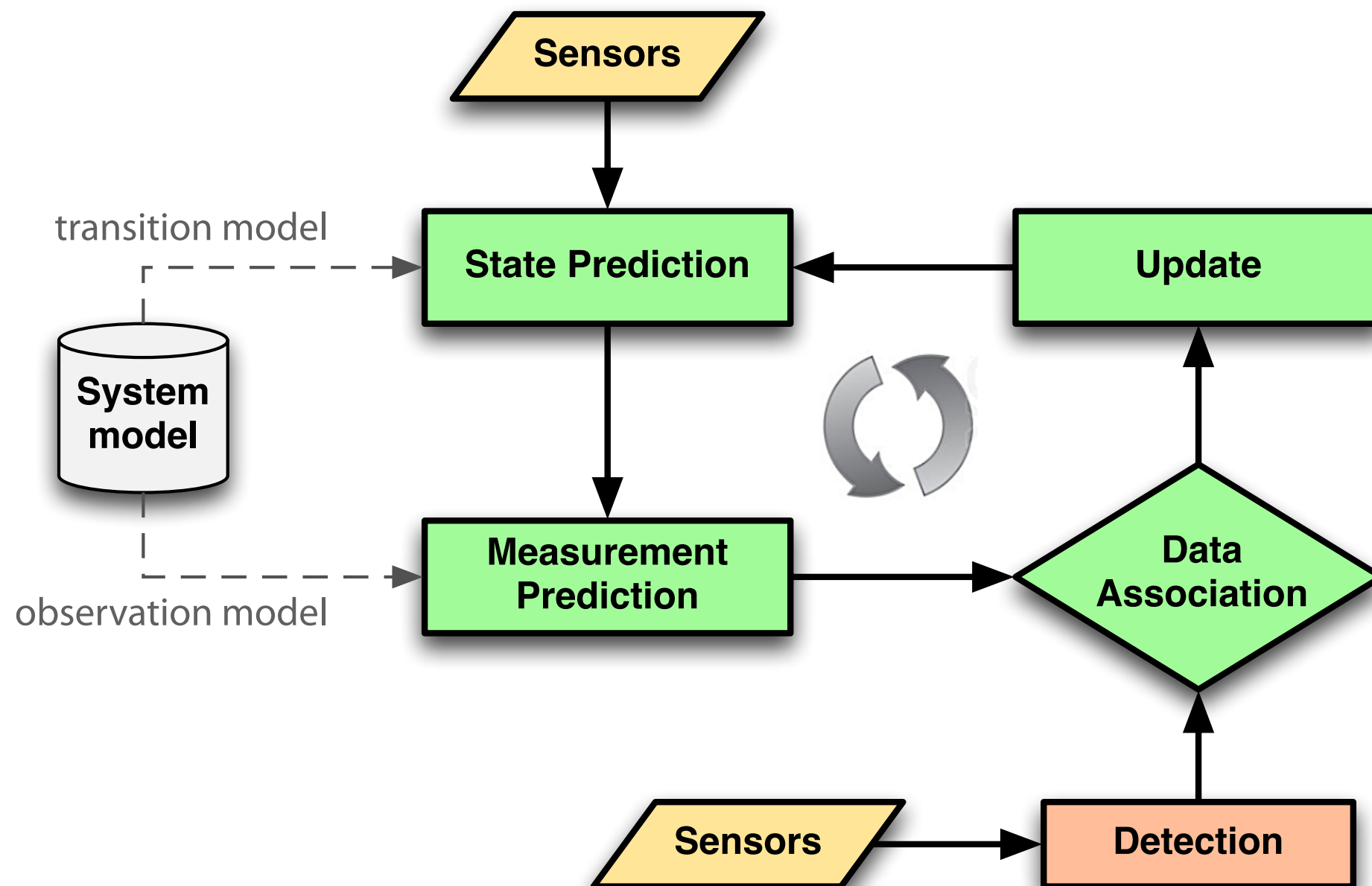
- We have derived the Kalman filter starting from **probabilistic graphical models**, Markov chains, and the state space model as a generic temporal model with latent variables. We have then considered HMMs for discrete and LDS for continuous latent variables. They share the same inference tasks of filtering, smoothing, prediction and most likely sequence
- Filtering in LDS has lead us to the Kalman filter as the linear-Gaussian version of the recursive Bayes filter
- This is a **very modern, unifying view** onto the Kalman filter. The filter has been developed in the late 1950s, long before graphical models had been discovered. HMMs have also been developed independently in the 1960s
- The Kalman filter has countless applications and is of **significant practical importance**: optimal tracking of rockets and satellites (was/is used in the Apollo program and the ISS), autopilots in aircrafts, weather forecasting, tracking for air traffic control, visual surveillance, etc.



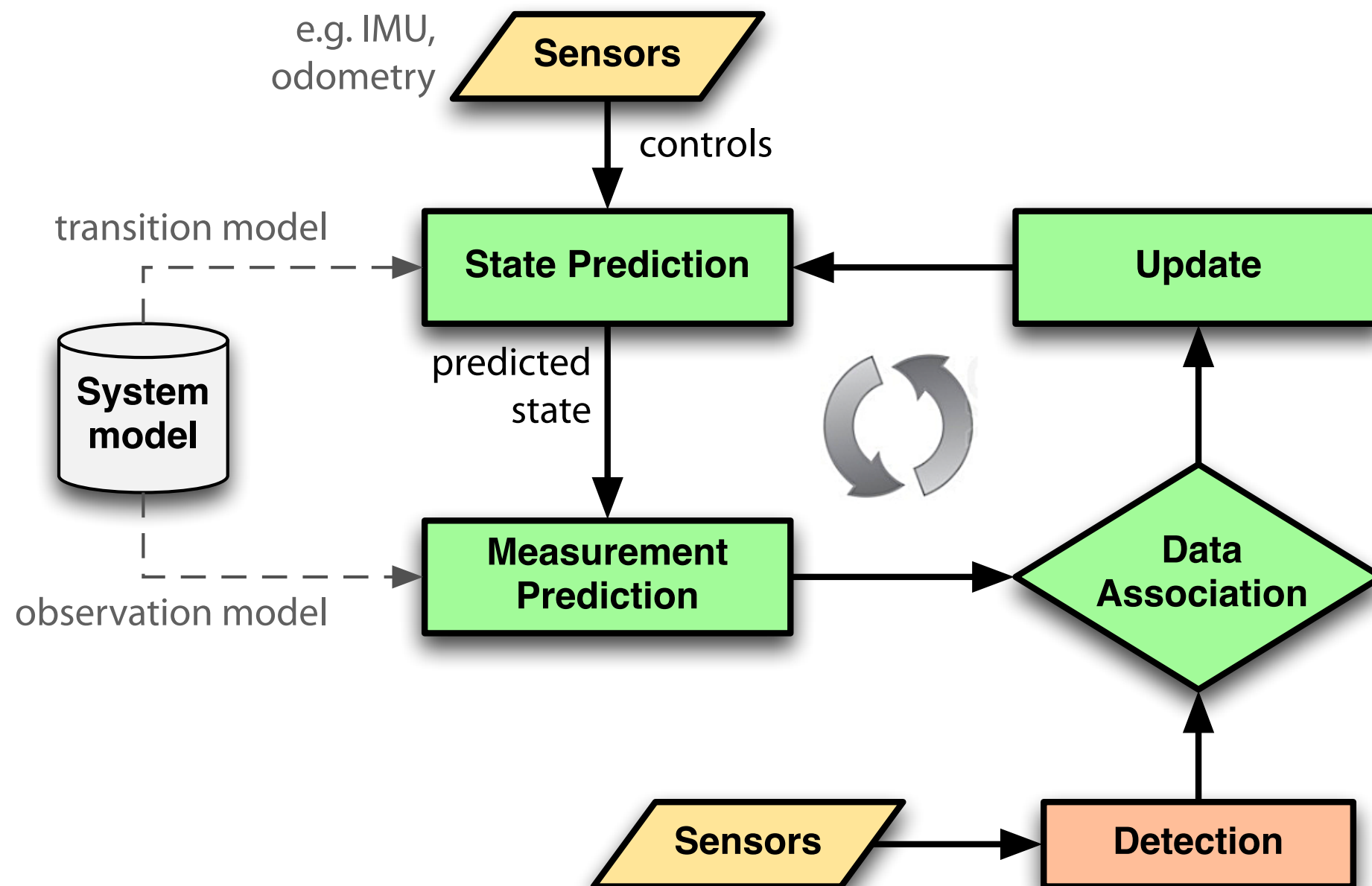
# Kalman Filter Cycle



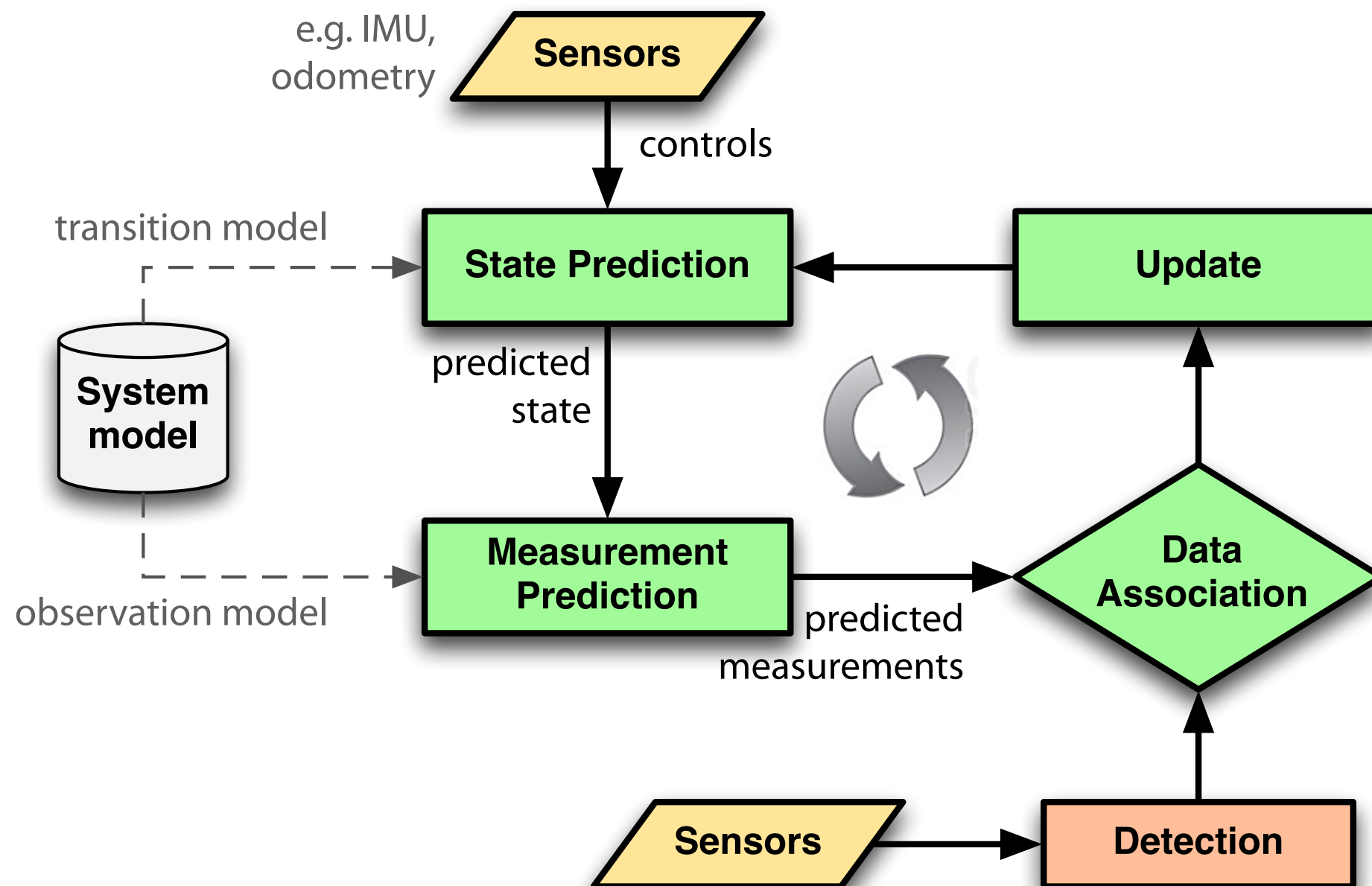
## Kalman Filter Cycle



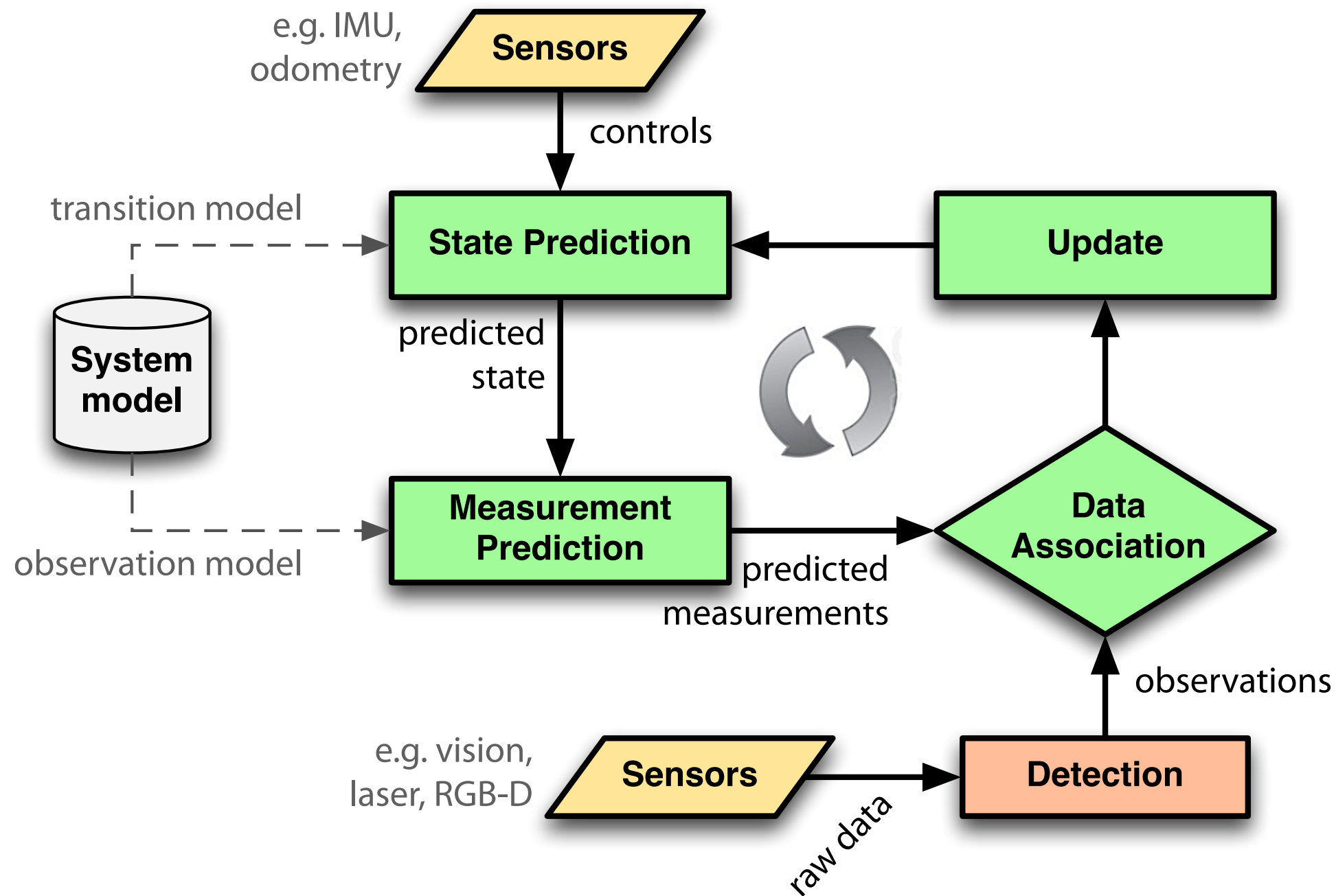
## Kalman Filter Cycle



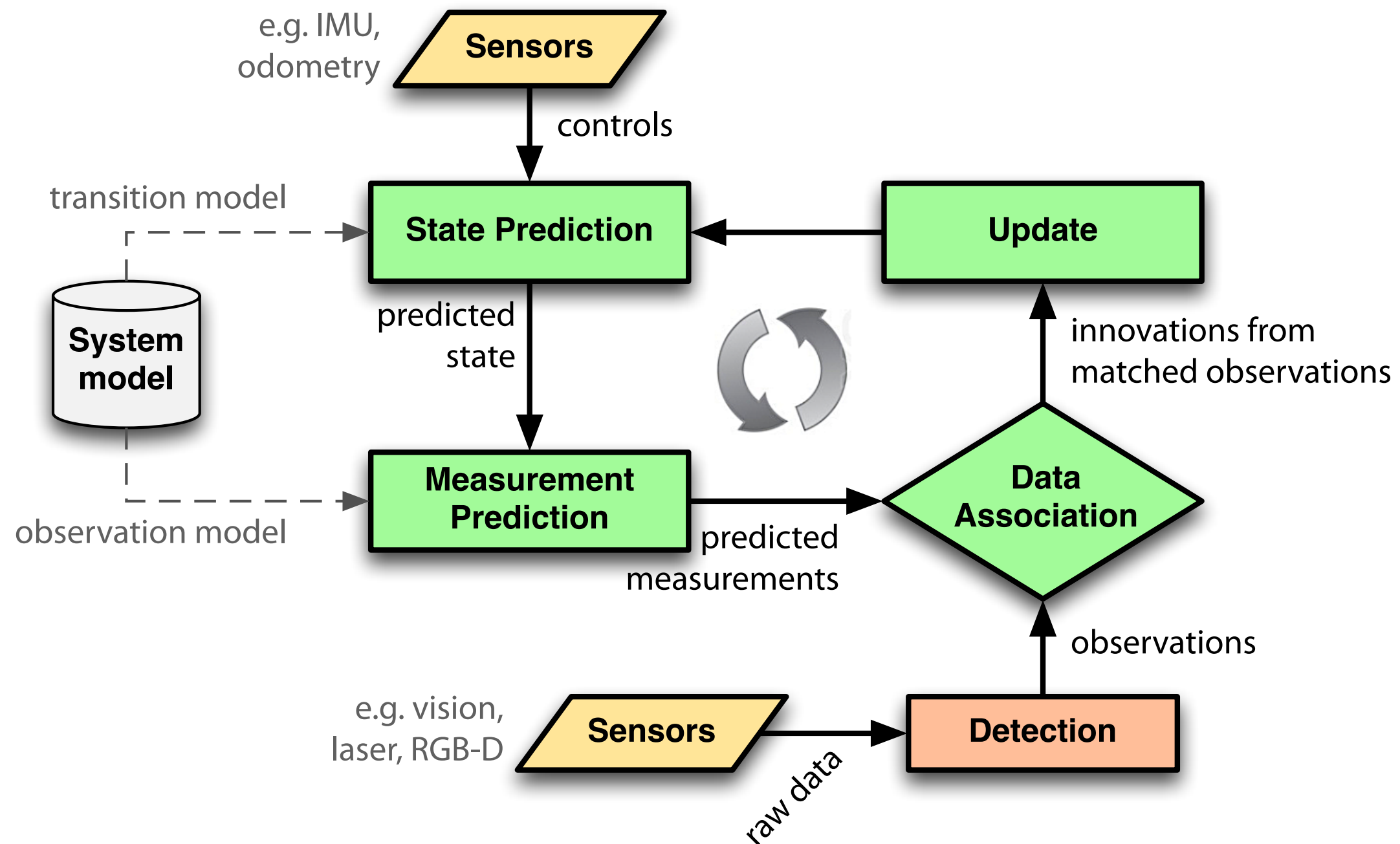
# Kalman Filter Cycle



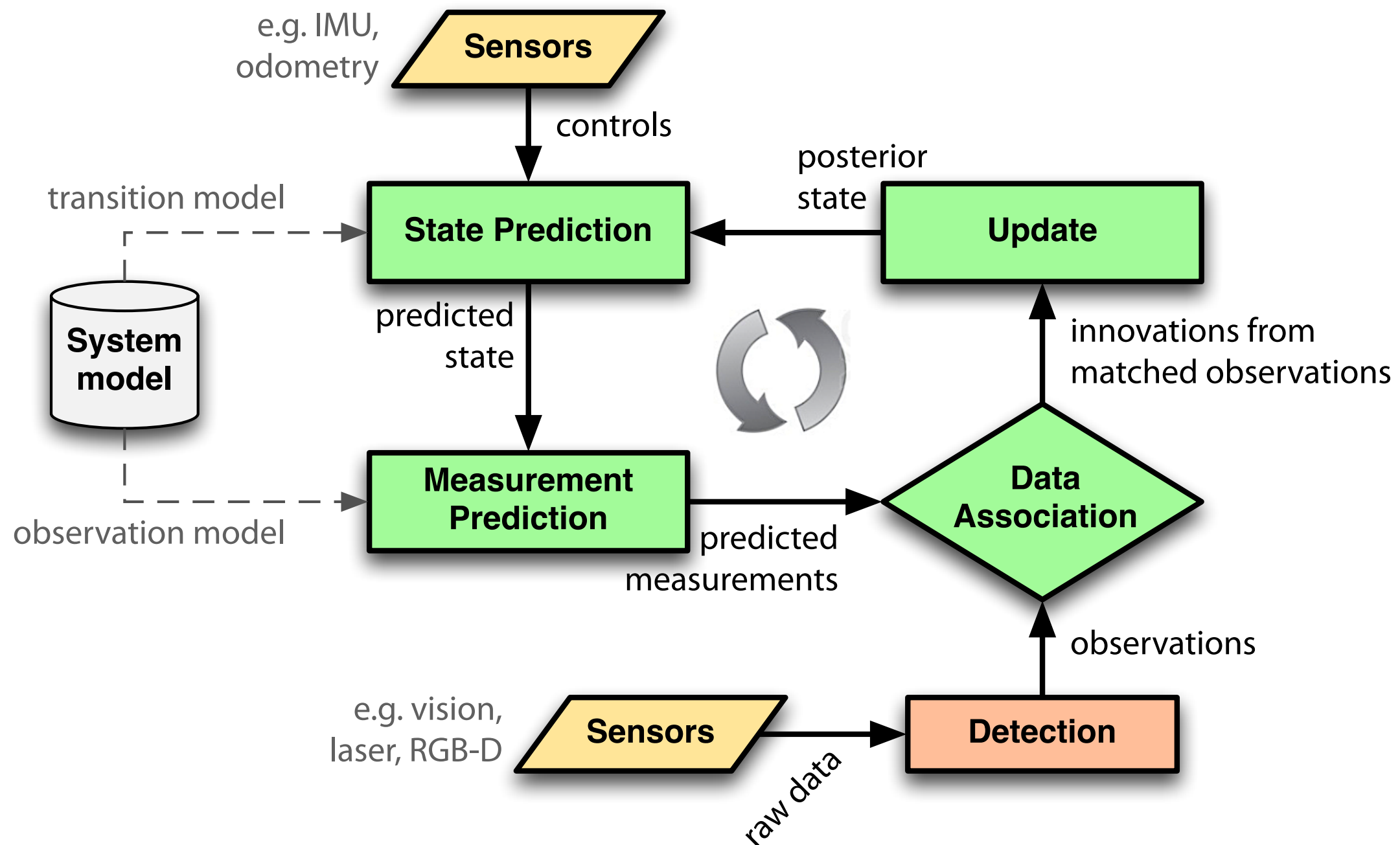
# Kalman Filter Cycle



# Kalman Filter Cycle



## Kalman Filter Cycle



## Kalman Filter Cycle (1/4): State Prediction

- State prediction is a **one-step prediction** of the state and its associated state covariance
- **Without controls**

