

# **무향 칼만 필터**

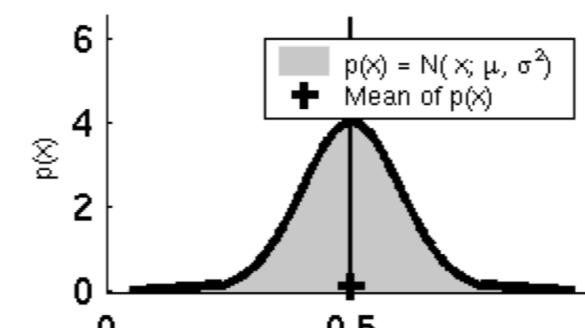
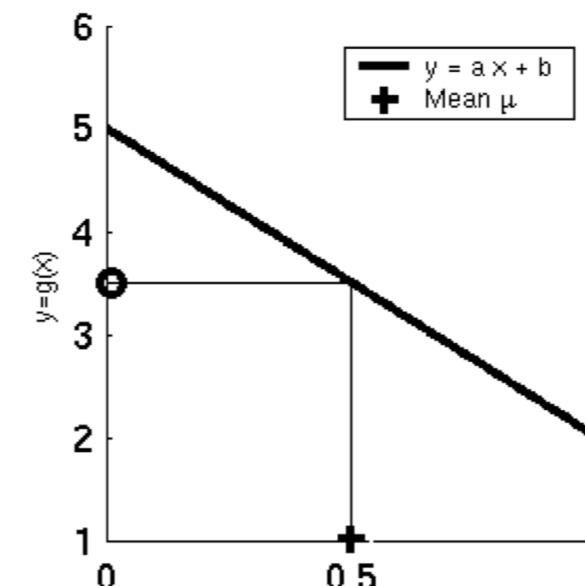
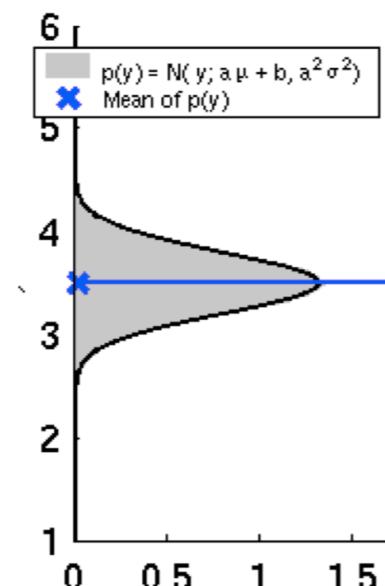
**문태봉**

**2020.03.11 (수)**

# 칼만 필터

- 칼만 필터는 언제 적용 가능한가?

- 시스템 모델: 선형 모델
- 상태, 측정 확률 분포: 가우시안 분포



$$\hat{x}_k^- = f(\hat{x}_{k-1})$$

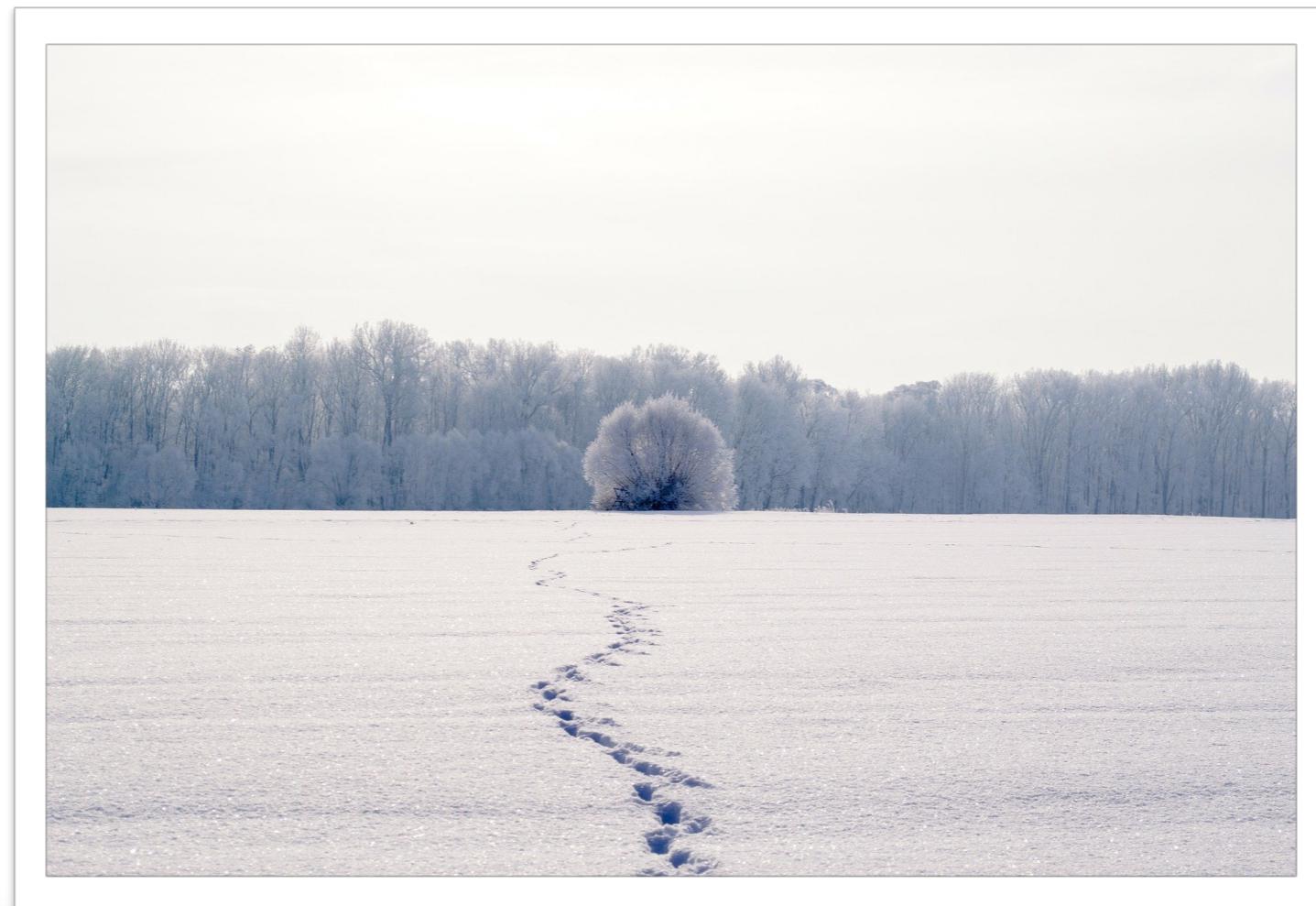
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-))$$

# 칼만 필터

- 칼만 필터는 언제 적용 가능한가?

- 시스템 모델: 선형 모델
- 상태, 측정 확률 분포: 가우시안 분포

- 하지만, 현실은 비선형적이다!



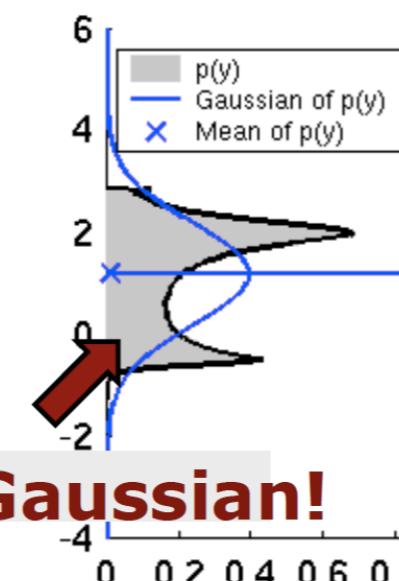
# 칼만 필터

- 칼만 필터는 언제 적용 가능한가?

- 시스템 모델: 선형 모델
- 상태, 측정 확률 분포: 가우시안 분포

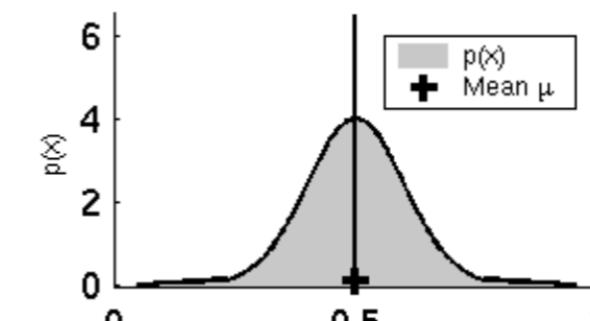
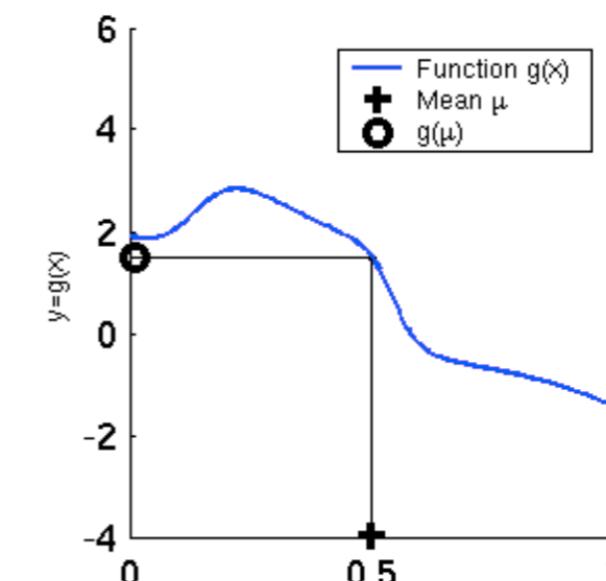
- 하지만, 현실은 비선형적이다!

가우시안 붕괴



$p(y)$ : 비선형 변환 후 출력

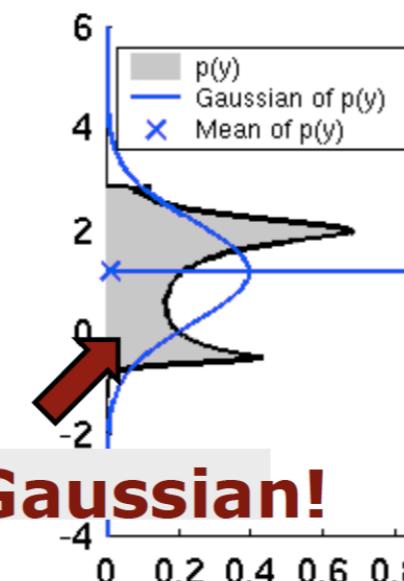
Gaussian of  $p(y)$ : 가우시안으로 변환한 최선의 결과



# 칼만 필터

- 칼만 필터는 언제 적용 가능한가?
  - 시스템 모델: 선형 모델
  - 상태, 측정 확률 분포: 가우시안 분포
- 재귀적으로 비선형 함수가 적용되면 가우시안 분포는?

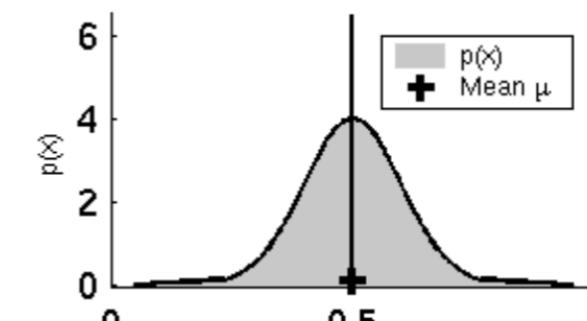
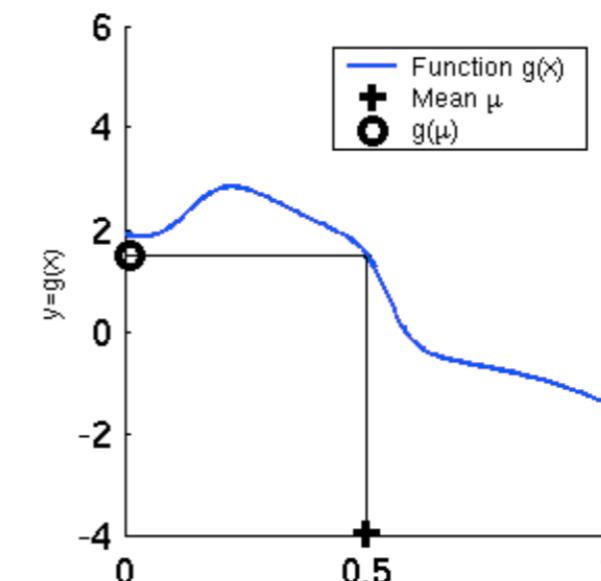
가우시안 더 붕괴



**Non-Gaussian!**

$p(y)$ : 비선형 변환 후 출력

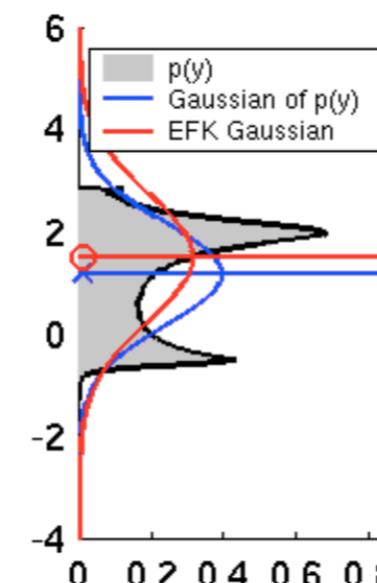
Gaussian of  $p(y)$ : 가우시안으로 변환한 최선의 결과



# 칼만 필터

- 칼만 필터는 언제 적용 가능한가?
  - 시스템 모델: 선형 모델
  - 상태, 측정 확률 분포: 가우시안 분포
- 확장 칼만 필터: 비선형 함수  $\xrightarrow{\text{테일러 근사}}$  선형 함수

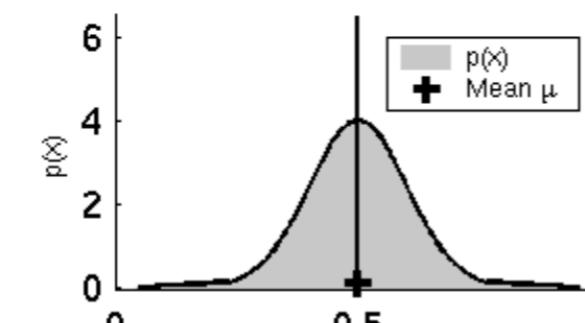
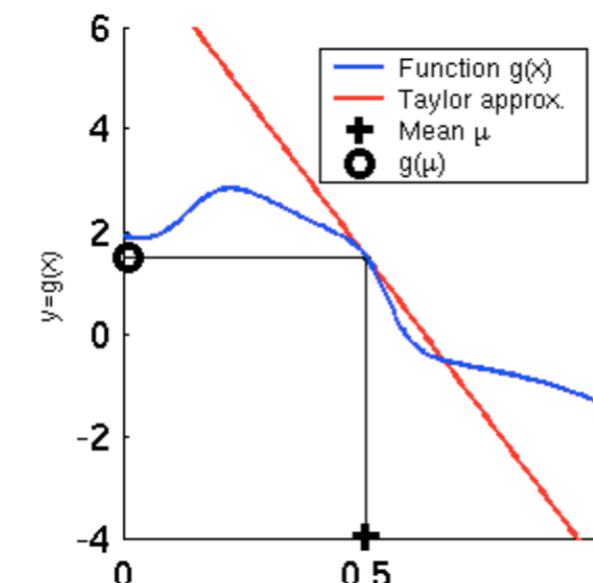
가우시안 유지



$p(y)$ : 비선형 변환 후 출력

Gaussian of  $p(y)$ : 가우시안으로 변환한 최선의 결과

EKF Gaussian: EKF에 의한 선형 변환 후 출력



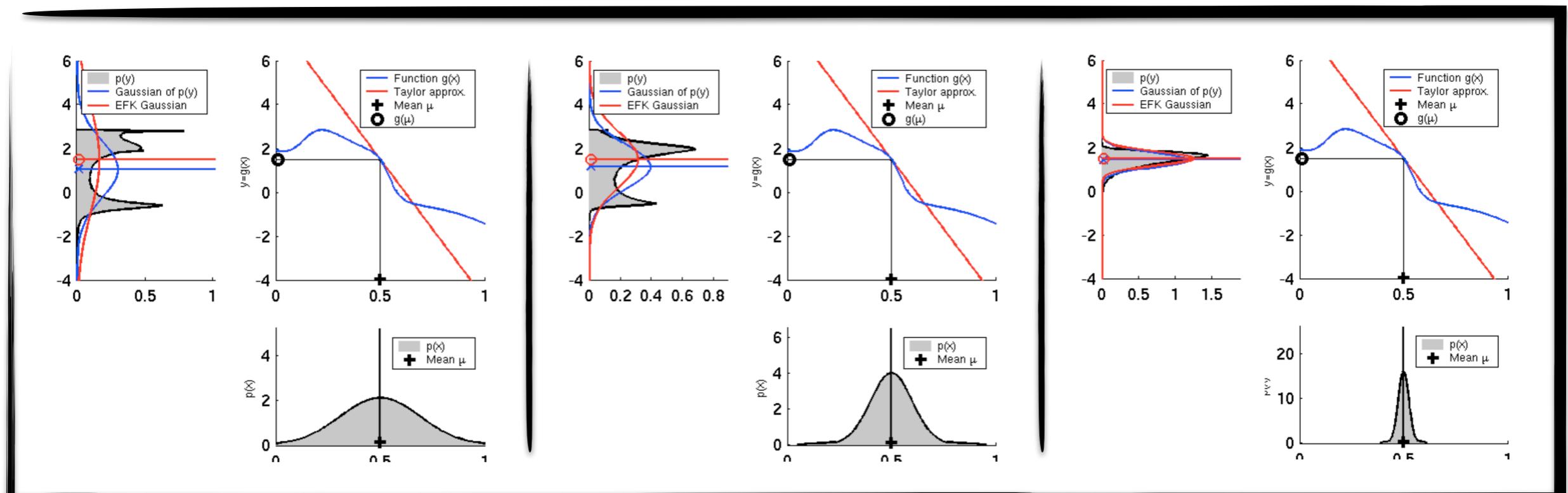
# 칼만 필터

- 칼만 필터는 언제 적용 가능한가?

