EKF, UKF

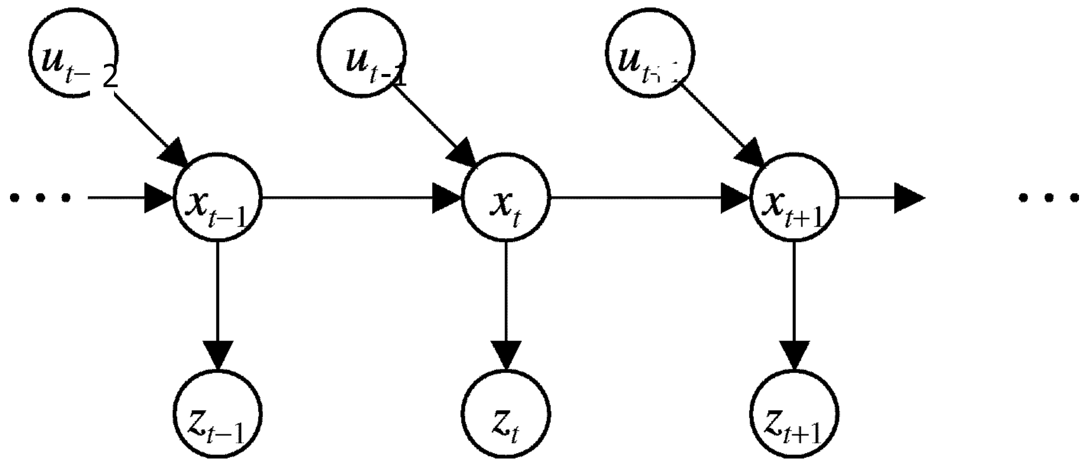
Pieter Abbeel
UC Berkeley EECS

Many slides adapted from Thrun, Burgard and Fox, Probabilistic Robotics

Kalman Filter

- Kalman Filter = special case of a Bayes' filter with dynamics model and sensory model being linear Gaussian:

$$X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$$



$$\begin{aligned} X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\ Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t) \end{aligned}$$

Kalman Filtering Algorithm

- At time 0: $X_0 \sim \mathcal{N}(\mu_{0|0}, \Sigma_{0|0})$
- For t = 1, 2, …
  - Dynamics update:

$$\begin{aligned} \mu_{t+1|0:t} &= A_t \mu_{t|0:t} + B_t u_t \\ \Sigma_{t+1|0:t} &= A_t \Sigma_{t|0:t} A_t^\top + Q_t \end{aligned}$$

  - Measurement update:

$$\begin{aligned} K_{t+1} &= \Sigma_{t+1|0:t} C_{t+1}^\top (C_{t+1} \Sigma_{t+1|0:t} C_{t+1}^\top + R_{t+1})^{-1} \\ \mu_{t+1|0:t+1} &= \mu_{t+1|0:t} + K_{t+1}(z_{t+1} - (C_{t+1}\mu_{t+1|0:t} + d)) \\ \Sigma_{t+1|0:t+1} &= (I - K_{t+1} C_{t+1}) \Sigma_{t+1|0:t} \end{aligned}$$

Nonlinear Dynamical Systems

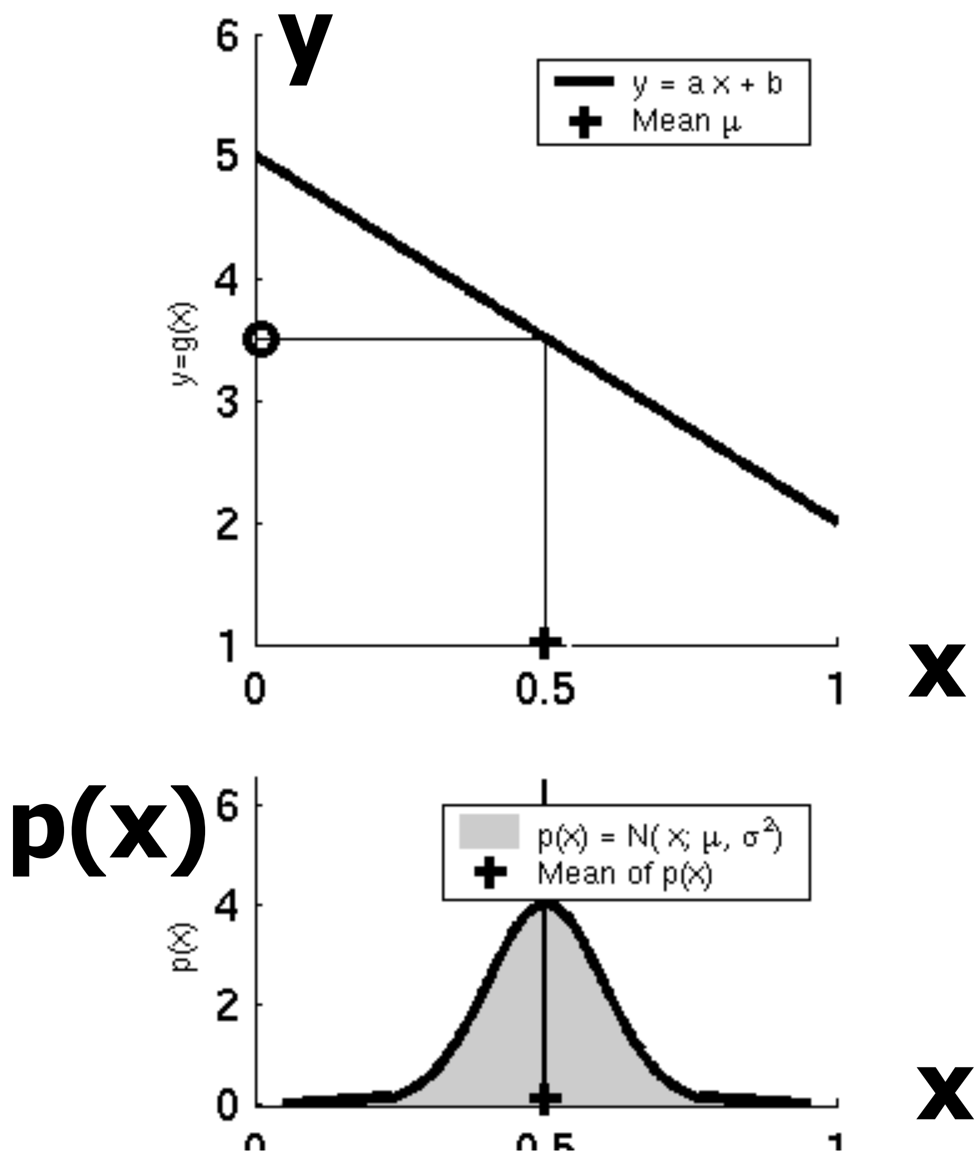
- Most realistic robotic problems involve nonlinear functions:

$$
\begin{aligned}
X_{t+1} &= f_t(X_t, u_t) + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\
Z_t &= h_t(X_t) + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
\end{aligned}
$$