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

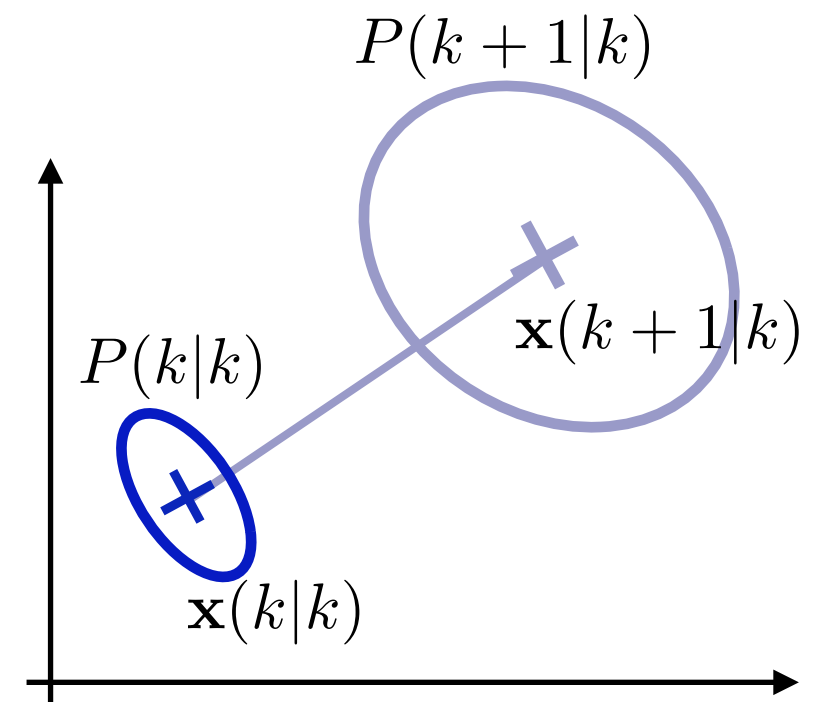
$$P(k+1|k) = F P(k|k) F^T + Q$$

- **With controls**

$$\mathbf{x}(k+1|k) = F \mathbf{x}(k|k) + G \mathbf{u}(k+1)$$

$$P(k+1|k) = F P(k|k) F^T + G U(k+1) G^T + Q$$

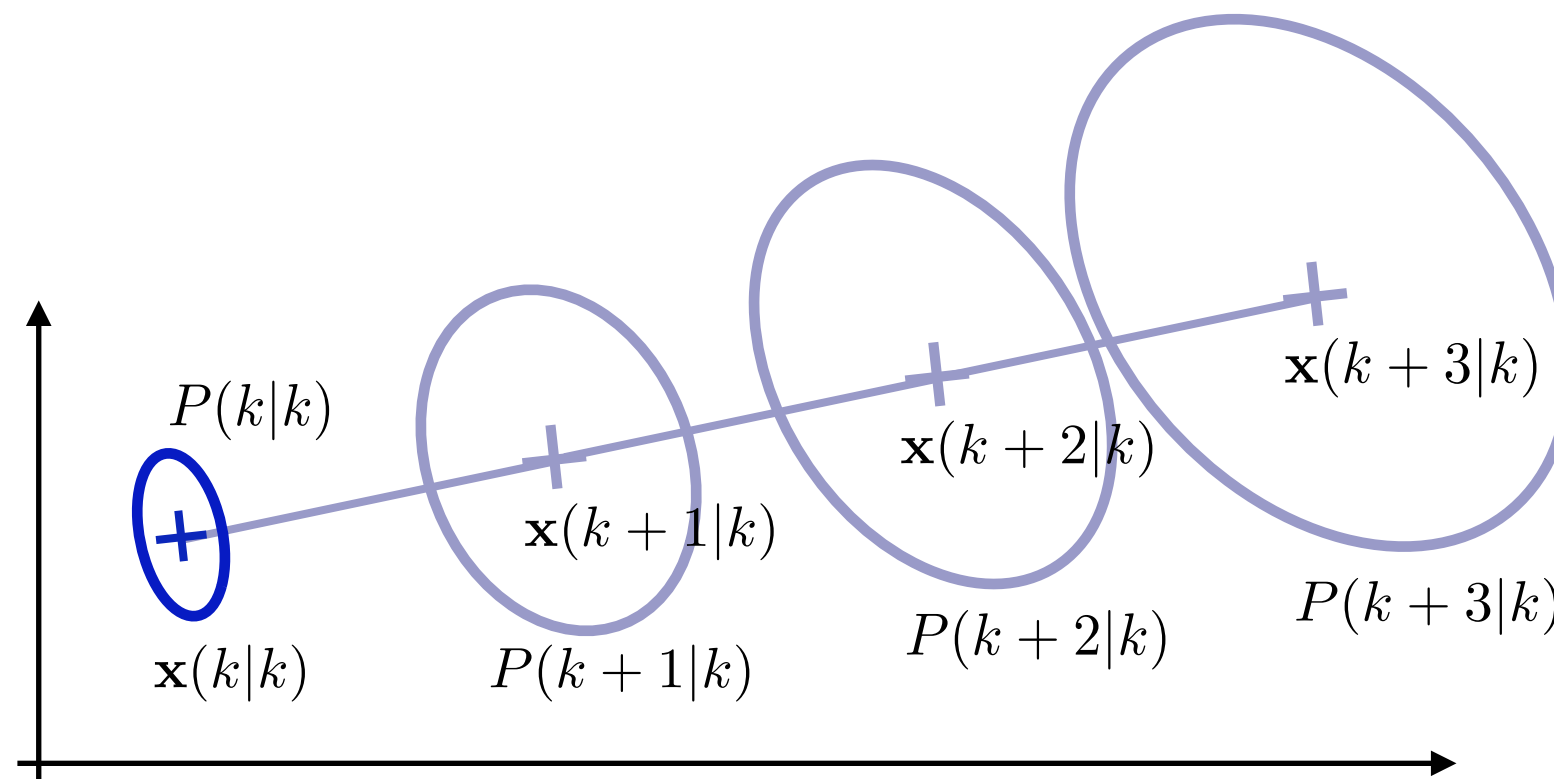
- State prediction projects the system's state into the future **without new observations**
- The error term  $\mathbf{v} \sim \mathcal{N}_{\mathbf{v}}(0, Q)$  in the transition model **injects new uncertainty** every time. Thus, the state prediction's uncertainty **grows**





## Kalman Filter Cycle (1/4): State Prediction

- General  $k$ -step prediction corresponds to the **LDS inference task of prediction**



- The growth of prediction's uncertainty **continues without bounds**. Over time, the state prediction “blurs” towards a **uniform distribution**

## Kalman Filter Cycle (2/4): Measurement Prediction

- Measurement prediction uses the predicted state to compute a predicted measurement  $\hat{\mathbf{z}}$  which hypothesizes **where to expect** the next observation
- Often, this is simply a coordinate frame transform. States are typically represented in some **global** (world) **coordinates** whereas observations are represented in **local sensor coordinates**

$$\begin{aligned}\hat{\mathbf{z}}(k+1) &= H \mathbf{x}(k+1|k) \\ \nu(k+1) &= \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1) \\ S(k+1) &= H P(k+1|k) H^T + R\end{aligned}$$

- The **innovation**  $\nu$  (pronounced n(y)öo like “new”) is the **error** between predicted and actual observation. It has the same dimension than the observations
- The **innovation covariance matrix**  $S$  is its associated uncertainty

## Kalman Filter Cycle (3/4): Data Association

- If there is a **single object state** to estimate and every observation is an observation of that object, then there is **no** data association problem
- Suppose there are **several states** to estimate or the observations are subject to **origin uncertainty** (e.g. sensor may produce false negatives, false positives, or measurements of unknown object identity). Then there is uncertainty about which object generated which observation
- This problem is called **data association** and consists in finding the **correct assignments of predicted to actual observations**
- Only correctly assigned prediction-observation pairs produce meaningful innovations and, in turn, accurate posterior state estimates. Incorrect associations may cause the filter to **diverge** and **lose track**
- An assignment of a prediction to an observation is called **pairing**

## Kalman Filter Cycle (3/4): Data Association

- How can we know when the pairing of prediction  $i$  and observation  $j$  is **correct**? By a **statistical compatibility** test:
- Given  $\nu_{ij}$ ,  $S_{ij}$ , the innovation and innovation covariance of pairing  $ij$ , we compute the **Mahalanobis distance** (skipping time indices)

$$d_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

and compare it against a threshold from a cumulative  $\chi^2$  distribution

- If the test

$$d_{ij}^2 \leq \chi_{n,\alpha}^2$$

significance level

degrees of freedom

holds, then **statistical compatibility** of the pairing on the significance level  $\alpha$  is given ( $\alpha$  is usually 0.95 or 0.99)

## Kalman Filter Cycle (4/4): Update

- In the update step, the **Kalman gain**

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

and the **posterior state estimates** are computed

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

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

- The Kalman filter averages the prediction of the system's state with a new observation using a **weighted average**
- More weights is put onto variables with better (i.e. smaller) estimated uncertainty. Such estimates are “trusted” more

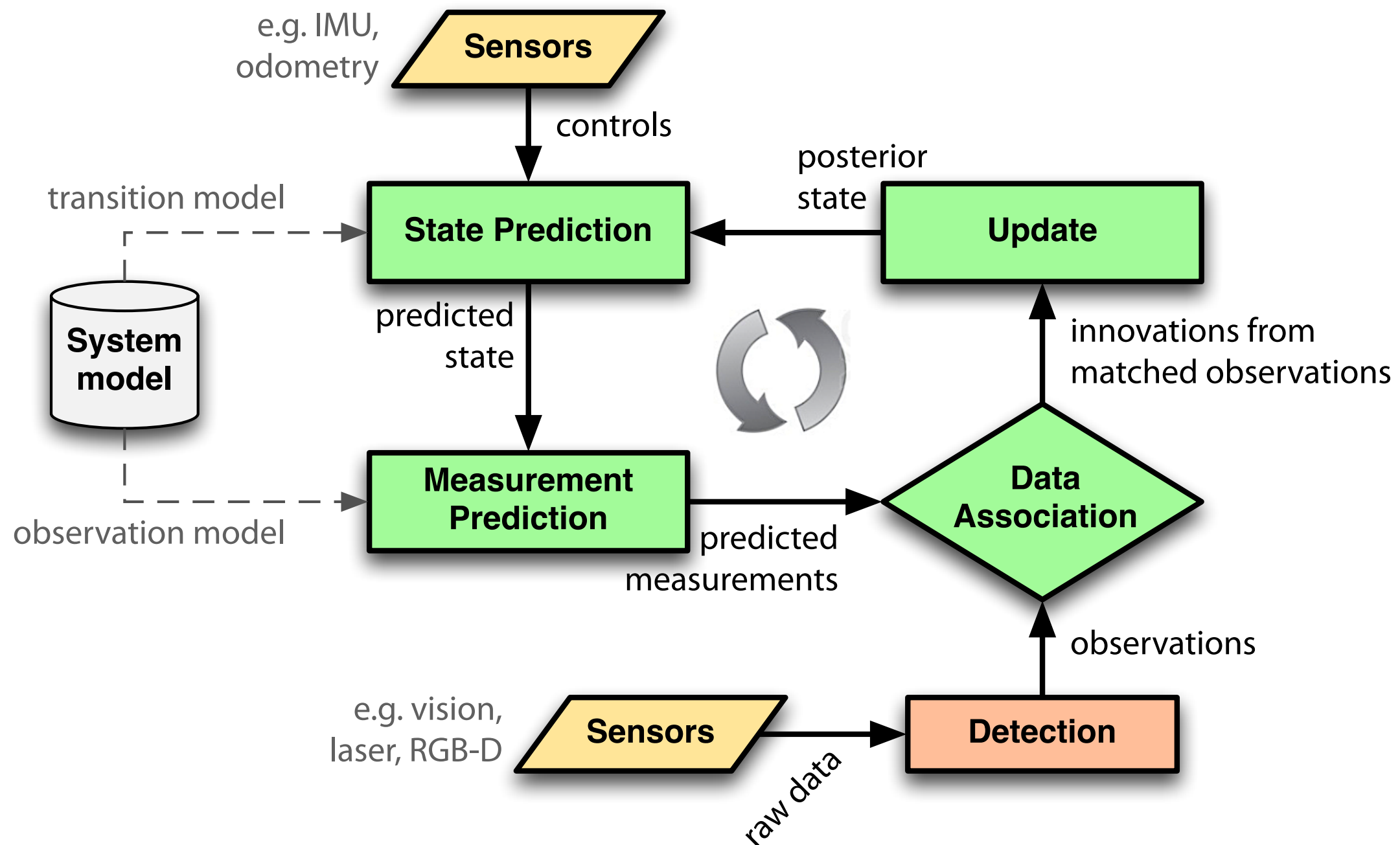
## Kalman Filter Cycle (4/4): Update

- It is common to discuss the filter's behavior in terms of **gain**

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

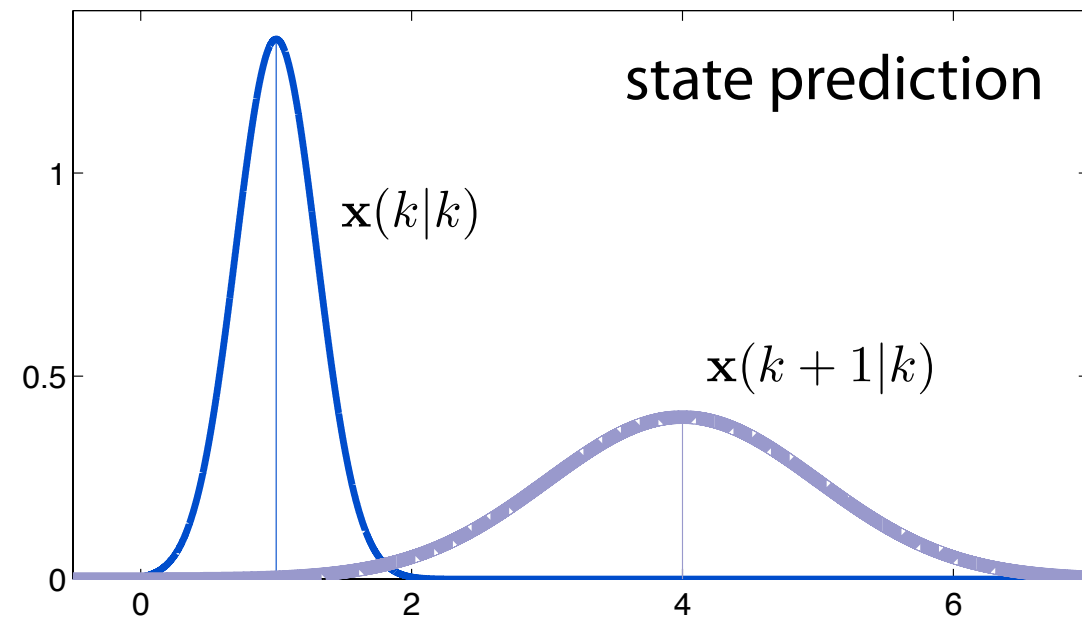
- With a **high gain**, the filter places **more weight on the measurements**, and thus follows them more closely
  - The innovation covariance  $S$  is small (e.g. observations are certain) and/or the predicted state covariance  $P(k+1|k)$  is large (e.g. due to a poor transition model)
- With a **low gain**, the filter **follows the state predictions** (process model) **more closely**, smoothing out noise but decreasing the responsiveness
  - The predicted state covariance  $P(k+1|k)$  is small (e.g. due to an accurate transition model) and/or the innovation covariance  $S$  is large (e.g. observations are uncertain)
- At the extremes, a **gain of one** causes the filter to ignore the state prediction, while a **gain of zero** causes the observations to be ignored