- 시스템 모델: 선형 모델
- 상태, 측정 확률 분포: 가우시안 분포

- 확장 칼만 필터: 비선형 함수  $\xrightarrow{\text{테일러 근사}}$  선형 함수

공분산  $\downarrow$  = 가우시안 유지  $\uparrow$



# 칼만 필터

- 칼만 필터는 언제 적용 가능한가?
  - 시스템 모델: 선형 모델
  - 상태, 측정 확률 분포: 가우시안 분포
- 확장 칼만 필터: 비선형 함수  $\xrightarrow{\text{테일러 근사}}$  선형 함수
- 더 좋은 방법은 없을까?

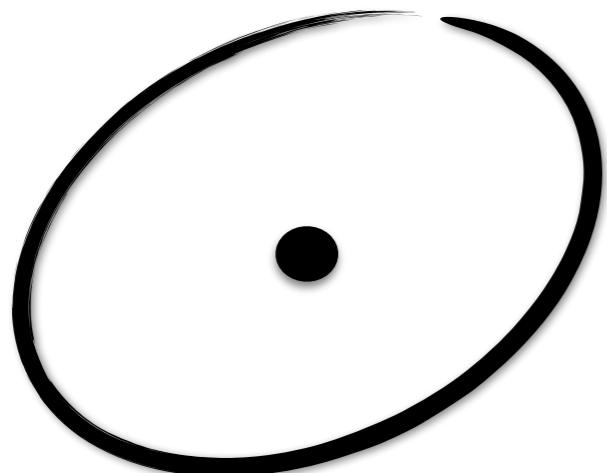
무향 변환 + 칼만 필터

# 무향 변환

# 확장 칼만 필터 (예측 단계)

현재

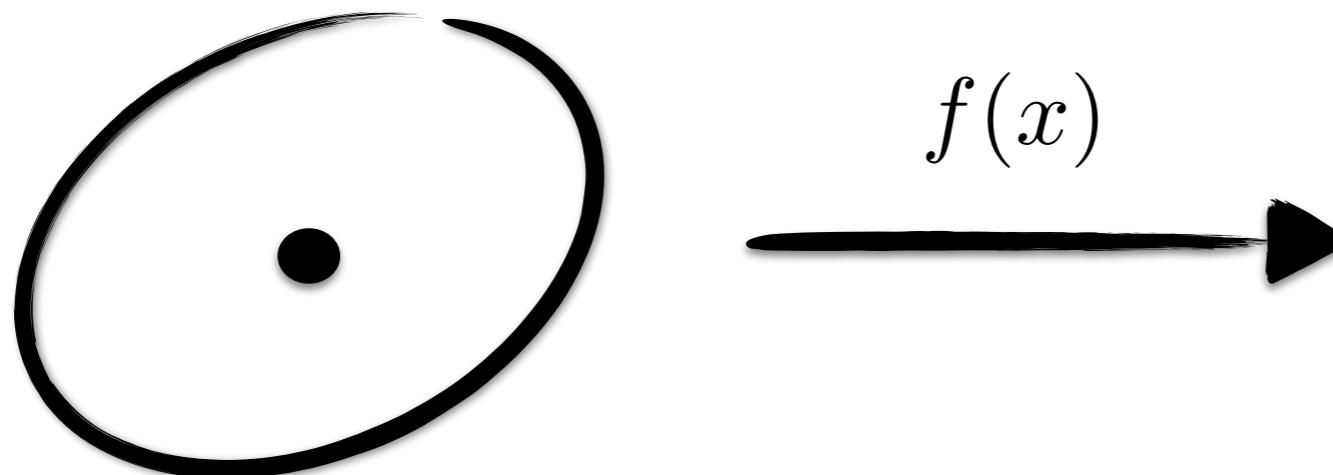
$$x \sim \text{Gauss}(\mu, \Sigma)$$



# 확장 칼만 필터 (예측 단계)

현재

$$x \sim \text{Gauss}(\mu, \Sigma)$$

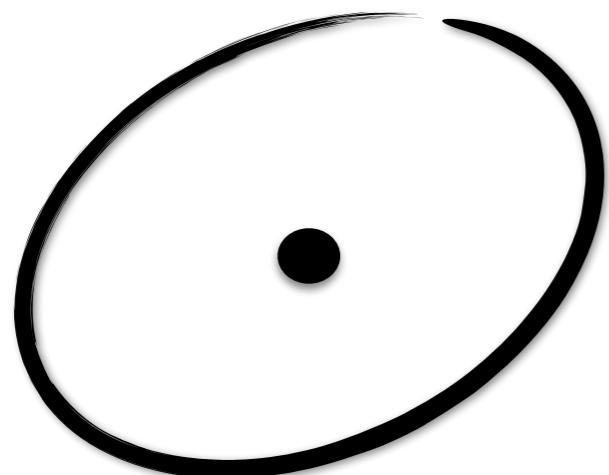


다음 상태를 예측하기 위해 현재 상태를 변환해야 한다

# 확장 칼만 필터 (예측 단계)

현재

$$x \sim \text{Gauss}(\mu, \Sigma)$$



$$\begin{array}{c} Ax \\ \longrightarrow \\ APA^T \end{array}$$

비선형 함수

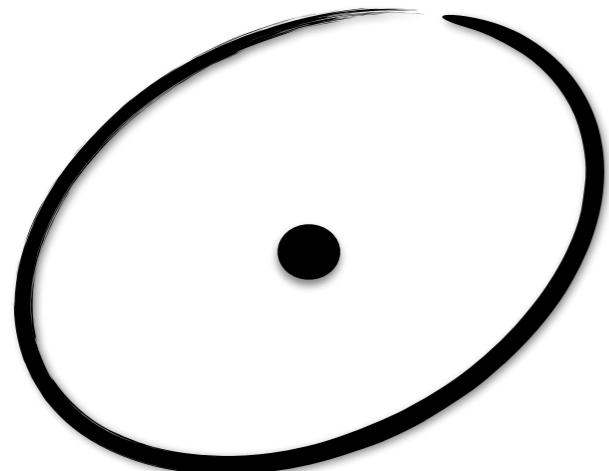
$$\xrightarrow{\text{테일러 근사}}$$

선형 함수

# 확장 칼만 필터 (예측 단계)

현재

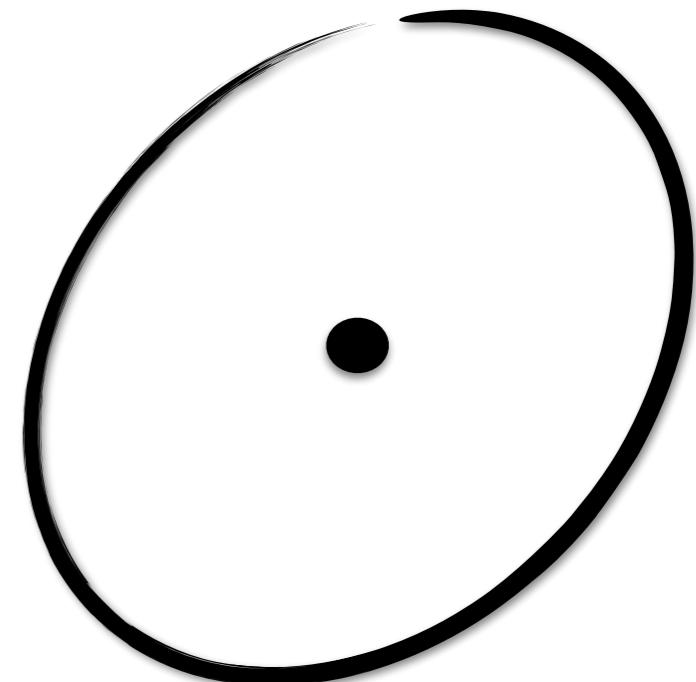
$$x \sim \text{Gauss}(\mu, \Sigma)$$



다음

$$y \sim \text{Gauss}(\mu', \Sigma')$$

$$\begin{array}{c} Ax \\ \longrightarrow \\ APA^T \end{array}$$

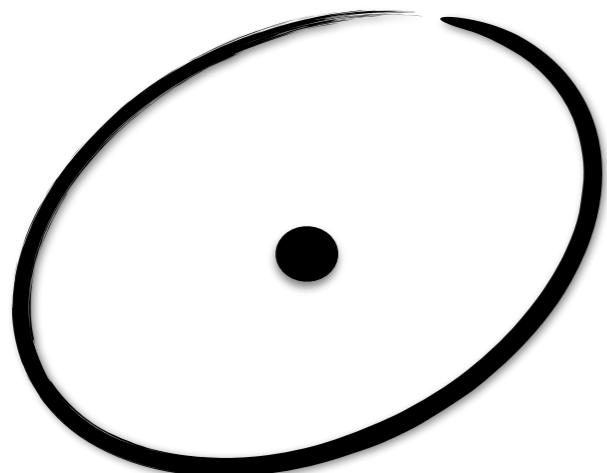


다음 상태를 가우시안 분포로 예측한다

# 무향 칼만 필터 (예측 단계)

현재

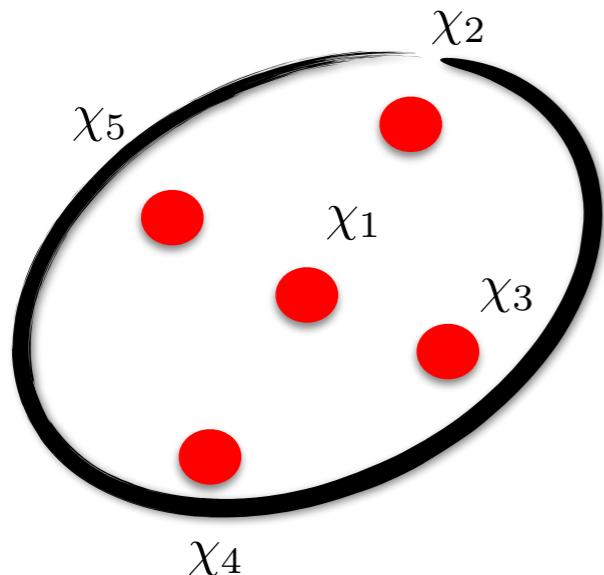
$$x \sim \text{Gauss}(\mu, \Sigma)$$



# 무향 칼만 필터 (예측 단계)

현재

$$x \sim \text{Gauss}(\mu, \Sigma)$$



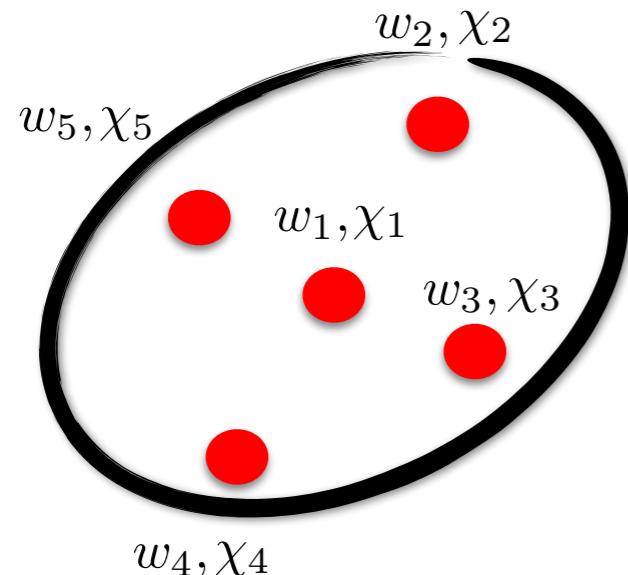
가우시안을 표현하는 최소한의 대표값 (시그마 포인트)을 계산한다

어떻게?

# 무향 칼만 필터 (예측 단계)

현재

$$x \sim \text{Gauss}(\mu, \Sigma)$$



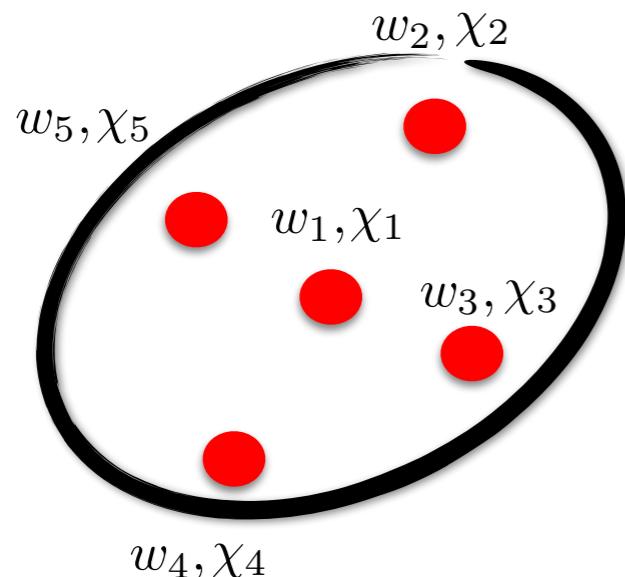
시그마 포인트의 가중치를 계산한다

어떻게?

