MODULE 2 LESSON 3

GOING NONLINEAR: THE EXTENDED KALMAN FILTER

Recap | The Linear Kalman Filter

Linear motion / process model

$$\underset{\overset{\uparrow}{\text{current state}}}{\mathbf{x}_k} = \mathbf{F}_{k-1}\underset{\overset{\uparrow}{\text{previous state}}}{\mathbf{x}_{k-1}} + \mathbf{G}_{k-1}\underset{\overset{\uparrow}{\text{inputs}}}{\mathbf{u}_{k-1}} + \underset{\overset{\uparrow}{\text{process noise}}}{\mathbf{w}_{k-1}}$$

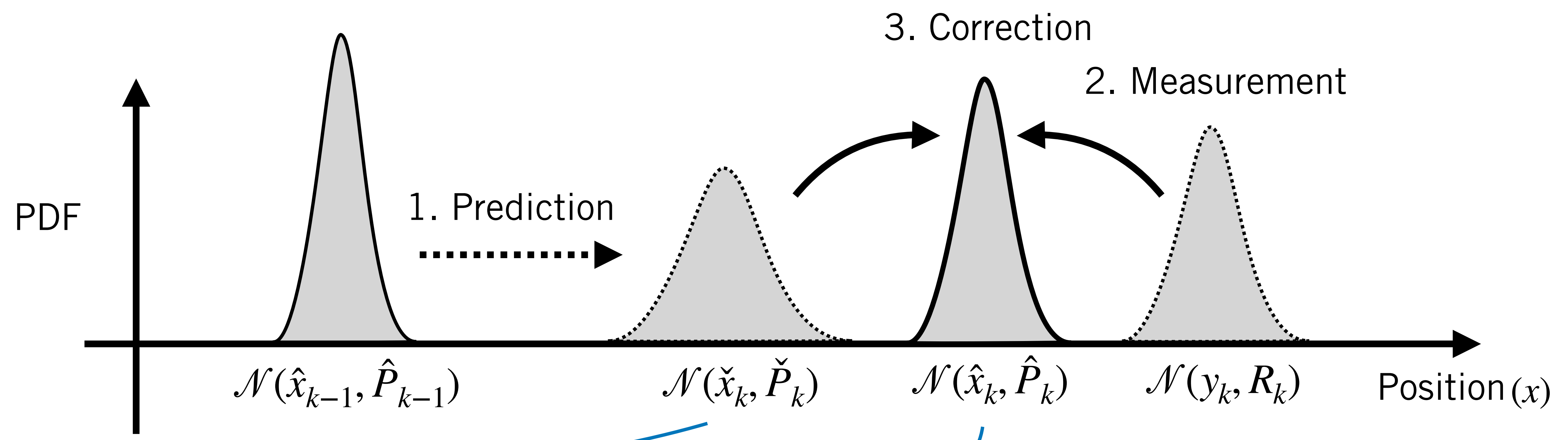
$$\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$$

Linear measurement model

$$\underset{\overset{\nearrow}{\text{measurement}}}{\mathbf{y}_k} = \mathbf{H}_k\underset{\overset{\uparrow}{\text{state}}}{\mathbf{x}_k} + \underset{\overset{\nwarrow}{\text{measurement noise}}}{\mathbf{v}_k}$$

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$$

Linear discrete-time Kalman Filter:


3. Correction
2. Measurement
PDF
1. Prediction
$\mathcal{N}(\hat{x}_{k-1}, \hat{P}_{k-1})$
$\mathcal{N}(\check{x}_k, \check{P}_k)$
$\mathcal{N}(\hat{x}_k, \hat{P}_k)$
$\mathcal{N}(y_k, R_k)$
Position (x)

1. Prediction

$$\begin{cases} \check{\mathbf{x}}_k = \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \\ \check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \end{cases}$$

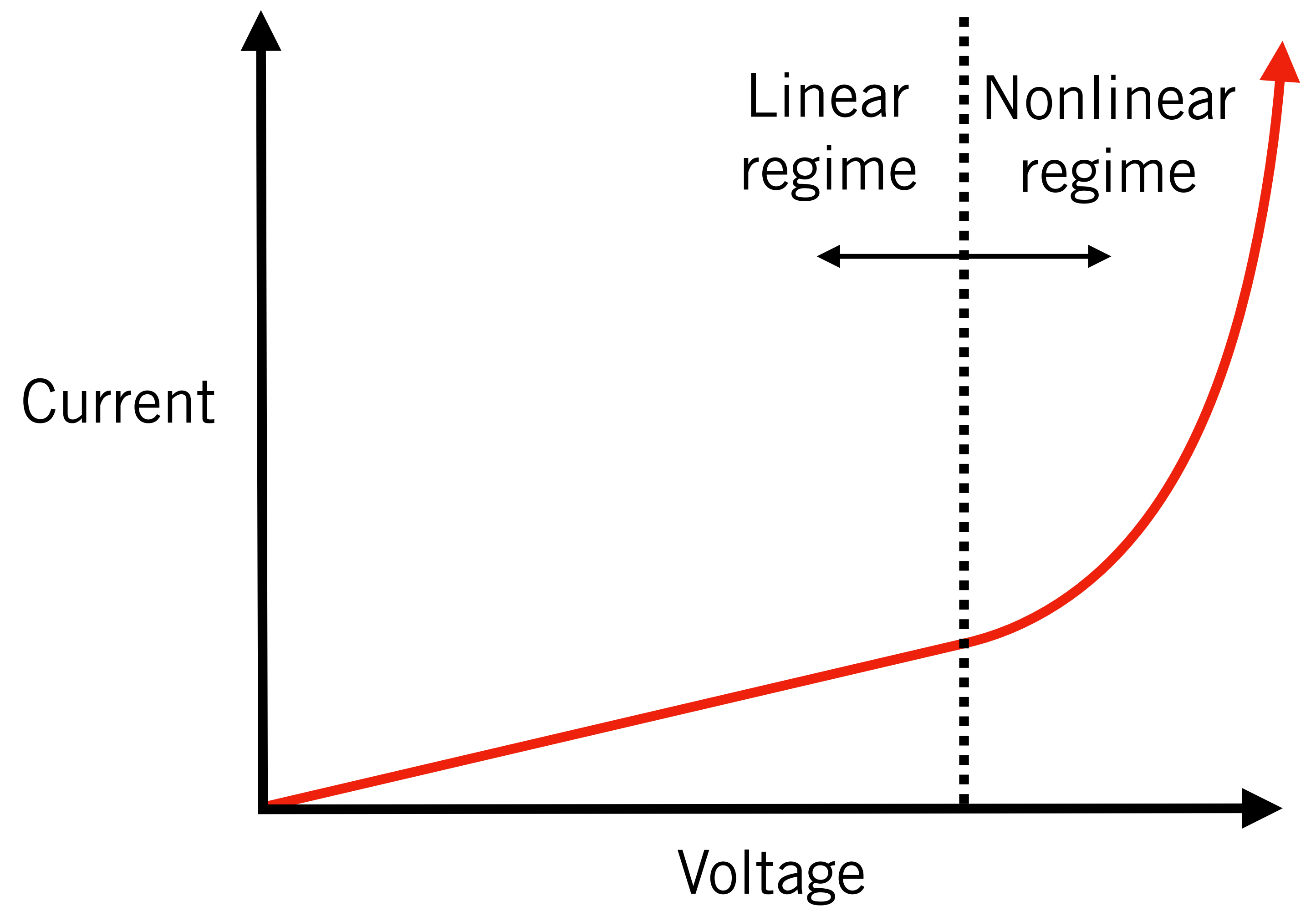
2/3. Measurement & Correction

$$\begin{cases} \mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\check{\mathbf{x}}_k) \\ \hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k \end{cases}$$

Nonlinear Kalman Filtering

Linear systems do not exist in reality!