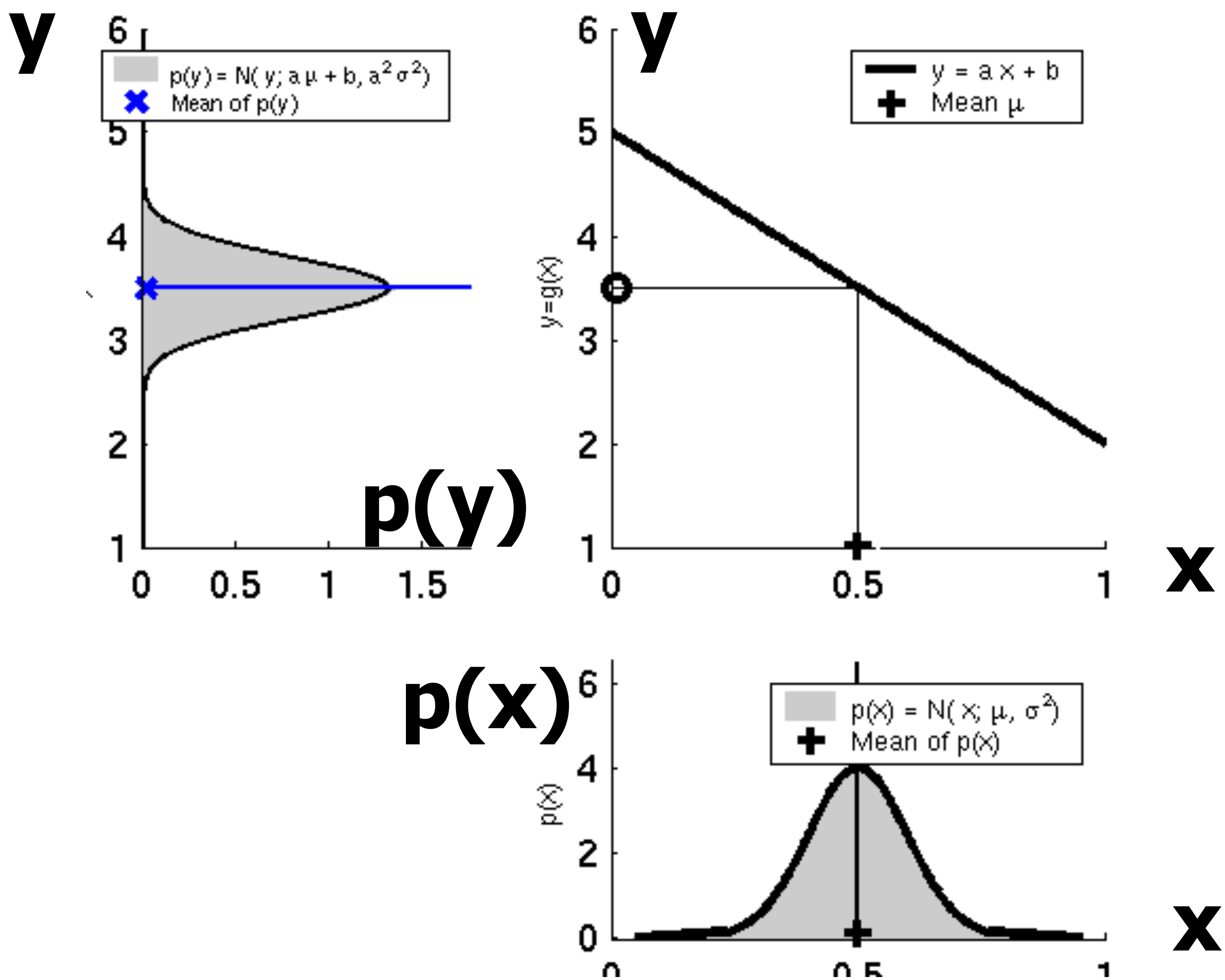
  - Versus linear setting:

$$
\begin{aligned}
X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\
Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
\end{aligned}
$$