# 무향 칼만 필터 (예측 단계)

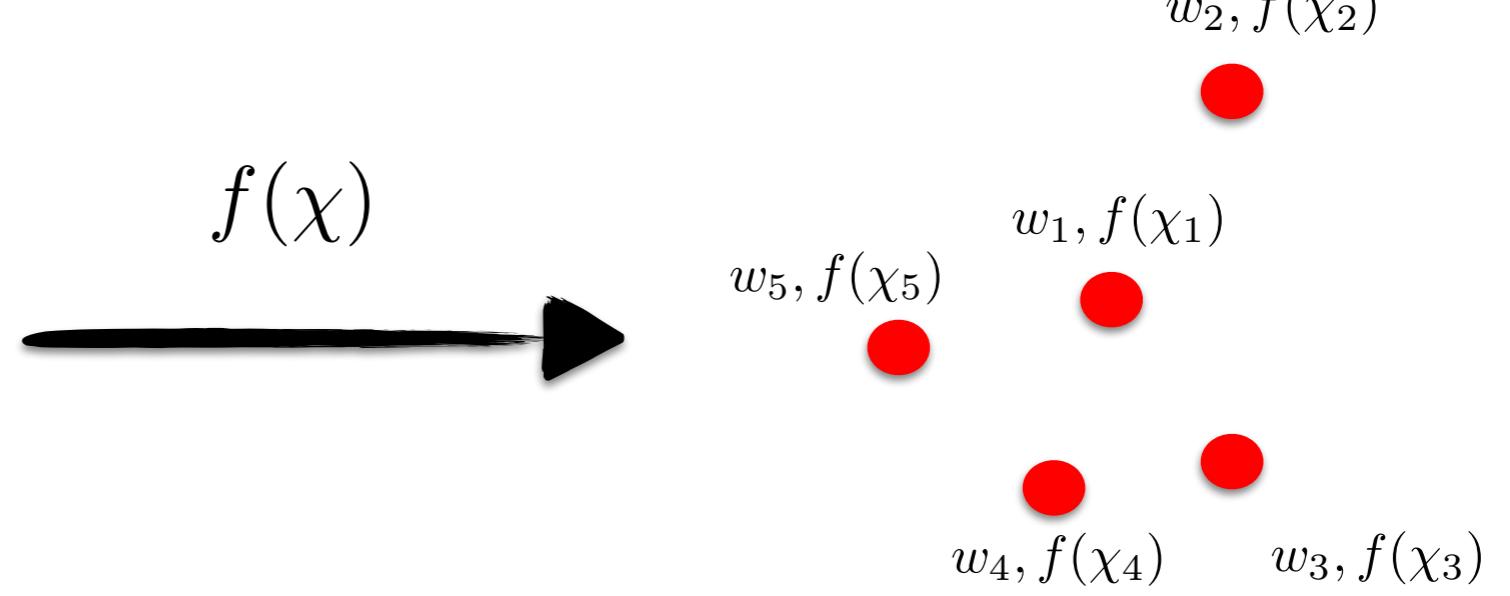
현재

$$x \sim \text{Gauss}(\mu, \Sigma)$$



다음

$$f(\chi)$$

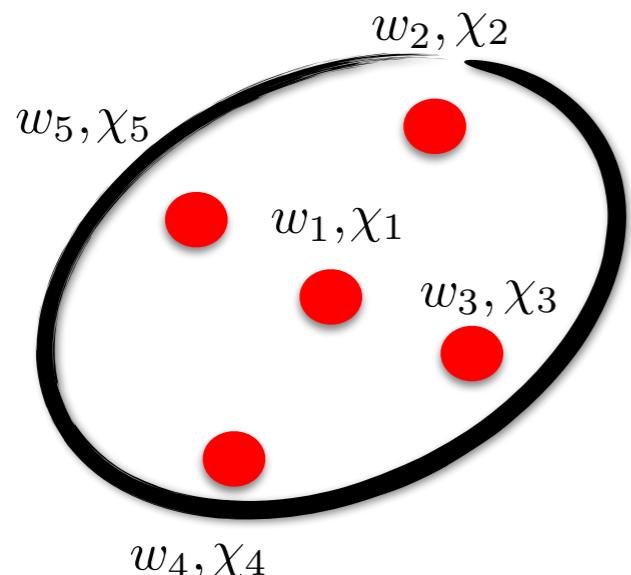


시그마 포인트를 비선형 변환한다

# 무향 칼만 필터 (예측 단계)

현재

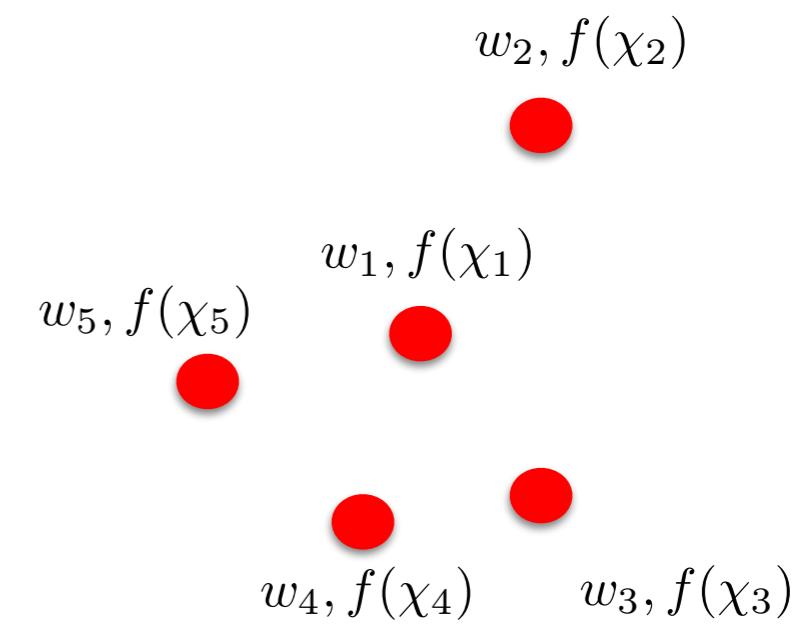
$$x \sim \text{Gauss}(\mu, \Sigma)$$



$$f(\chi)$$



다음



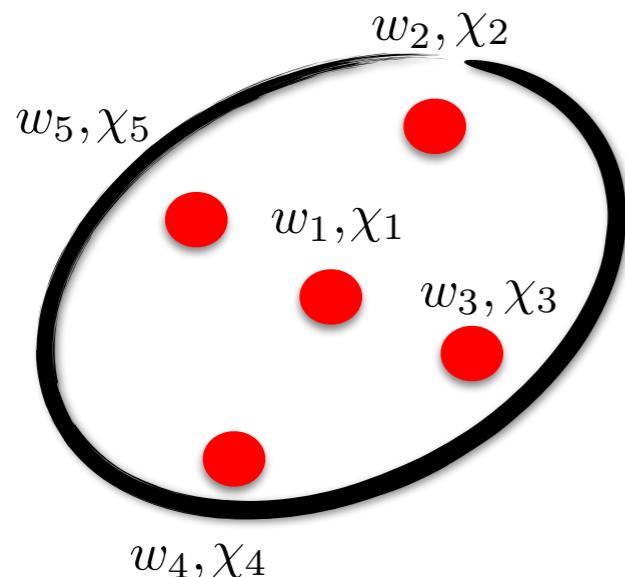
$$\mu' = \sum_{i=1}^5 w_i f(\chi_i)$$

$$\Sigma' = \sum_{i=1}^5 w_i \{f(\chi_i) - \mu'\} \{f(\chi_i) - \mu'\}^T$$

# 무향 칼만 필터 (예측 단계)

현재

$$x \sim \text{Gauss}(\mu, \Sigma)$$

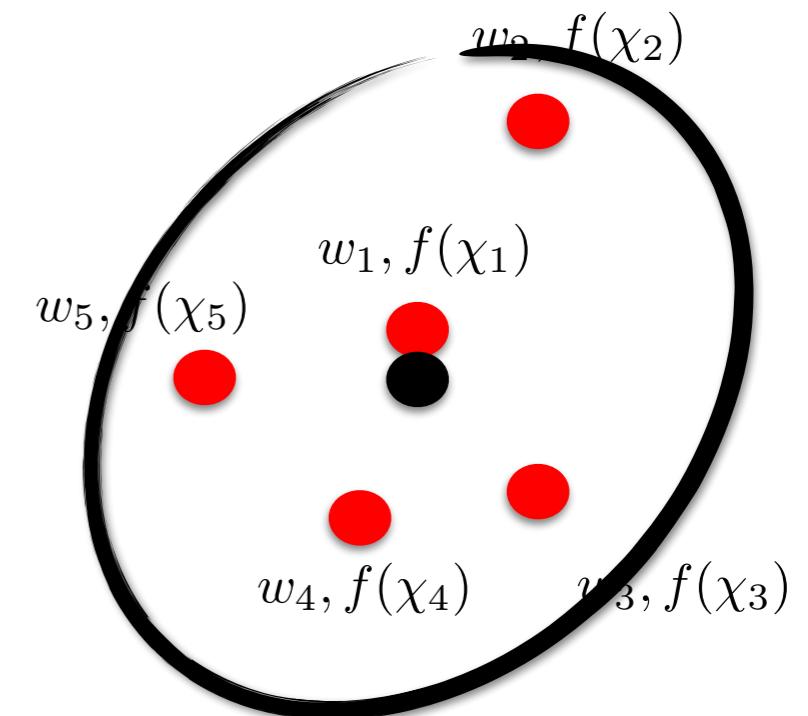


$$f(\chi)$$



다음

$$y \sim \text{Gauss}(\mu', \Sigma')$$



다음 상태를 가우시안 분포로 예측한다

# 시그마 포인트와 가중치

- 시그마 포인트는 어떻게 선택할까?
- 시그마 포인트의 가중치는 어떻게 선택할까?

# 시그마 포인트와 가중치

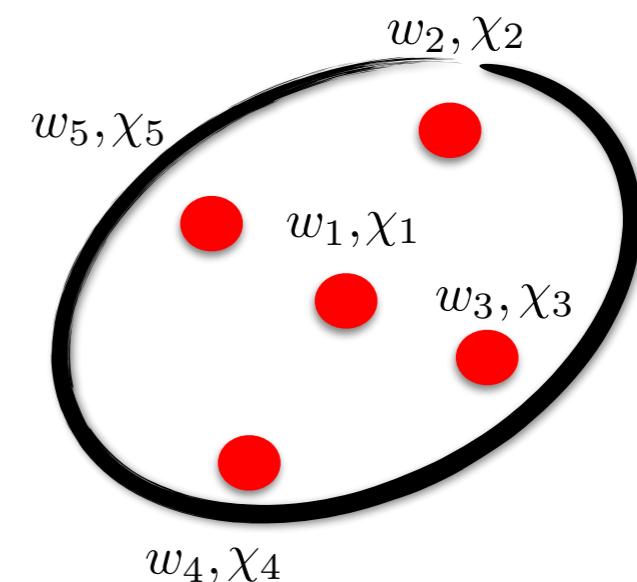
- 시그마 포인트는 어떻게 선택할까?
- 시그마 포인트의 가중치는 어떻게 선택할까?
- 세 조건을 만족하도록 시그마 포인트와 가중치를 선택한다!

$$\sum_i w_i = 1$$

$$\mu = \sum_i w_i \chi_i$$

$$\Sigma = \sum_i w_i \{\chi_i - \mu\} \{\chi_i - \mu\}^T$$

$$x \sim \text{Gauss}(\mu, \Sigma)$$



# 시그마 포인트와 가중치

- 시그마 포인트는 어떻게 선택할까?
- 시그마 포인트의 가중치는 어떻게 선택할까?
- 세 조건을 만족하도록 시그마 포인트와 가중치를 선택한다!

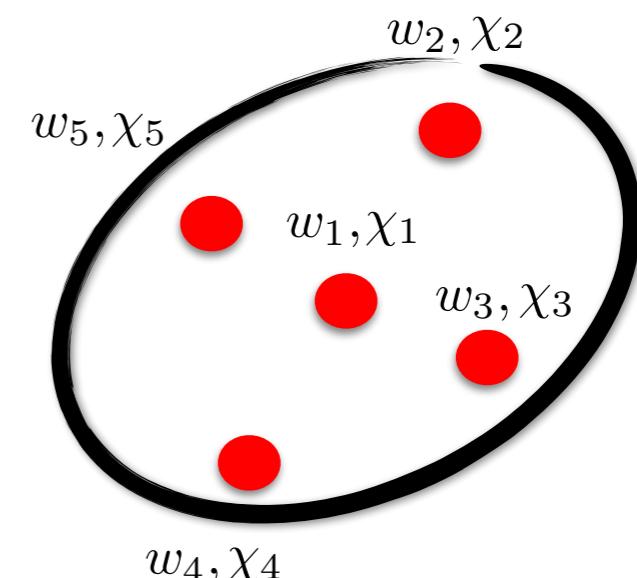
시그마 포인트와 가중치로부터 원래 가우시안 분포를 추정 가능하도록!

$$\sum_i w_i = 1$$

$$\mu = \sum_i w_i \chi_i$$

$$\Sigma = \sum_i w_i \{\chi_i - \mu\} \{\chi_i - \mu\}^T$$

$$x \sim \text{Gauss}(\mu, \Sigma)$$



# 시그마 포인트와 가중치

- 시그마 포인트는 어떻게 선택할까?
- 시그마 포인트의 가중치는 어떻게 선택할까?
- 세 조건을 만족하도록 시그마 포인트와 가중치를 선택한다!

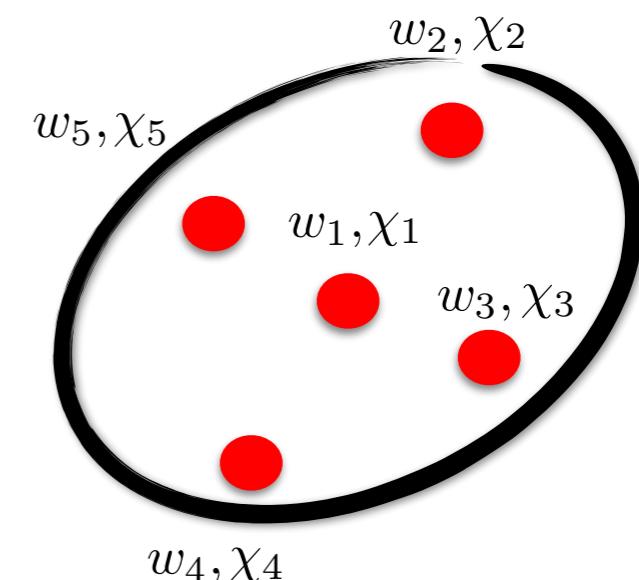
시그마 포인트와 가중치로부터 원래 가우시안 분포를 추정 가능하도록!

$$\sum_i w_i = 1$$

$$\mu = \sum_i w_i \chi_i$$

$$\Sigma = \sum_i w_i \{\chi_i - \mu\} \{\chi_i - \mu\}^T$$

$$x \sim \text{Gauss}(\mu, \Sigma)$$



- 여러 해가 존재 가능하다!

하이퍼 파라미터에 따라 결과를 살펴보자!

## 시그마 포인트와 가중치

- 세 조건을 만족하기 위한 시그마 포인트와 가중치

$$\chi_1 = \mu$$

$$\chi_{i+1} = \mu + \sqrt{(n+\kappa)\Sigma_i}$$

$$\chi_{i+n+1} = \mu - \sqrt{(n + \kappa)\Sigma_i} \quad (i = 1, 2, \dots, n)$$

차원  스케일링 파라미터  열 벡터 

```

def sigma_points(mu, Sigma, kappa):
    n = len(mu)
    Xi = np.zeros((n, 2*n+1))
    W = np.zeros(2*n+1)

    Xi[:, 0] = mu
    W[0] = kappa / (n + kappa)

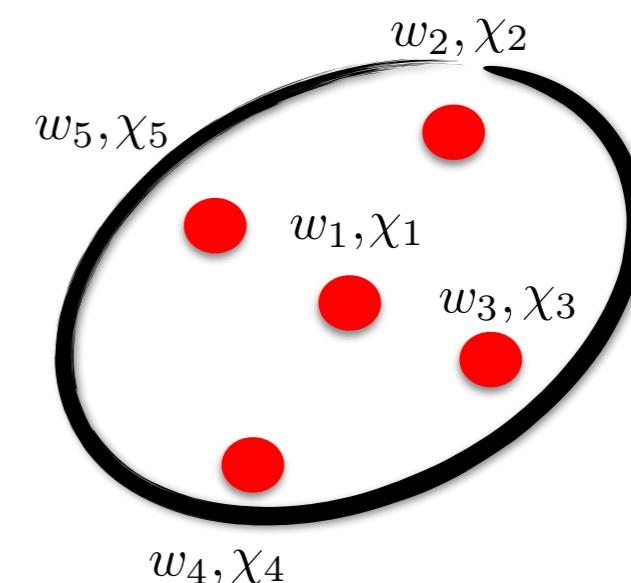
    U = cholesky((n + kappa)*Sigma)

    for i in range(n):
        Xi[:, i+1] = mu + U[:, i]
        Xi[:, n+i+1] = mu - U[:, i]
        W[i+1] = 1 / (2*(n+kappa))
        W[n+i+1] = W[i+1]

    return Xi, W