Ohm's Law: $I = V/R$



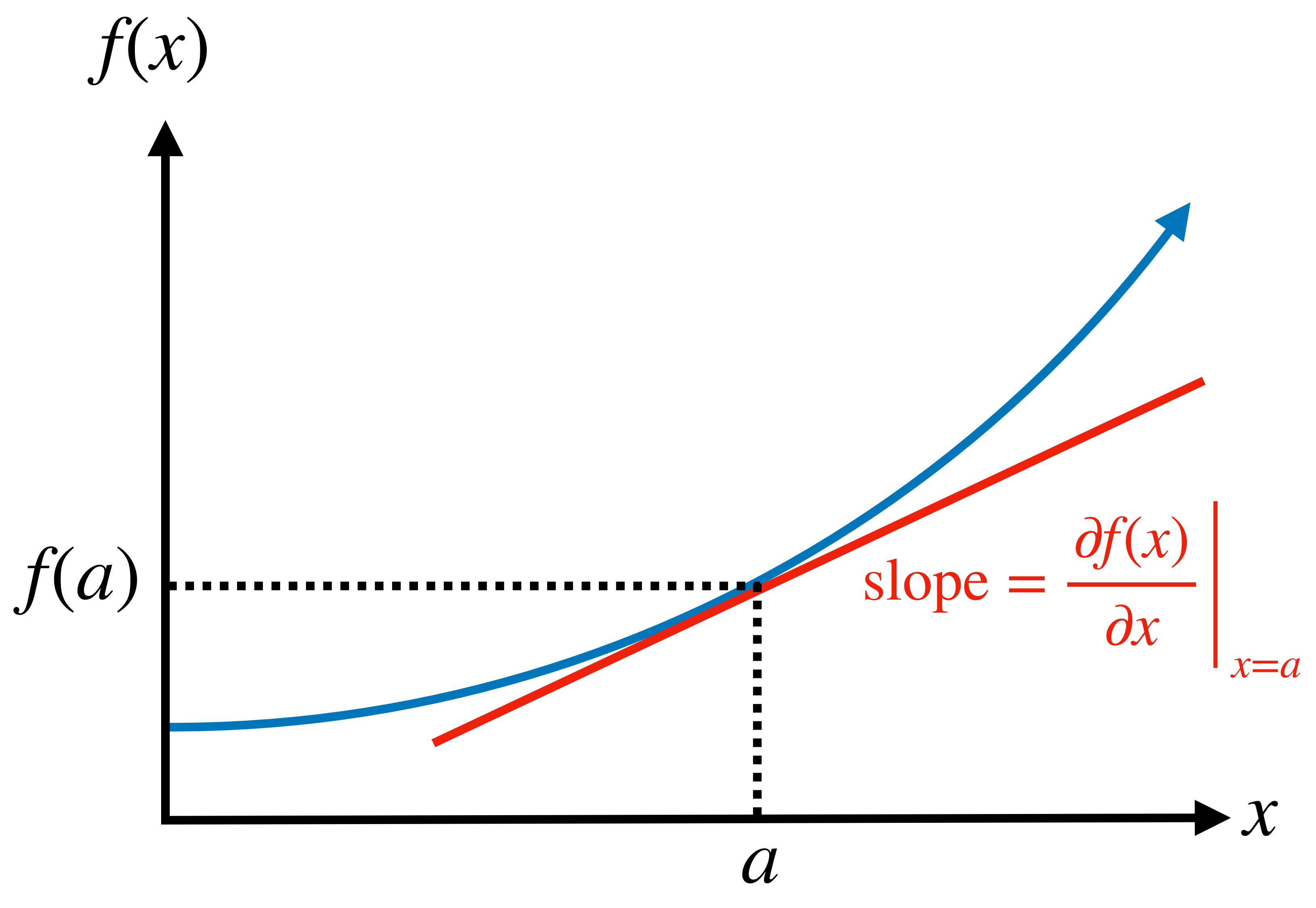
How can we adapt the Kalman Filter to *nonlinear* discrete-time systems?

$$\underset{\substack{\uparrow\\ \text{current}\\ \text{state}}}{\mathbf{x}_k} = \mathbf{f}_{k-1}(\underset{\substack{\nearrow\\ \text{previous}\\ \text{state}}}{\mathbf{x}_{k-1}}, \underset{\substack{\uparrow\\ \text{inputs}}}{\mathbf{u}_{k-1}}, \underset{\substack{\nwarrow\\ \text{process}\\ \text{noise}}}{\mathbf{w}_{k-1}})$$

$$\underset{\substack{\nearrow\\ \text{measurement}}}{\mathbf{y}_k} = \mathbf{h}_k(\underset{\substack{\uparrow\\ \text{state}}}{\mathbf{x}_k}, \underset{\substack{\nwarrow\\ \text{measurement}\\ \text{noise}}}{\mathbf{v}_k})$$

EKF | Linearizing a Nonlinear System

Choose an operating point *a* and approximate the nonlinear function by a tangent line at that point



$f(x)$
$f(a)$
slope $= \left.\frac{\partial f(x)}{\partial x}\right|_{x=a}$
x
a

Mathematically, we compute this linear approximation using a first-order Taylor expansion:

$$\underbrace{\boxed{f(x) \approx f(a) + \left.\frac{\partial f(x)}{\partial x}\right|_{x=a} (x-a)}}_{\text{First-order terms}} \underbrace{+ \frac{1}{2!}\left.\frac{\partial^2 f(x)}{\partial x^2}\right|_{x=a} (x-a)^2 + \frac{1}{3!}\left.\frac{\partial^2 f(x)}{\partial x^2}\right|_{x=a} (x-a)^3 + \dots}_{\text{Higher-order terms}}$$

EKF | Linearizing a Nonlinear System

For the EKF, we choose the operating point to be our most recent state estimate, our known input, and zero noise:

Linearized motion model

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1}) \approx \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) + \underbrace{\left.\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1},\mathbf{0}}}_{\mathbf{F}_{k-1}} (\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}) + \underbrace{\left.\frac{\partial \mathbf{f}_{k-1}}{\partial \mathbf{w}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1},\mathbf{0}}}_{\mathbf{L}_{k-1}} \mathbf{w}_{k-1}$$