## Kalman Filter Cycle



## Kalman Filter Cycle

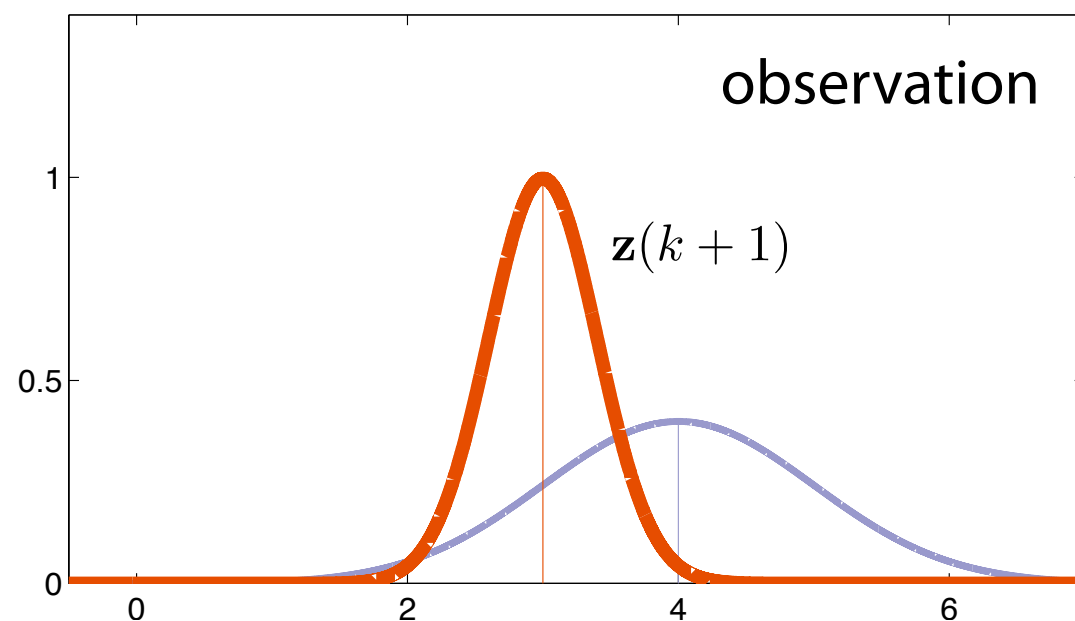
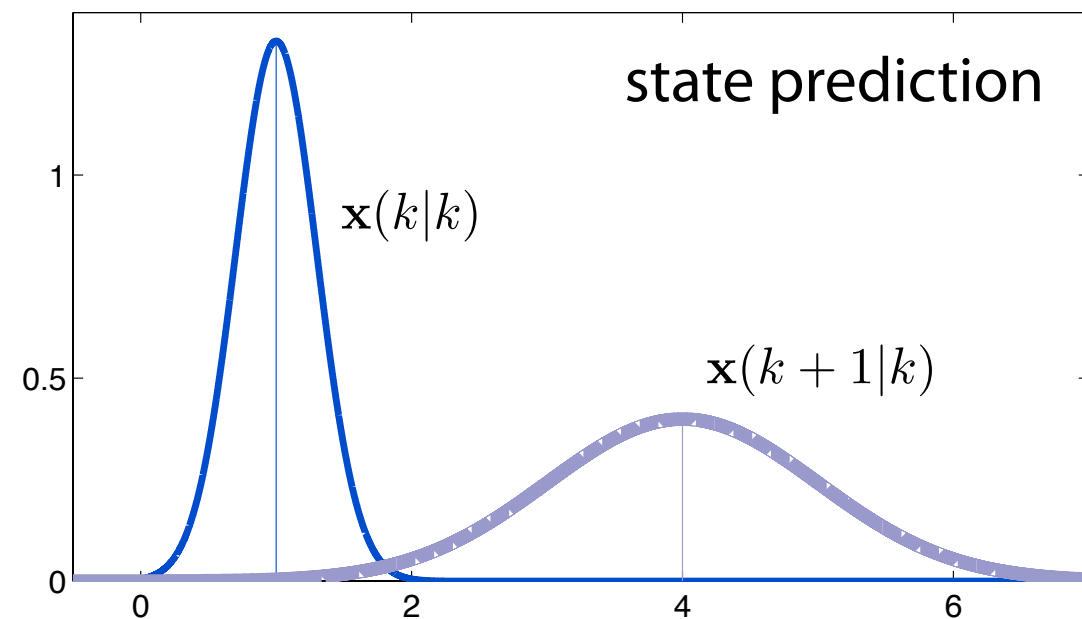
- A one-dimensional example
- For simplicity, we ignore measurement prediction by assuming a trivial observation model  $H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$  (e.g. (when state and observations are in same coordinate frame))





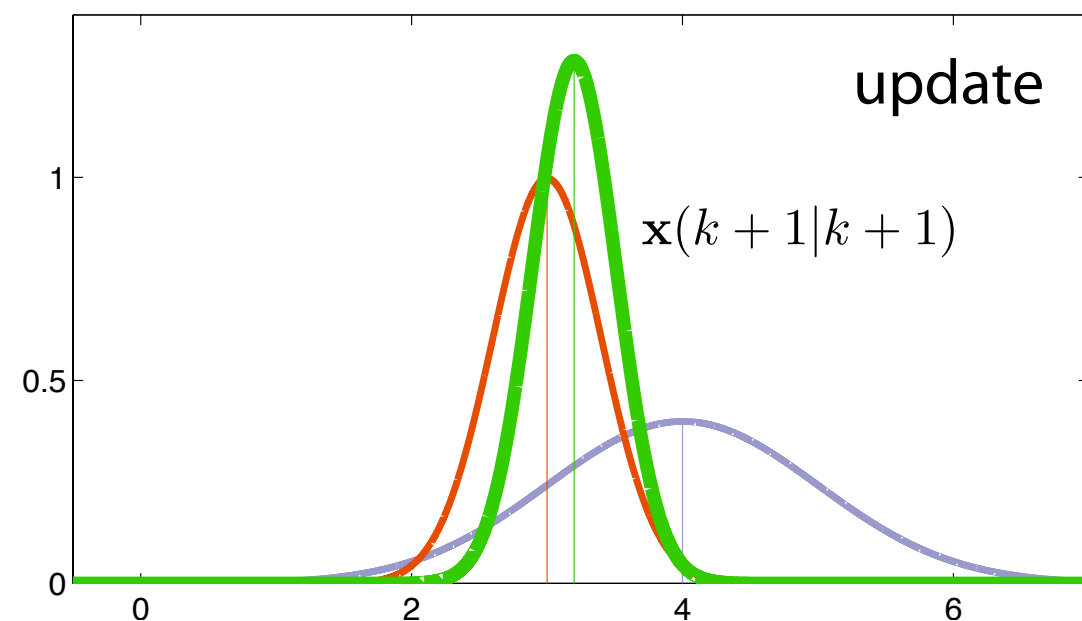
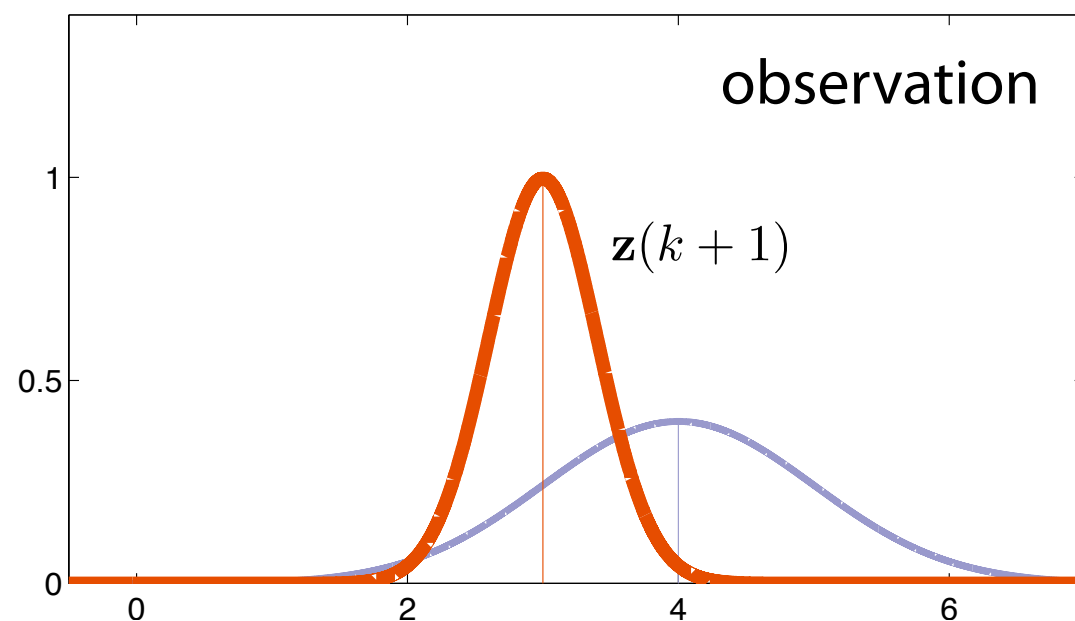
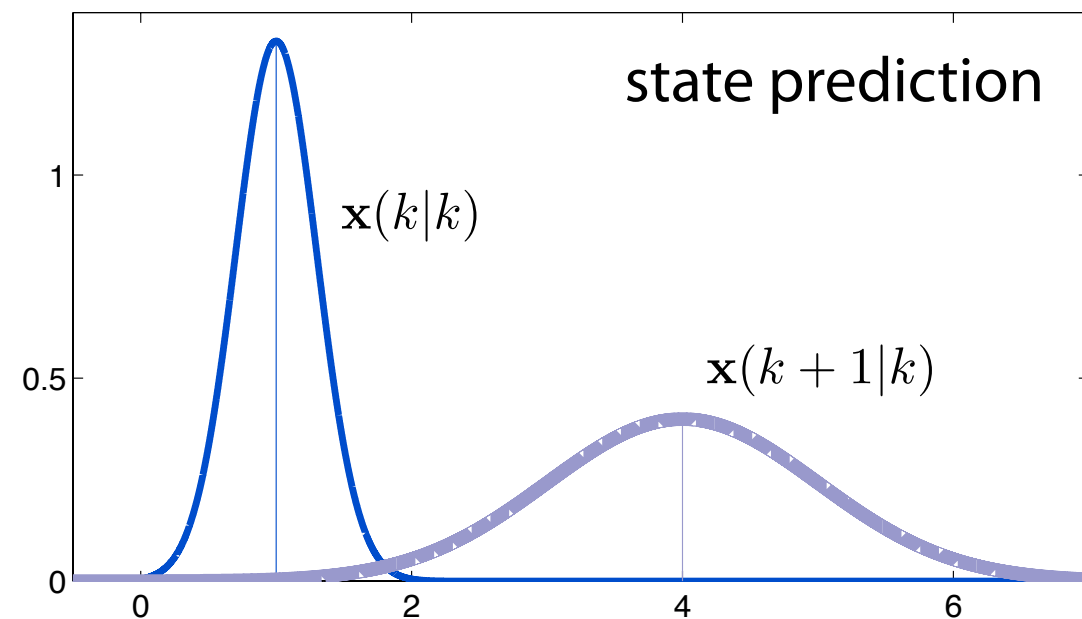
## Kalman Filter Cycle

- A one-dimensional example
- For simplicity, we ignore measurement prediction by assuming a trivial observation model  $H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$  (e.g. (when state and observations are in same coordinate frame))



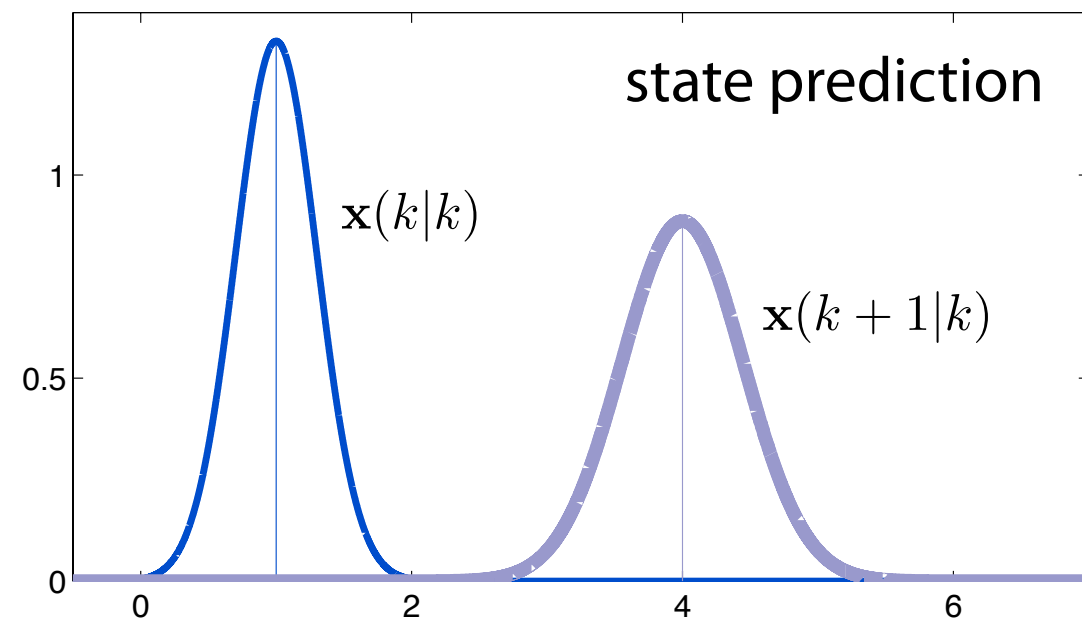
## Kalman Filter Cycle

- A one-dimensional example
- For simplicity, we ignore measurement prediction by assuming a trivial observation model  $H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$  (e.g. (when state and observations are in same coordinate frame))



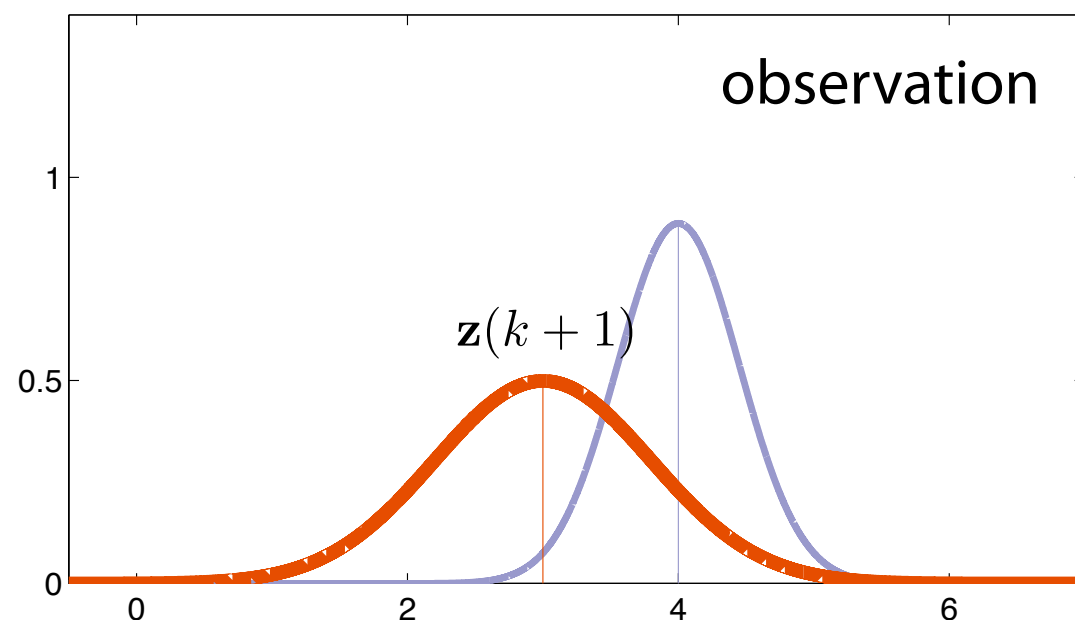
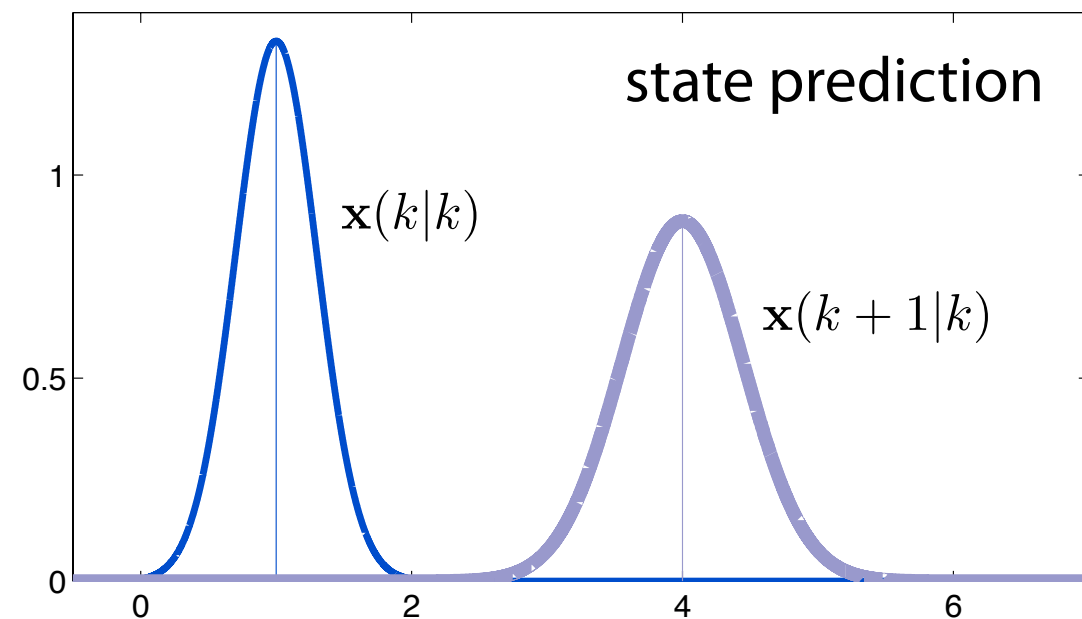
## Kalman Filter Cycle

- A one-dimensional example
- For simplicity, we ignore measurement prediction by assuming a trivial observation model  $H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$  (e.g. (when state and observations are in same coordinate frame))



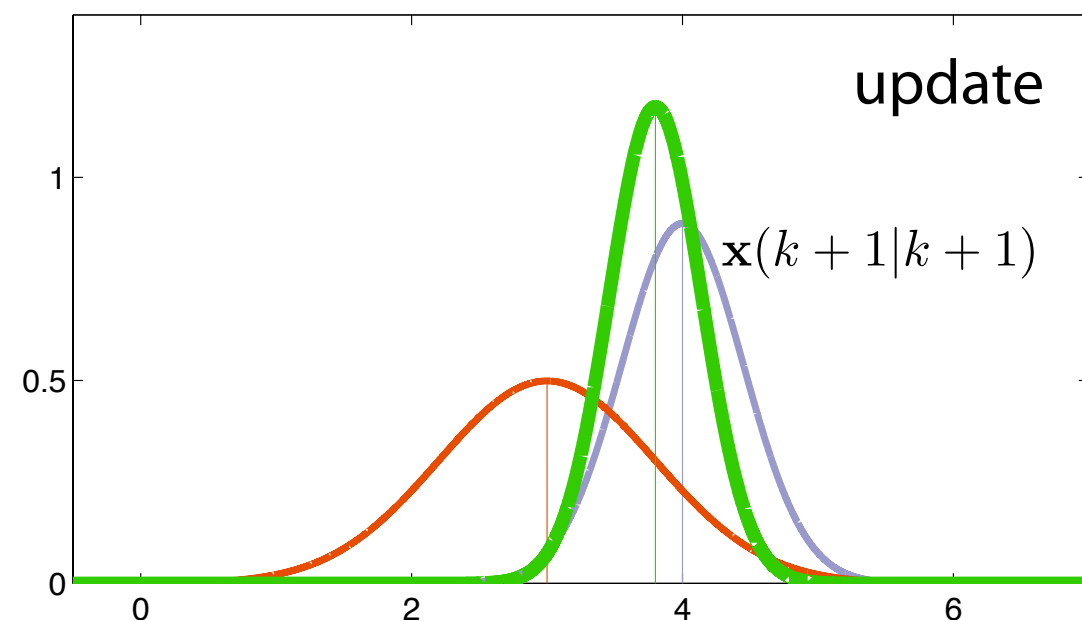
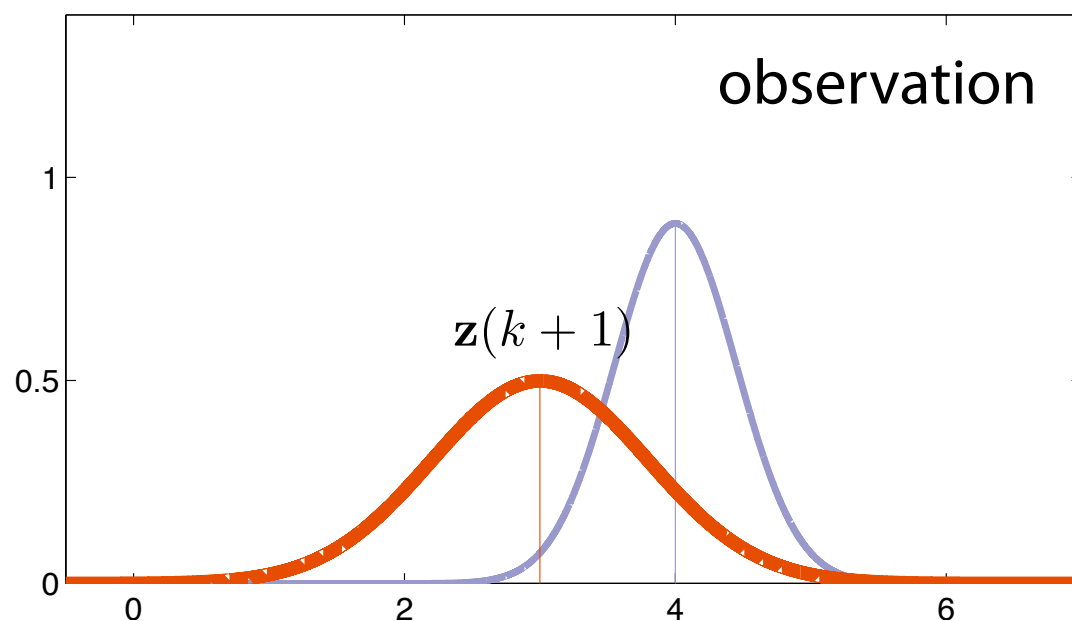
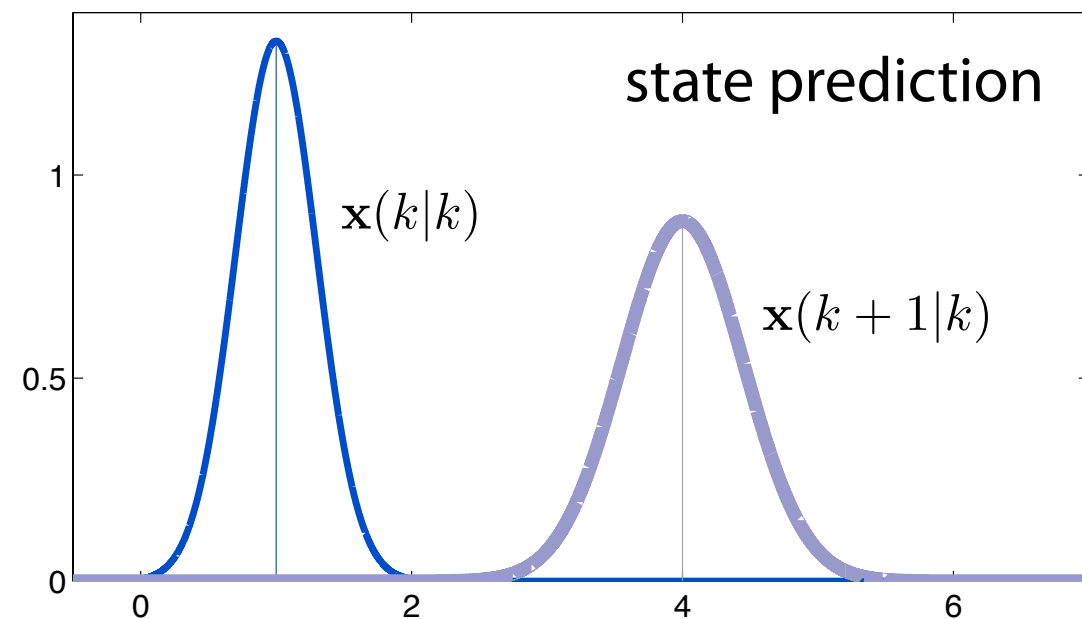
## Kalman Filter Cycle