```

$$x \sim Gauss(\mu, \Sigma)$$



# 시그마 포인트와 가중치

- 세 조건을 만족하기 위한 시그마 포인트와 가중치

$$w_1 = \frac{\kappa}{n + \kappa}$$

$$w_{i+1} = w_{i+n+1} = \frac{1}{2(n + \kappa)} \quad (i = 1, 2, \dots, n)$$

```
def sigma_points(mu, Sigma, kappa):
    n = len(mu)
    Xi = np.zeros((n, 2*n+1))
    W = np.zeros(2*n+1)

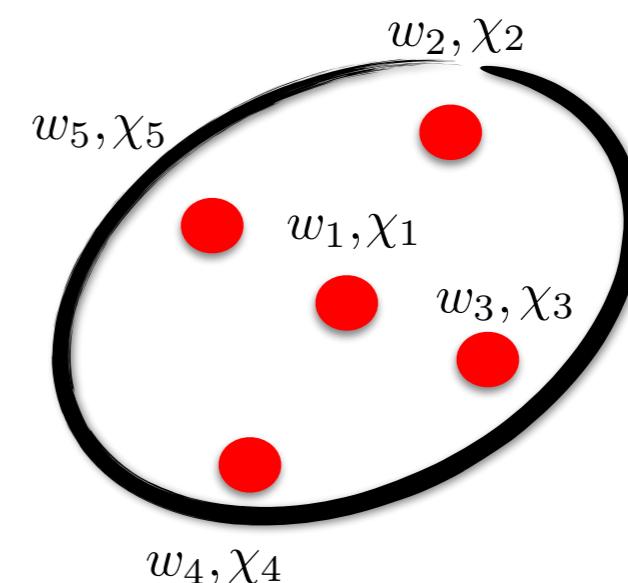
    Xi[:, 0] = mu
    W[0] = kappa / (n + kappa)

    U = cholesky((n + kappa)*Sigma)

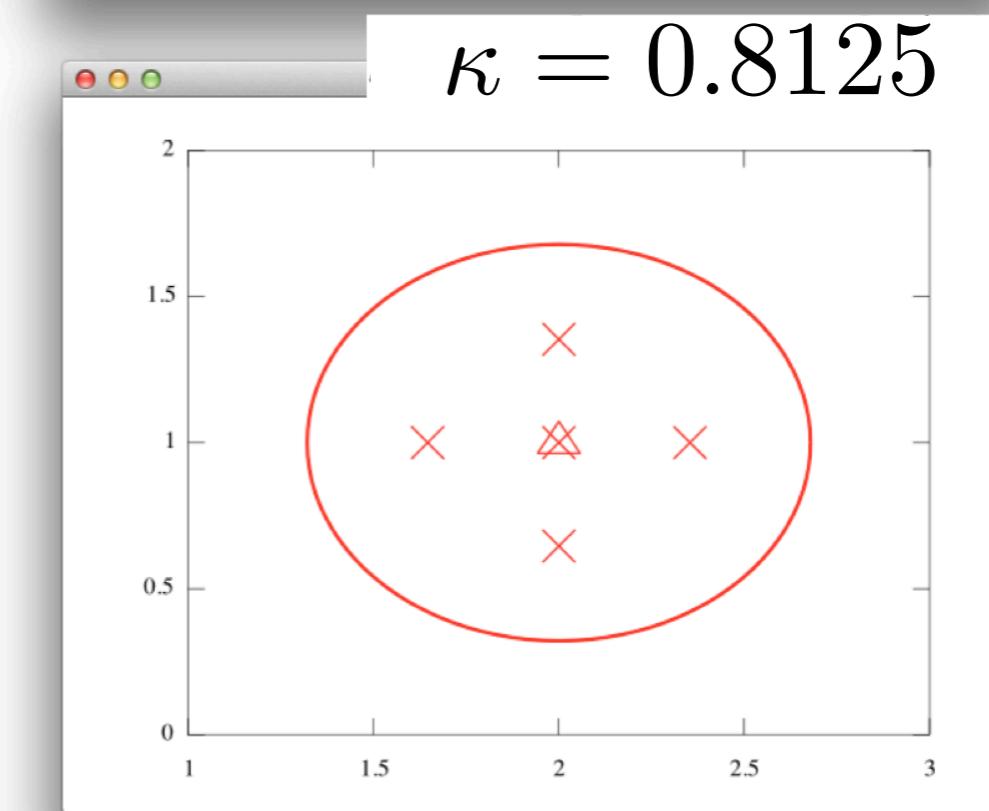
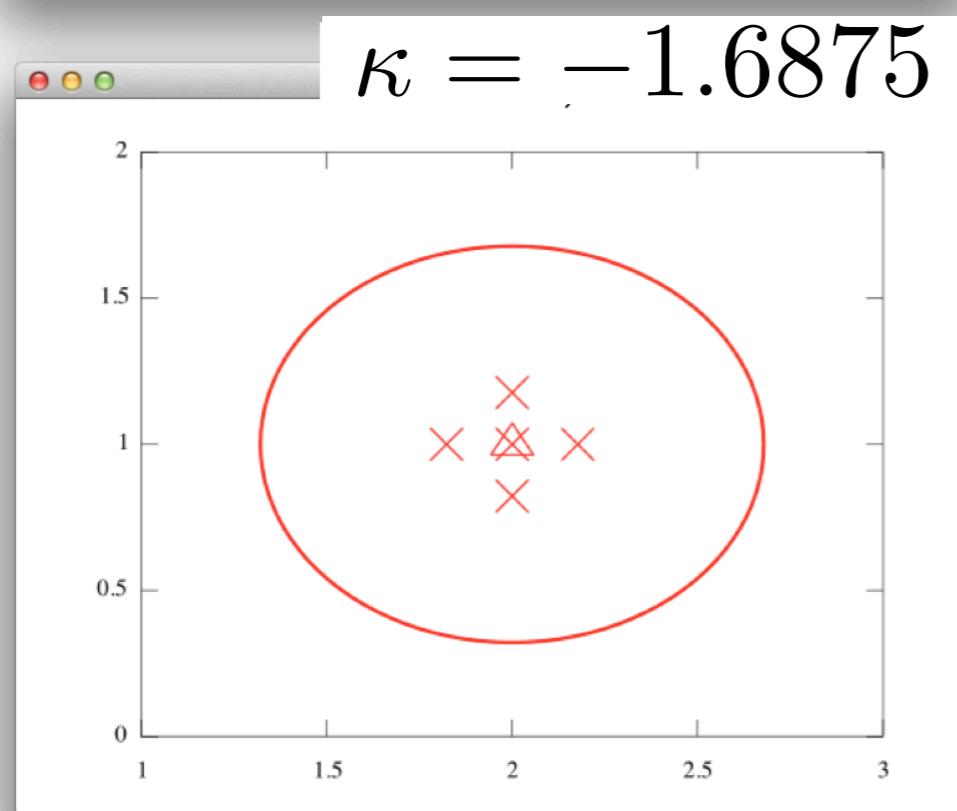
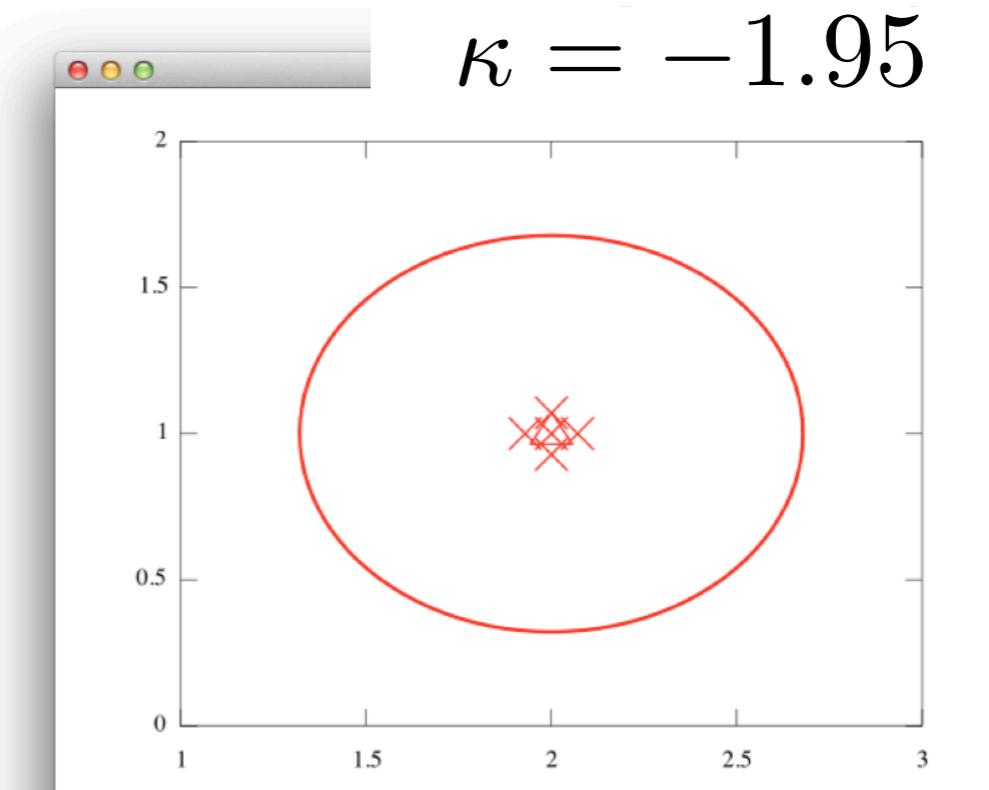
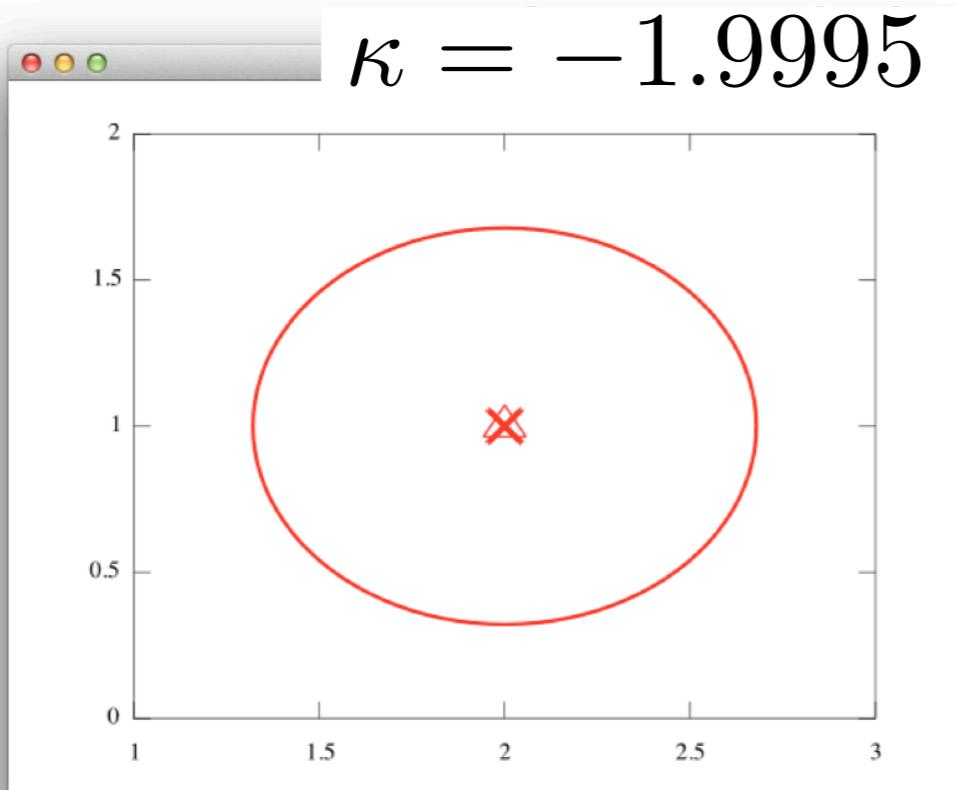
    for i in range(n):
        Xi[:, i+1] = mu + U[:, i]
        Xi[:, n+i+1] = mu - U[:, i]
        W[i+1] = 1 / (2*(n+kappa))
        W[n+i+1] = W[i+1]

    return Xi, W
```

$$x \sim \text{Gauss}(\mu, \Sigma)$$



# 스케일링 파라미터



# 실습: 시그마 포인트와 가중치로부터 원래 가우시안 추정

```
def sigma_points(mu, Sigma, kappa):
    n = len(mu)
    Xi = np.zeros((n, 2*n+1))
    W = np.zeros(2*n+1)

    Xi[:, 0] = mu
    W[0] = kappa / (n + kappa)

    U = cholesky((n + kappa)*Sigma)

    for i in range(n):
        Xi[:, i+1] = mu + U[:, i]
        Xi[:, n+i+1] = mu - U[:, i]
        W[i+1] = 1 / (2*(n+kappa))
        W[n+i+1] = W[i+1]

    return Xi, W
```

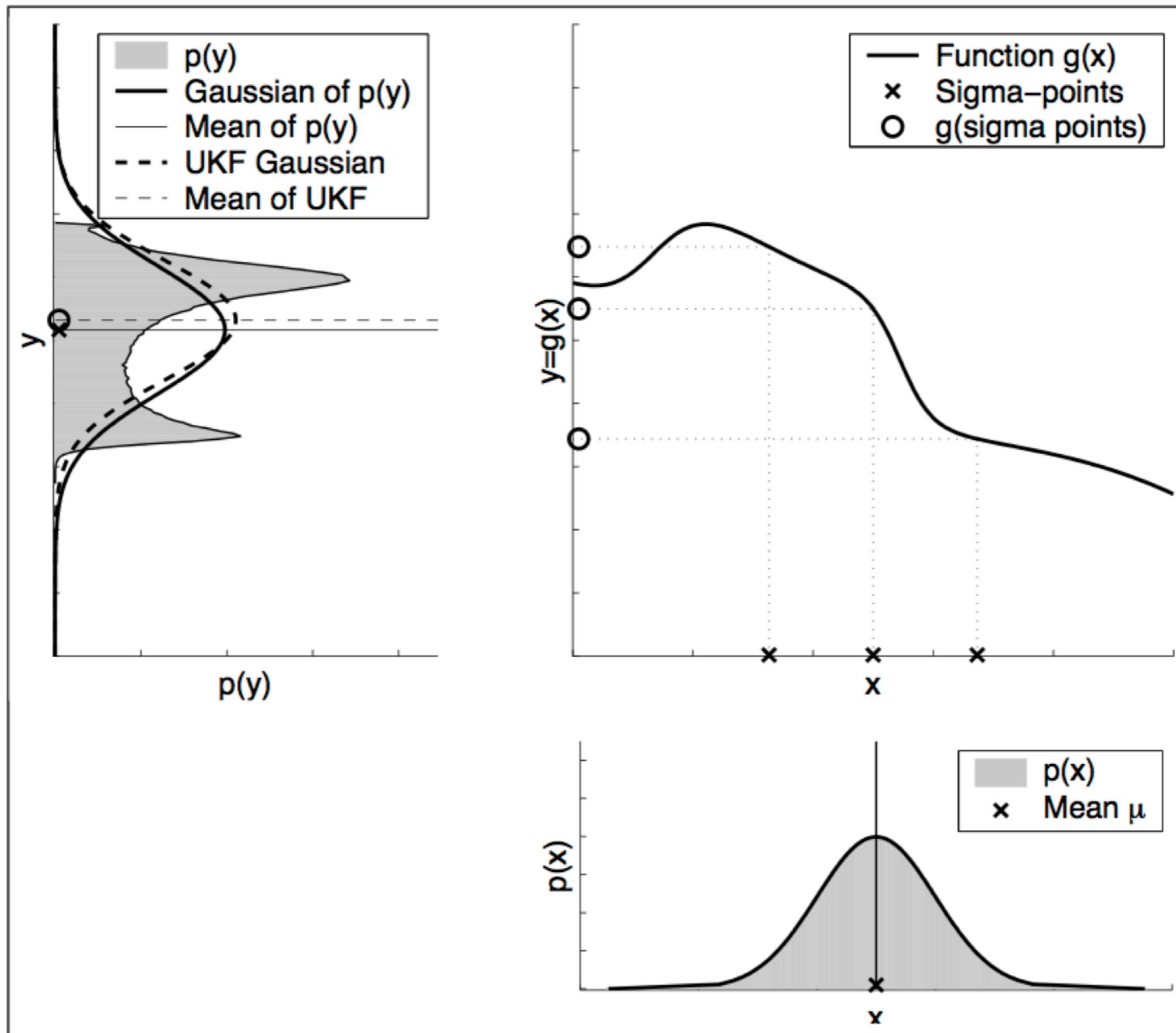
```
def UT(Xi, W, noiseCov=0):
    mean = np.sum(W * Xi, axis=1)
    cov = W * (Xi - mean.reshape(-1, 1)) @ (Xi - mean.reshape(-1, 1)).T
    return mean, cov + noiseCov
```

```
mu = np.array([5, 3])
Sigma = np.array([[7, 0], [0, 18]])
kappa = np.array([2])
Xi, W = sigma_points(mu, Sigma, kappa)
Xi, W
```