Linearized measurement model

$$\mathbf{y}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{v}_k) \approx \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \underbrace{\left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}\right|_{\check{\mathbf{x}}_k,\mathbf{0}}}_{\mathbf{H}_k} (\mathbf{x}_k - \check{\mathbf{x}}_k) + \underbrace{\left.\frac{\partial \mathbf{h}_k}{\partial \mathbf{v}_k}\right|_{\check{\mathbf{x}}_k,\mathbf{0}}}_{\mathbf{M}_k} \mathbf{v}_k$$

We now have a linear system in state-space! The matrices $\mathbf{F}_{k-1}$, $\mathbf{L}_{k-1}$, $\mathbf{H}_k$, and $\mathbf{M}_k$ are called the *Jacobian matrices* of the system

EKF | Computing Jacobian Matrices

In vector calculus, a *Jacobian matrix* is the matrix of all first-order partial derivatives of a vector-valued function

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Intuitively, the Jacobian matrix tells you how fast each output of the function is changing along each input dimension

For example:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_1 + x_2 \\ x_1^2 \end{bmatrix} \quad \Longrightarrow \quad \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2x_1 & 0 \end{bmatrix}$$

EKF | Putting It All Together

With our linearized models and Jacobians, we can now use the Kalman Filter equations!

1 Linearized motion model

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) + \mathbf{F}_{k-1}\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}\right) + \mathbf{L}_{k-1}\mathbf{w}_{k-1}$$

3 Linearized measurement model

$$\mathbf{y}_k = \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \mathbf{H}_k\left(\mathbf{x}_k - \check{\mathbf{x}}_k\right) + \mathbf{M}_k\mathbf{v}_k$$

2 Prediction

nonlinear

$$\check{\mathbf{x}}_k = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0})$$

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\hat{\mathbf{P}}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T$$

4 Optimal gain

$$\mathbf{K}_k = \check{\mathbf{P}}_k\mathbf{H}_k^T(\mathbf{H}_k\check{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T)^{-1}$$

5 Correction

nonlinear

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \mathbf{K}_k\underbrace{(\mathbf{y}_k - \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}))}_{\text{measurement residue}}$$

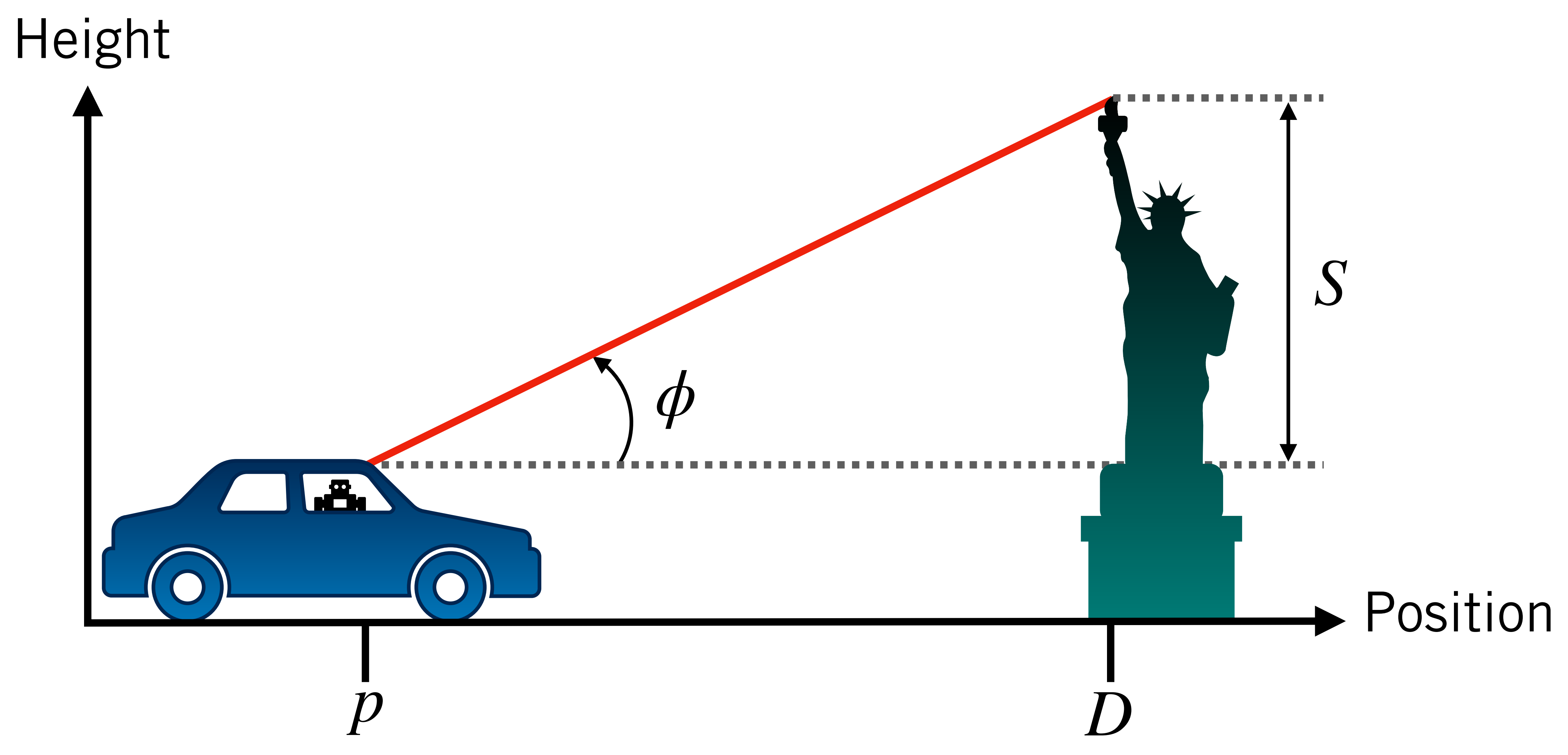
$$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k$$

$\check{\mathbf{x}}_k$	Prediction (given motion model) at time *k*
$\hat{\mathbf{x}}_k$	Corrected prediction (given measurement) at time *k*

noise Jacobians

Mostly L & M are identity matrices.