- A one-dimensional example
- For simplicity, we ignore measurement prediction by assuming a trivial observation model  $H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$  (e.g. (when state and observations are in same coordinate frame))



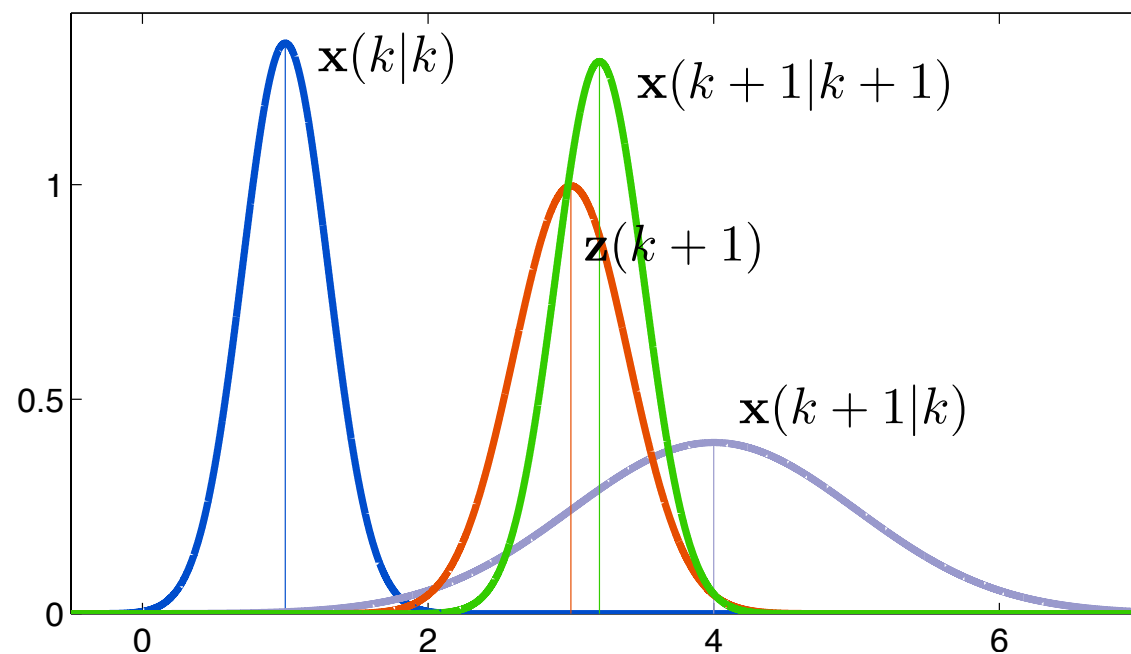
## Kalman Filter Cycle

- A one-dimensional example
- For simplicity, we ignore measurement prediction by assuming a trivial observation model  $H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$  (e.g. (when state and observations are in same coordinate frame))

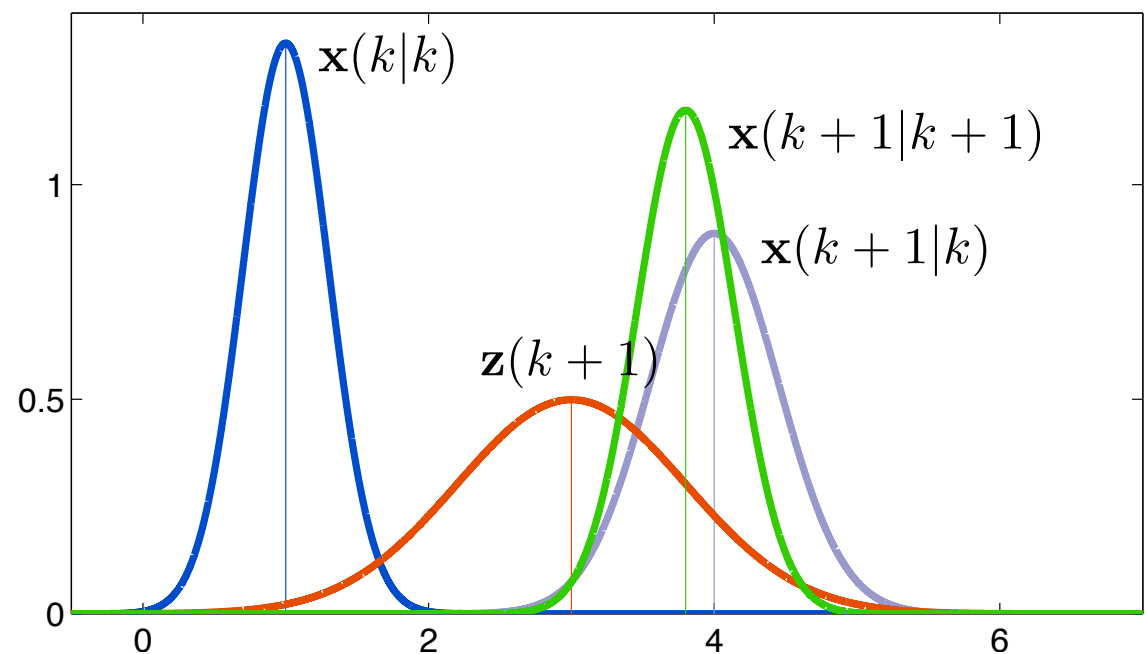


## Kalman Filter Cycle

- A one-dimensional example



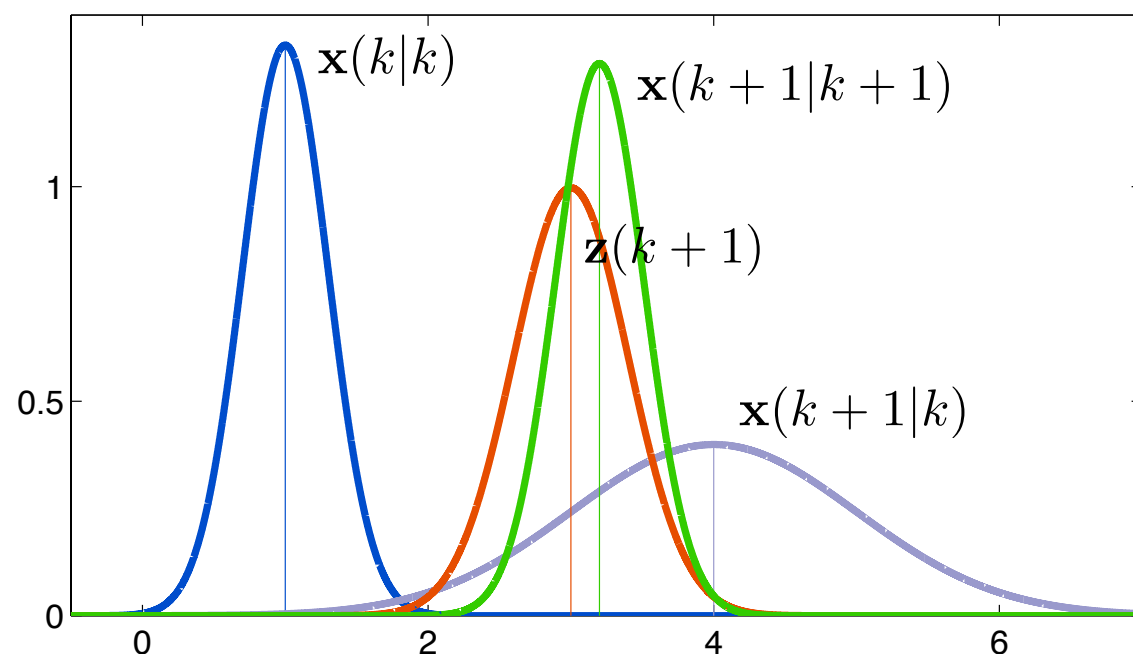
**Large** process noise, **small** observation noise. Leads to **high Kalman gain** and an update that follows the observations more closely



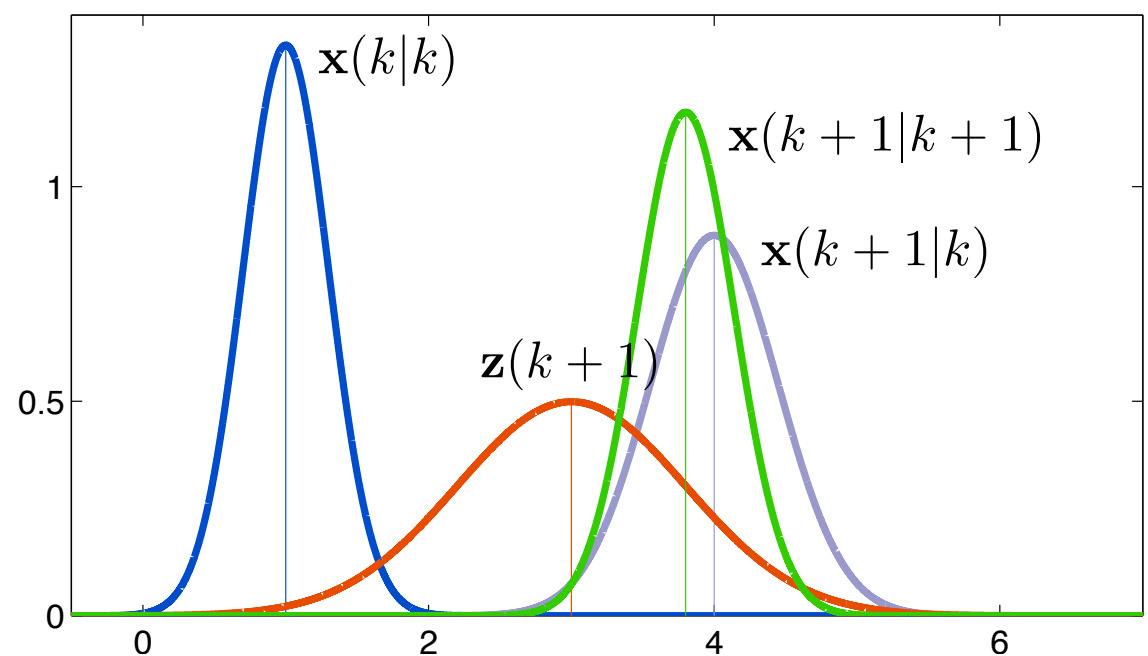
**Small** process noise, **large** observation noise. Leads to **low Kalman gain** and an update that follows the state prediction more closely

## Kalman Filter Cycle

- A one-dimensional example



**Large** process noise, **small** observation noise. Leads to **high Kalman gain** and an update that follows the observations more closely



**Small** process noise, **large** observation noise. Leads to **low Kalman gain** and an update that follows the state prediction more closely



**It's a weighted average!**

## Kalman Filter Example

- Let us return to our **ball example**
- This time we want to **track the ball** where “tracking” means estimating the **ball’s position and velocity** in an **online** fashion
- Note that, before, we have used the example to **demonstrate the LDS representation**, that is, the ability of the LDS model to describe the evolution of a dynamical system observed through an uncertain observation model. We have relied on physics to model the process and added noise in both, the system dynamics and the observations
- Now, when we want to **track the ball**, the only available knowledge about the ball are noisy  $(x, y)$ -observations that arrive one at a time





## Kalman Filter Example

- Suppose we also have some knowledge about the physics of throwing objects into the air and sensing them with a sensor
- This knowledge gives us parameters  $F, G, H$
- But suppose further that we **do not know anything** about the thrower, the thrown object (ball, paper airplane, model aircraft), or the environmental conditions (wind, rain)
- This is a typical situation in Kalman filtering: the transition and observation models are only known to some degree of accuracy. Then, the process and observation noise covariances  $Q$  and  $R$  have to cater for both, the **inherent uncertainty** of the system dynamics and observation process (e.g. due to unforeseen disturbances or sensor noise) and the **lack of accurate model knowledge** (a.k.a. mismodeling effects)



## Kalman Filter Example

- In a first approach we choose a very generic process model **without input** (we do not know anything about the thrown object)



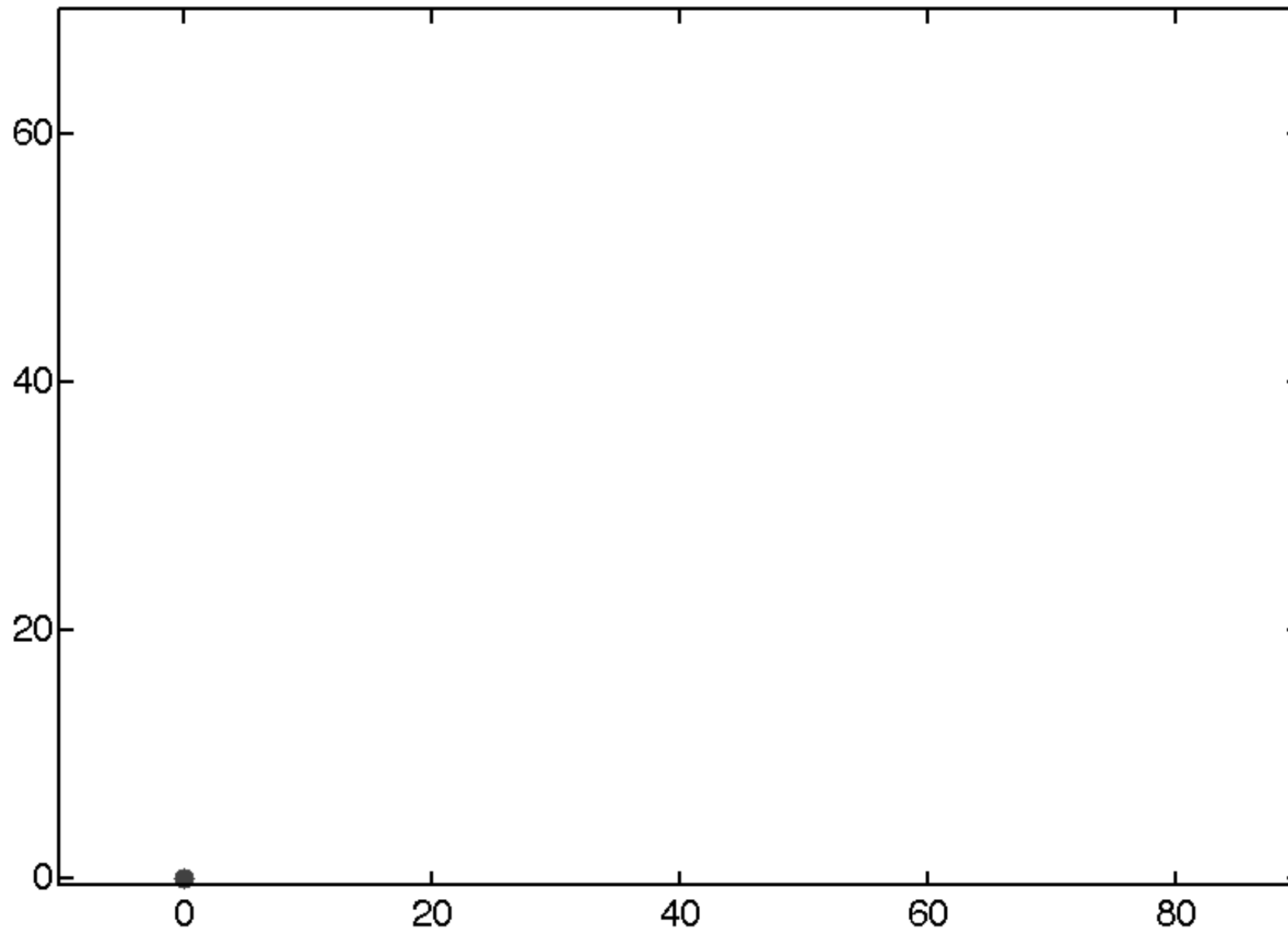
$$F = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{u} = 0 \quad H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- We choose  $Q$ ,  $R$ , and prior covariance  $P_0$  conservatively (i.e. large)

$$Q = \begin{pmatrix} 2.5 & 0 & 0 & 0 \\ 0 & 2.5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix} \quad R = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \quad \mathbf{x}_0 = \begin{pmatrix} 1 \\ 2 \\ 10 \\ 20 \end{pmatrix} \quad P_0 = \begin{pmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{pmatrix}$$

- Also, we **do not** perform a **statistical compatibility test** and accept all sensor readings as originating from the thrown object (no false positives)

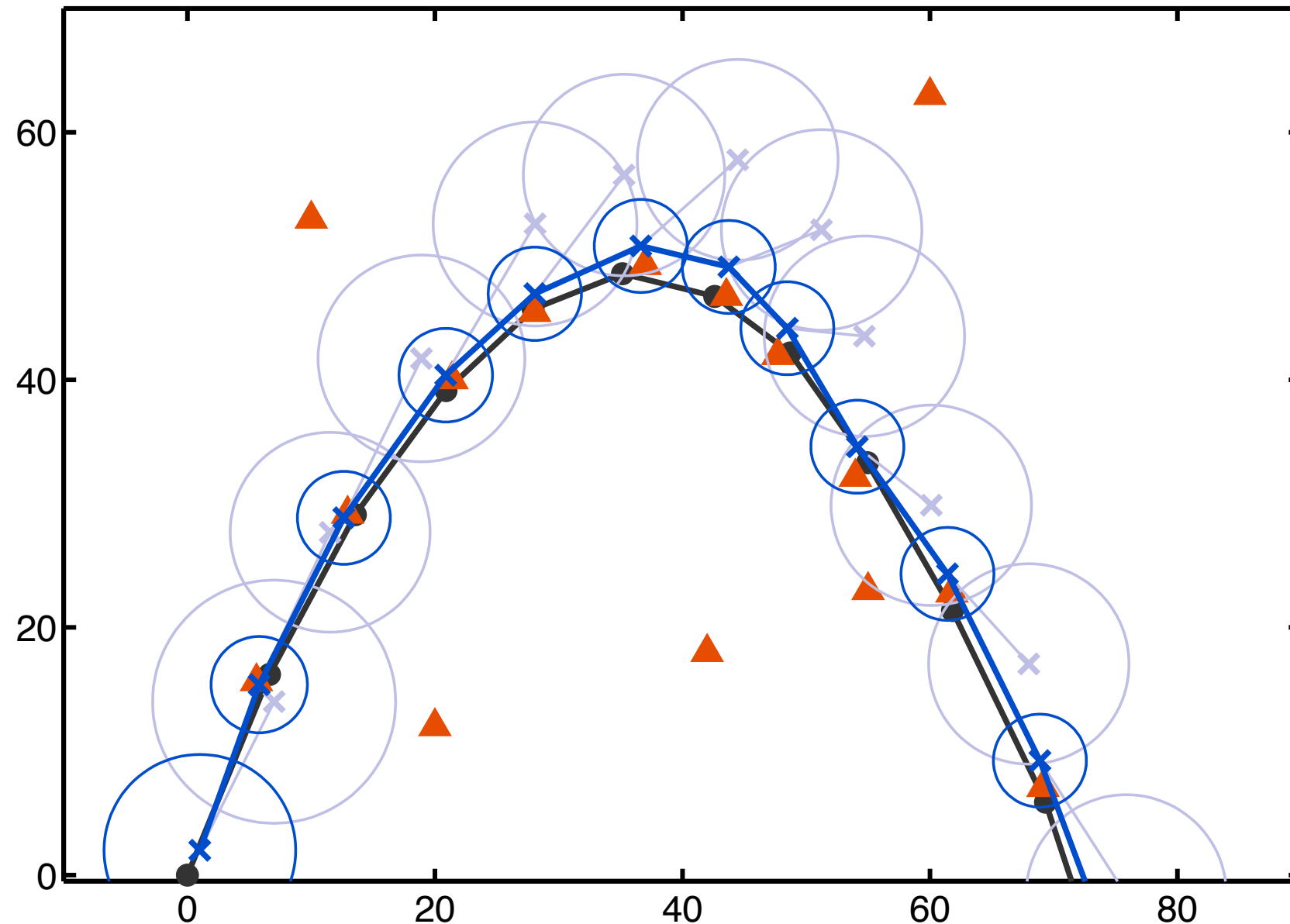
## Kalman Filter Example



Ground truth    Observations    State estimates    State predictions



## Kalman Filter Example



- **Poor** state predictions
- **Poor** velocity estimates
- **Low** tracking accuracy