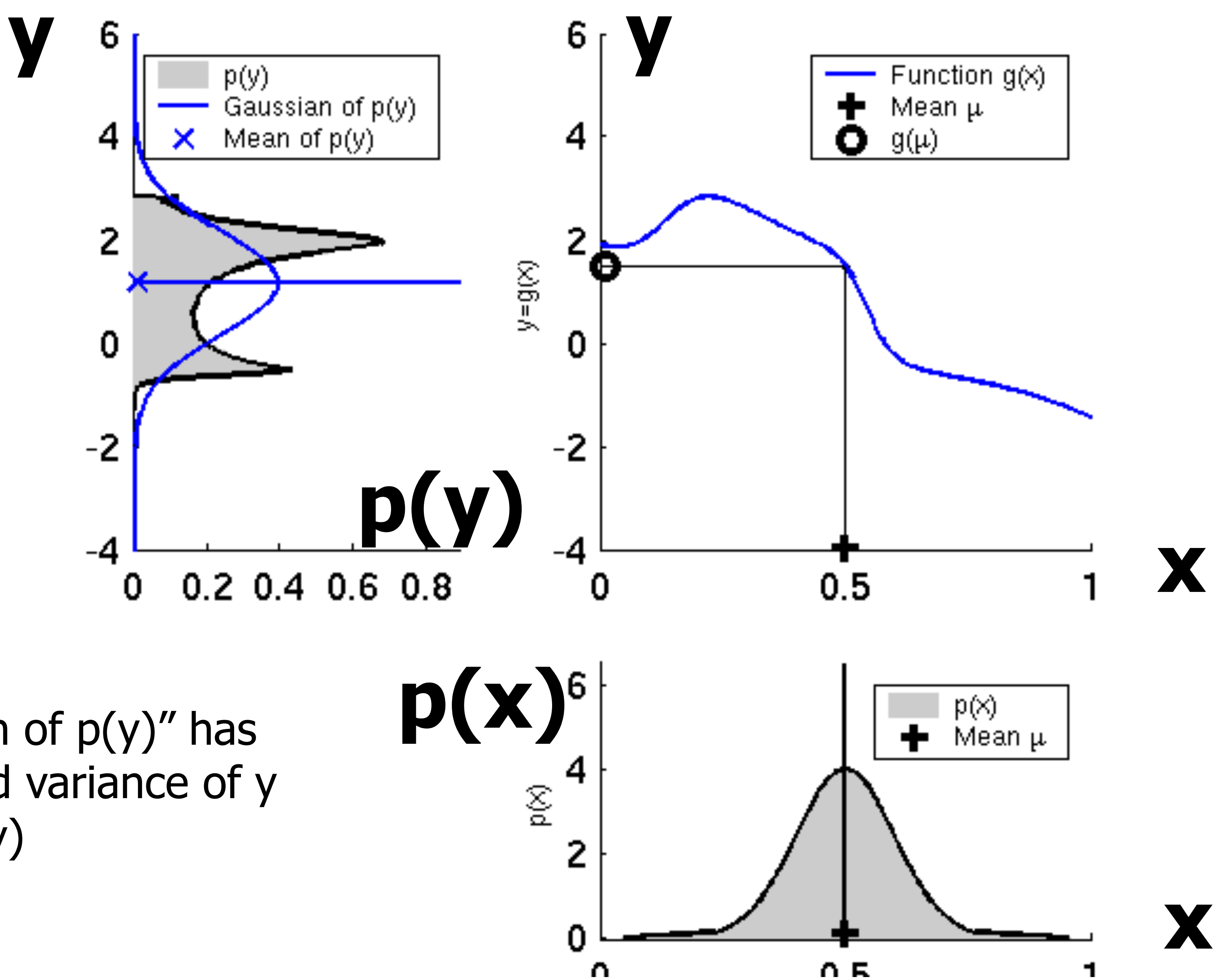
Linearity Assumption Revisited



Linearity Assumption Revisited

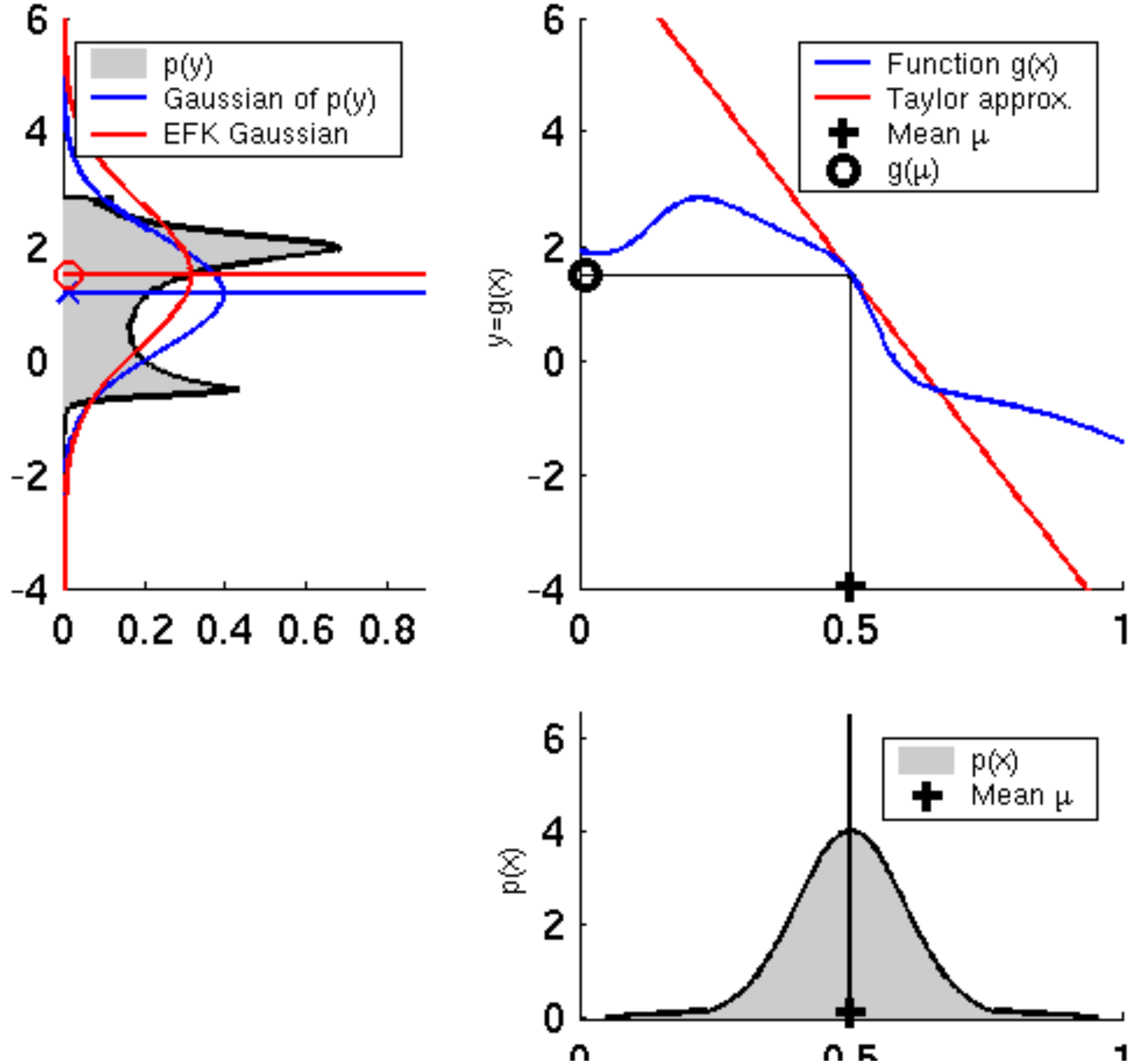


Non-linear Function

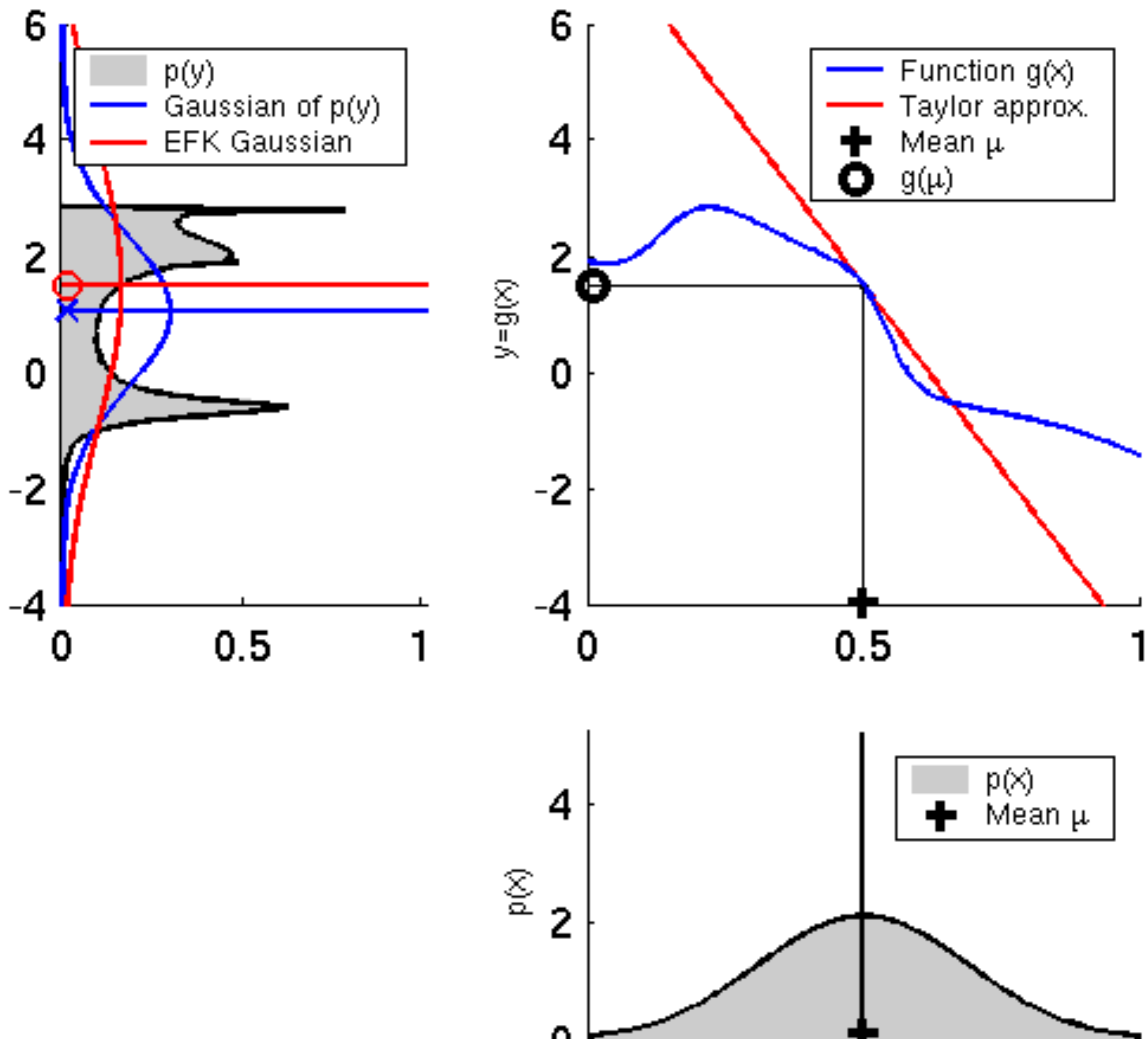


"Gaussian of p(y)" has mean and variance of y under p(y)

EKF Linearization (1)