EKF | Short Example



$$\mathbf{x} = \begin{bmatrix} p \\ \dot{p} \end{bmatrix} \qquad \mathbf{u} = \ddot{p}$$

S and *D* are known in advance

Motion/Process model

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

$$= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \mathbf{x}_{k-1} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \mathbf{u}_{k-1} + \mathbf{w}_{k-1}$$

Landmark measurement model

$$y_k = \phi_k = h(p_k, v_k)$$

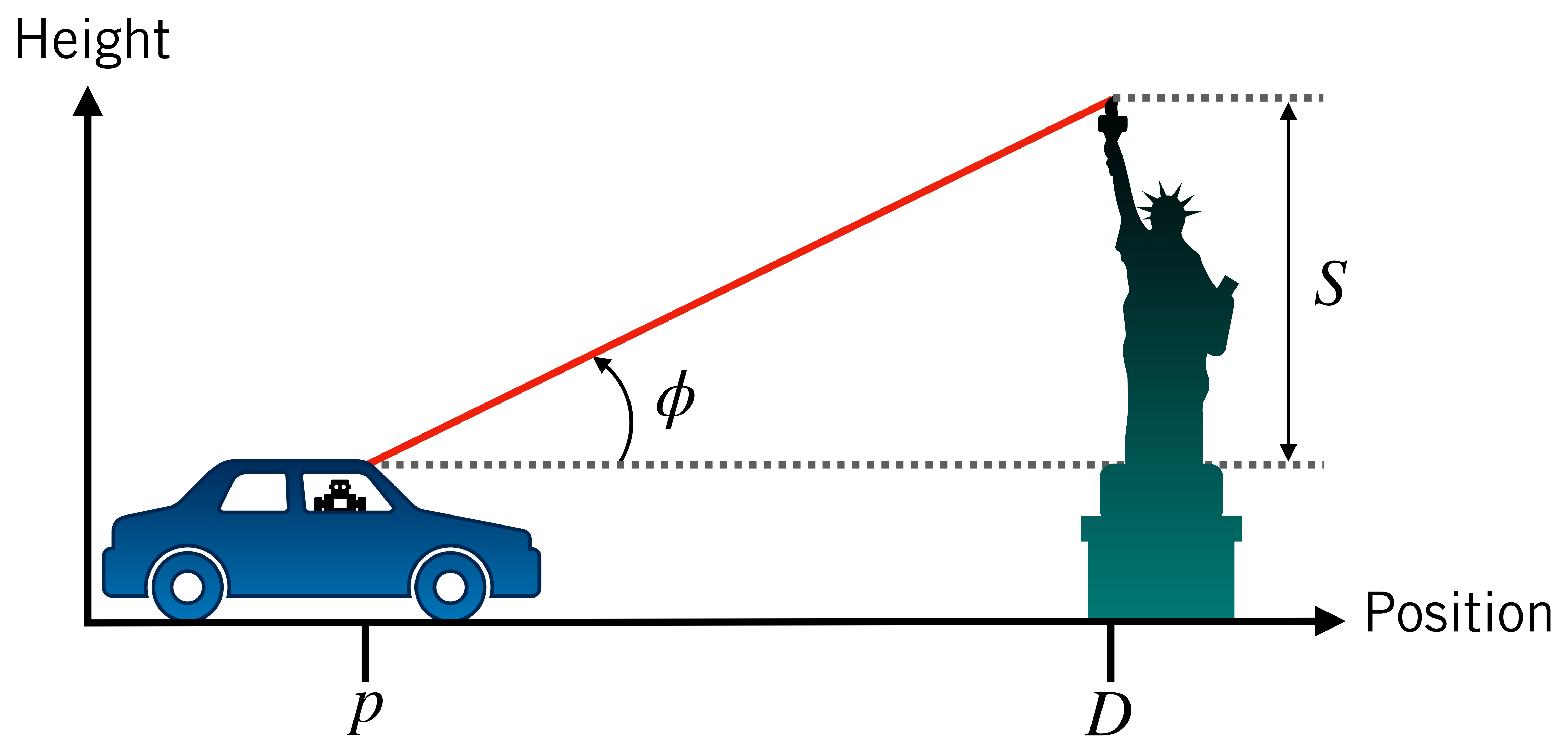
$$= \tan^{-1}\left(\frac{S}{D - p_k}\right) + v_k$$

Ex: Camera

Noise densities

$$v_k \sim \mathcal{N}(0,\ 0.01) \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, (0.1)\mathbf{1}_{2\times 2})$$

EKF | Short Example



Motion model Jacobians

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1},\mathbf{0}} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

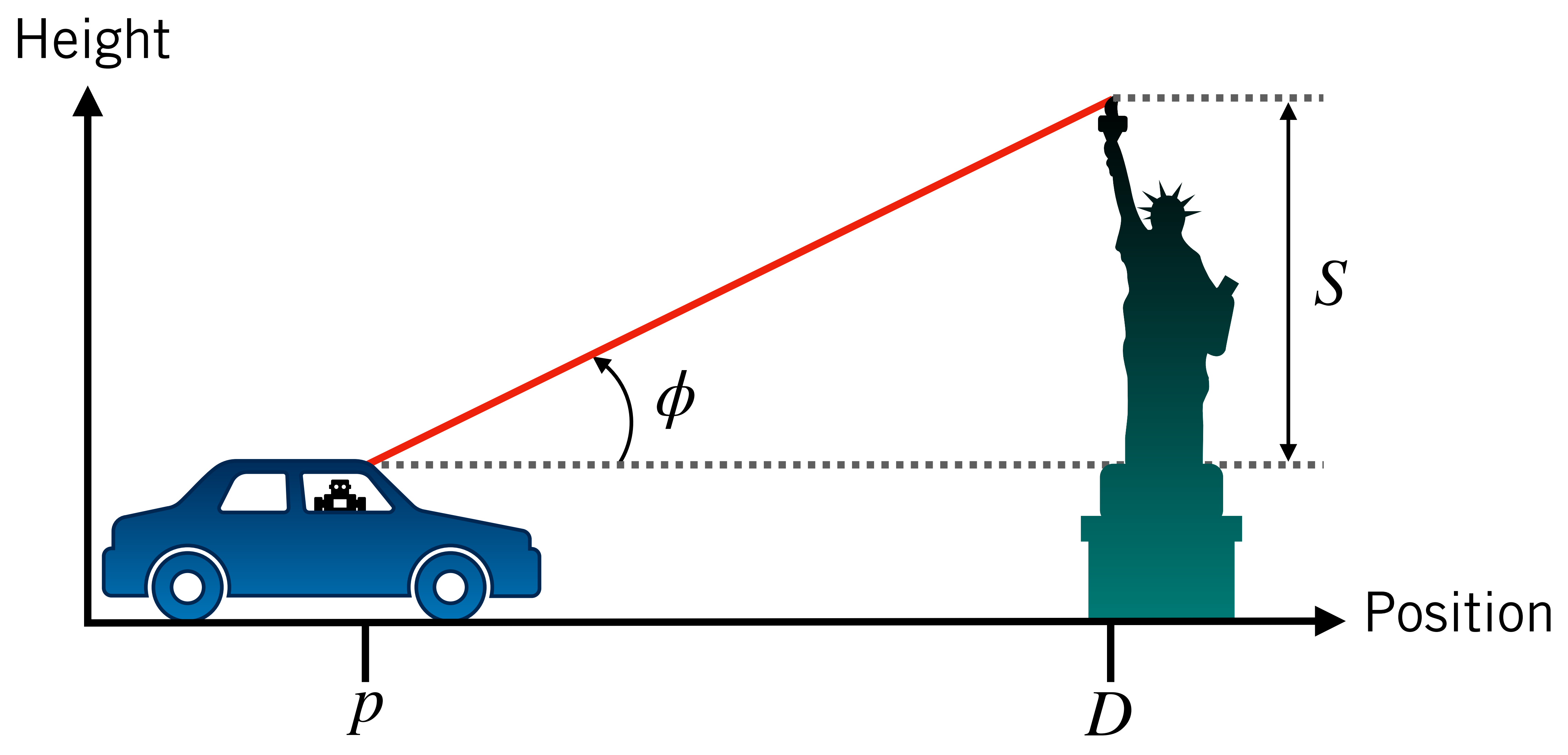
$$\mathbf{L}_{k-1} = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{w}_{k-1}}\right|_{\hat{\mathbf{x}}_{k-1},\mathbf{u}_{k-1},\mathbf{0}} = \mathbf{1}_{2\times 2}$$