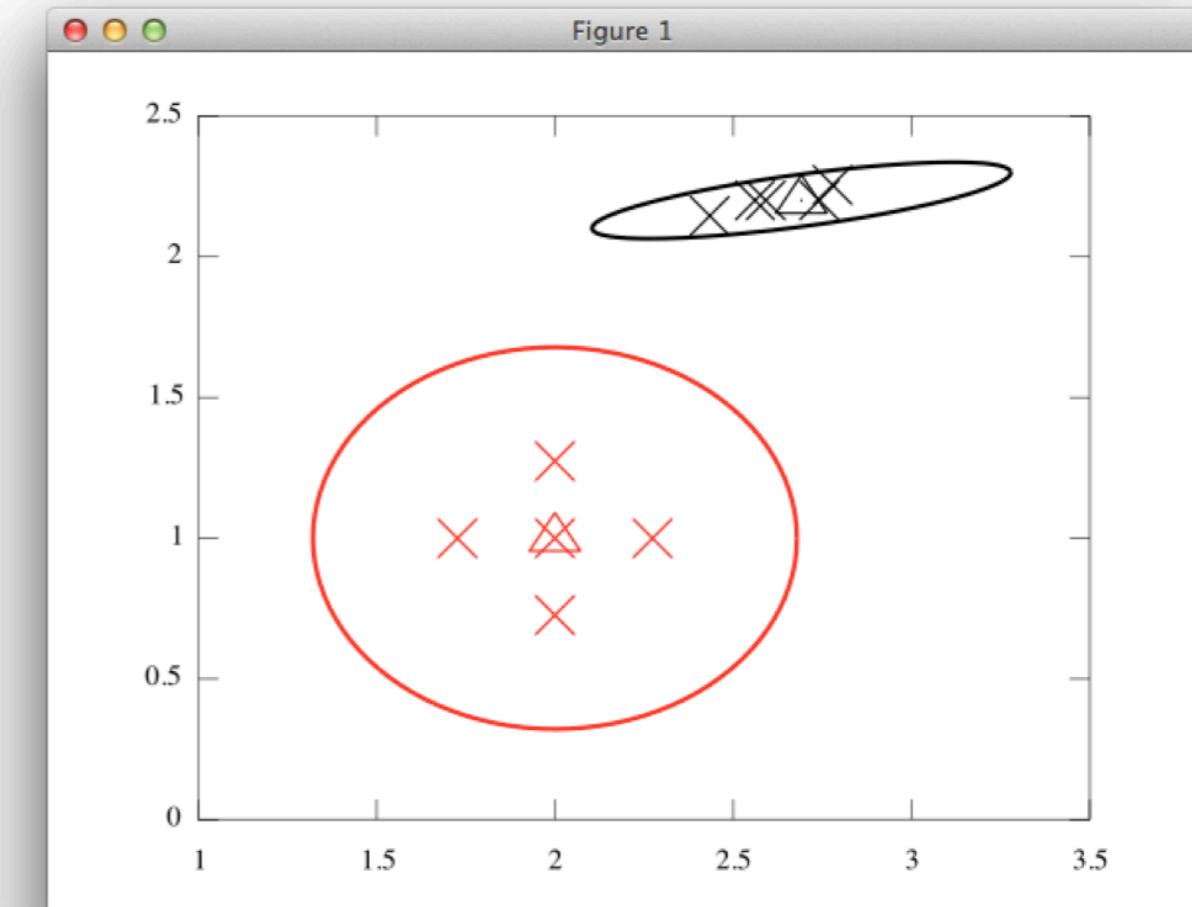
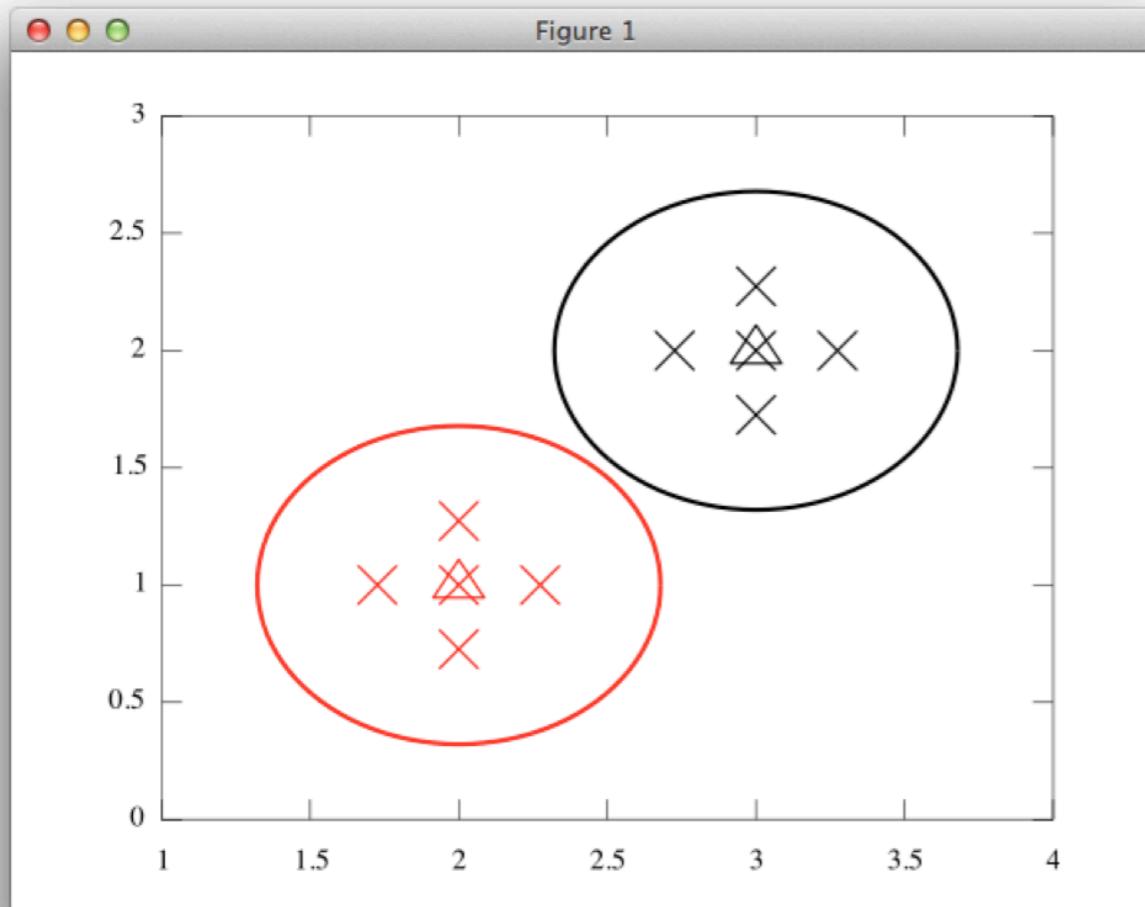
```
mean, cov = UT(Xi, W)
print("mean:")
print(mean, "\n")
print("cov :")
print(cov)
```

[https://github.com/tbmoon/kalman\\_filter/blob/master/asset/10.week/UT.ipynb](https://github.com/tbmoon/kalman_filter/blob/master/asset/10.week/UT.ipynb)

# 무향 변환



# 무향 변환



$$g((x, y)^T) = \begin{pmatrix} x + 1 \\ y + 1 \end{pmatrix}^T$$

$$g((x, y)^T) = \begin{pmatrix} 1 + x + \sin(2x) + \cos(y) \\ 2 + 0.2y \end{pmatrix}^T$$

# 무향 칼만 필터

<http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/pdf/slam06-ukf.pdf>

# 확장 칼만 필터 -> 무향 칼만 필터

0. 초기값 선정

$$\hat{x}_0, P_0$$

1. 추정값, 오차 공분산 예측

$$\hat{x}_k^- = f(\hat{x}_{k-1})$$

$$P_k^- = AP_{k-1}A^T + Q$$

2. 칼만 이득 계산

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

3. 추정값 계산

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-))$$

4. 오차 공분산 계산

$$P_k = P_k^- - K_k H P_k^-$$

0. 초기값 선정

$$\hat{x}_0, P_0$$

1. 추정값, 측정값, 오차 공분산 예측

$$(\chi_i, w_i) \leftarrow (\hat{x}_{k-1}, P_{k-1}, \kappa)$$

$$(\hat{x}_k^-, P_k^-) \leftarrow UT(f(\chi_i), w_i, Q)$$

$$(\hat{z}_k, P_z) \leftarrow UT(h(\chi_i), w_i, R)$$

2. 칼만 이득 계산

$$P_{xz} = \sum_{i=1}^{2n+1} w_i \{f(\chi_i) - \hat{x}_k^-\} \{h(\chi_i) - \hat{z}_k\}^T$$

$$K_k = P_{xz} P_z^{-1}$$

3. 추정값 계산

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - \hat{z}_k)$$

4. 오차 공분산 계산

$$P_k = P_k^- - K_k P_z K_k^T$$

## EKF Algorithm

```
1:  Extended_Kalman_filter( $\mu_{t-1}, \Sigma_{t-1}, z_t$ ):  
2:       $\bar{\mu}_t = g(\mu_{t-1})$   
3:       $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$   
4:       $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$   
5:       $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$   
6:       $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$   
7:      return  $\mu_t, \Sigma_t$ 
```

## EKF to UKF – Prediction

Unscented

1: ~~Extended\_Kalman\_filter~~( $\mu_{t-1}, \Sigma_{t-1}, z_t$ ):

2:  $\bar{\mu}_t$  = replace this by sigma point  
3:  $\bar{\Sigma}_t$  = propagation of the motion

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

5:  $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

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

7: return  $\mu_t, \Sigma_t$

## UKF Algorithm – Prediction

- 1: Unscented\_Kalman\_filter( $\mu_{t-1}, \Sigma_{t-1}, \cdot, z_t$ ):
- 2:  $\mathcal{X}_{t-1} = (\mu_{t-1} - \mu_{t-1} + \sqrt{(n + \lambda)\Sigma_{t-1}}) \quad \mu_{t-1} - \sqrt{(n + \lambda)\Sigma_{t-1}})$
- 3:  $\bar{\mathcal{X}}_t^* = g(\mathcal{X}_{t-1})$
- 4:  $\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$
- 5:  $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^T + R_t$

## EKF to UKF – Correction

Unscented

1: ~~Extended\_Kalman\_filter~~( $\mu_{t-1}, \Sigma_{t-1}, z_t$ ):

2:  $\bar{\mu}_t =$  replace this by sigma point  
3:  $\bar{\Sigma}_t =$  propagation of the motion

use sigma point propagation for the expected observation and Kalman gain

5:  $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$

6:  $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$

7: return  $\mu_t, \Sigma_t$

28

## UKF Algorithm – Correction (1)

- 6:  $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n + \lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n + \lambda)\bar{\Sigma}_t})$
- 7:  $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$
- 8:  $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$
- 9:  $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$
- 10:  $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$
- 11:  $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$

## UKF Algorithm – Correction (1)

$$6: \quad \bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n+\lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n+\lambda)\bar{\Sigma}_t})$$

$$7: \quad \bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$$

$$8: \quad \hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$$

$$9: \quad S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t) (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$$

$$10: \quad \bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t) (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$$

$$11: \quad K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$$

$$K_t = \underbrace{\bar{\Sigma}_t}_{\bar{\Sigma}_t^{x,z}} \underbrace{H_t^T}_{S_t} (\underbrace{H_t \bar{\Sigma}_t H_t^T}_{S_t} + Q_t)^{-1}$$

(from EKF)

30

## UKF Algorithm – Correction (2)

```

6:    $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n + \lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n + \lambda)\bar{\Sigma}_t})$ 
7:    $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$ 
8:    $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$ 
9:    $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$ 
10:   $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t)(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$ 
11:   $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$ 
12:   $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$ 
13:   $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$ 
14:  return  $\mu_t, \Sigma_t$ 

```

## UKF Algorithm – Correction (2)

6:  $\bar{\mathcal{X}}_t = (\bar{\mu}_t \quad \bar{\mu}_t + \sqrt{(n + \lambda)\bar{\Sigma}_t} \quad \bar{\mu}_t - \sqrt{(n + \lambda)\bar{\Sigma}_t})$

7:  $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$

8:  $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$

9:  $S_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t) (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T + Q_t$

10:  $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t) (\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t)^T$

11:  $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$

12:  $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$

13:  $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^T$

14: return  $\mu_t, \Sigma_t$

$$\begin{aligned}\Sigma_t &= (I - K_t H_t) \bar{\Sigma}_t \\ &= \bar{\Sigma}_t - K_t H_t \bar{\Sigma}_t \\ &= \bar{\Sigma}_t - K_t (\Sigma^{x,z})^T \\ &= \bar{\Sigma}_t - K_t (\Sigma^{x,z} S_t^{-1} S_t)^T \\ &= \bar{\Sigma}_t - K_t (K_t S_t)^T \\ &= \bar{\Sigma}_t - K_t S_t^T K_t^T \\ &= \bar{\Sigma}_t - K_t S_t K_t^T\end{aligned}$$

(see next slide)