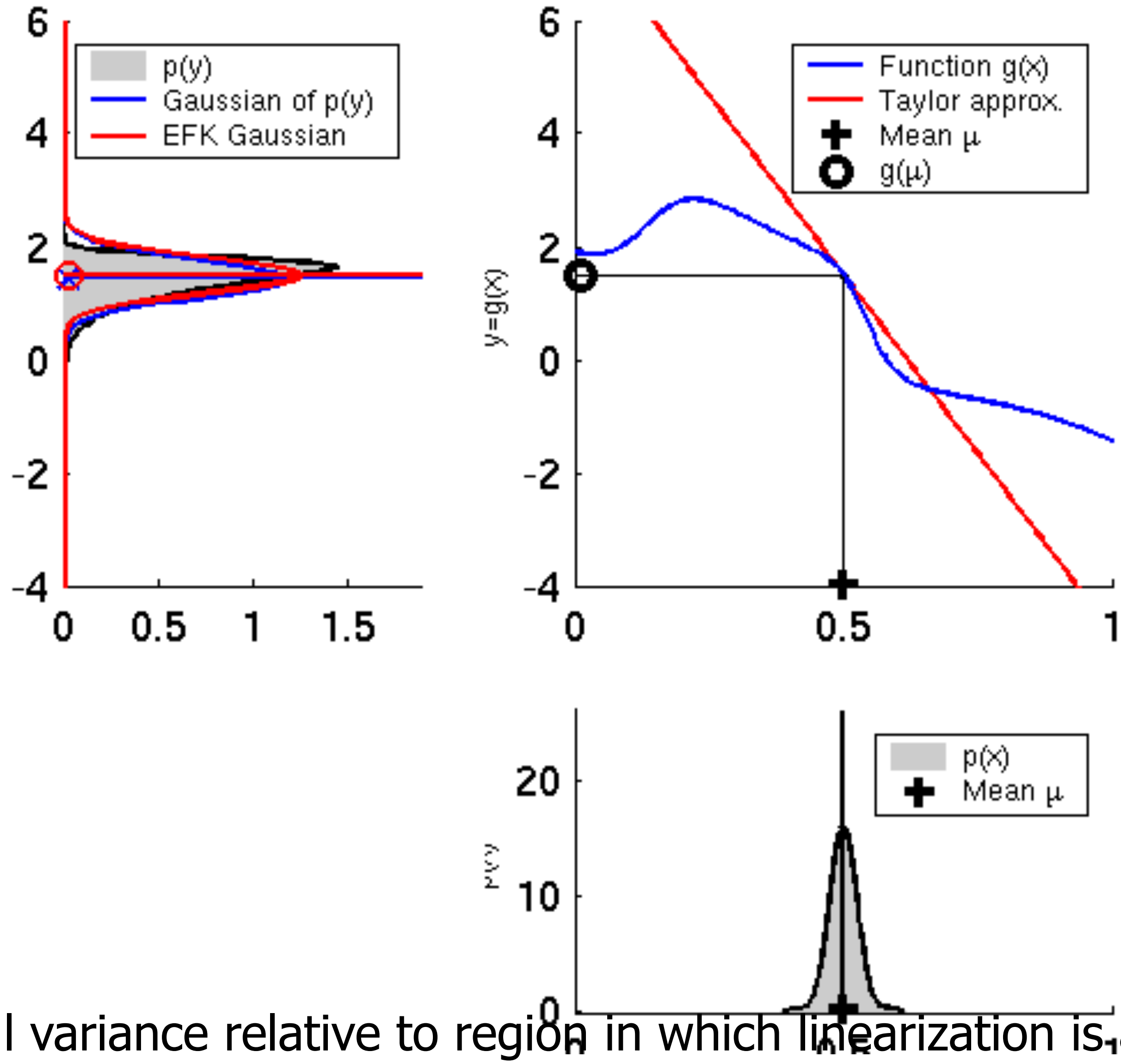


EKF Linearization (2)



p(x) has high variance relative to region in which linearization is accurate.

EKF Linearization (3)



p(x) has small variance relative to region in which linearization is accurate.

EKF Linearization: First Order Taylor Series Expansion

- **Dynamics model:** for x_t "close to" μ_t we have:

$$\begin{aligned} f_t(x_t, u_t) &\approx f_t(\mu_t, u_t) + \frac{\partial f_t(\mu_t, u_t)}{\partial x_t}(x_t - \mu_t) \\ &= f_t(\mu_t, u_t) + F_t(x_t - \mu_t) \end{aligned}$$

- **Measurement model:** for x_t "close to" μ_t we have:

$$\begin{aligned} h_t(x_t) &\approx h_t(\mu_t) + \frac{\partial h_t(\mu_t)}{\partial x_t}(x_t - \mu_t) \\ &= h_t(\mu_t) + H_t(x_t - \mu_t) \end{aligned}$$

EKF Linearization: Numerical

$$\begin{aligned} f_t(x_t, u_t) &\approx f_t(\mu_t, u_t) + \frac{\partial f_t(\mu_t, u_t)}{\partial x_t}(x_t - \mu_t) \\ &= f_t(\mu_t, u_t) + F_t(x_t - \mu_t) \end{aligned}$$

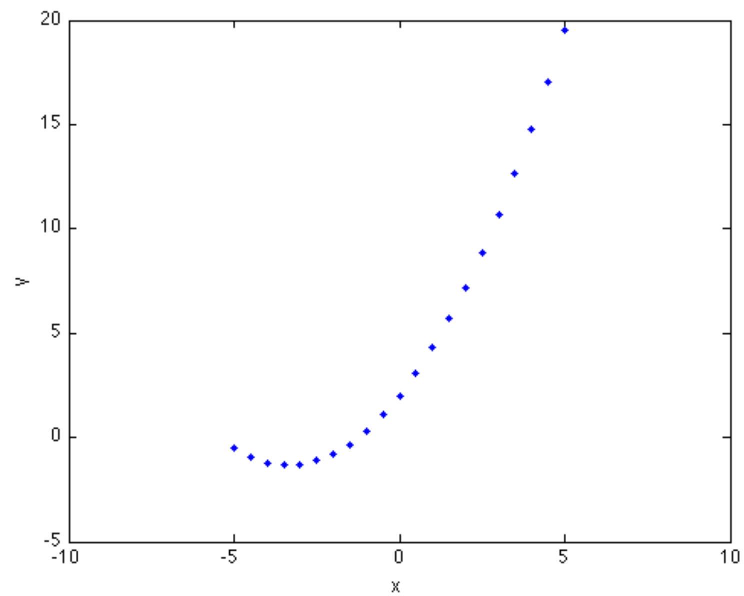
- Numerically compute F_t column by column:

$$\text{for } i = 1, \ldots, n \qquad F_t(:, i) = \frac{f_t(\mu_t + \varepsilon e_i, u_t) - f_t(\mu_t - \varepsilon e_i, u_t)}{2\varepsilon}$$

  - Here e_i is the basis vector with all entries equal to zero, except for the i't entry, which equals 1.
  - If wanting to approximate F_t as closely as possible then ϵ is chosen to be a small number, but not too small to avoid numerical issues

Ordinary Least Squares

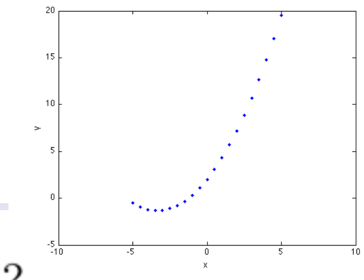
- Given: samples $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$



- Problem: find function of the form $f(x) = a_0 + a_1 x$ that fits the samples as well as possible in the following sense:

$$\min_{a_0, a_1} \frac{1}{2} \sum_{i=1}^{m} (a_0 + a_1 x^{(i)} - y^{(i)})^2$$

Ordinary Least Squares



- Recall our objective: $\min_{a_0, a_1} \frac{1}{2} \sum_{i=1}^{m} (a_0 + a_1 x^{(i)} - y^{(i)})^2$
- Let's write this in vector notation:
  - $\bar{x}^{(i)} = \begin{bmatrix} 1 \\ x^{(i)} \end{bmatrix} \quad a = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$ giving: $\min_{a} \frac{1}{2} \sum_{i=1}^{m} (\bar{x}^{(i)\top} a - y^{(i)})^2$
- Set gradient equal to zero to find extremum:

$$\begin{aligned} 0 = \nabla_a(\ldots) &= \sum_{i=1}^{m} \bar{x}^{(i)} (\bar{x}^{(i)\top} a - y^{(i)}) \\ &= \left(\sum_{i=1}^{m} \bar{x}^{(i)} \bar{x}^{(i)\top} \right) a - \sum_{i=1}^{m} \bar{x}^{(i)} y^{(i)} \\ &= \bar{X} \bar{X}^\top a - \bar{X} y \end{aligned}$$

$$\boxed{a = (\bar{X} \bar{X}^\top)^{-1} \bar{X} y}$$

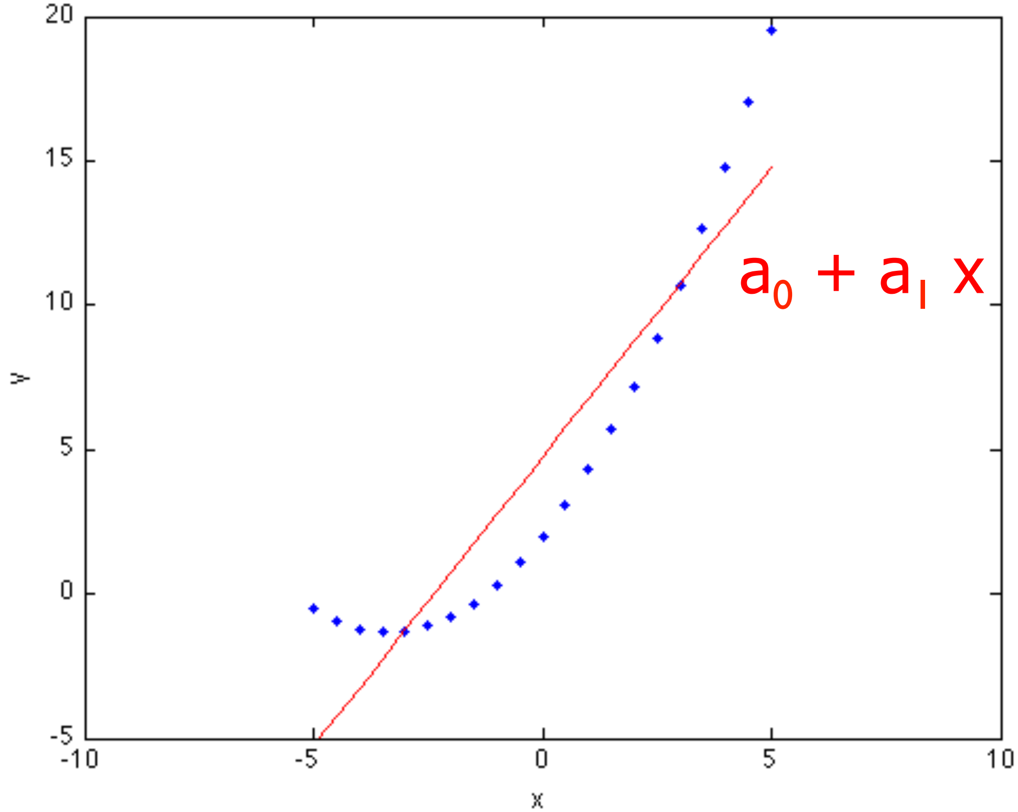
$$\bar{X} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \end{bmatrix}$$