Measurement model Jacobians

$$\mathbf{H}_k = \left.\frac{\partial h}{\partial \mathbf{x}_k}\right|_{\check{\mathbf{x}}_k,\mathbf{0}} = \begin{bmatrix} \frac{S}{(D-\check{p}_k)^2 + S^2} & 0 \end{bmatrix}$$

$$M_k = \left.\frac{\partial h}{\partial v_k}\right|_{\check{\mathbf{x}}_k,\mathbf{0}} = 1$$

EKF | Short Example



Data

$$\hat{\mathbf{x}}_0 \sim \mathcal{N}\left(\begin{bmatrix}0\\5\end{bmatrix}, \begin{bmatrix}0.01 & 0\\0 & 1\end{bmatrix}\right)$$

$$\Delta t = 0.5 \text{ s}$$

$$u_0 = -2\ [m/s^2] \qquad y_1 = \pi/6\ [\text{rad}]$$

$$S = 20\ [m] \qquad D = 40\ [m]$$

Using the Extended Kalman Filter equations, what is our updated position?

$$\hat{p}_1$$

EKF | Short Example Solution

Prediction

$$\check{\mathbf{x}}_1 = \mathbf{f}_0(\hat{\mathbf{x}}_0, \mathbf{u}_0, \mathbf{0})$$

$$\begin{bmatrix} \check{p}_1 \\ \check{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} (-2) = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix}$$

This is the same result as in the linear Kalman Filter example because the motion model is already linear!

$$\check{\mathbf{P}}_1 = \mathbf{F}_0 \hat{\mathbf{P}}_0 \mathbf{F}_0^T + \mathbf{L}_0 \mathbf{Q}_0 \mathbf{L}_0^T$$

$$\check{\mathbf{P}}_1 = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.01 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

EKF | Short Example Solution

Correction

$$\mathbf{K}_1 = \check{\mathbf{P}}_1\mathbf{H}_1^T(\mathbf{H}_1\check{\mathbf{P}}_1\mathbf{H}_1^T + \mathbf{M}_1\mathbf{R}_1\mathbf{M}_1^T)^{-1}$$

$$= \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 0.011 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 0.011 & 0 \end{bmatrix} \begin{bmatrix} 0.36 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} 0.011 \\ 0 \end{bmatrix} + 1(0.01)(1) \right)^{-1}$$

$$= \begin{bmatrix} 0.40 \\ 0.55 \end{bmatrix}$$

$$\hat{\mathbf{x}}_1 = \check{\mathbf{x}}_1 + \mathbf{K}_1(\mathbf{y}_1 - \mathbf{h}_1(\check{\mathbf{x}}_1, \mathbf{0}))$$

$$\begin{bmatrix} \hat{p}_1 \\ \hat{\dot{p}}_1 \end{bmatrix} = \begin{bmatrix} 2.5 \\ 4 \end{bmatrix} + \begin{bmatrix} 0.40 \\ 0.55 \end{bmatrix}(0.52 - 0.49) = \begin{bmatrix} 2.51 \\ 4.02 \end{bmatrix}$$

Bonus!

$$\hat{\mathbf{P}}_1 = (\mathbf{1} - \mathbf{K}_1\mathbf{H}_1)\check{\mathbf{P}}_1$$

$$= \begin{bmatrix} 0.36 & 0.50 \\ 0.50 & 1.1 \end{bmatrix}$$

ϕ Angle changes slowly didn't provide much information

Summary | Extended Kalman Filter (EKF)

- The EKF uses *linearization* to adapt the Kalman filter to nonlinear systems
- Linearization works by computing a local linear approximation to a nonlinear function about a chosen operating point
- Linearization relies on computing *Jacobian matrices*, which contain all the first-order partial derivatives of a function