Ground truth    Observations    State estimates    State predictions

## Kalman Filter Example

- Now we learn that the sensor produces **false alarms** (false positives)
- Thus, we cannot trust all observations to originate from the thrown object
- We have to make a **statistical compatibility test**. We choose a significance level of 0.99

$$d_{ij}^2 \leq \chi_{n,\alpha}^2$$

with

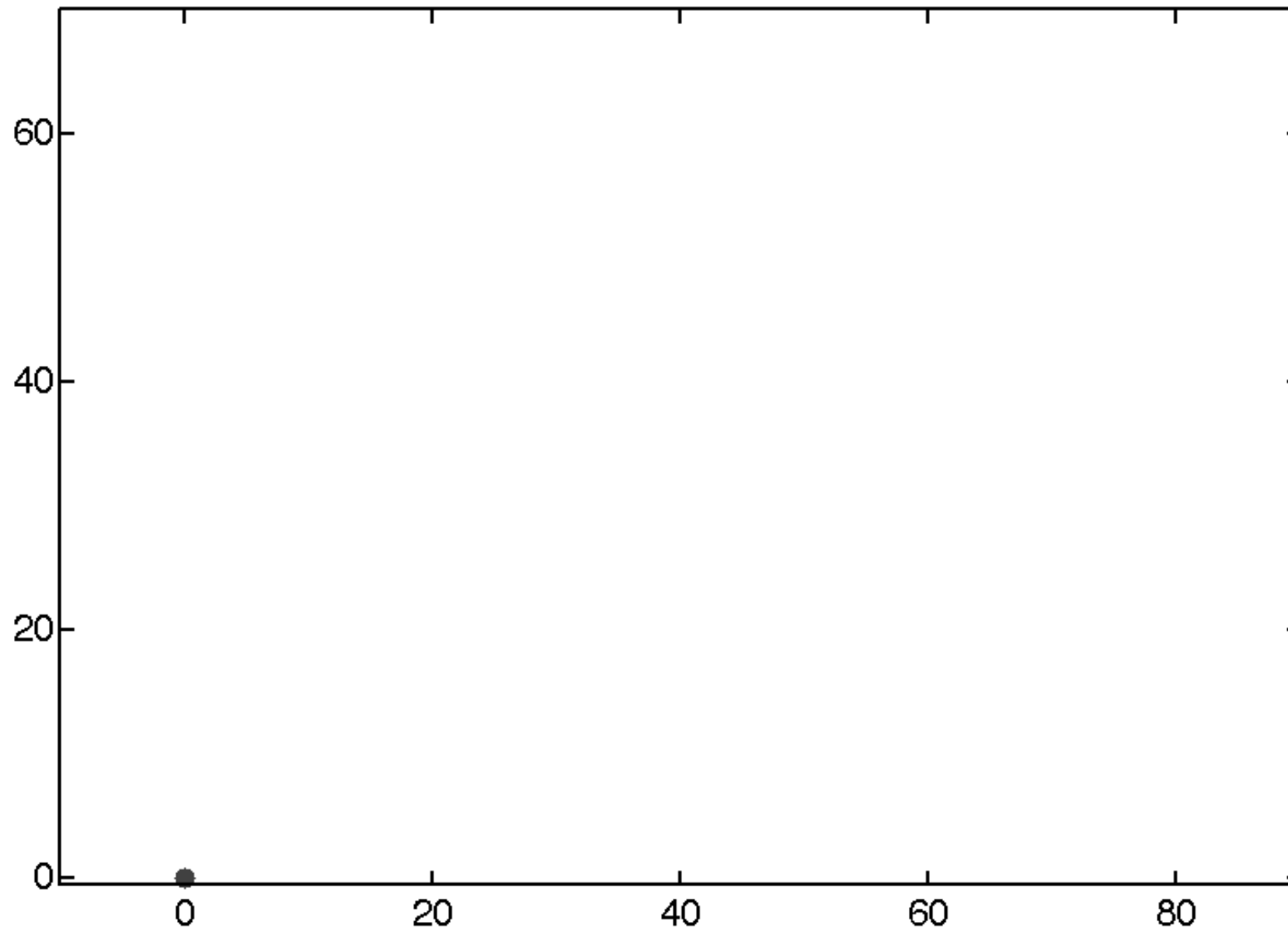
$$d_{ij}^2 = \nu_{ij}^T S_{ij}^{-1} \nu_{ij}$$

- All other parameters remain unchanged



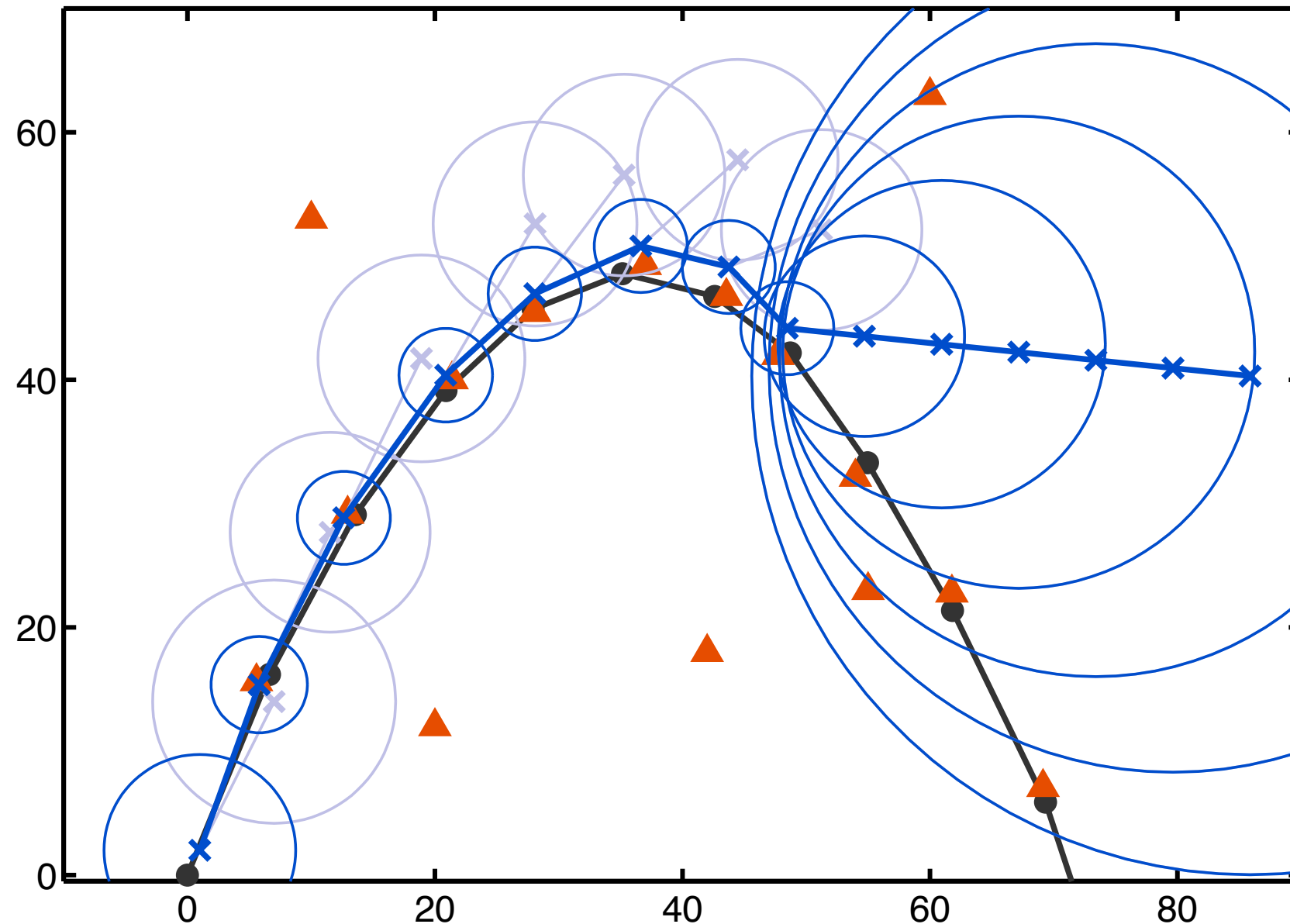


## Kalman Filter Example



Ground truth    Observations    State estimates    State predictions

## Kalman Filter Example



Ground truth    Observations    State estimates    State predictions



- Filter loses track and diverges
- State is recursively predicted without update

## Kalman Filter Example

- Now we learn that the thrown object is a **ball**  
– not a paper airplane or motorized model aircraft. We **refine our process model** by adding the gravity force as input

$$\mathbf{u} = -g \quad G = \begin{pmatrix} 0 \\ \Delta t^2/2 \\ 0 \\ \Delta t \end{pmatrix}$$

- The new transition model

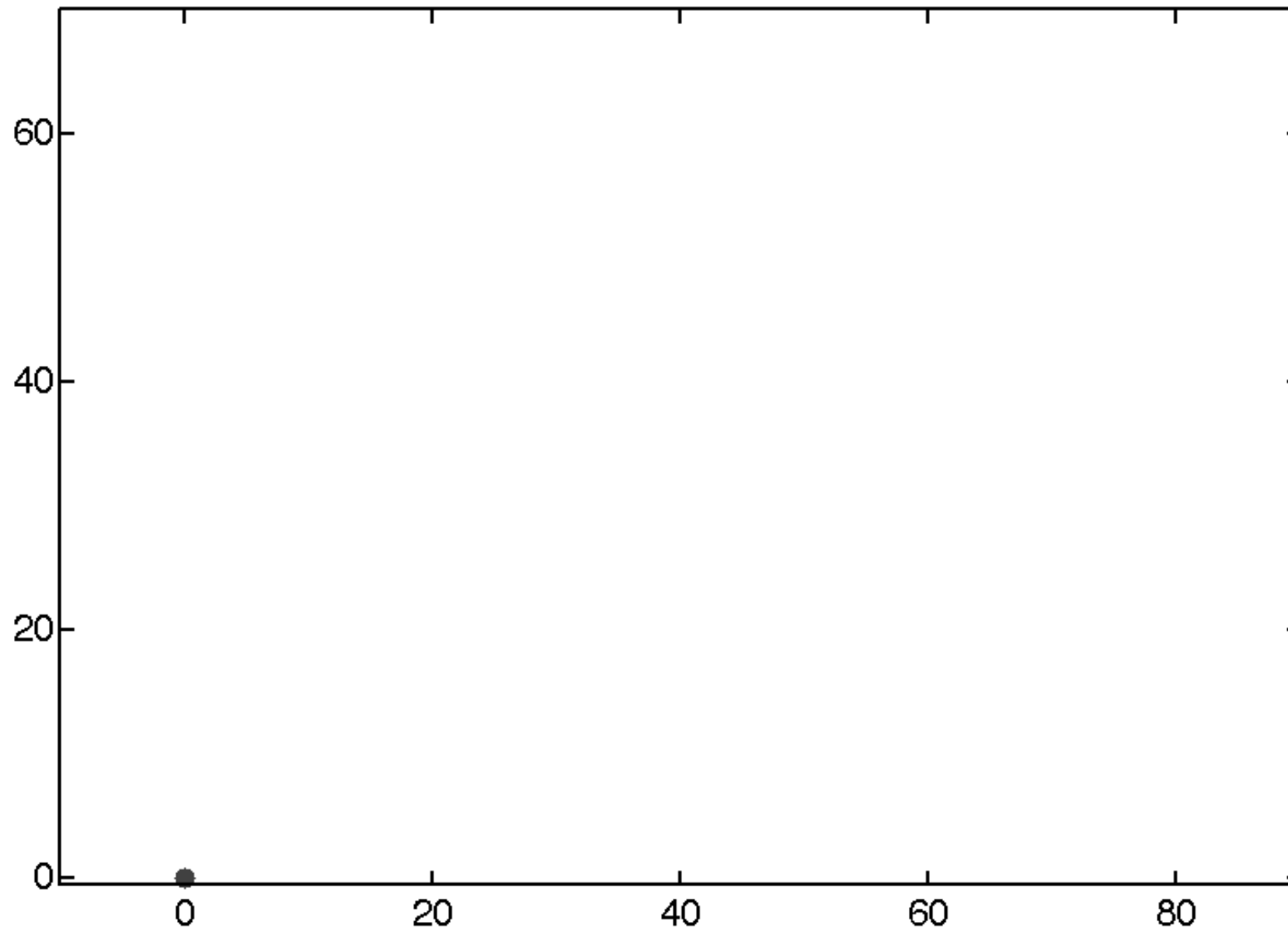
$$\mathbf{x}(k+1|k) = F \mathbf{x}(k|k) + G \mathbf{u}(k+1)$$

- We also employ a specific **ball detector** with low false alarm rate. Anyway, we still perform the compatibility test
- All other parameters remain unchanged





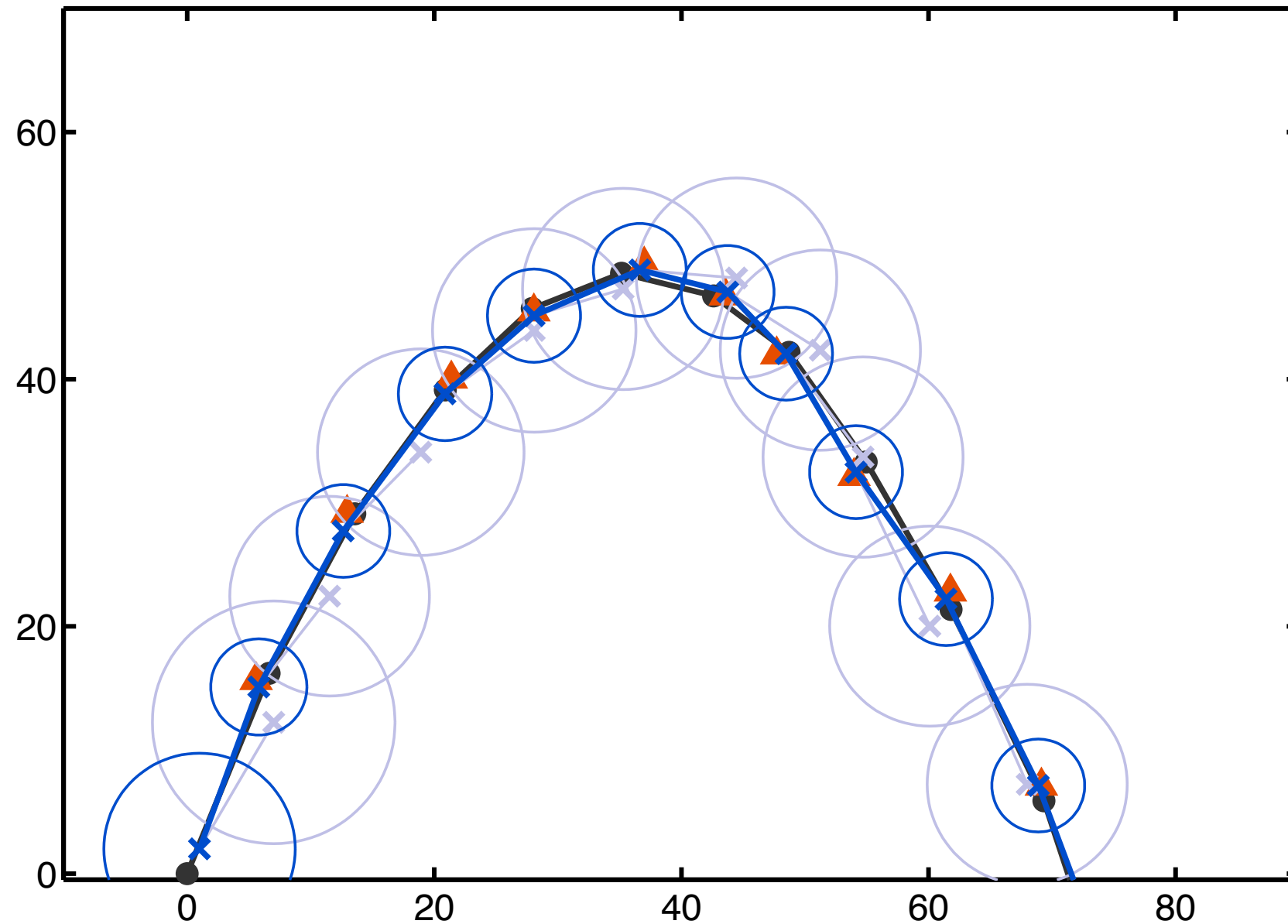
## Kalman Filter Example



Ground truth    Observations    State estimates    State predictions



## Kalman Filter Example



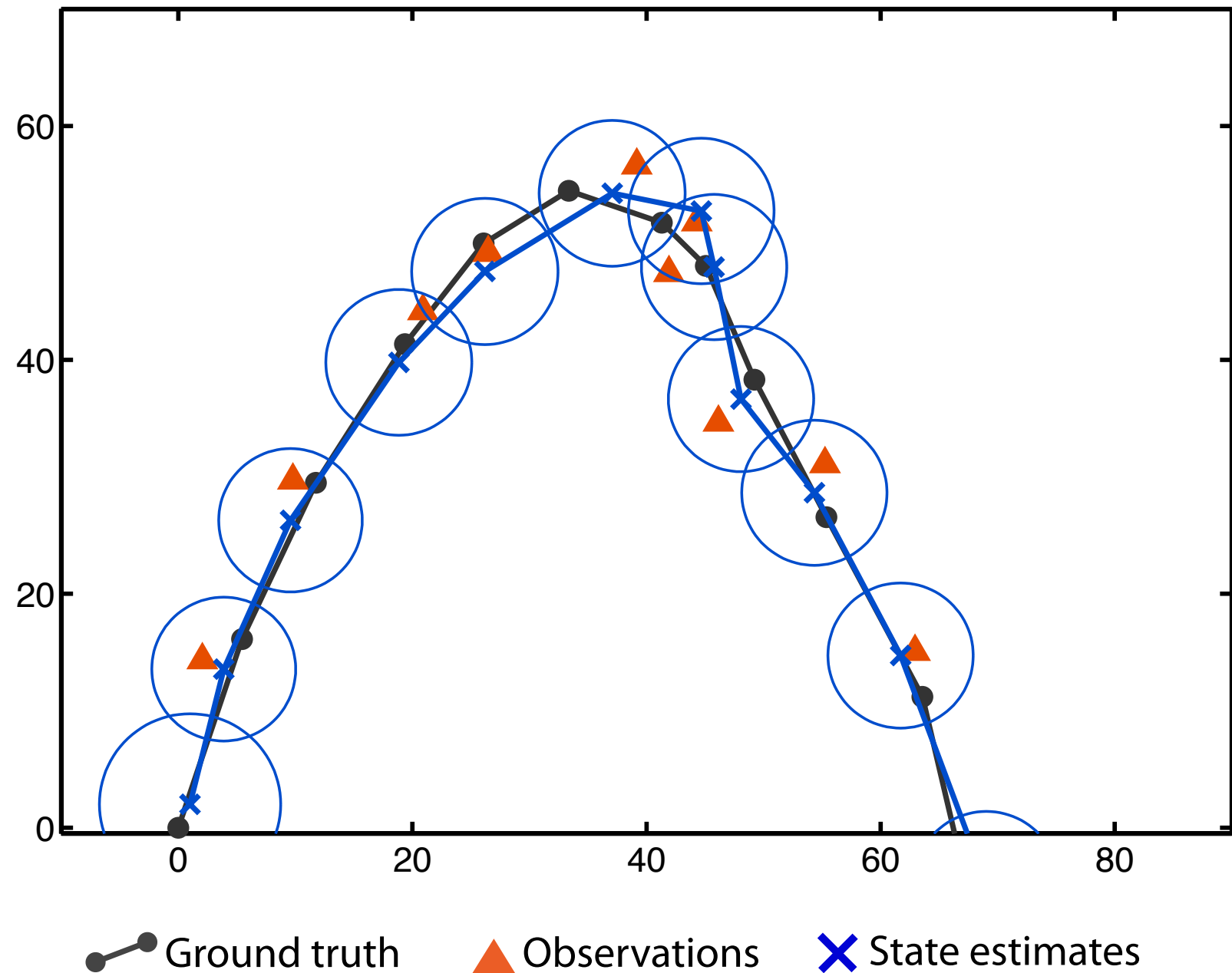
- **Good** state predictions
- **Good** velocity estimates
- **Good** tracking accuracy