$$y^\top = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}$$

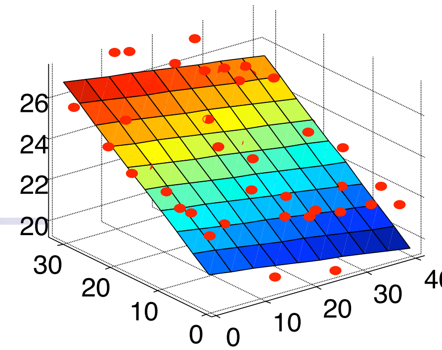
(See the Matrix Cookbook for matrix identities, including derivatives.)

Ordinary Least Squares

- For our example problem we obtain a = [4.75; 2.00]



Ordinary Least Squares



- More generally: $x^{(i)} \in \mathbb{R}^n$

$$\min_{a_0, a_1, a_2, \ldots, a_n} \frac{1}{2} \sum_{i=1}^{m} (a_0 + a_1 x_1^{(i)} + a_2 x_2^{(i)} + \ldots + a_n x_n^{(i)} - y^{(i)})^2$$

- In vector notation:
  - $\bar{x}^{(i)} = \begin{bmatrix} 1 \\ x^{(i)} \end{bmatrix}, \; a = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$ gives: $\min_{a} \frac{1}{2} \sum_{i=1}^{m} (\bar{x}^{(i)\top} a - y^{(i)})^2$

- Set gradient equal to zero to find extremum (exact same derivation as two slides back):

$$\boxed{a = (\bar{X}\bar{X}^\top)^{-1}\bar{X}y}$$

$$\bar{X} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \end{bmatrix}$$

$$y^\top = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}$$

Vector Valued Ordinary Least Squares Problems

- So far have considered approximating a scalar valued function from samples $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$ with $x^{(i)} \in \mathbb{R}^n, y^{(i)} \in \mathbb{R}$
- A vector valued function is just many scalar valued functions and we can approximate it the same way by solving an OLS problem multiple times. Concretely, let $y^{(i)} \in \mathbb{R}^p$ then we have:

$$\text{Find } a_0 \in \mathbb{R}^p, A \in \mathbb{R}^{n \times p}, \text{ such that } \forall i = 1, \ldots, m \quad a_0 + Ax^{(i)} \approx y^{(i)}.$$

- In our vector notation:

$$\bar{x}^{(i)\top} = \begin{bmatrix} 1 & x^{(i)\top} \end{bmatrix}, \bar{A} = \begin{bmatrix} a_0 & A \end{bmatrix},$$
$$\text{Find } \bar{A} \text{ such that } \forall i = 1, \ldots, m \quad \bar{A}\bar{x}^{(i)} \approx y^{(i)}.$$

- This can be solved by solving a separate ordinary least squares problem to find each row of $\bar{A}$

Vector Valued Ordinary Least Squares Problems

$y^{(i)} \in \mathbb{R}^p$

- Solving the OLS problem for each row gives us:

$$(\bar{A}_{j,:})^\top = (\bar{X}\bar{X}^\top)^{-1}\bar{X}y_j^{(0,\ldots,m)}$$

$$y_j^{(0,\ldots,m)} = \begin{bmatrix} y_j^{(0)} & y_j^{(1)} & \cdots & y_j^{(m)} \end{bmatrix}^\top$$

- Each OLS problem has the same structure. We have

$$\bar{A}^\top = (\bar{X}\bar{X}^\top)^{-1}\bar{X}Y$$

$$\begin{aligned} Y &= \begin{bmatrix} y_1^{(0,\ldots,m)} & y_2^{(0,\ldots,m)} & \cdots & y_p^{(0,\ldots,m)} \end{bmatrix} \\ &= \begin{bmatrix} y_1^{(0)} & y_2^{(0)} & \cdots & y_p^{(0)} \\ y_1^{(1)} & y_2^{(1)} & \cdots & y_p^{(1)} \\ & \cdots & & \\ y_1^{(m)} & y_2^{(m)} & \cdots & y_p^{(m)} \end{bmatrix} \end{aligned}$$

Vector Valued Ordinary Least Squares and EKF Linearization

- Approximate $x_{t+1} = f_t(x_t, u_t)$

  with affine function $a_0 + F_t x_t$

  by running least squares on samples from the function:

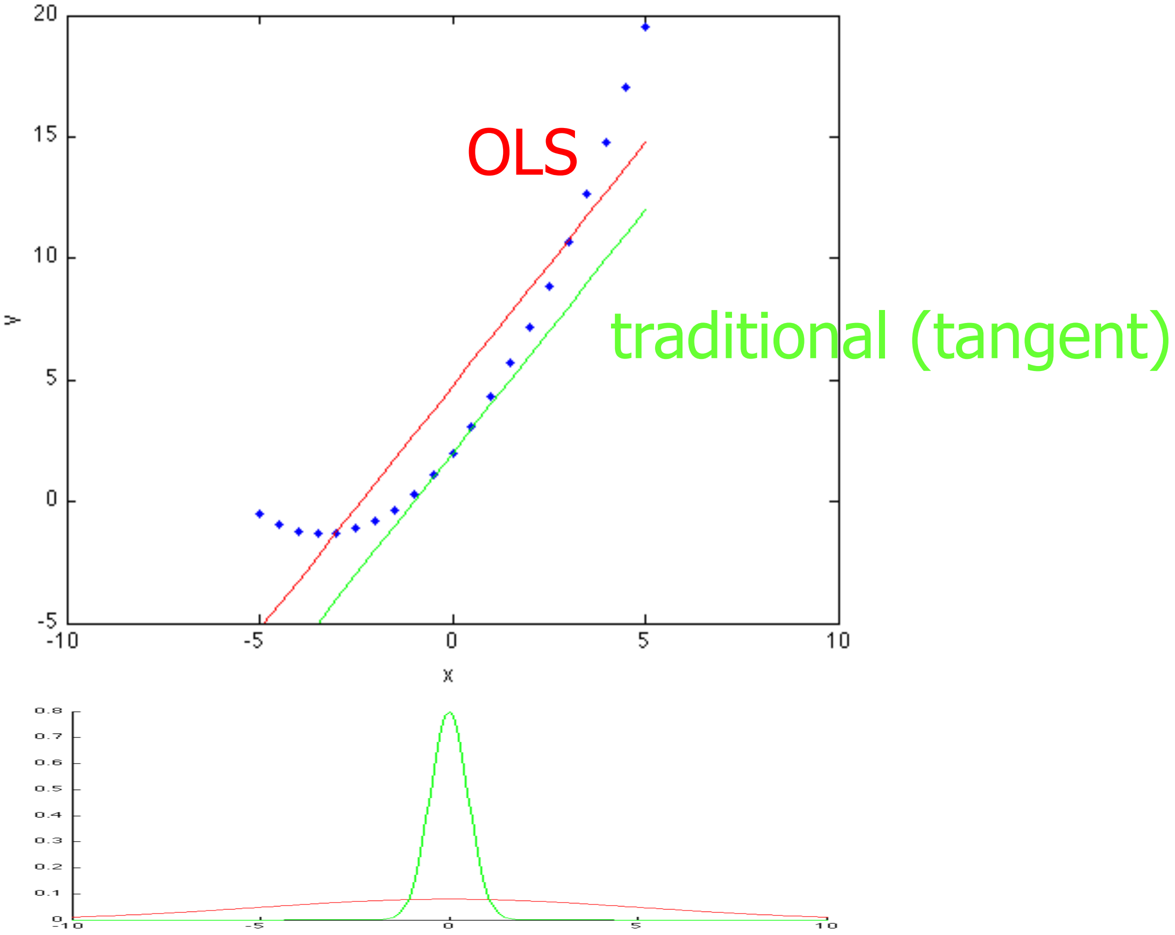
$\{(x_t^{(1)}, y^{(1)}=f_t(x_t^{(1)},u_t), (x_t^{(2)}, y^{(2)}=f_t(x_t^{(2)},u_t), \ldots, (x_t^{(m)}, y^{(m)}=f_t(x_t^{(m)},u_t)\}$