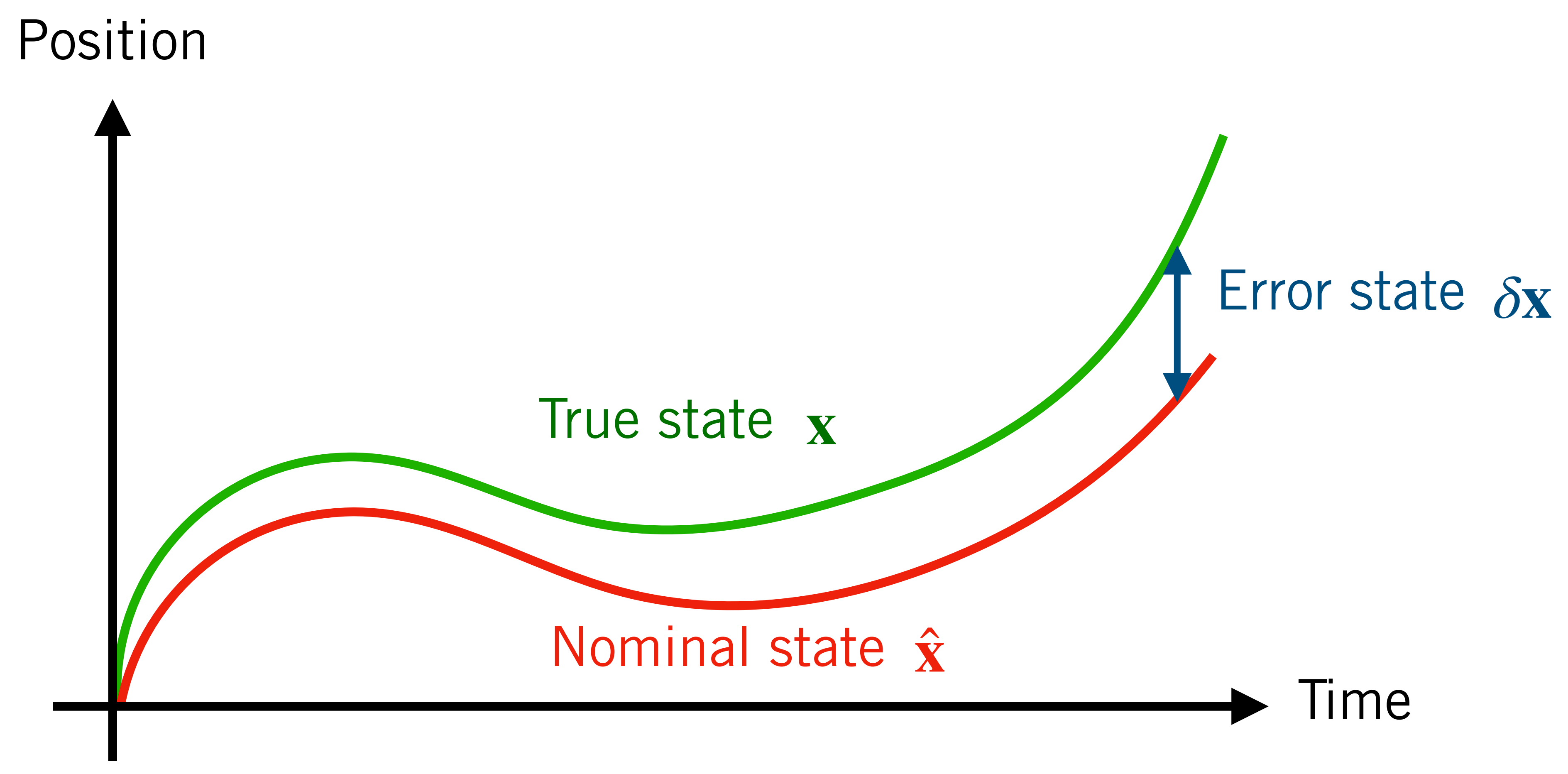
MODULE 2 LESSON 4

AN IMPROVED EKF: THE ERROR-STATE EXTENDED KALMAN

What’s in a State?

We can think of the vehicle state as composed of two parts:

$$\underset{\text{True State}}{\mathbf{x}} = \underset{\substack{\text{Nominal State}\\(\text{“Large”})}}{\hat{\mathbf{x}}} + \underset{\substack{\text{Error State}\\(\text{“Small”})}}{\delta\mathbf{x}}$$


Position
Error state $\delta\mathbf{x}$
True state $\mathbf{x}$
Nominal state $\hat{\mathbf{x}}$
Time

- We can continuously update the *nominal state* by integrating the motion model
- Modelling errors and process noise accumulate into the *error state*

The Error-State Extended Kalman Filter

The Error-State Extended Kalman Filter estimates the error state directly and uses it as a correction to the nominal state:

Linearized motion model

$$\mathbf{x}_k = \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0}) + \mathbf{F}_{k-1}\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}\right) + \mathbf{L}_{k-1}\mathbf{w}_{k-1}$$

$$\Downarrow$$

$$\underbrace{\mathbf{x}_k - \mathbf{f}_{k-1}(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}, \mathbf{0})}_{\delta\mathbf{x}_k} = \mathbf{F}_{k-1}\underbrace{\left(\mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}\right)}_{\delta\mathbf{x}_{k-1}} + \mathbf{L}_{k-1}\mathbf{w}_{k-1}$$

Error state

Linearized measurement model

$$\mathbf{y}_k = \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \mathbf{H}_k\left(\mathbf{x}_k - \check{\mathbf{x}}_k\right) + \mathbf{M}_k\mathbf{v}_k$$

$$\Downarrow$$

$$\mathbf{y}_k = \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}) + \mathbf{H}_k\underbrace{\left(\mathbf{x}_k - \check{\mathbf{x}}_k\right)}_{\delta\mathbf{x}_k} + \mathbf{M}_k\mathbf{v}_k$$

Error state

The Error-State Extended Kalman Filter

Loop:

1. Update nominal state with motion model

$$\check{\mathbf{x}}_k = \mathbf{f}_{k-1}(\underset{\uparrow}{\mathbf{x}_{k-1}}, \mathbf{u}_{k-1}, \mathbf{0})$$

This could be $\check{\mathbf{x}}_{k-1}$ or $\hat{\mathbf{x}}_{k-1}$

The Error-State Extended Kalman Filter

Loop:

1. Update nominal state with motion model
2. Propagate uncertainty

$$\check{\mathbf{P}}_k = \mathbf{F}_{k-1}\underset{\uparrow}{\mathbf{P}_{k-1}}\mathbf{F}_{k-1}^T + \mathbf{L}_{k-1}\mathbf{Q}_{k-1}\mathbf{L}_{k-1}^T$$

This could be $\check{\mathbf{P}}_{k-1}$ or $\hat{\mathbf{P}}_{k-1}$

The Error-State Extended Kalman Filter

Loop:

1. Update nominal state with motion model
2. Propagate uncertainty
3. If a measurement is available:
    1. Compute Kalman Gain

$$\mathbf{K}_k = \check{\mathbf{P}}_k \mathbf{H}_k^T (\mathbf{H}_k \check{\mathbf{P}}_k \mathbf{H}_k^T + \mathbf{R})^{-1}$$

The Error-State Extended Kalman Filter

Loop:

1. Update nominal state with motion model
2. Propagate uncertainty
3. If a measurement is available:
    1. Compute Kalman Gain
    2. Compute error state

$$\delta\hat{\mathbf{x}}_k = \mathbf{K}_k(\mathbf{y}_k - \mathbf{h}_k(\check{\mathbf{x}}_k, \mathbf{0}))$$

The Error-State Extended Kalman Filter

Loop:

1. Update nominal state with motion model
2. Propagate uncertainty
3. If a measurement is available:
    1. Compute Kalman Gain
    2. Compute error state
    3. Correct nominal state

$$\hat{\mathbf{x}}_k = \check{\mathbf{x}}_k + \delta\hat{\mathbf{x}}_k$$

The Error-State Extended Kalman Filter

Loop:

1. Update nominal state with motion model
2. Propagate uncertainty
3. If a measurement is available:
    1. Compute Kalman Gain
    2. Compute error state
    3. Correct nominal state
    4. Correct state covariance

$$\hat{\mathbf{P}}_k = (\mathbf{1} - \mathbf{K}_k\mathbf{H}_k)\check{\mathbf{P}}_k$$

Why Use the ES-EKF?

1. **Better performance compared to the vanilla EKF**
   The "small" error state is more amenable to linear filtering than the "large" nominal state, which can be integrated nonlinearly

2. **Easy to work with constrained quantities (e.g., rotations in 3D)**
   We can also break down the state using a generalized composition operator

$$\underset{\text{True State}}{\mathbf{x}} = \underset{\substack{\text{Nominal State} \\ \text{(Overparametrized,} \\ \text{constrained)}}}{\hat{\mathbf{x}}} \oplus \underset{\substack{\text{Error State} \\ \text{(Minimal parametrization,} \\ \text{unconstrained)}}}{\delta\mathbf{x}}$$

Summary | The Error-State EKF (ES-EKF)

- The error-state formulation separates the state into a “large” nominal state and a “small” error state.
- The ES-EKF uses local linearization to estimate the error state and uses it to correct the nominal state.
- The ES-EKF can perform better than the vanilla EKF, and provides a natural way to handle constrained quantities like rotations in 3D.