Ground truth    Observations    State estimates    State predictions

## Kalman Filter Example

### Why “Filtering”?

- The Kalman filter **reduces the noise** of the observations
- Hence the name **filtering**
- Rooted in early works in **signal processing** where the goal is to filter out the noise in a signal



## Kalman Filter

- Under the linear-Gaussian assumptions, the Kalman filter is the optimal solution to the recursive Bayes filtering problem. No algorithm can do better than the Kalman filter under these conditions
- Concretely, the Kalman filter is the **optimal minimum mean squared error (MMSE) estimator**
- If we define the estimation error to be

$$\tilde{\mathbf{x}} = \mathbf{x} - \overset{\triangle}{\hat{\mathbf{x}}} \quad \text{the ground truth}$$

then, “optimal” means that the algorithm processes observations in a way that the state estimates minimize the **minimum squared error (MSE)**

$$\text{MSE}(\tilde{\mathbf{x}}) = E[\tilde{\mathbf{x}}^2] = E[(\mathbf{x} - \overset{\triangle}{\hat{\mathbf{x}}})^2]$$

## Extended Kalman Filter

- But what if the (very strong) linear Gaussian assumption is not met? What if the process model or the observation models are **nonlinear**?
- This brings us to the **Extended Kalman filter** (EKF) that can deal with nonlinear process and nonlinear observation models
- While our regular LDS model was

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + G_k \mathbf{u}_k + \mathbf{v}_k$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{w}_k$$

the EKF makes **no linearity assumptions** about the those models

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

$$\mathbf{z}_k = h(k, \mathbf{x}_k) + \mathbf{w}_k$$

## Extended Kalman Filter

- Again, for notation simplicity, we make the assumption of **time-invariant** models (extension is straightforward)

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

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{w}_k$$

- All other variables (e.g. initial states) and assumptions (e.g. mutually independent noise terms) are the same than in the Kalman filter
- The **main consequence** of this extension concerns the way how the uncertainties of states and observations are propagated through the new nonlinear models
- So let us return to the problem of **error propagation**, now for **nonlinear functions**

## Error Propagation

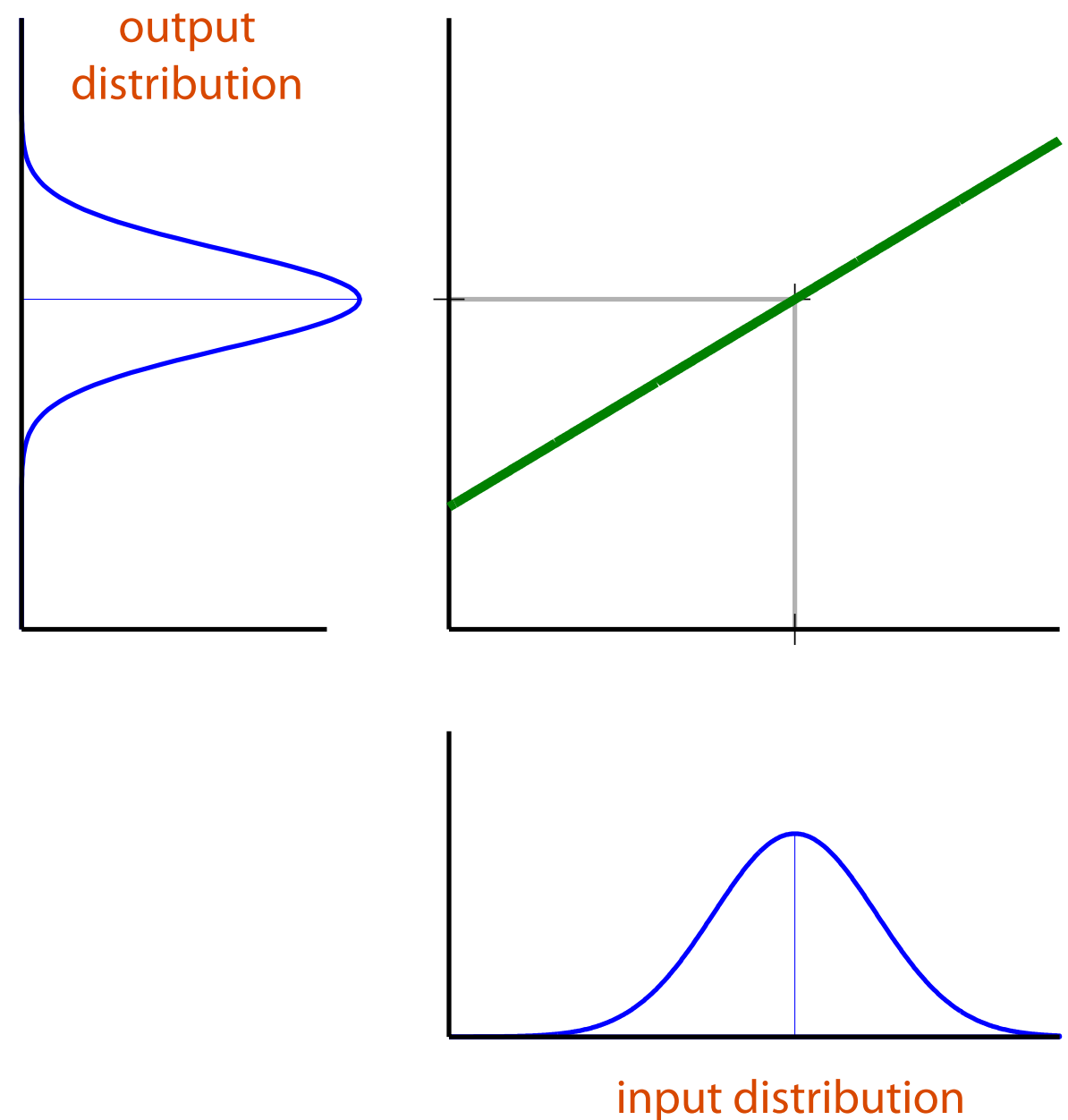
- We have seen that transferring a **Gaussian** random variable across a **linear function** results again in a **Gaussian** with parameters

$$\mathbf{y} \sim \mathcal{N}_{\mathbf{y}}[A \boldsymbol{\mu}_{\mathbf{x}} + b, A \Sigma_{\mathbf{x}} A^T]$$

- The relationship for the output covariance matrix

$$\Sigma_y = A \Sigma_x A^T$$

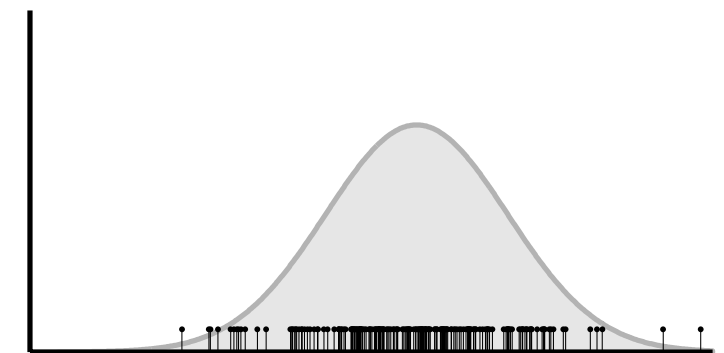
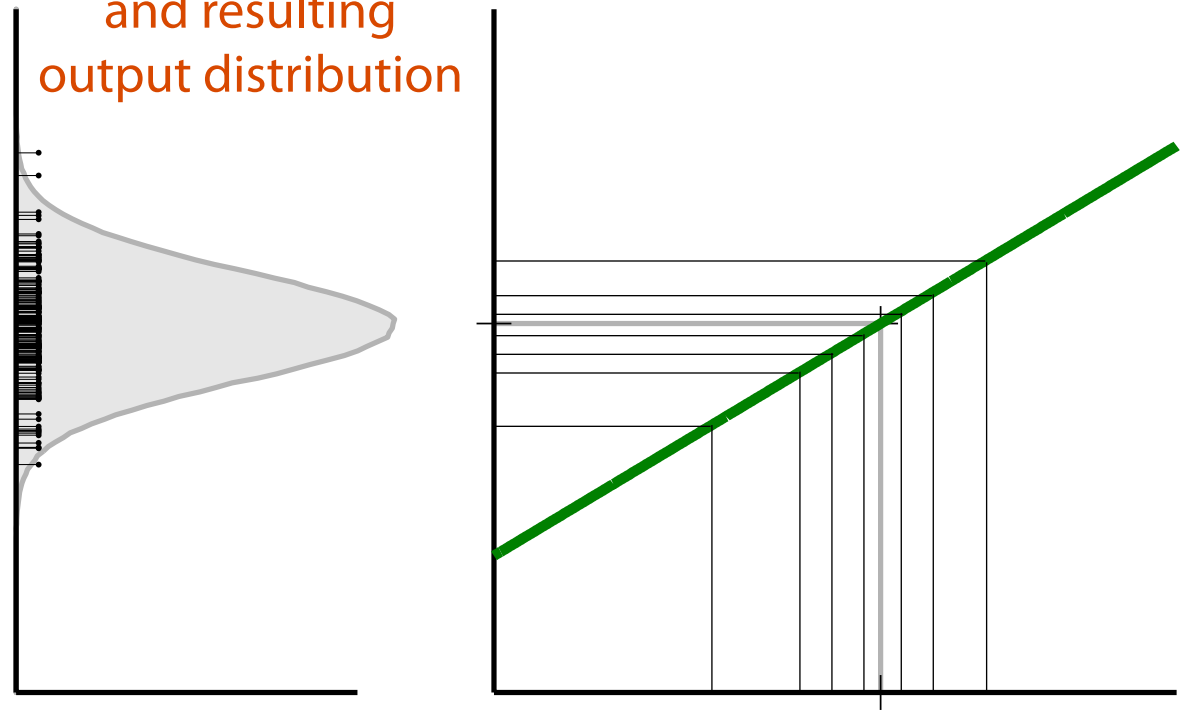
is called **error propagation law**



## Error Propagation

- A different approach to the propagation of uncertainty is **Monte Carlo error propagation**
- Relies on a non-parametric **sample-based** representation of uncertainty
- Error propagation is done by simply transferring **each sample**
- Here, we can **draw samples** from the input distribution and propagate, histogram and normalize them at the output
- This gives the **output distribution**

transferred samples  
and resulting  
output distribution

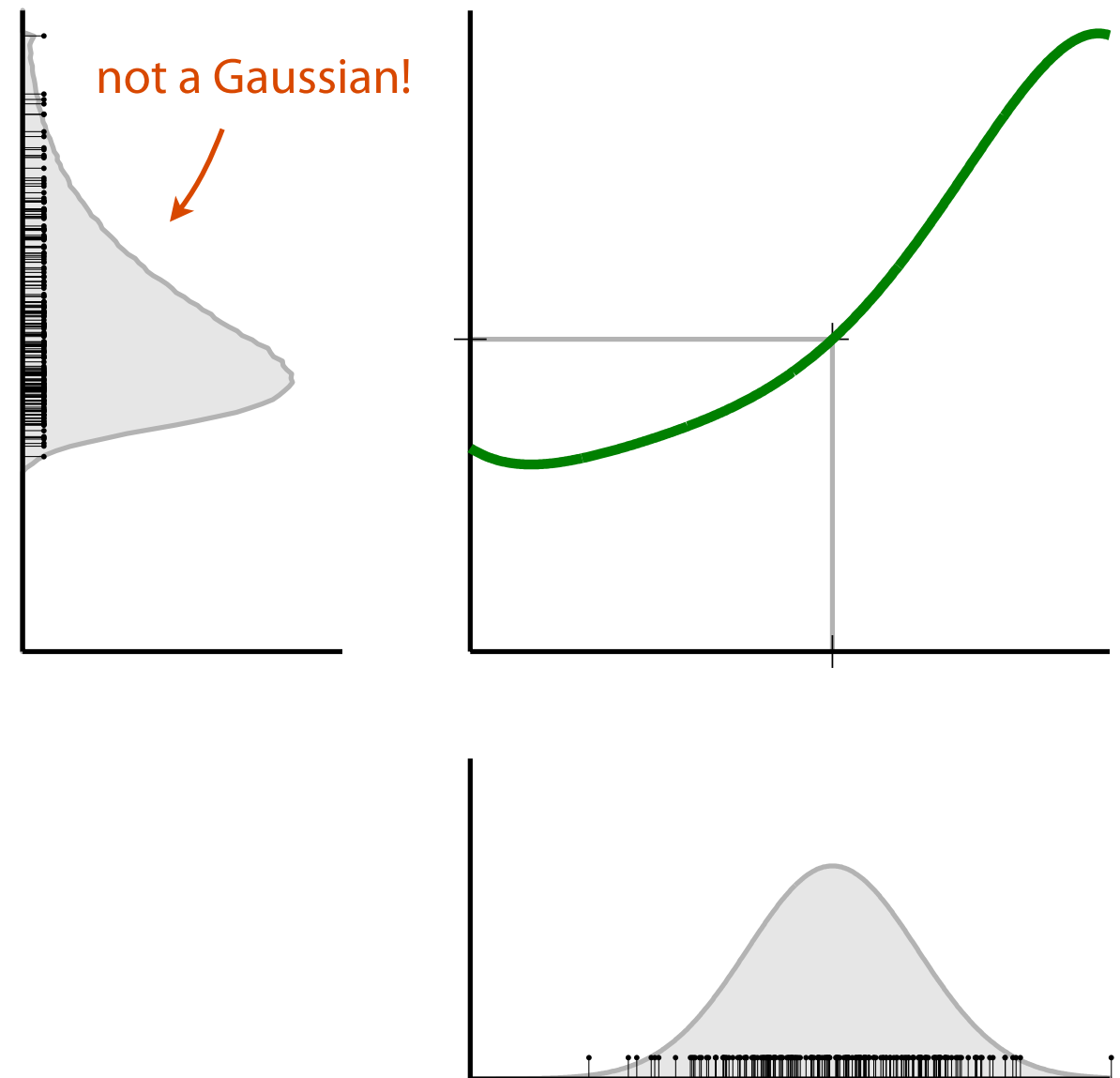


samples drawn from  
input distribution



## Error Propagation

- Monte Carlo error propagation is great to show what happens when the function is **nonlinear**
- The output distribution is **not a Gaussian anymore!**
- Monte Carlo error propagation has the advantage of being **general** but is computationally **expensive** particularly in high dimensions
- Many samples are needed to achieve good accuracy

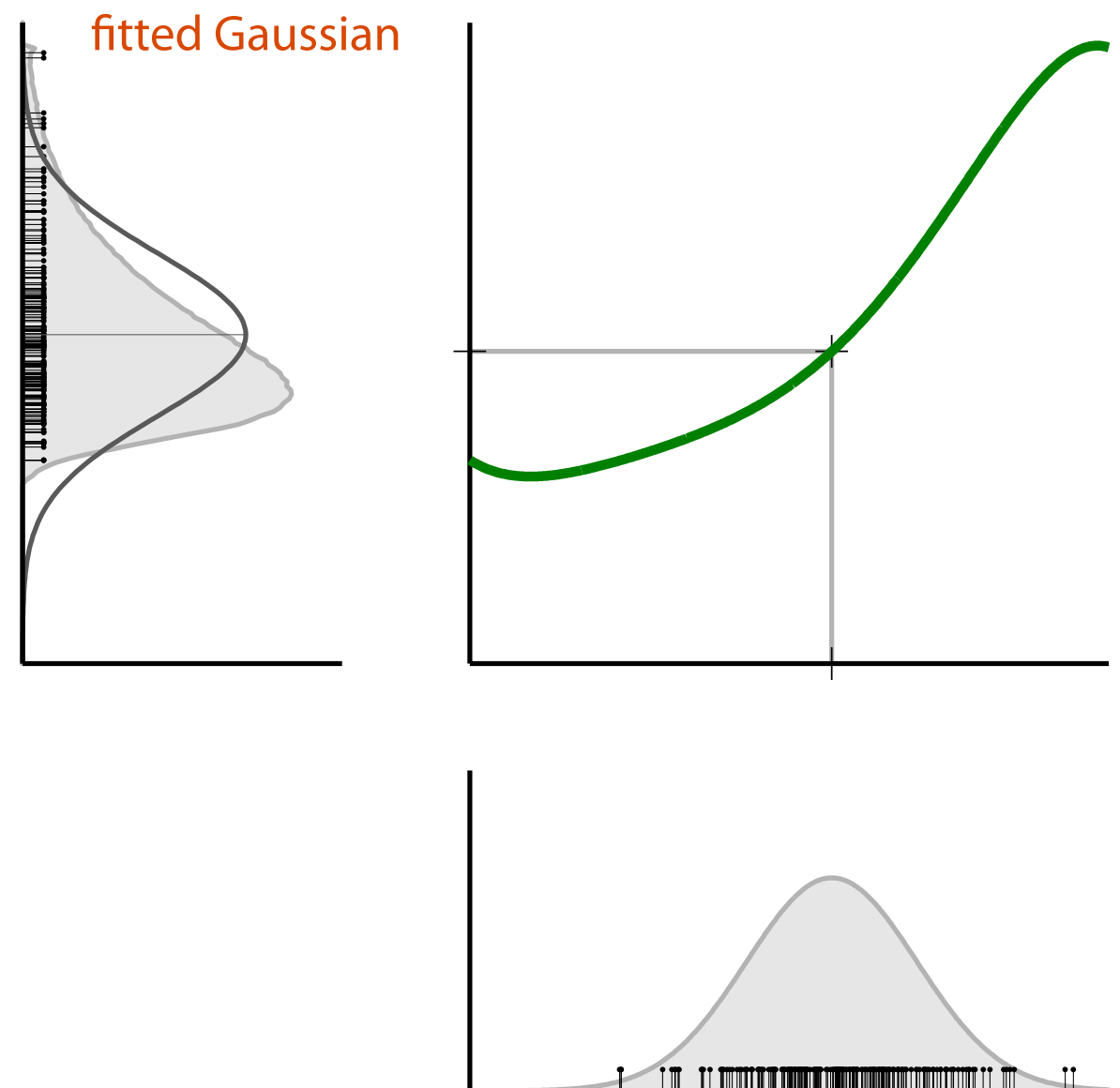


## Error Propagation

- If Gaussian distributions are required – which is the case in Kalman filtering – we can **fit the parameters**  $(\mu_y, \Sigma_y)$  of a normal distribution to the  $N$  propagated samples
- With  $\mathbf{x}^{[i]}$  being a sample

$$\mu_y = \frac{1}{N} \sum_{i=1}^N \mathbf{x}^{[i]} \quad \leftarrow \text{sample mean and covariance}$$
$$\Sigma_y = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}^{[i]} - \mu)(\mathbf{x}^{[i]} - \mu)^T$$

- This is the **best** maximum likelihood **estimate** of the Gaussian output distribution