$$\begin{bmatrix} a_0 & F_t \end{bmatrix}^\top = \bar{A}^\top = (\bar{X}\bar{X}^\top)^{-1}\bar{X}Y$$

- Similarly for $z_{t+1} = h_t(x_t)$

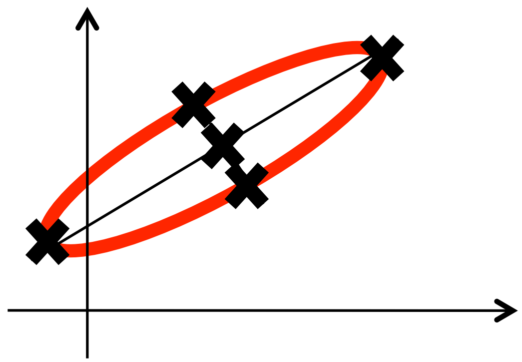
OLS and EKF Linearization: Sample Point Selection

- OLS vs. traditional (tangent) linearization:



OLS Linearization: choosing samples points

- Perhaps most natural choice:



  - $\mu_t, \mu_t + \Sigma_t^{1/2}, \mu_t - \Sigma_t^{1/2}$
  - reasonable way of trying to cover the region with reasonably high probability mass

Analytical vs. Numerical Linearization

- Numerical (based on least squares or finite differences) could give a more accurate "regional" approximation. Size of region determined by evaluation points.
- Computational efficiency:
  - Analytical derivatives can be cheaper or more expensive than function evaluations
- Development hint:
  - Numerical derivatives tend to be easier to implement
  - If deciding to use analytical derivatives, implementing finite difference derivative and comparing with analytical results can help debugging the analytical derivatives

EKF Algorithm

- At time 0: $X_0 \sim \mathcal{N}(\mu_{0|0}, \Sigma_{0|0})$
- For t = 1, 2, …
  - Dynamics update: $f_t(x_t, u_t) \approx a_{0,t} + F_t(x_t - \mu_{t|0:t})$

$$
\begin{aligned}
(a_{0,t}, F_t) &= \text{linearize}(f_t, \mu_{t|0:t}, \Sigma_{t|0:t}, u_t) \\
\mu_{t+1|0:t} &= a_{0,t} \\
\Sigma_{t+1|0:t} &= F_t \Sigma_{t|0:t} F_t^\top + Q_t
\end{aligned}
$$

  - Measurement update: $h_{t+1}(x_{t+1}) \approx c_{0,t+1} + H_{t+1}(x_{t+1} - \mu_{t+1|0:t})$

$$
\begin{aligned}
(c_{0,t+1}, H_{t+1}) &= \text{linearize}(h_{t+1}, \mu_{t+1|0:t}, \Sigma_{t+1|0:t}) \\
K_{t+1} &= \Sigma_{t+1|0:t} H_{t+1}^\top (H_{t+1} \Sigma_{t+1|0:t} H_{t+1}^\top + R_{t+1})^{-1} \\
\mu_{t+1|0:t+1} &= \mu_{t+1|0:t} + K_{t+1}(z_{t+1} - c_{0,t+1}) \\
\Sigma_{t+1|0:t+1} &= (I - K_{t+1} H_{t+1}) \Sigma_{t+1|0:t}
\end{aligned}
$$

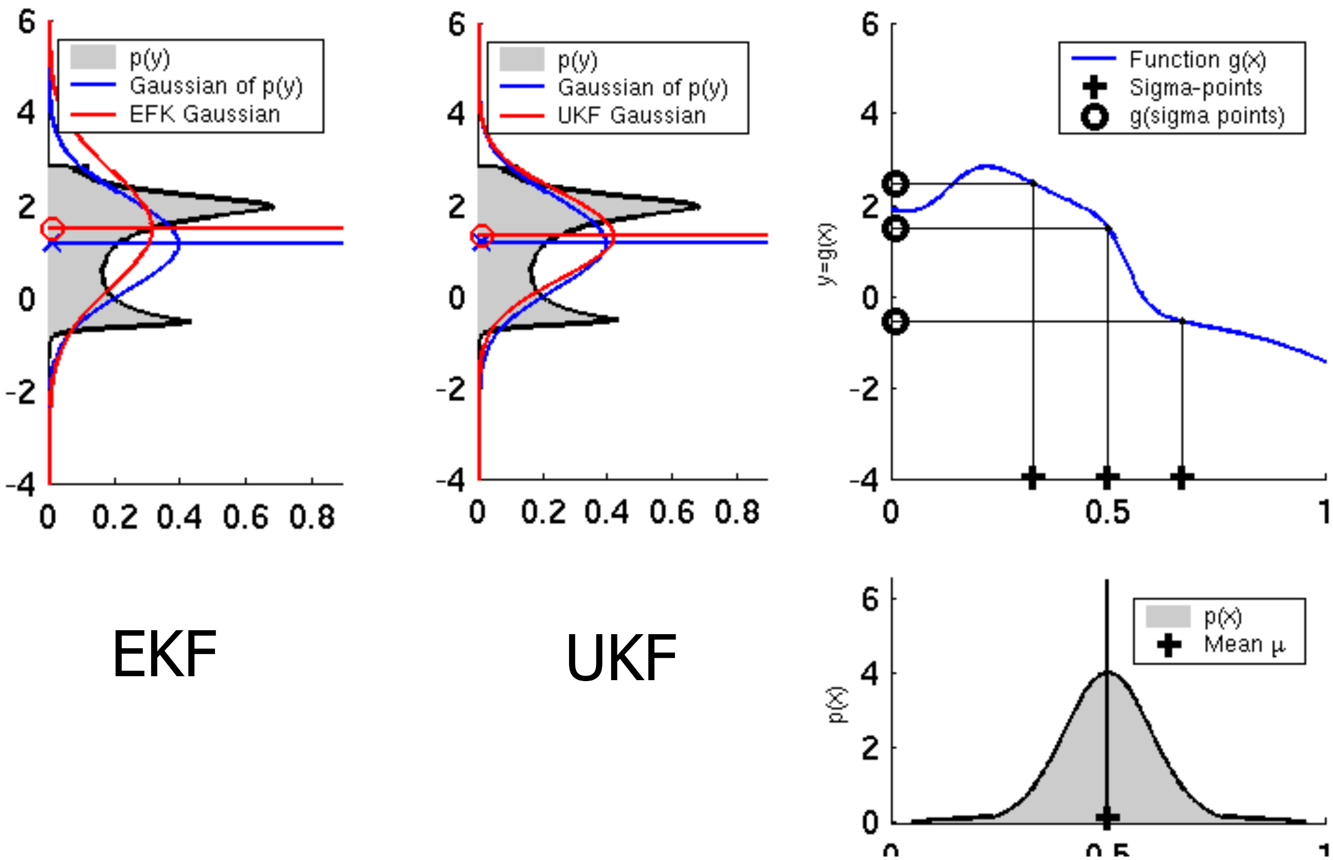
EKF Summary

- Highly efficient: Polynomial in measurement dimensionality k and state dimensionality n:

  $$O(k^{2.376} + n^2)$$

- Not optimal!
- Can diverge if nonlinearities are large!
- Works surprisingly well even when all assumptions are violated!