Error state is closer to linear.

MODULE 2 LESSON 5

LIMITATIONS OF THE EXTENDED KALMAN FILTER

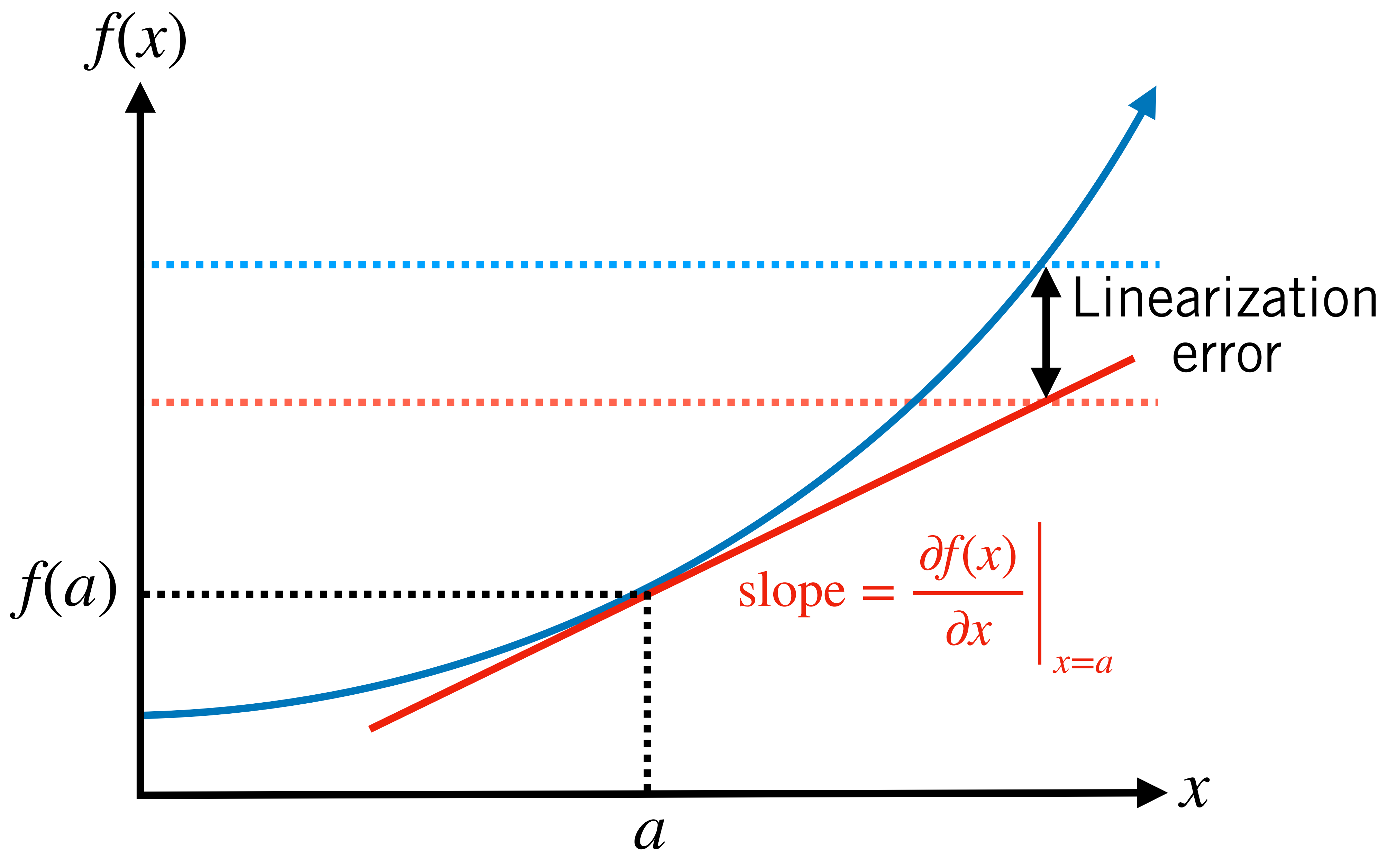
Limitations of the EKF | Linearization error

The EKF works by *linearizing* the nonlinear motion and measurement models to update the mean and covariance of the state

The difference between the linear approximation and the nonlinear function is called *linearization error*

In general, linearization error depends on

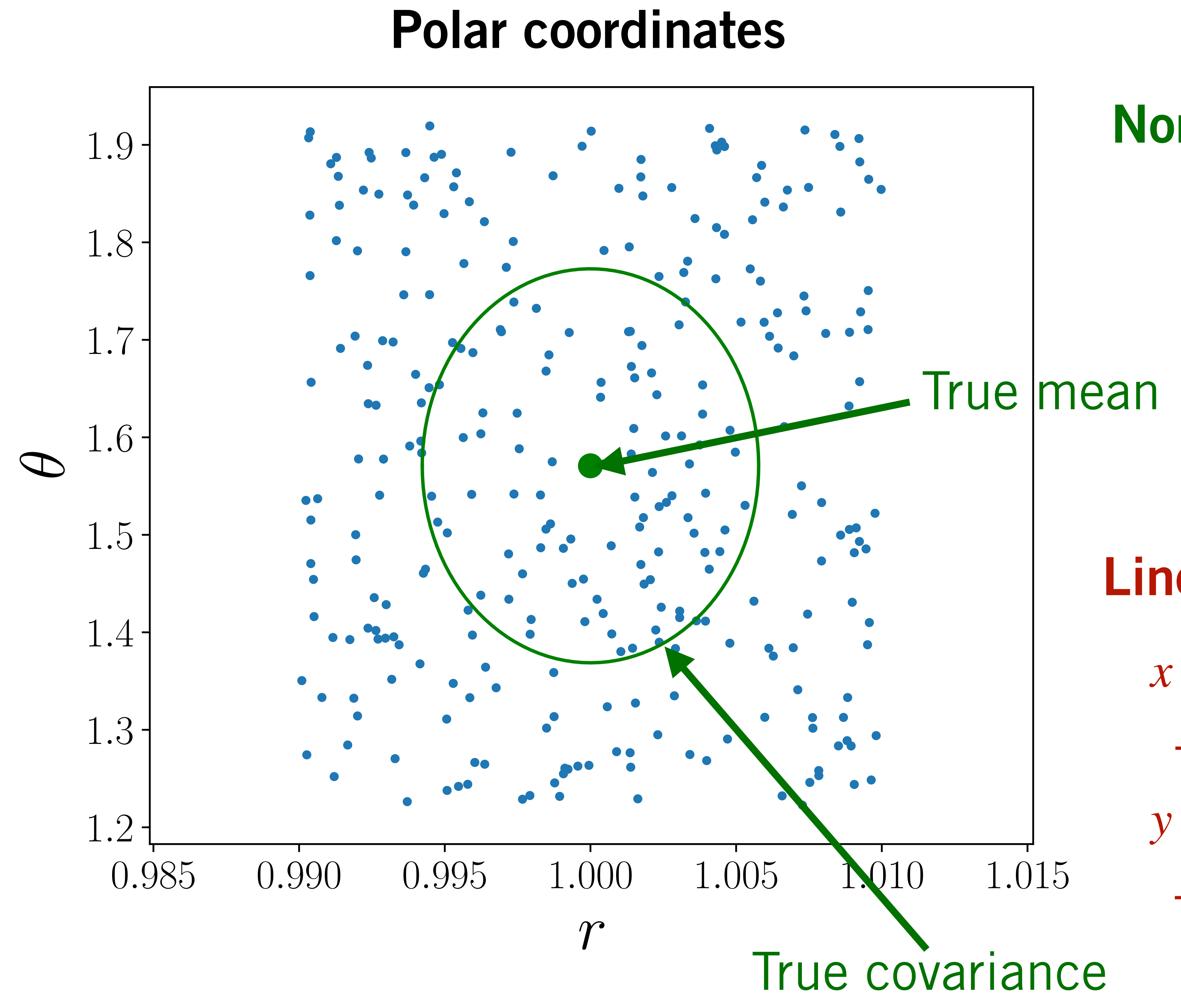
1. How nonlinear the function is
2. How far away from the operating point the linear approximation is being used