## Error Propagation

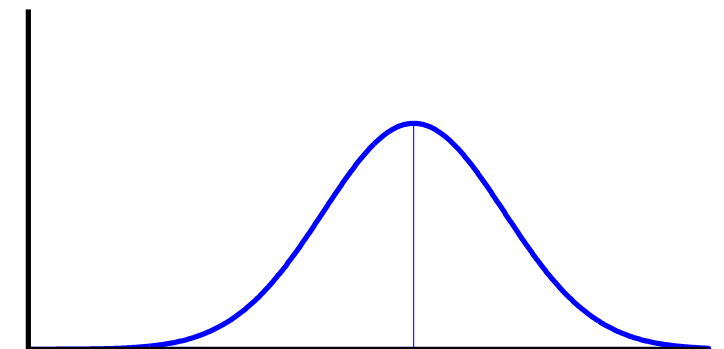
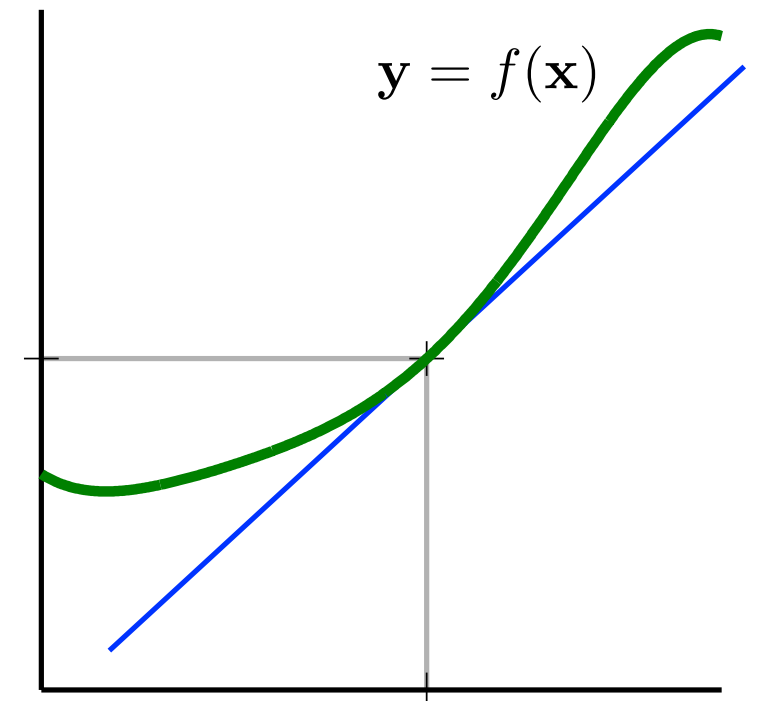
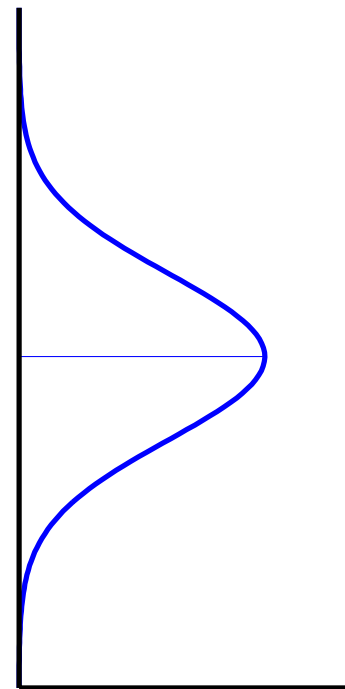
- Because Monte Carlo methods may be costly, we consider the following approach: we represent the nonlinear function

$$y = f(\mathbf{x})$$

by a **Taylor series expansion**

$$y = f(\boldsymbol{\mu}_{\mathbf{x}}) + \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}}) + \frac{1}{2} \frac{\partial^2 f}{\partial \mathbf{x}^2}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{x}})^2 + \dots$$

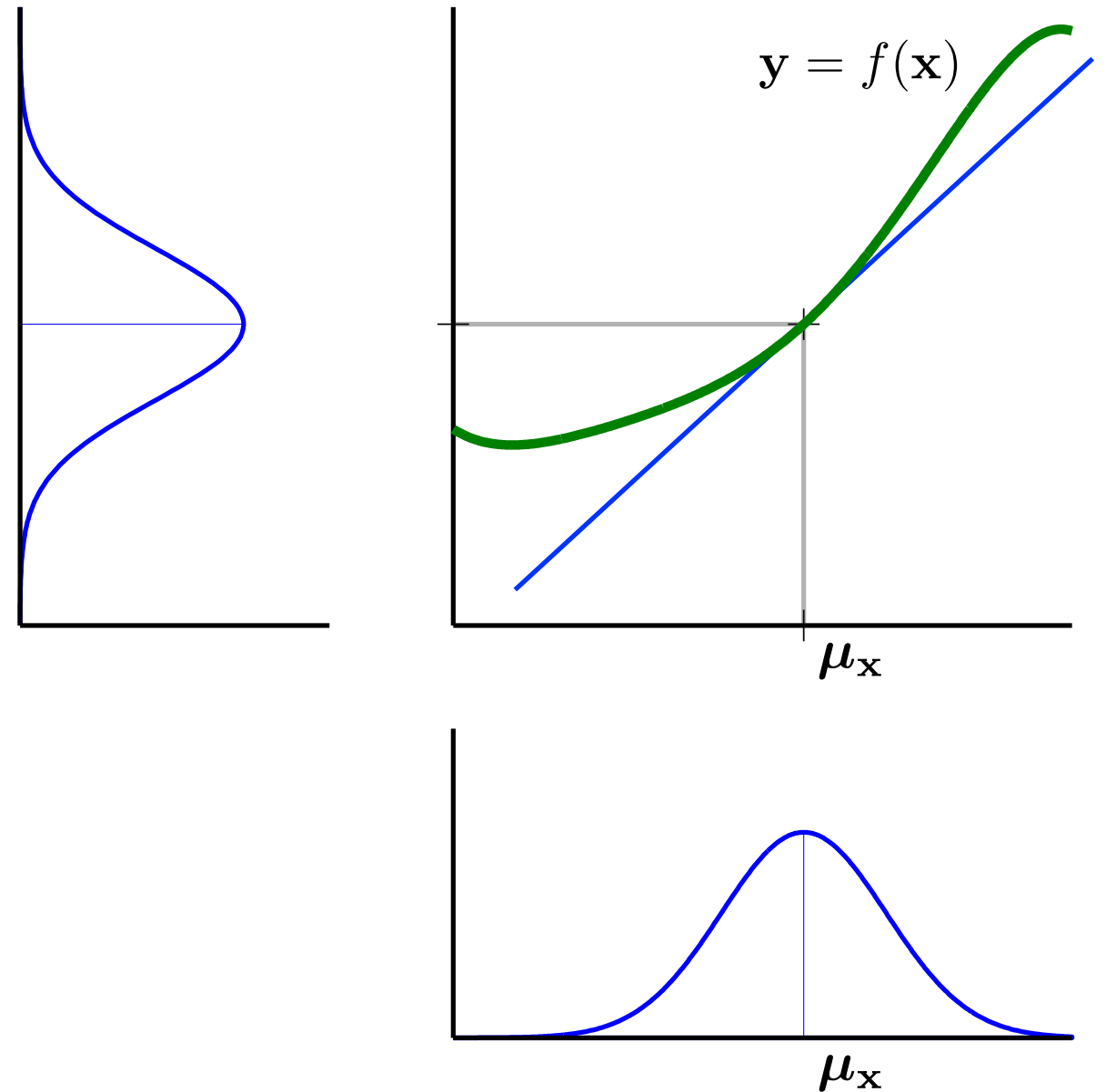
- Then, we truncate the series after the first-order term. This corresponds to a **linearization** of  $f$



## Error Propagation

- This approach is called **first-order error propagation**
- Second (or higher) order error propagation is rarely used because the higher order terms are typically complex to derive (e.g. Hessian)
- We linearize always around the most probable value, i.e. the **mean**

$$y \approx f(\mu_x) + \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\mathbf{x}=\mu_x} (\mathbf{x} - \mu_x)$$



## Error Propagation

- For one dimension we have

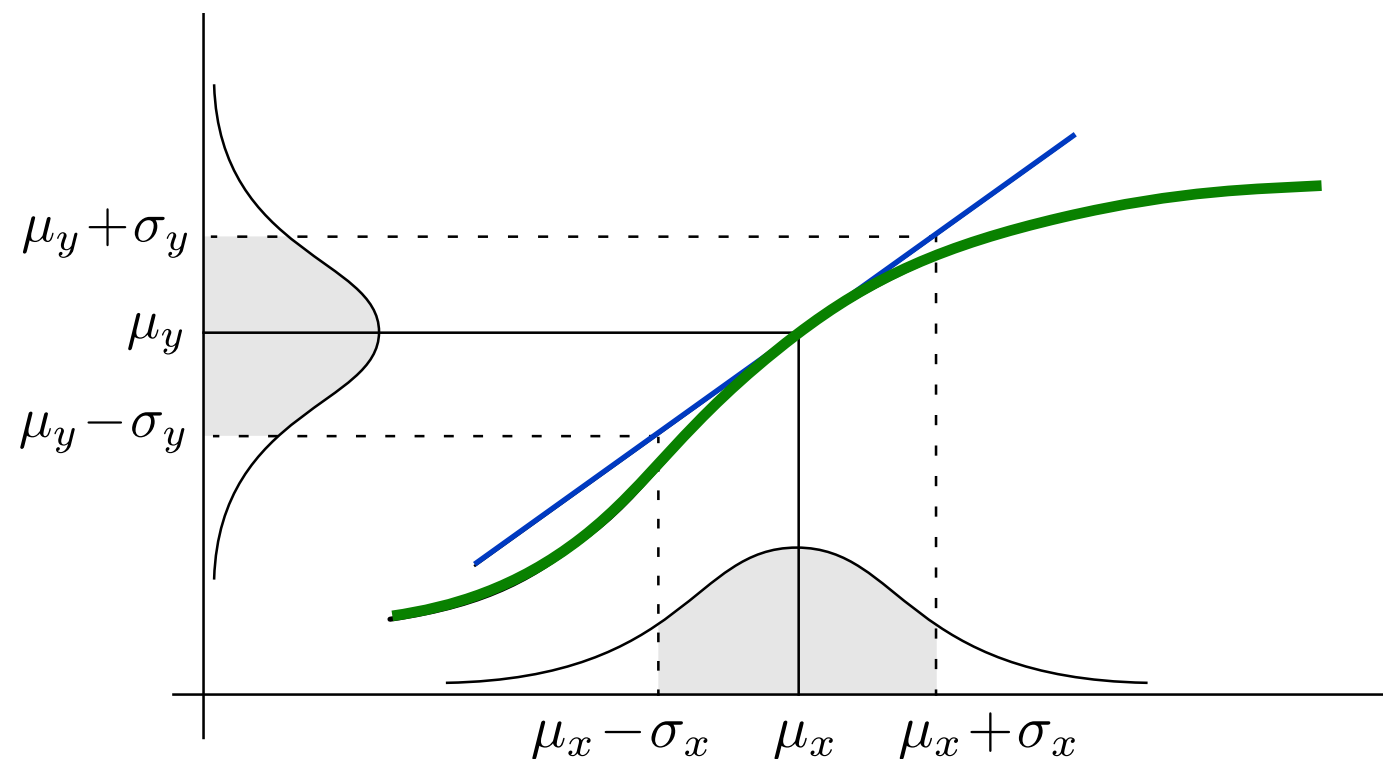
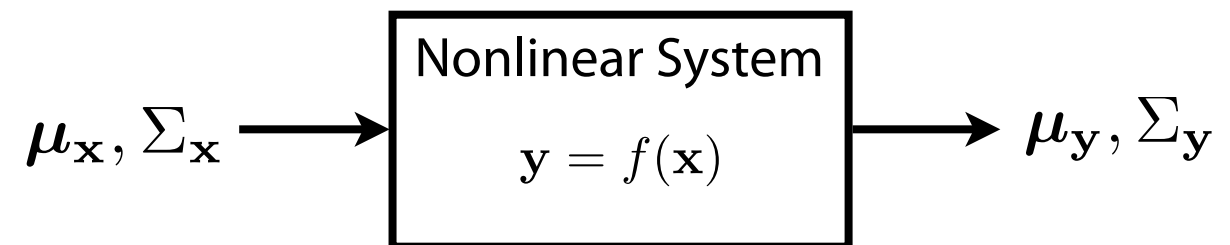
$$y \approx f(\mu_x) + \left. \frac{\partial f}{\partial x} \right|_{x=\mu_x} (x - \mu_x)$$

- Looking for the parameters of the output distribution  $\mu_y, \sigma_y^2$  we find immediately

$$\mu_y = f(\mu_x)$$

$$\sigma_y^2 = \left( \frac{\partial f}{\partial x} \right)^2 \sigma_x^2$$

from slope =  $\frac{\Delta y}{\Delta x}$



## Error Propagation

- How does this scale to  $n$  dimensions?
- The “ $n$ -dimensional derivative” is known as the **Jacobian matrix**. The Jacobian is defined as the outer product of vector-valued function and gradient operator

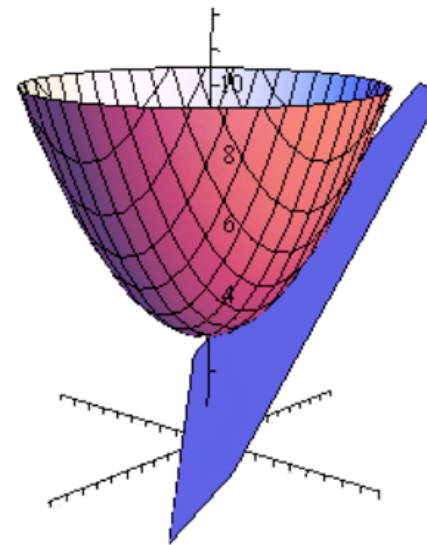
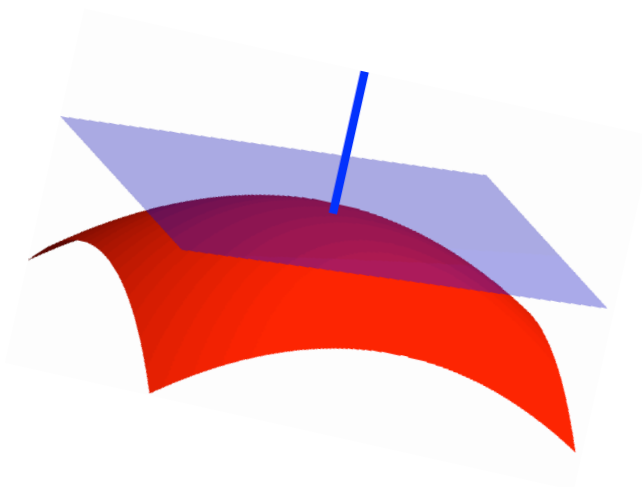
$$F = f(\mathbf{x}) \cdot \nabla_{\mathbf{x}}^T$$

$$F = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \cdots & \frac{\partial}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

with  $\nabla_{\mathbf{x}} = \left( \frac{\partial}{\partial x_1} \quad \frac{\partial}{\partial x_2} \quad \cdots \quad \frac{\partial}{\partial x_n} \right)^T$  being the **gradient operator** of first-order derivatives with respect to  $\mathbf{x}$

## Error Propagation

- The Jacobian gives the **orientation of the tangent plane** to a vector-valued function at a given point
- Generalizes the gradient of a scalar function



- **Non-square** matrix in general (e.g. EKF observation model Jacobian)
- For higher-order error propagation, the **Hessian** is the matrix of second-order partial derivatives of a function describing the **local curvature**

## Error Propagation

- For one dimension, we found  $\sigma_y^2 = \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2$ . Rearranging gives

$$\sigma_y^2 = \left(\frac{\partial f}{\partial x}\right) \sigma_x^2 \left(\frac{\partial f}{\partial x}\right)$$

- For  $n$  dimensions, it can be shown that the output covariance is given by

$$\Sigma_y = F \Sigma_x F^T$$

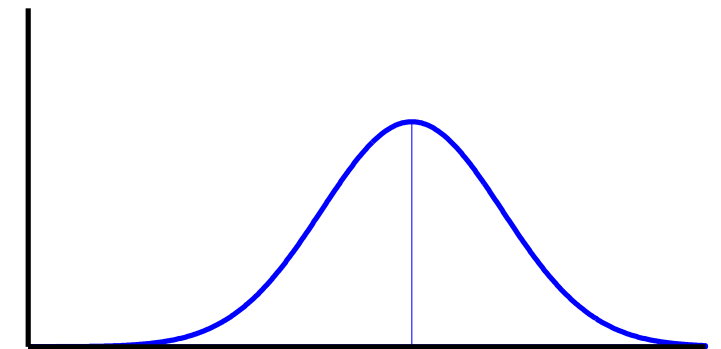
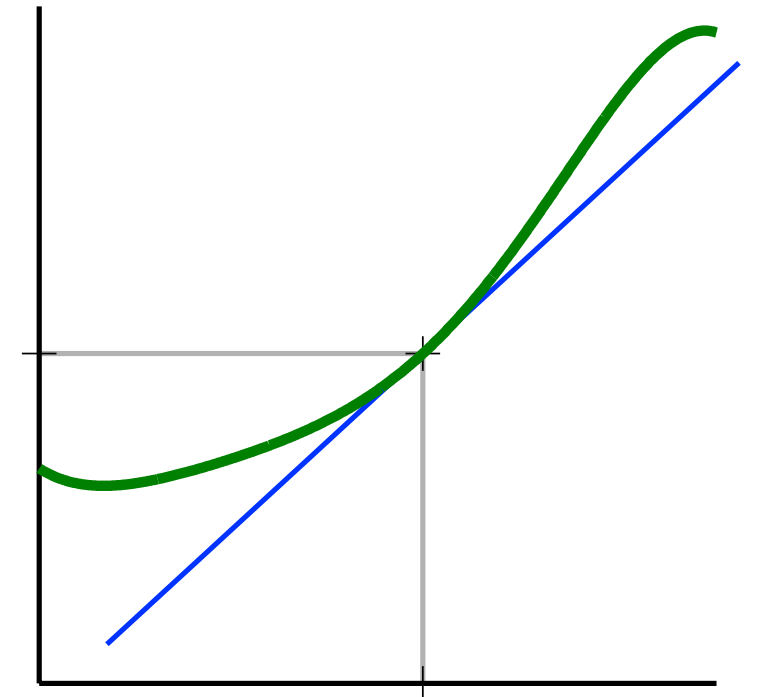
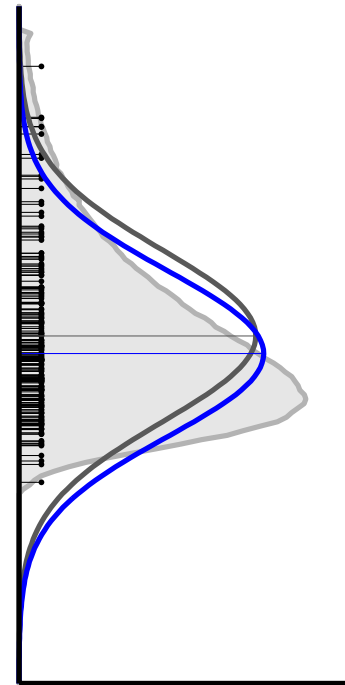
where  $F$  is the **Jacobian matrix** of the nonlinear function  $f$  linearized around the mean of  $\mathbf{x}$

- Thus, we have the **same expression** for **exact** error propagation across linear functions and **approximate** error propagation through nonlinear functions



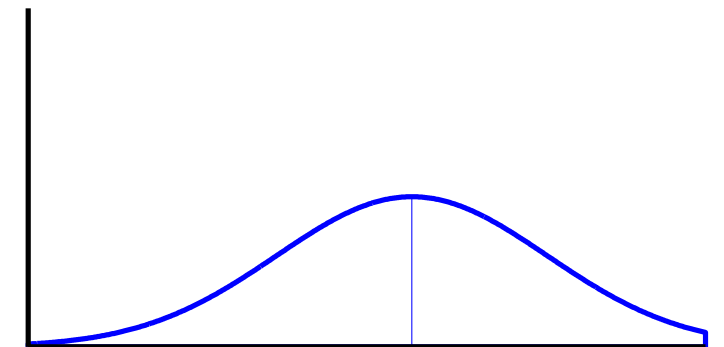
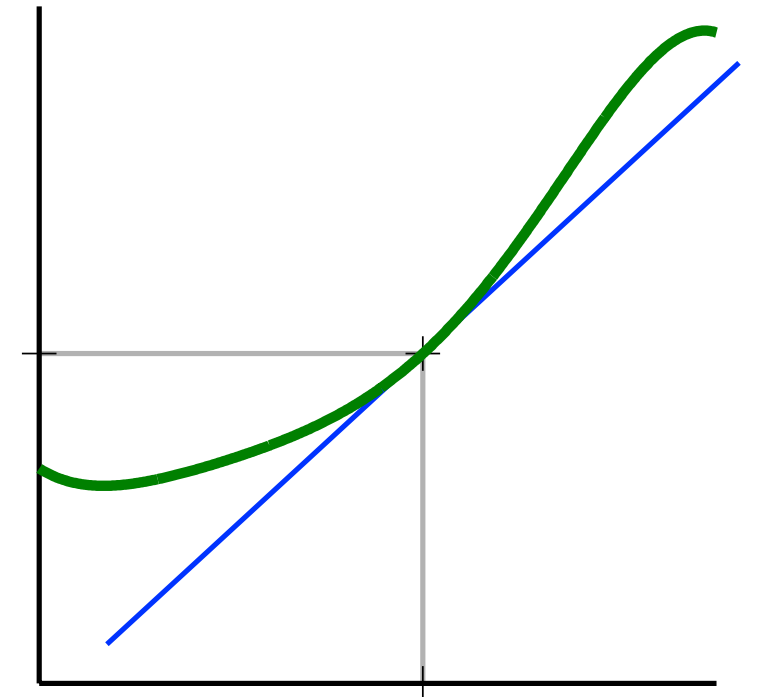
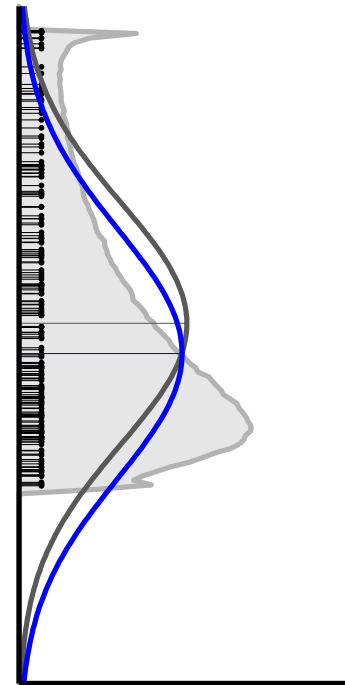
## Error Propagation