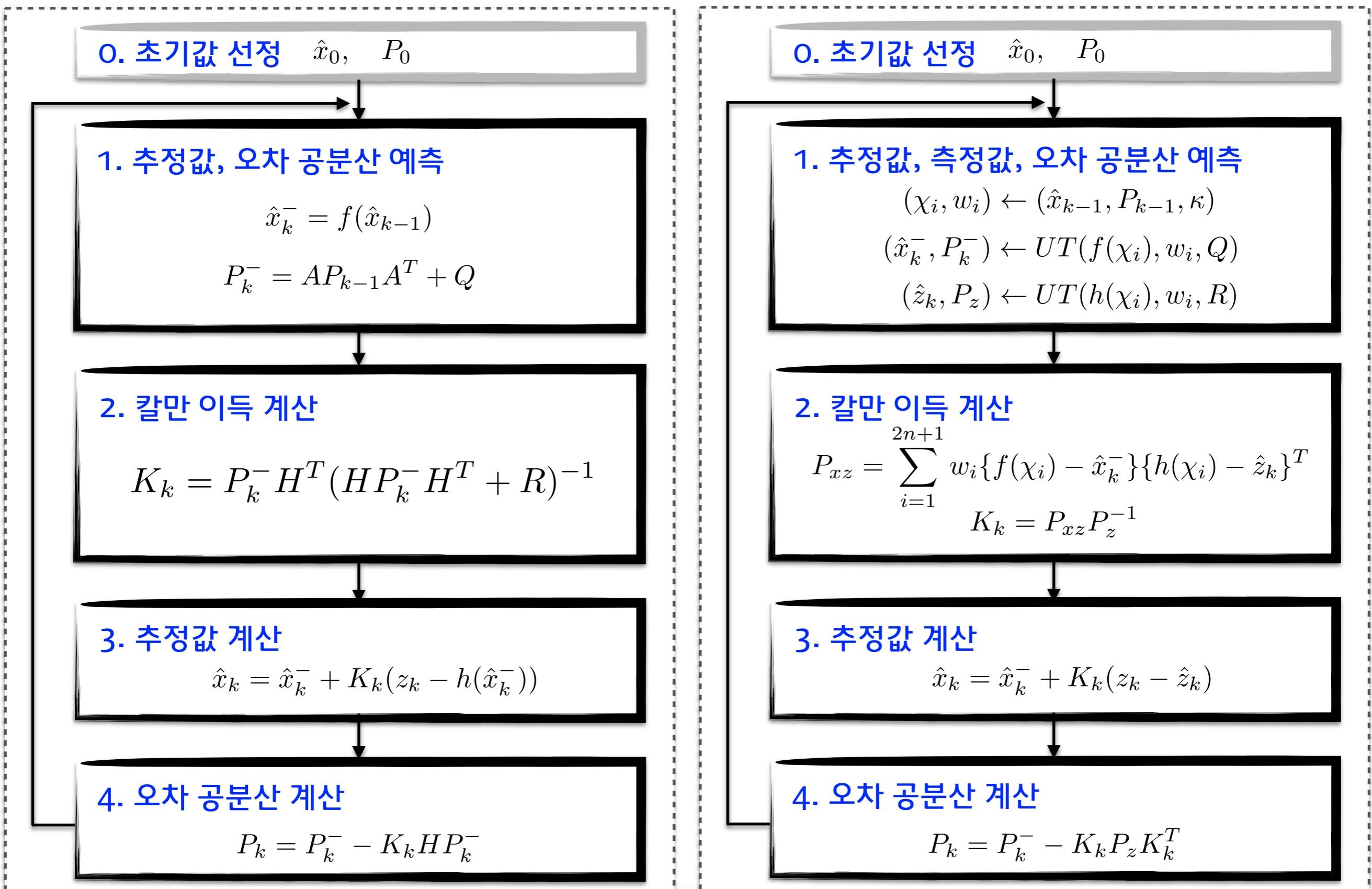
32

## From EKF to UKF – Computing the Covariance

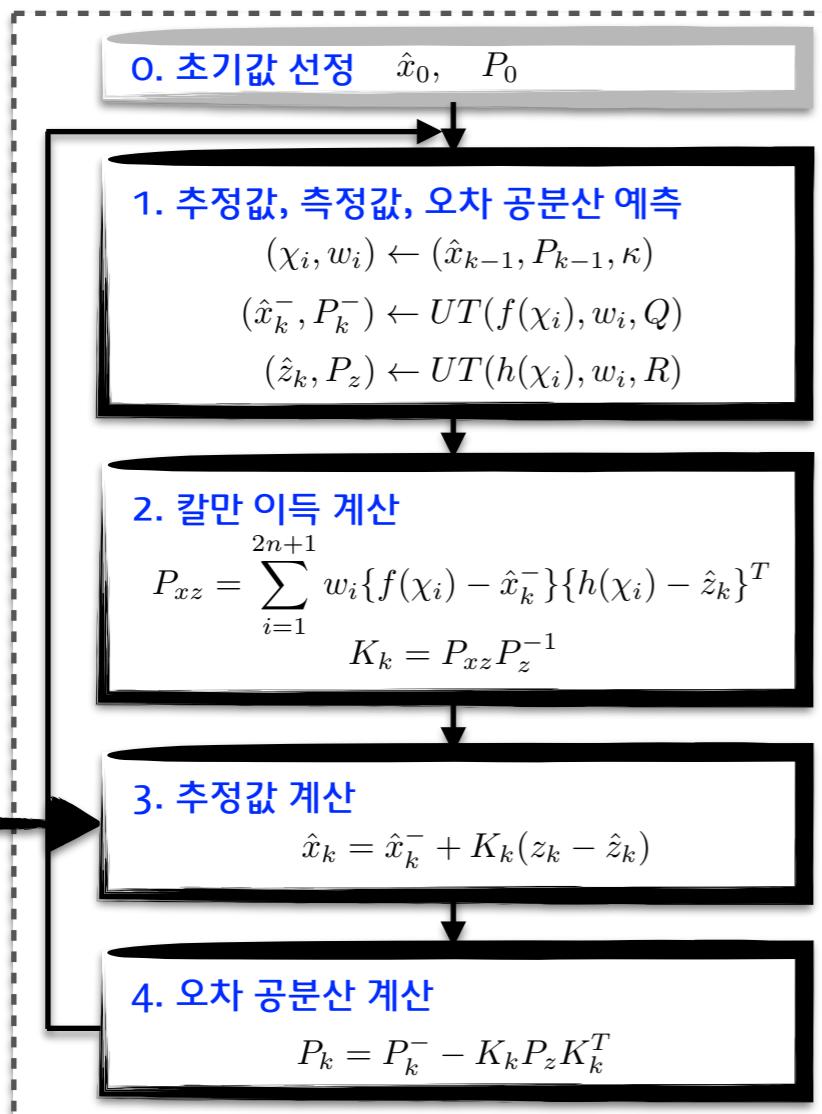
$$\begin{aligned}
 \Sigma_t &= (I - K_t H_t) \bar{\Sigma}_t \\
 &= \bar{\Sigma}_t - K_t \underline{H_t \bar{\Sigma}_t} \\
 &= \bar{\Sigma}_t - K_t (\bar{\Sigma}^{x,z})^T \\
 &= \bar{\Sigma}_t - K_t (\bar{\Sigma}^{x,z} S_t^{-1} S_t)^T \\
 &= \bar{\Sigma}_t - K_t (\underline{K_t S_t})^T \\
 &= \bar{\Sigma}_t - K_t S_t^T K_t^T \\
 &= \bar{\Sigma}_t - K_t S_t K_t^T
 \end{aligned}$$

33

# 확장 칼만 필터 -> 무향 칼만 필터

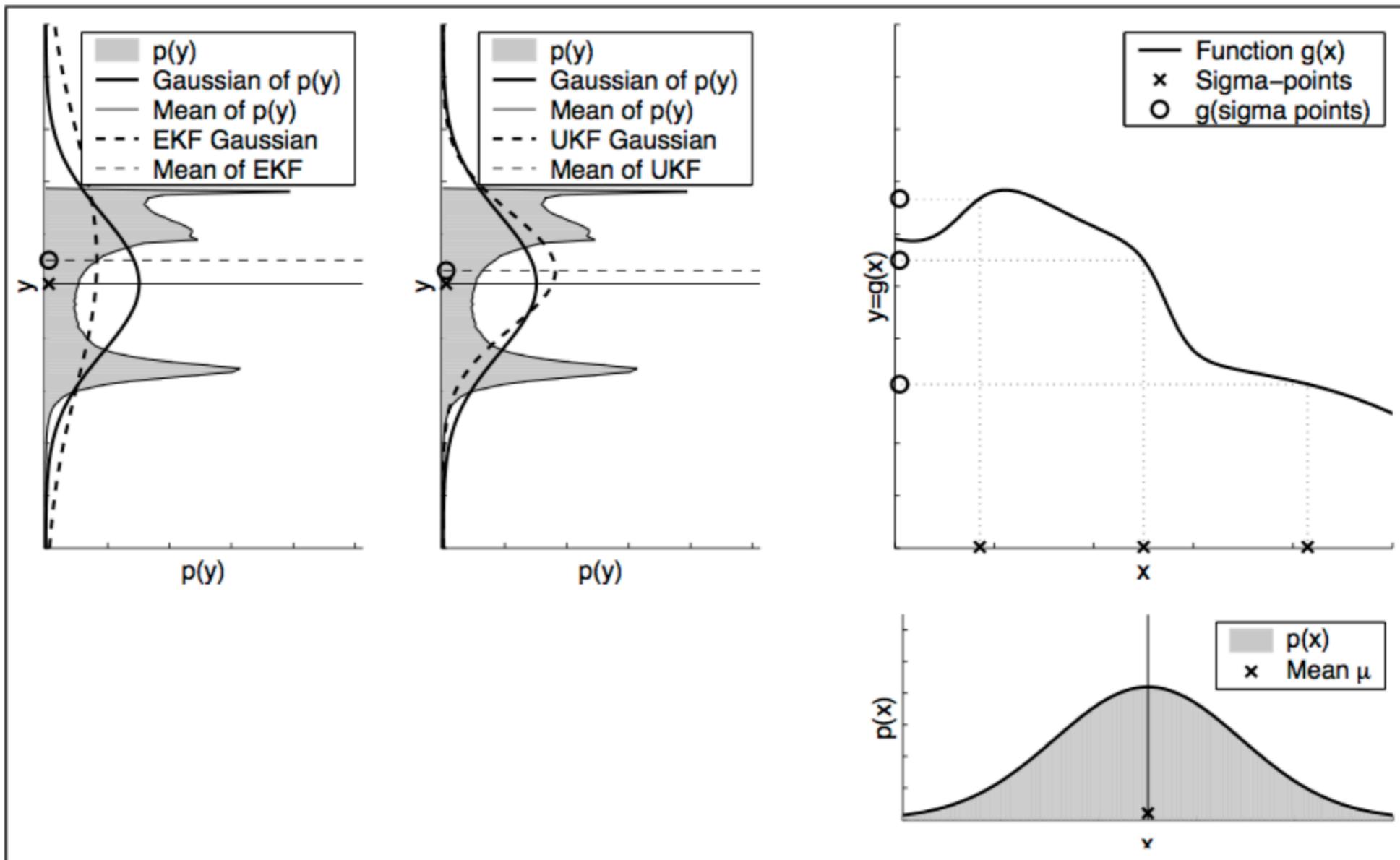


# 무향 칼만 필터



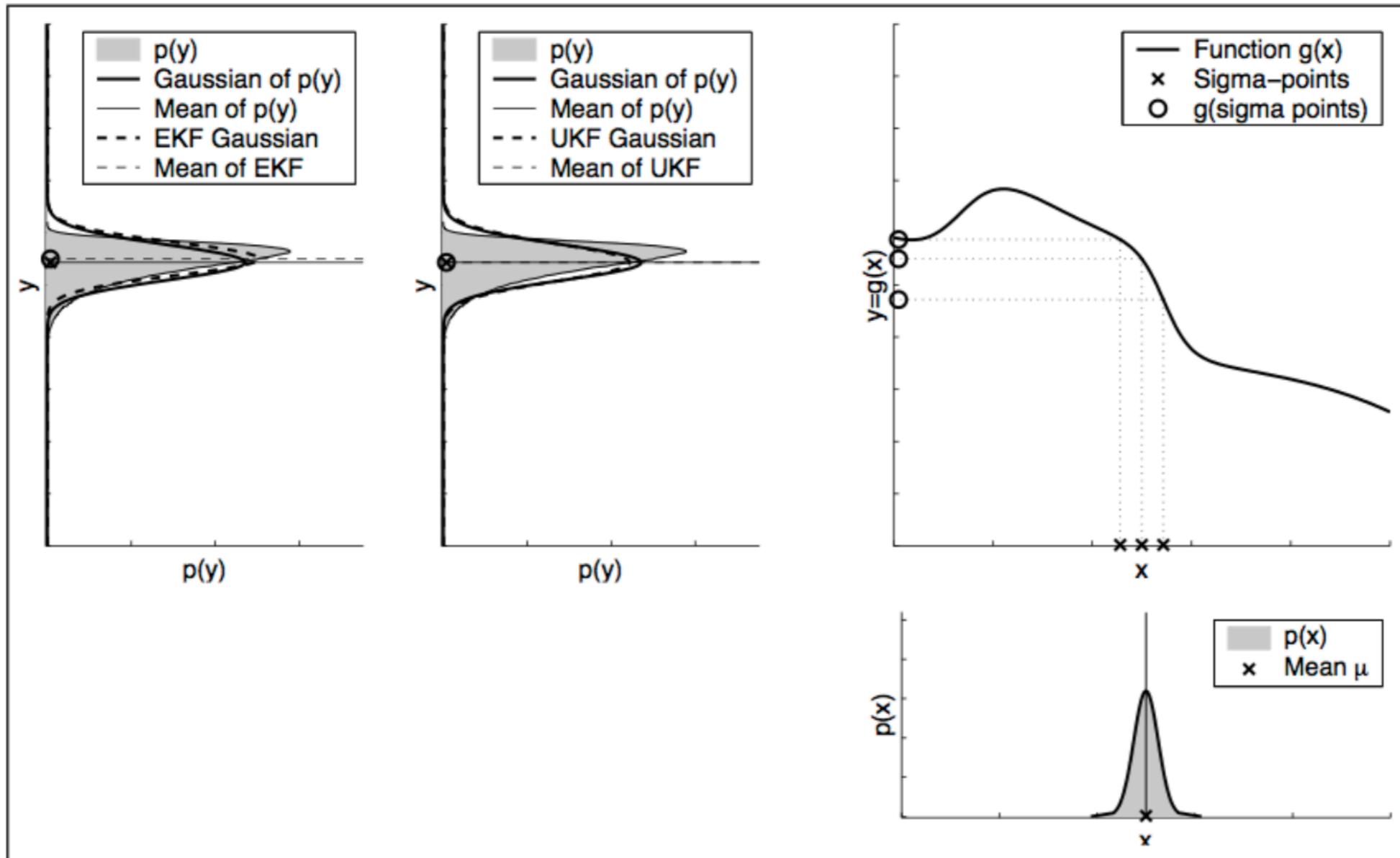
```
def unscented_kalman_filter(z_meas, x_esti, P):  
    """Unscented Kalman Filter Algorithm."""  
    # (1) Sample Sigma Points and Weights.  
    Xi, W = sigma_points(x_esti, P, kappa)  
  
    # (2) Predict Mean and Error Covariance of States.  
    fXi = fx(Xi)  
    x_pred, P_x = UT(fXi, W, Q)  
  
    # (3) Calculate Mean and Error Covariance for the Expected Observation.  
    hXi = hx(fXi)  
    z_pred, P_z = UT(hXi, W, R)  
  
    # (4) Calculate Off Diagonal Elements of Error Covariance and Kalman Gain.  
    Pxz = W * (fXi - x_pred.reshape(-1, 1)) @ (hXi - z_pred.reshape(-1, 1)).T  
    K = Pxz @ inv(P_z)  
  
    # (5) Estimate Mean and Error Covariance of States.  
    x_esti = x_pred + K @ (z_meas - z_pred)  
    P = P_x - K @ P_z @ K.T  
  
    return x_esti, P
```

## UKF vs. EKF



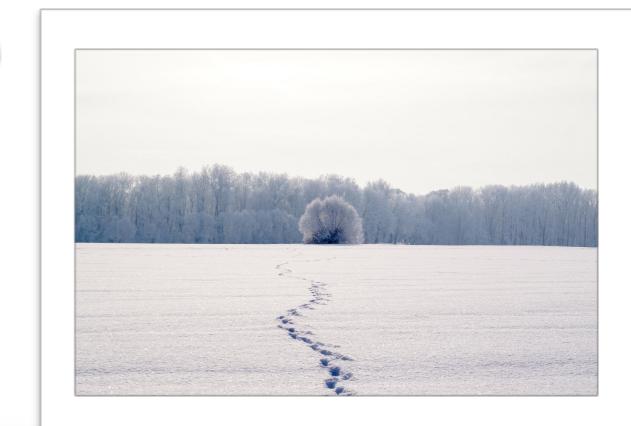
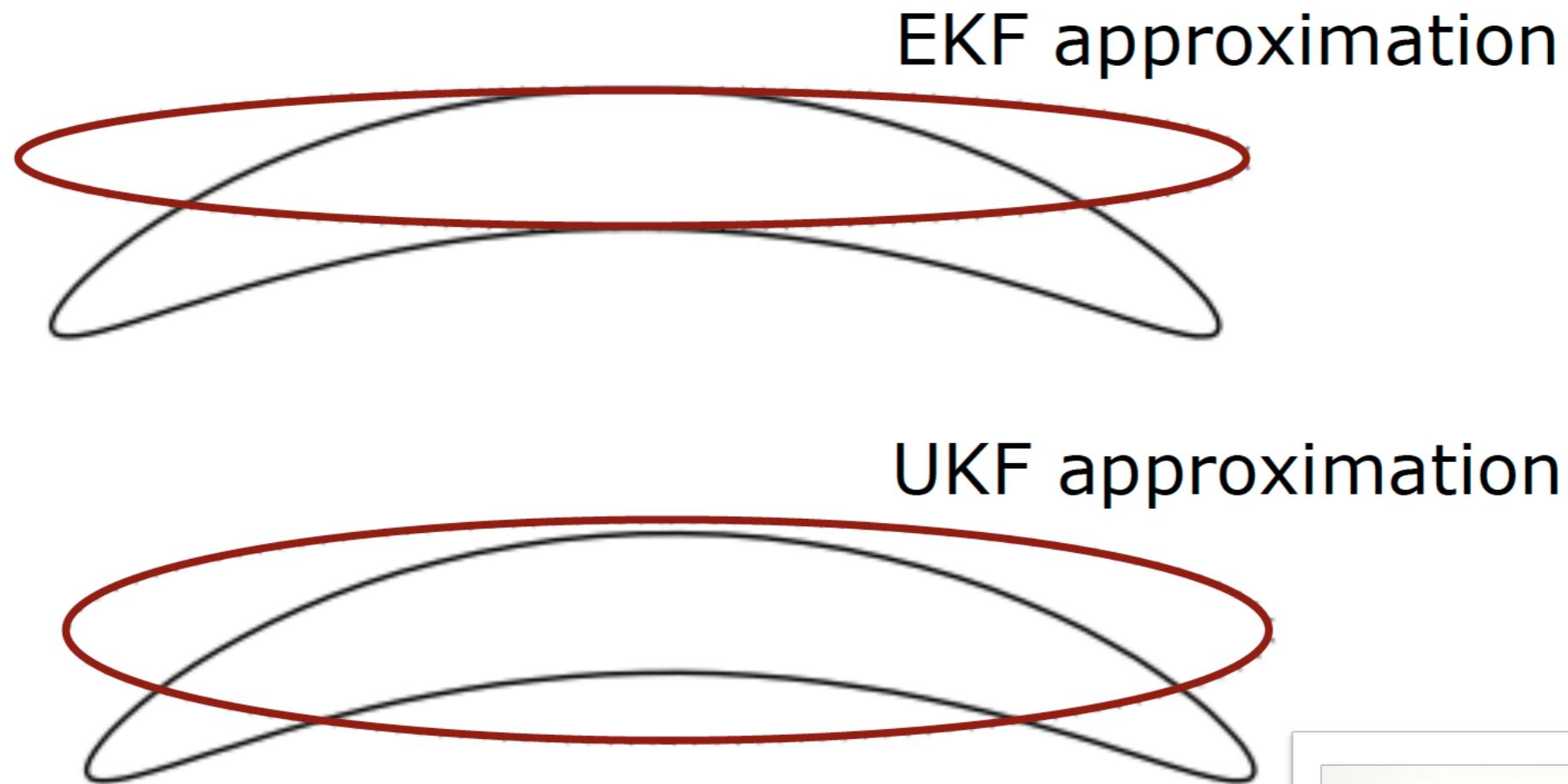
34