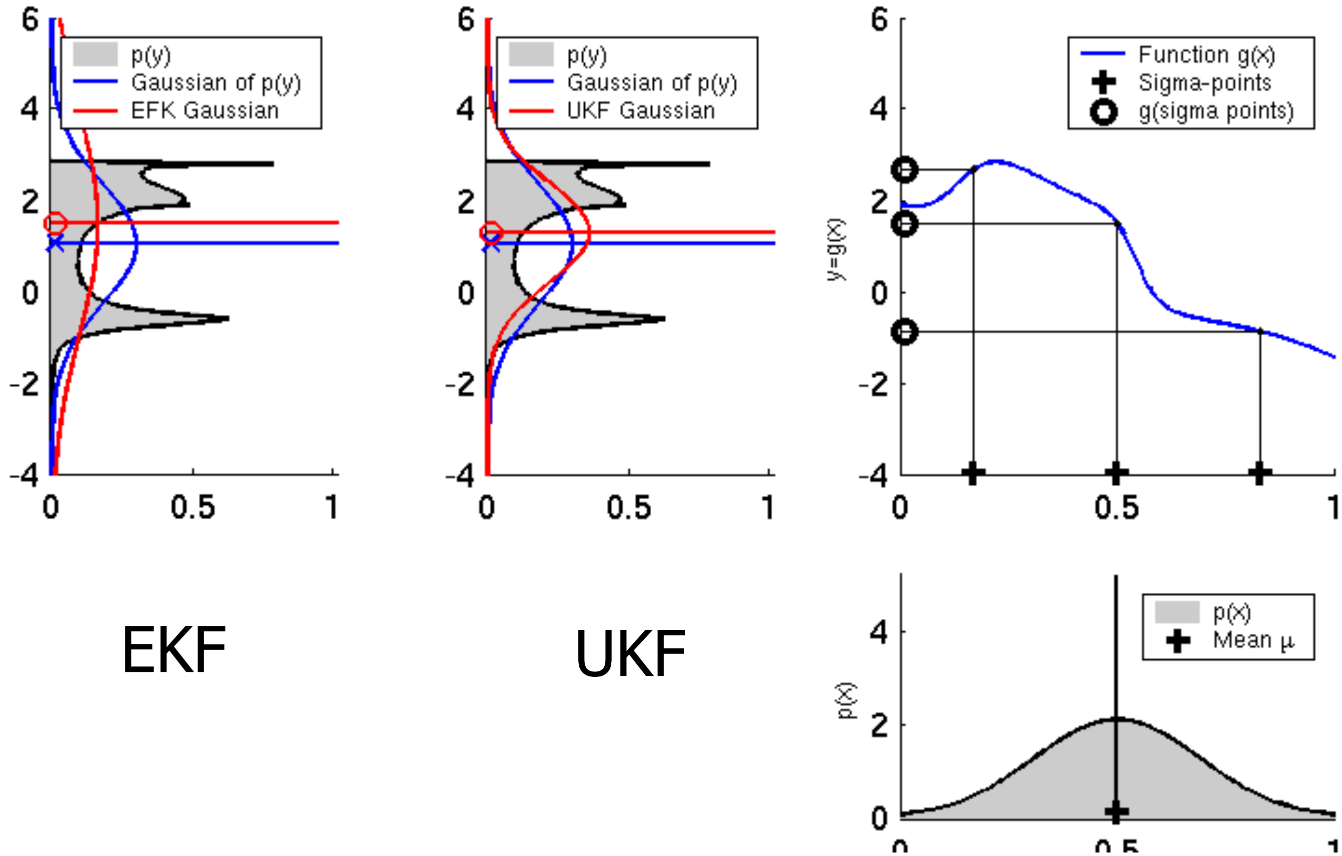
Linearization via Unscented Transform



EKF

UKF

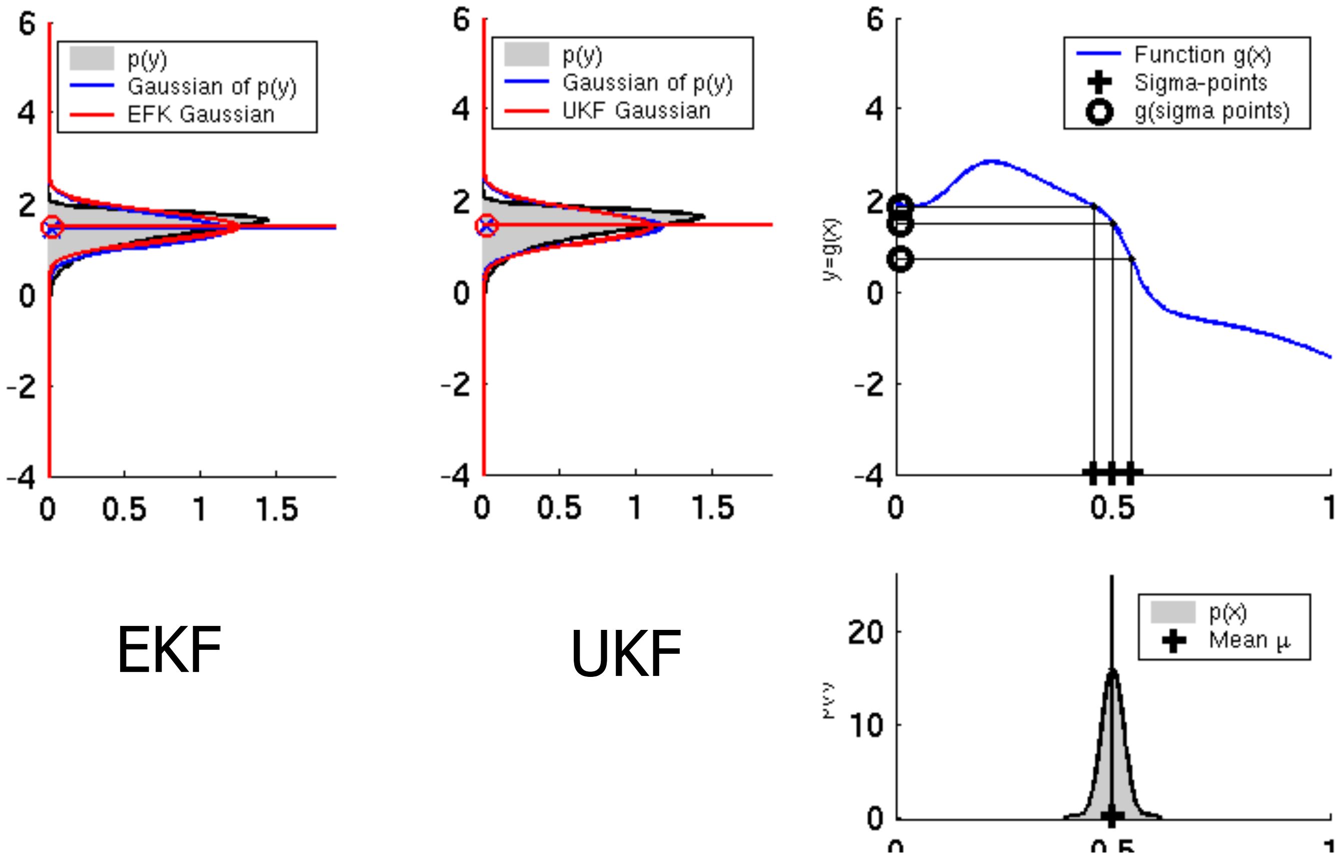
UKF Sigma-Point Estimate (2)