- How good is the **approximation**?
- Let us **visually** examine the approximation accuracy of first-order error propagation
- **Medium-sized** input covariance
- True distribution is slightly asymmetric, medium error from sample mean and sample variance



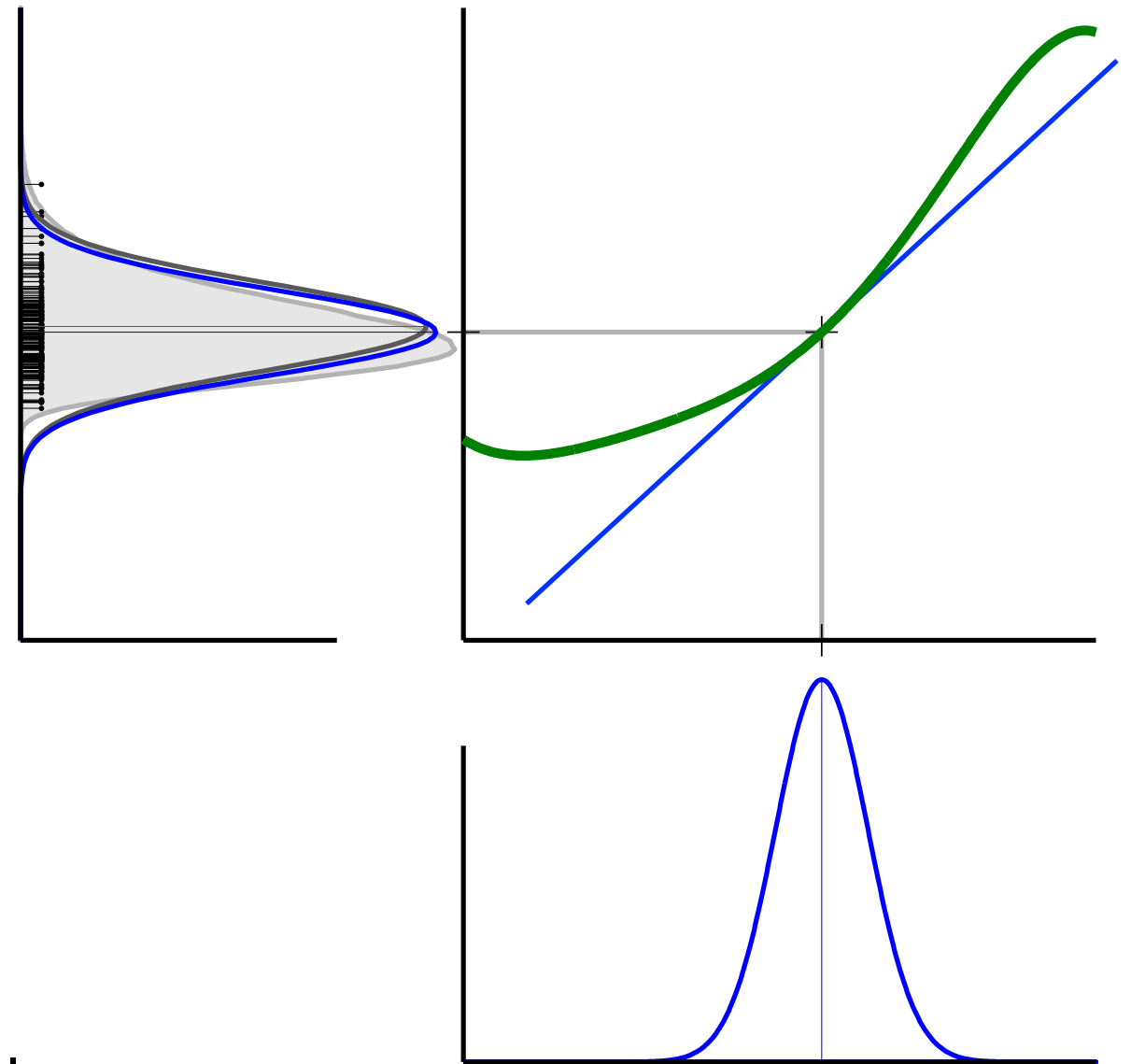
## Error Propagation

- How good is the **approximation**?
- Let us **visually** examine the accuracy of the approximation of first-order error propagation
- **Large** input covariance
- True distribution is arbitrarily shaped, has three modes, large error from sample moments
- Normal distribution is a poor model



## Error Propagation

- How good is the **approximation**?
- Let us **visually** examine the accuracy of the approximation of first-order error propagation
- **Small** input covariance
- Good correspondence of all distributions (true, fitted, first-order propagated)
- Normal distribution is a good model



## Kalman Filter

- State prediction

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

transition model

$$P(k+1|k) = F P(k|k) F^T + Q$$

- Measurement prediction

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

observation model

$$\nu(k+1) = \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1)$$

innovation

$$S(k+1) = H P(k+1|k) H^T + R$$

innovation covariance

- Update

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

Kalman gain

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

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

## Extended Kalman Filter

- State prediction

$$\begin{aligned}\mathbf{x}(k+1|k) &= f(\mathbf{x}(k|k)) \\ P(k+1|k) &= F P(k|k) F^T + Q\end{aligned}$$

Jacobian of  $f$

transition model

- Measurement prediction

$$\begin{aligned}\hat{\mathbf{z}}(k+1) &= h(\mathbf{x}(k+1|k)) \\ \nu(k+1) &= \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1) \\ S(k+1) &= H P(k+1|k) H^T + R\end{aligned}$$

Jacobian of  $h$

observation model

innovation

innovation covariance

- Update

$$\begin{aligned}K(k+1) &= P(k+1|k) H^T S(k+1)^{-1} \\ \mathbf{x}(k+1|k+1) &= \mathbf{x}(k+1|k) + K(k+1) \nu(k+1) \\ P(k+1|k+1) &= (\mathbf{I} - K(k+1) H) P(k+1|k)\end{aligned}$$

Kalman gain

## Extended Kalman Filter

- Jacobians are most often **time-varying** as the partial derivatives are **functions of the state**. We thus reintroduce the time index

$$\begin{aligned}\mathbf{x}(k+1|k) &= f(k, \mathbf{x}(k|k)) \\ P(k+1|k) &= F(k) P(k|k) F(k)^T + Q\end{aligned}$$

(the same for observation model and innovation covariance)

- In case of a **control input**, there will be **two Jacobians**, one  $n_x \times n_x$  Jacobian with partial derivatives with respect to  $\mathbf{x}$ ,  $F_{\mathbf{x}}(k)$ , and one  $n_x \times n_u$  Jacobian with partial derivatives with respect to  $\mathbf{u}$ ,  $F_{\mathbf{u}}(k)$

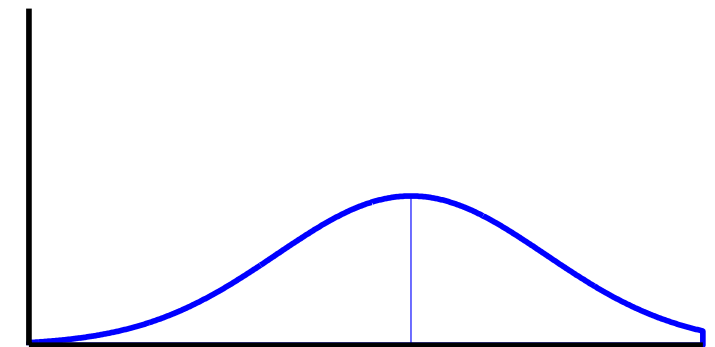
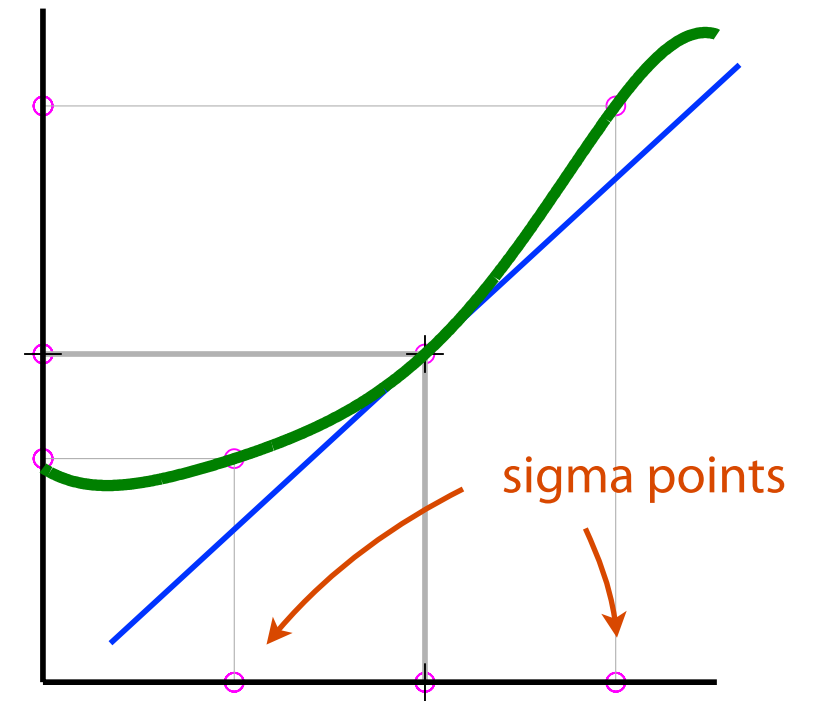
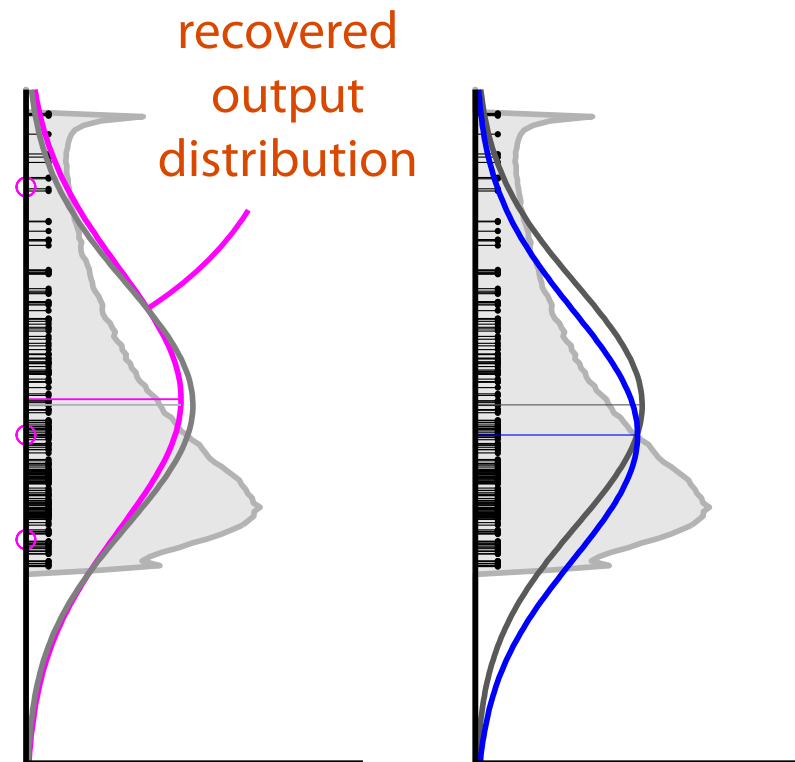
$$\begin{aligned}\mathbf{x}(k+1|k) &= f(k, \mathbf{x}(k|k), \mathbf{u}(k+1)) \\ P(k+1|k) &= F_{\mathbf{x}}(k) P(k|k) F_{\mathbf{x}}(k)^T + F_{\mathbf{u}}(k) U(k+1) F_{\mathbf{u}}(k)^T + Q\end{aligned}$$

## Unscented Transform

- The **unscented transform** is an alternative technique that has interesting properties for error propagation through nonlinear functions
- **Main idea:** rather than approximating a known function  $f$  by linearization and propagating an imprecisely-known probability distribution, use the exact nonlinear function and apply it to an approximating probability distribution
- It computes so called **sigma points**, cleverly chosen “samples” of the input distribution, that capture its mean and covariance information
- Output distribution is then recovered from the propagated sigma points
- Having a given mean and covariance in  $n$  dimensions, one requires **only**  $n+1$  sigma points to **fully encode** the mean and covariance information
- Can be viewed as a deterministic and minimal sampling technique

## Unscented Transform

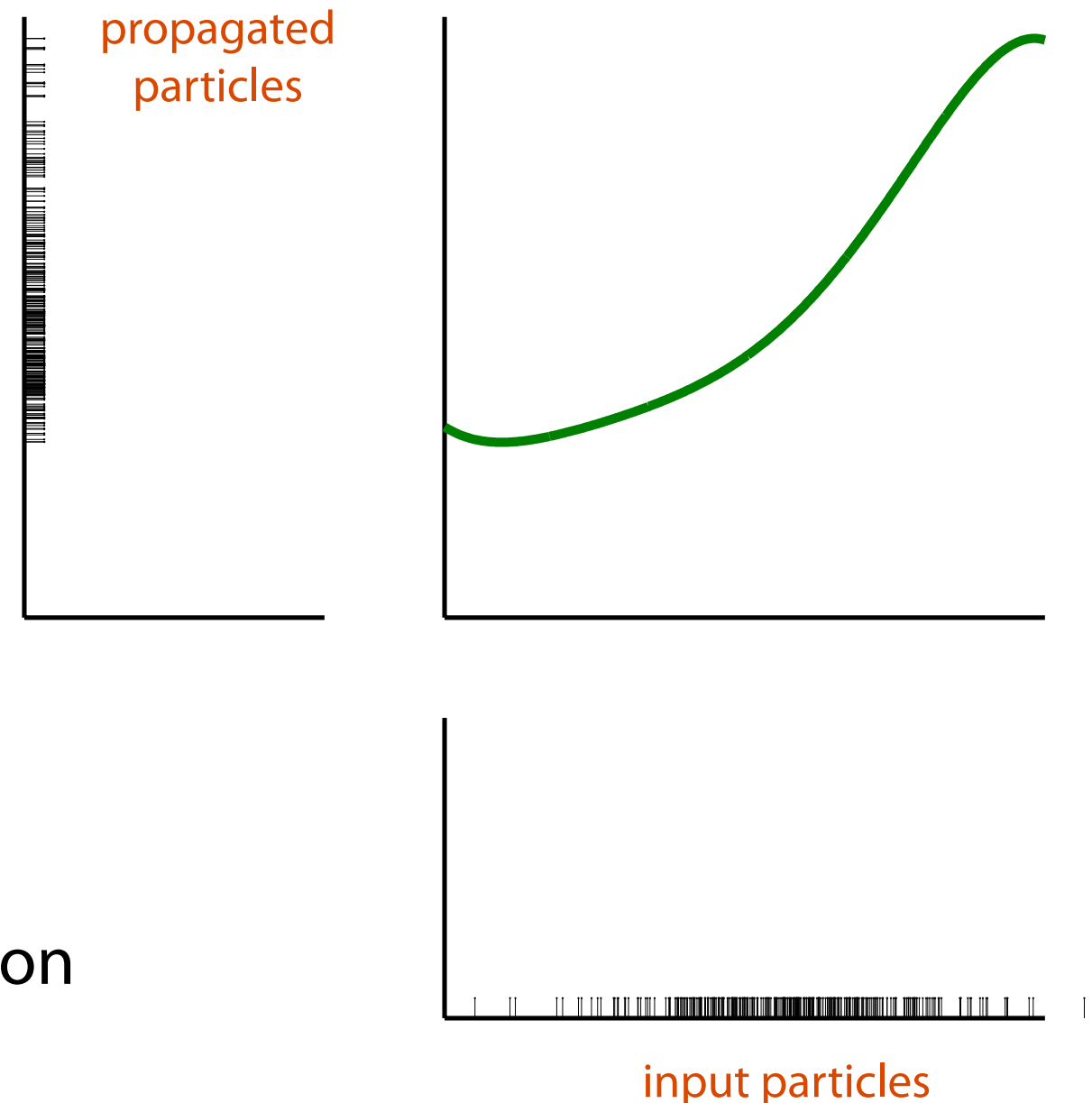
- **More accurate** than first-order error propagation particularly for highly nonlinear functions  $f$
- Works also with **non-differentiable** functions
- **No need** to derive and implement the linearizing Jacobians
- EKF with unscented transform error propagation is known as the **Unscented Kalman filter (UKF)**





## Particle Filter

- If the true distributions take **any form**, we may need to abandon the Gaussian model assumption
- Using a sample-based representation of uncertainty, the **particle filter** (PF) is a realization of the recursive Bayes filter **without any assumptions** on the underlying distributions and system models
- Basically an on-line density estimation algorithm
- There are good tutorials on PFs



## Kalman Filter: Discussion

- The Kalman filter is the workhorse of **linear** state estimation and filtering
  - It is the optimal algorithm, easy to implement and computationally very efficient
- The EKF is one of the most widely used filtering algorithms for **nonlinear** systems
  - Works very well as long as uncertainties remain small with respect to the degree of nonlinearities (when the system is “almost linear on the time scale of the updates”)
  - It may underestimate the true covariance matrix and diverge more quickly due to modeling or initialization errors – problems that mainly arise from its first-order error propagation
  - Good example: EKF robot localization. Counter-example: EKF SLAM
- The UKF relies on the **unscented transform** that provides more accurate error propagation for nonlinear models

## Summary

- We have considered **linear dynamical systems** (LDS), temporal probability models under the linear-Gaussian assumption with continuous state and observation variables
- LDS are defined by the three parameters transition/process model, observation model and prior
- The four inference tasks of HMM also exist for LDS. We have considered **filtering**, **prediction** and **most likely sequence** (smoothing has been skipped for time reasons – there is also a Kalman smoother)
- Using Gaussian LDS parameters, the recursive Bayes filter becomes the **Kalman filter**, a widely applied estimation technique for linear systems
- The Kalman filter is basically a recursive **weighted average** of the prediction and observation

## Summary

- The extended Kalman filter (EKF) can deal with **nonlinear** process and observation models. It relies on first-order error propagation which models the system as **locally linear** in regions around the respective means
- The EKF works well as long as nonlinearities within those local regions are small, i.e. as long as first-order error propagation gives Gaussian state distributions that are a reasonable approximation to the true posterior
- The **unscented transform**, and the resulting unscented Kalman filter (UKF), uses a deterministic sampling strategy of the input distribution for improved error propagate across less well-behaved and/or highly nonlinear functions
- The particle filter uses a sample-based representation of uncertainty. It is an instance of the recursive Bayes filter **without any modeling assumptions** on the underlying distributions and system models

## Sources and Further Reading

These slides follow roughly the derivation of LDS and Kalman filtering by Russell and Norvig [1] (chapter 15) and Bishop [2] (chapter 13). A comprehensive treatment of non-parametric filtering (histogram filter and particle filter), particularly for robotics, is given by Thrun et al. [3]. The tutorial by Maybeck [4] is a good start, the textbook by Bar-Shalom et al. [5] is a comprehensive treatment of Kalman filters.

- [1] S. Russell, P. Norvig, "Artificial Intelligence: A Modern Approach", 3rd edition, Prentice Hall, 2009. See <http://aima.cs.berkeley.edu>
- [2] C.M. Bishop, "Pattern Recognition and Machine Learning", Springer, 2nd ed., 2007. See <http://research.microsoft.com/en-us/um/people/cmbishop/prml>
- [3] S. Thrun, W. Burgard, D. Fox, "Probabilistic Robotics", MIT Press, 2005
- [4] P.S. Maybeck, "The Kalman filter: An introduction to concepts", In I.J. Cox et al. (ed.), Autonomous Robot Vehicles, Springer, 1990
- [5] Y. Bar-Shalom, X. Rong Li, T. Kirubarajan, "Estimation with Applications to Tracking and Navigation", Wiley, 2001
- [6] S.J.D. Prince, "Computer vision: models, learning and inference", Cambridge University Press, 2012. See [www.computervisionmodels.com](http://www.computervisionmodels.com)