## UKF vs. EKF (Small Covariance)



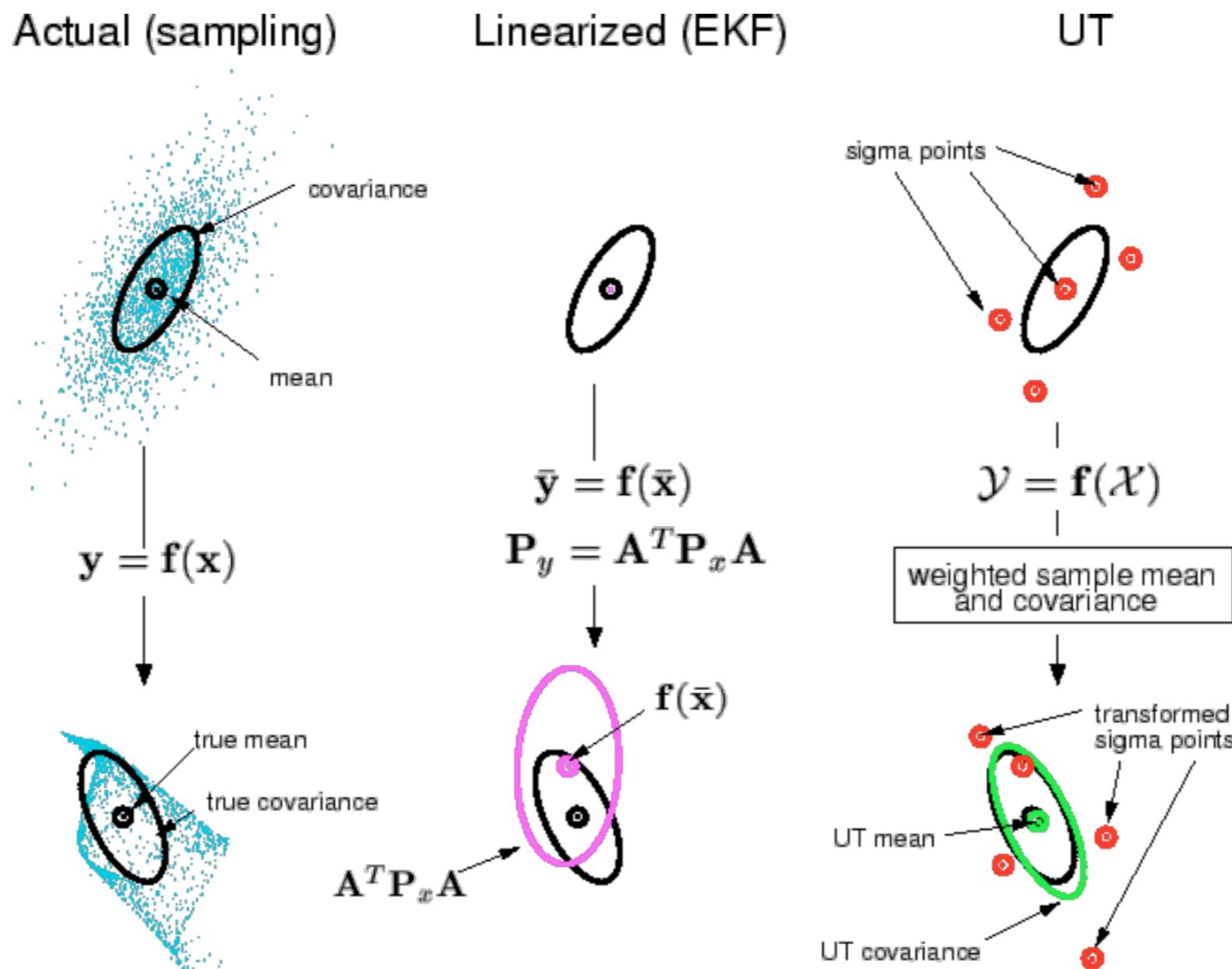
35

## UKF vs. EKF – Banana Shape



# 무향 칼만 필터 vs. 확장 칼만 필터

## UKF vs. EKF



Courtesy: E.A. Wan and R. van der Merwe

37

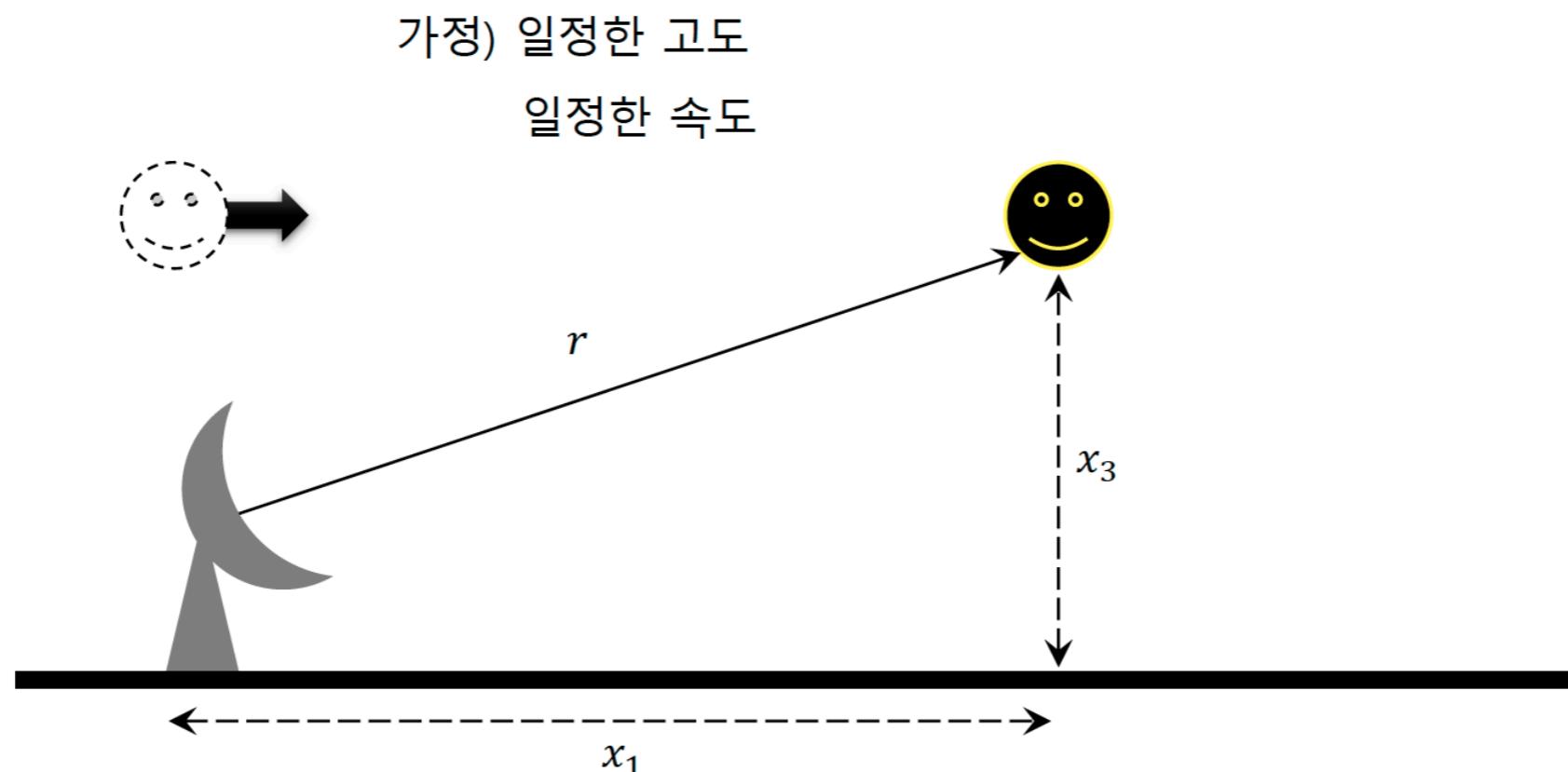
# 예제 실습

참고 자료: 광수님 확장 칼만 필터 발표

# 예제 1: 레이다 추적

Ch 12. 확장 칼만 필터

## 12.3 예제 1 : 레이다 추적



2

# 예제 1: 레이다 추적

## 12.3 예제 1 : 레이다 추적

### ● 상태변수

$$x = \begin{Bmatrix} \text{수평거리} \\ \text{이동속도} \\ \text{고도} \end{Bmatrix} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix}$$

### ● 측정 모델 (비선형)

$$\begin{aligned} r &= \sqrt{x_1^2 + x_3^2} + v \\ &\equiv h(x) + v \end{aligned}$$

측정 잡음

### ● 시스템 모델

$$\begin{Bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} + \begin{Bmatrix} 0 \\ w_1 \\ w_2 \end{Bmatrix}$$

시스템 잡음

$$\equiv Ax + w$$

$\dot{x}_1 = x_2$  : 이동속도는 수평 거리의 변화율(미분)

$\dot{x}_2 = w_1$  : 일정한 이동속도 ( $\dot{x}_2=0$ )

$\dot{x}_3 = w_2$  : 일정한 고도 ( $\dot{x}_3=0$ )

# 예제 1: 레이다 추적

## 12.3 예제 1 : 레이다 추적

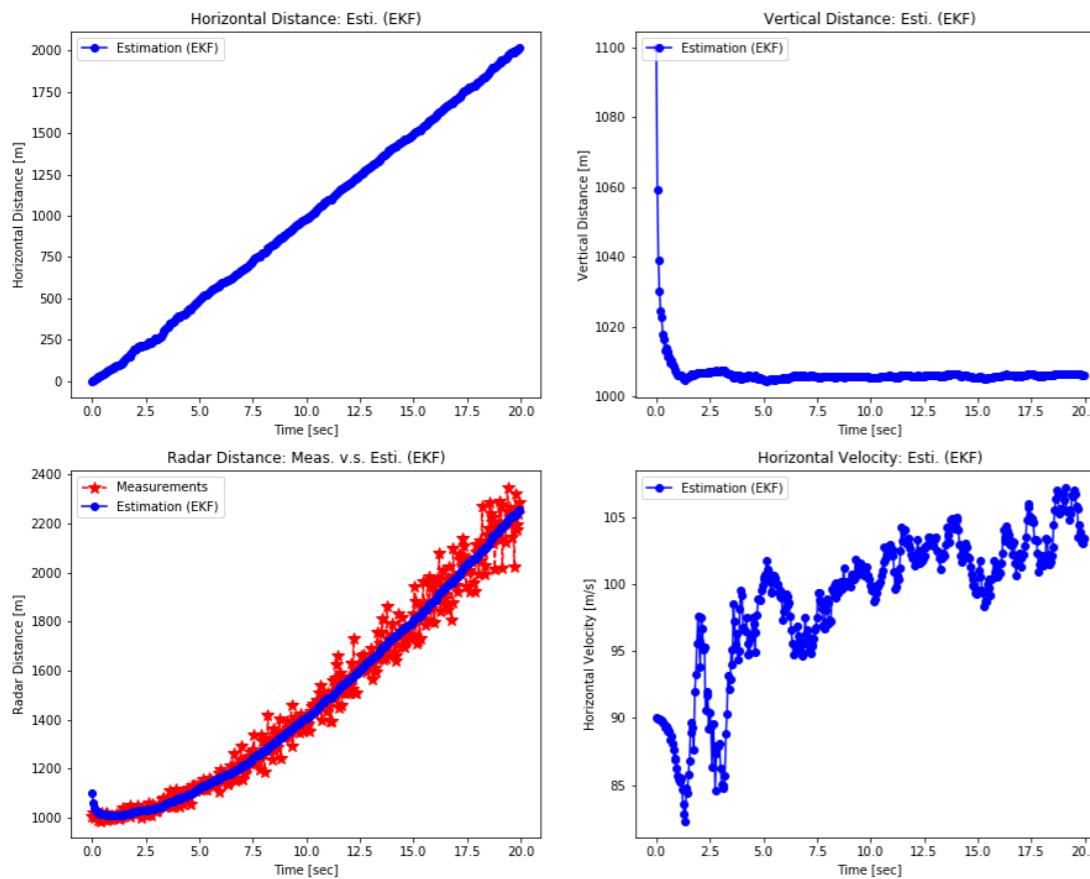
- EKF 구성

$$\begin{aligned}x_{k+1} &= x_k + Ax_k \cdot dt \\&= \underline{(I + A \cdot dt) \cdot x_k} \\A &= \underline{\quad}$$

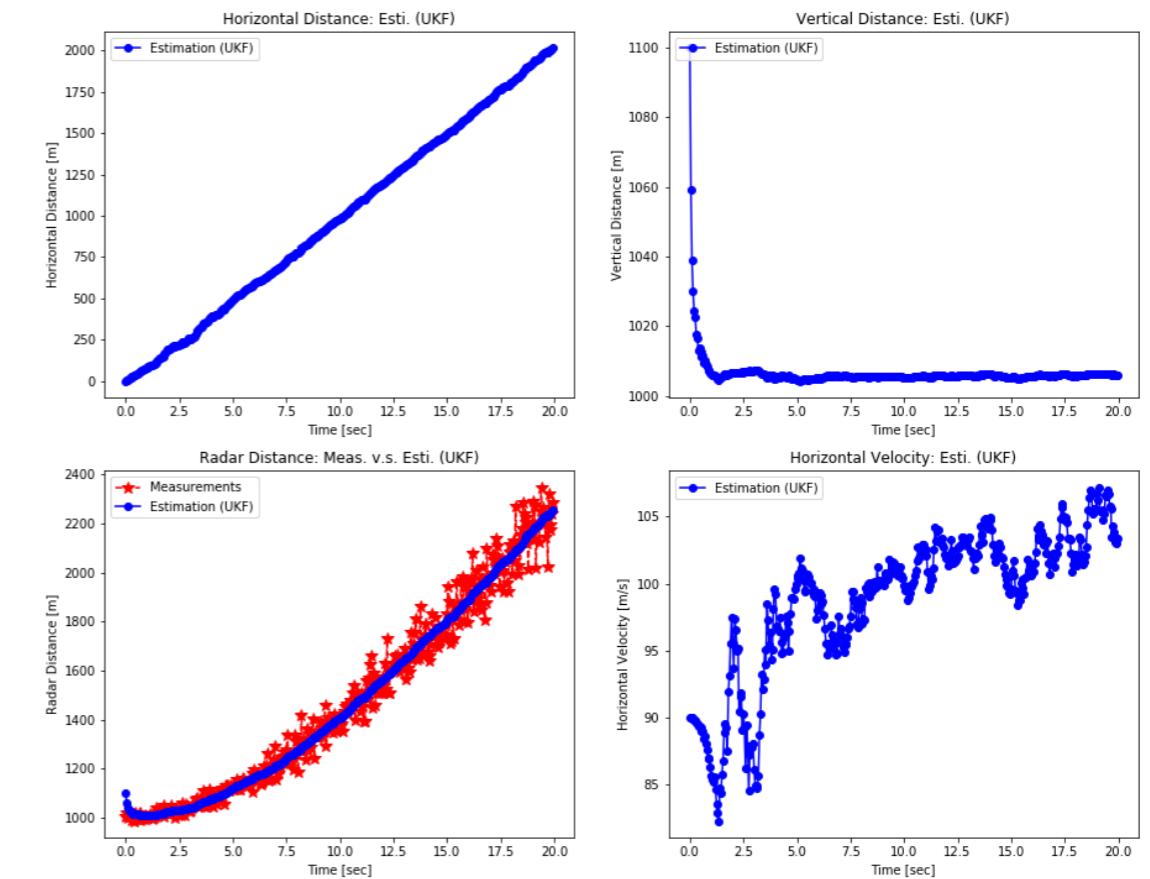
$$A = I + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot dt$$