EKF

UKF

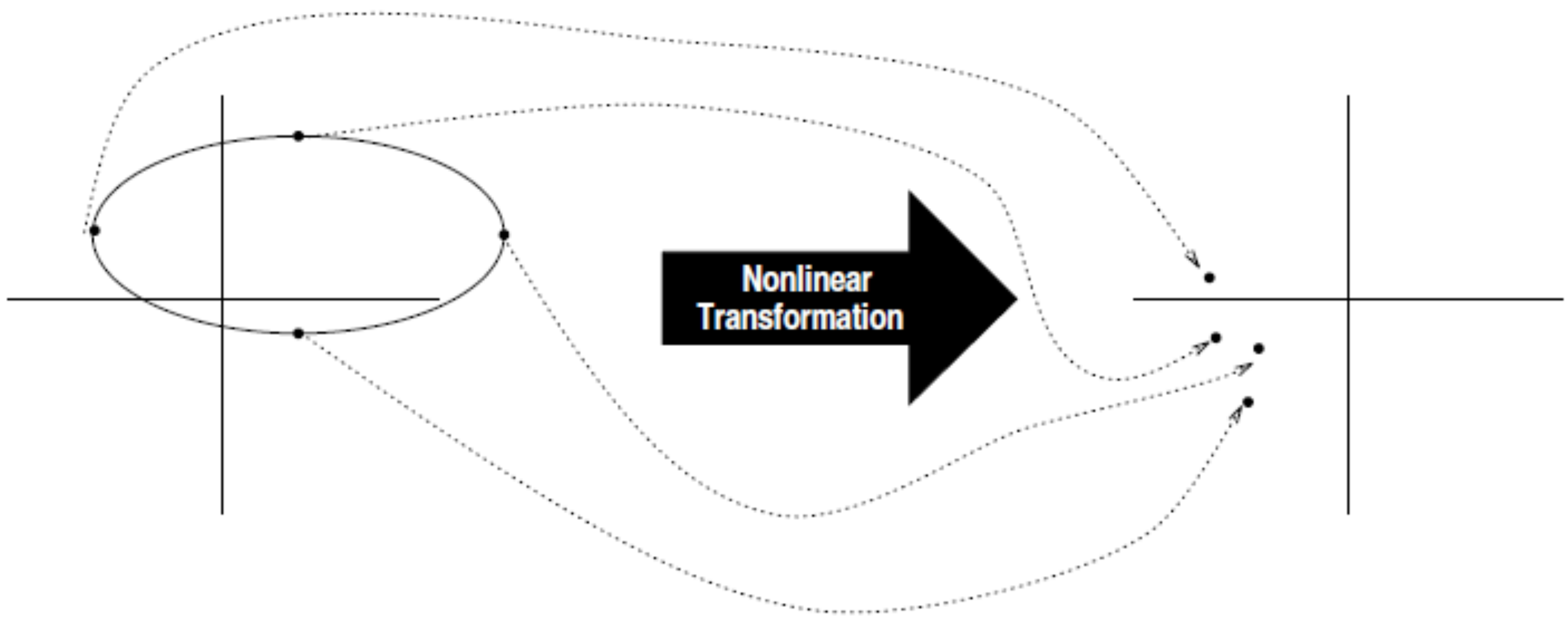
UKF Sigma-Point Estimate (3)



EKF

UKF

UKF Sigma-Point Estimate (4)



[Julier and Uhlmann, 1997]

UKF intuition why it can perform better

- Assume we know the distribution over X and it has a mean \bar{x}
- Y = f(X)

$$\mathbf{f}[\mathbf{x}] = \mathbf{f}[\bar{\mathbf{x}} + \boldsymbol{\delta}\mathbf{x}]$$
$$= \mathbf{f}[\bar{\mathbf{x}}] + \nabla\mathbf{f}\boldsymbol{\delta}\mathbf{x} + \frac{1}{2}\nabla^2\mathbf{f}\boldsymbol{\delta}\mathbf{x}^2 + \frac{1}{3!}\nabla^3\mathbf{f}\boldsymbol{\delta}\mathbf{x}^3 + \frac{1}{4!}\nabla^4\mathbf{f}\boldsymbol{\delta}\mathbf{x}^4 + \cdots$$

$$\bar{\mathbf{y}} = \mathbf{f}[\bar{\mathbf{x}}] + \frac{1}{2}\nabla^2\mathbf{f}\,\mathbf{P}_{xx} + \frac{1}{2}\nabla^4\mathbf{f}\,\mathrm{E}\left[\boldsymbol{\delta}\mathbf{x}^4\right] + \cdots$$

$$\mathbf{P}_{yy} = \nabla\mathbf{f}\,\mathbf{P}_{xx}(\nabla\mathbf{f})^T + \frac{1}{2\times 4!}\nabla^2\mathbf{f}\left(\mathrm{E}\left[\boldsymbol{\delta}\mathbf{x}^4\right] - \mathrm{E}\left[\boldsymbol{\delta}\mathbf{x}^2\mathbf{P}_{yy}\right] - \mathrm{E}\left[\mathbf{P}_{yy}\boldsymbol{\delta}\mathbf{x}^2\right] + \mathbf{P}_{yy}^2\right)(\nabla^2\mathbf{f})^T +$$
$$\frac{1}{3!}\nabla^3\mathbf{f}\mathrm{E}\left[\boldsymbol{\delta}\mathbf{x}^4\right](\nabla\mathbf{f})^T + \cdots.$$

- EKF approximates f by first order and ignores higher-order terms
- UKF uses f exactly, but approximates p(x).

Original unscented transform

- Picks a minimal set of sample points that match 1st, 2nd and 3rd moments of a Gaussian:

$$
\begin{aligned}
\mathcal{X}_0 &= \bar{\mathbf{x}} & W_0 &= \kappa/(n+\kappa) \\
\mathcal{X}_i &= \bar{\mathbf{x}} + \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_i &= 1/2(n+\kappa) \\
\mathcal{X}_{i+n} &= \bar{\mathbf{x}} - \left(\sqrt{(n+\kappa)\mathbf{P}_{xx}}\right)_i & W_{i+n} &= 1/2(n+\kappa)
\end{aligned}
$$

- \bar{x} = mean, P_{xx} = covariance, i → i'th column, $x \in \Re^n$
- κ : extra degree of freedom to fine-tune the higher order moments of the approximation; when x is Gaussian, n+κ = 3 is a suggested heuristic
- L = \sqrt{P_{xx}} can be chosen to be any matrix satisfying:
  - $L L^T = P_{xx}$

[Julier and Uhlmann, 1997]

Unscented Kalman filter

- Dynamics update:
  - Can simply use unscented transform and estimate the mean and variance at the next time from the sample points
- Observation update:
  - Use sigma-points from unscented transform to compute the covariance matrix between x_t and z_t. Then can do the standard update.

Algorithm Unscented_Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

1. $\mathcal{X}_{t-1} = \left(\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}}\right)$
2. $\bar{\mathcal{X}}_t^* = g(\mu_t, \mathcal{X}_{t-1})$
3. $\bar{\mu}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{X}}_t^{*[i]}$
4. $\bar{\Sigma}_t = \sum_{i=0}^{2n} w_c^{[i]} (\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)(\bar{\mathcal{X}}_t^{*[i]} - \bar{\mu}_t)^\top + R_t$
5. $\bar{\mathcal{X}}_t = \left(\bar{\mu}_t \quad \bar{\mu}_t + \gamma\sqrt{\bar{\Sigma}_t} \quad \bar{\mu}_t - \gamma\sqrt{\bar{\Sigma}_t}\right)$
6. $\bar{\mathcal{Z}}_t = h(\bar{\mathcal{X}}_t)$
7. $\hat{z}_t = \sum_{i=0}^{2n} w_m^{[i]} \bar{\mathcal{Z}}_t^{[i]}$
8. $S_t = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^\top + Q_t$
9. $\bar{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^{[i]} \left(\bar{\mathcal{X}}_t^{[i]} - \bar{\mu}_t\right)\left(\bar{\mathcal{Z}}_t^{[i]} - \hat{z}_t\right)^\top$
10. $K_t = \bar{\Sigma}_t^{x,z} S_t^{-1}$
11. $\mu_t = \bar{\mu}_t + K_t(z_t - \hat{z}_t)$
12. $\Sigma_t = \bar{\Sigma}_t - K_t S_t K_t^\top$
13. **return** μ_t, Σ_t

Here $L = \sqrt{\Sigma}$ can be chosen to be any $n \times n$ matrix satisfying:
$LL^\top = \Sigma$
Technically this is an abuse of notation for the symbol $\sqrt{\ }$.

[Table 3.4 in Probabilistic Robotics]

UKF Summary

- Highly efficient: Same complexity as EKF, with a constant factor slower in typical practical applications
- Better linearization than EKF: Accurate in first two terms of Taylor expansion (EKF only first term) + capturing more aspects of the higher order terms
- Derivative-free: No Jacobians needed
- Still not optimal!