$f(x)$
Linearization error
$f(a)$
$\text{slope} = \left.\frac{\partial f(x)}{\partial x}\right|_{x=a}$
x
a

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

Linearization Error | Example

Let’s look at an example of how linearization error affects the mean and covariance of a random variable transformed by a nonlinear function:

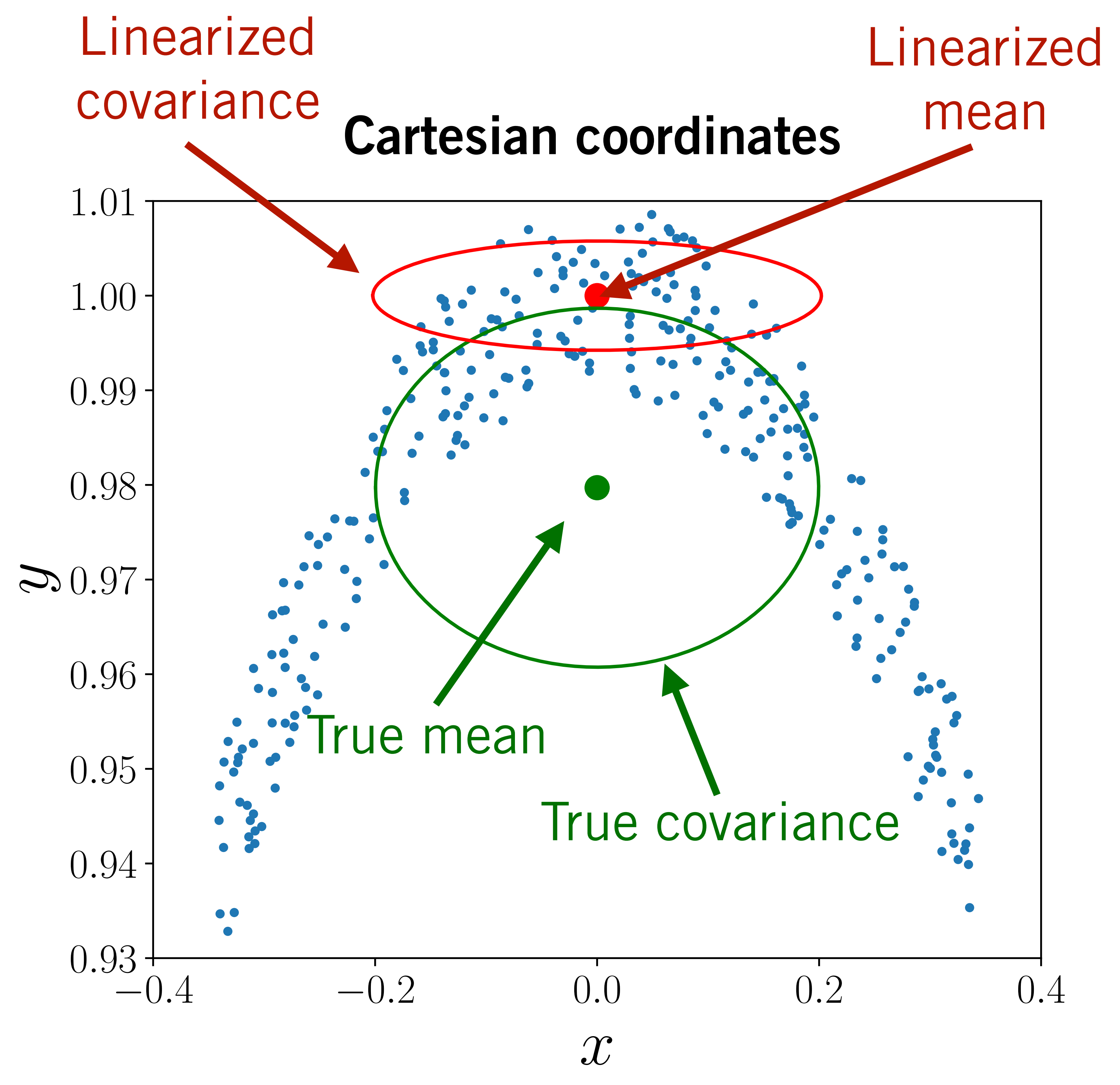


Nonlinear transformation

$$x = r\cos\theta$$
$$y = r\sin\theta$$

Linearized transformation

$$x \approx \bar{r}\cos\bar{\theta} + \cos\bar{\theta}\,(r-\bar{r}) - \bar{r}\sin\bar{\theta}\,(\theta-\bar{\theta})$$
$$y \approx \bar{r}\sin\bar{\theta} + \sin\bar{\theta}\,(r-\bar{r}) + \bar{r}\cos\bar{\theta}\,(\theta-\bar{\theta})$$



Limitations of the EKF | Linearization Error

The EKF is prone to linearization error when

1. The system dynamics are highly nonlinear
2. The sensor sampling time is slow relative how fast the system is evolving

How far away from the operating point

This has two important consequences:

1. The estimated mean state can become very different from the true state
2. The estimated state covariance can fail to capture the true uncertainty in the state

Linearization error can cause the estimator to be overconfident in a wrong answer!

Limitations of the EKF | Computing Jacobians

Computing Jacobian matrices for complicated nonlinear functions is also a common source of error in EKF implementations!

- Analytical differentiation is prone to human error
- Numerical differentiation can be slow and unstable
- Automatic differentiation (e.g., at compile time) can also behave unpredictably

What if one or more of our models is non-differentiable?

Do we really need linearization for nonlinear Kalman filtering?

Summary | Limitations of the EKF

- The EKF uses analytical local linearization and, as a result, is sensitive to linearization errors
- For highly nonlinear systems, the EKF estimate can *diverge* and become unreliable
- Computing complex Jacobian matrices is an error-prone process and must be done with substantial care