# 예제 1: 레이다 추적

EKF



UKF



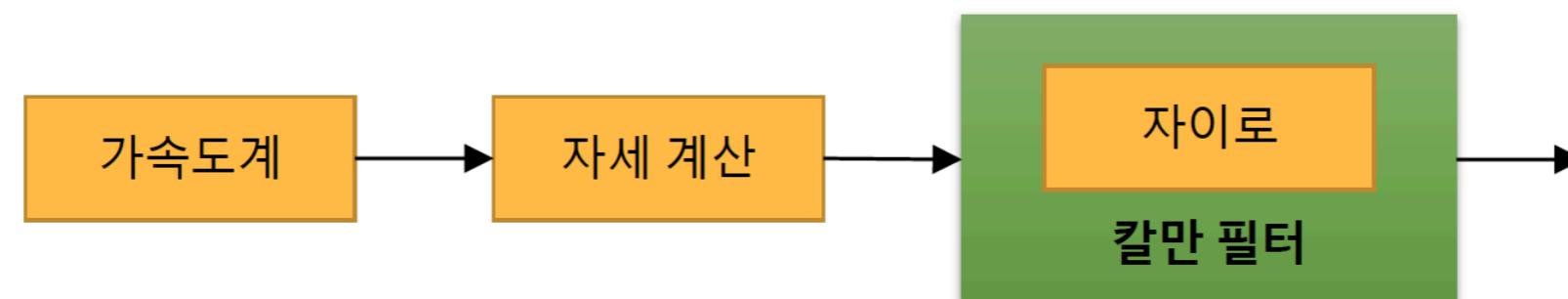
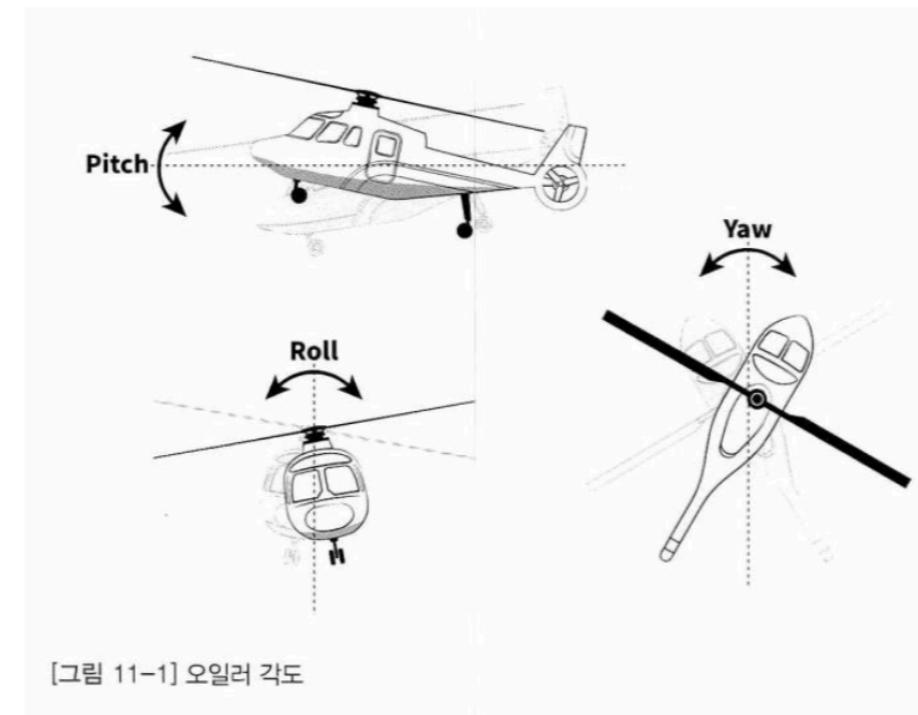
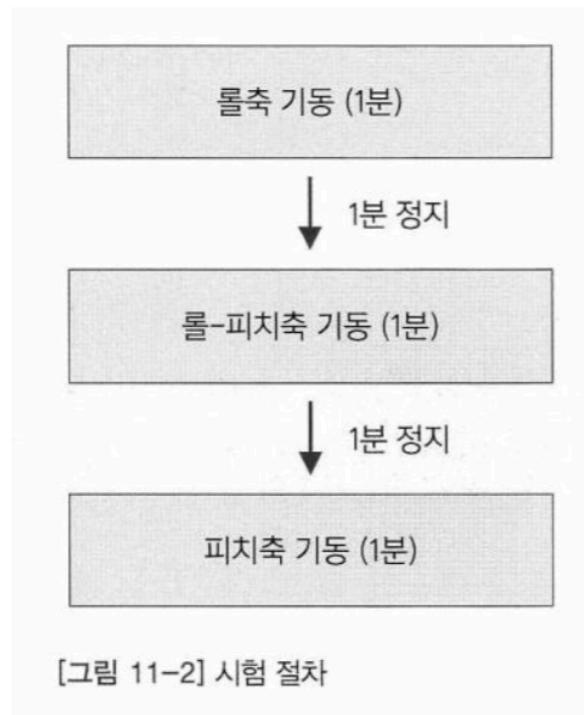
[https://github.com/tbmoon/kalman\\_filter/blob/master/Ch13.UnscentedKalmanFilter/radar\\_ukf.ipynb](https://github.com/tbmoon/kalman_filter/blob/master/Ch13.UnscentedKalmanFilter/radar_ukf.ipynb)

# 예제 2: 기울기 자세 측정하기

Ch 12. 확장 칼만 필터

## 12.4 예제 4 : 기울기 자세 측정하기

- Chapter 11) 자이로와 가속도계를 이용한 기울기 자세 측정 예제



7

# 예제 2: 기울기 자세 측정하기

Ch 12. 확장 칼만 필터

## 12.4 예제 4 : 기울기 자세 측정하기

### ● 상태변수

$$x = \begin{Bmatrix} roll \\ pitch \\ yaw \end{Bmatrix} = \begin{Bmatrix} \phi \\ \theta \\ \psi \end{Bmatrix}$$

### ● 측정 모델

$$\begin{aligned} z &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \cdot \begin{Bmatrix} \phi \\ \theta \\ \psi \end{Bmatrix} + v \\ &= Hx + v \end{aligned}$$

### ● 시스템 모델 (비선형)

$$\begin{aligned} \begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} &= \begin{bmatrix} 1 & \sin \phi \cdot \tan \theta & \cos \phi \cdot \tan \theta \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi / \cos \theta & \cos \phi / \cos \theta \end{bmatrix} \cdot \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} + w \\ &= \begin{bmatrix} p + q \cdot \sin \phi \cdot \tan \theta + r \cdot \cos \phi \cdot \tan \theta \\ q \cdot \cos \phi - r \cdot \sin \phi \\ q \cdot \sin \phi / \cos \theta + r \cdot \cos \phi / \cos \theta \end{bmatrix} + w \\ &\equiv f(x) + w \end{aligned}$$

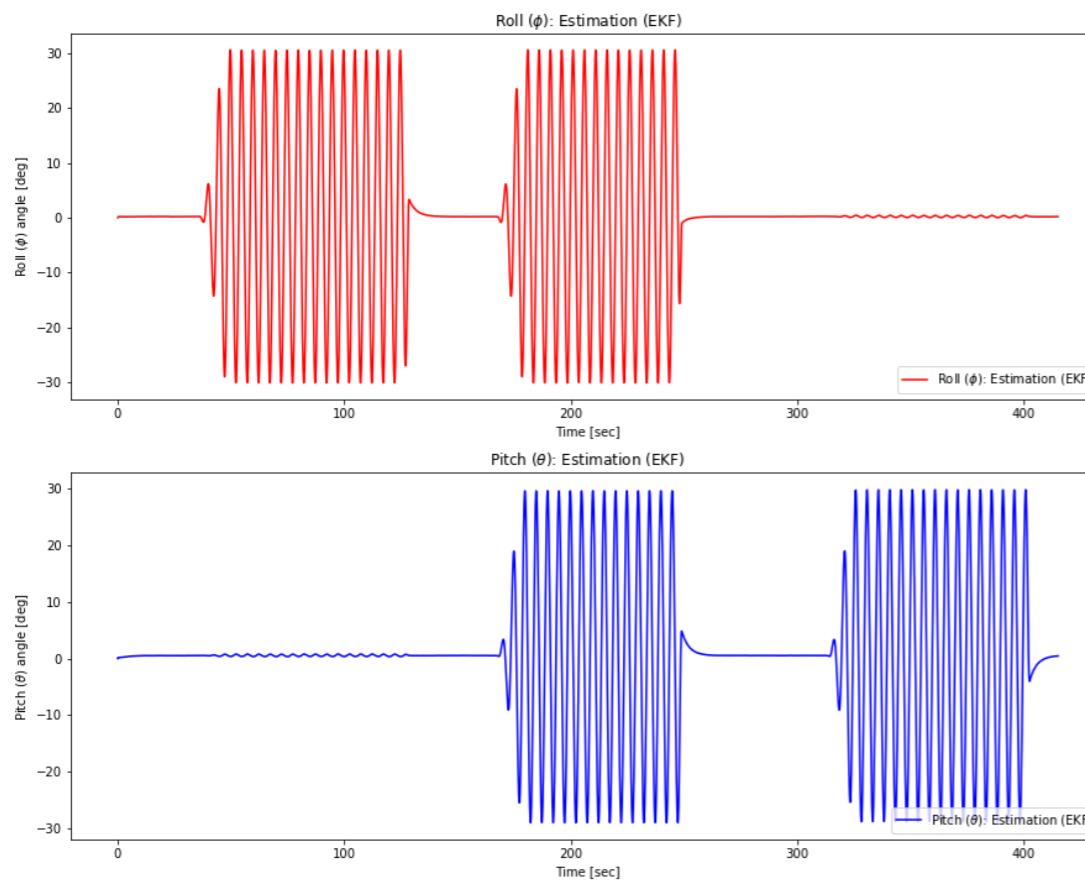
보정용 측정값)

가속도계로부터 얻은 룰, 피치각

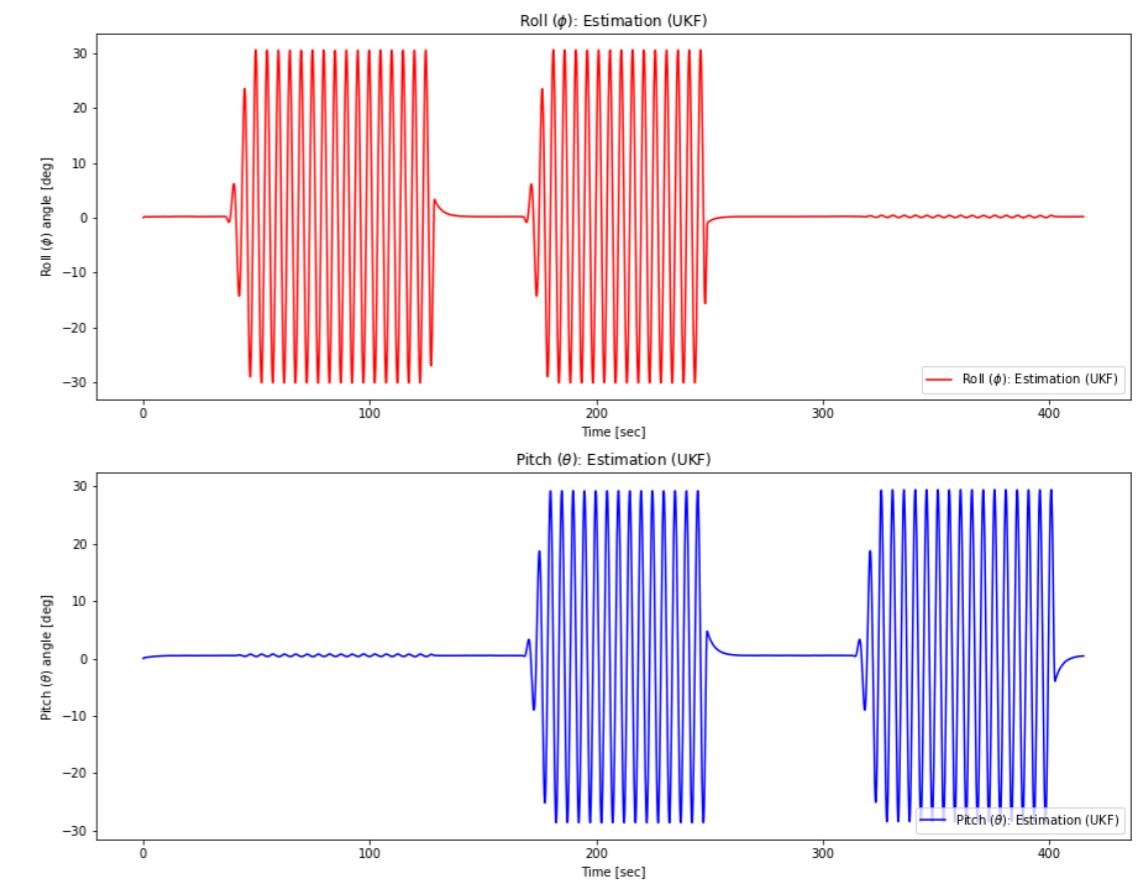
$$cf) \sec \theta = \frac{1}{\cos \theta}$$

# 예제 2: 기울기 자세 측정하기

EKF



UKF



[https://github.com/tbmoon/kalman\\_filter/blob/master/Ch13.UnscentedKalmanFilter/pose\\_orientation\\_fusion\\_ukf.ipynb](https://github.com/tbmoon/kalman_filter/blob/master/Ch13.UnscentedKalmanFilter/pose_orientation_fusion_ukf.ipynb)

# 무향 변환 정리

- 선형화 대신 무향 변환을 사용한다.
- 시그마 포인트를 선택한다.
- 하이퍼 파라미터 ( $\kappa$ )가 존재한다.
- 테일러 전개보다 더 좋은 근사를 제공한다.
- 무향 칼만 필터에는 예측과 업데이트 단계에서 무향 변환을 사용한다.

# 무향 칼만 필터 v.s. 확장 칼만 필터

- 선형 시스템에서 동일한 결과를 보인다.
- 비선형 시스템에서 더 좋은 결과를 보일 수 있다.
- 보통은 결과의 차이가 크지 않다.
- Jacobians 계산이 필요 없다.
- EKF 보다 느리다.
- 여전히 가우시안 확률 분포를 요구한다.

# 무향 칼만 필터 v.s. 확장 칼만 필터

- 선형 시스템에서 동일한 결과를 보인다.
- 비선형 시스템에서 더 좋은 결과를 보일 수 있다.
- 보통은 결과의 차이가 크지 않다.
- Jacobians 계산이 필요 없다.
- EKF 보다 느리다.
- 여전히 가우시안 확률 분포를 요구한다.

## 파티클 필터

# 참고 자료

- 칼만 필터는 어렵지 않아 (저자: 김성필 님)
- 파이썬으로 구현하는 칼만 필터
- Robot Mapping (Cyrill Stachniss)
- 확장 칼만 필터 발표 (박광수 님)

# 풀잎 스쿨 일정

10 주차 3월 11일 - 무향 칼만 필터

11 주차 3월 18일 - 파텍클 필터 (상윤님 / 주희님)

# 감사합니다