



NYU

# Introduction to Robot Intelligence

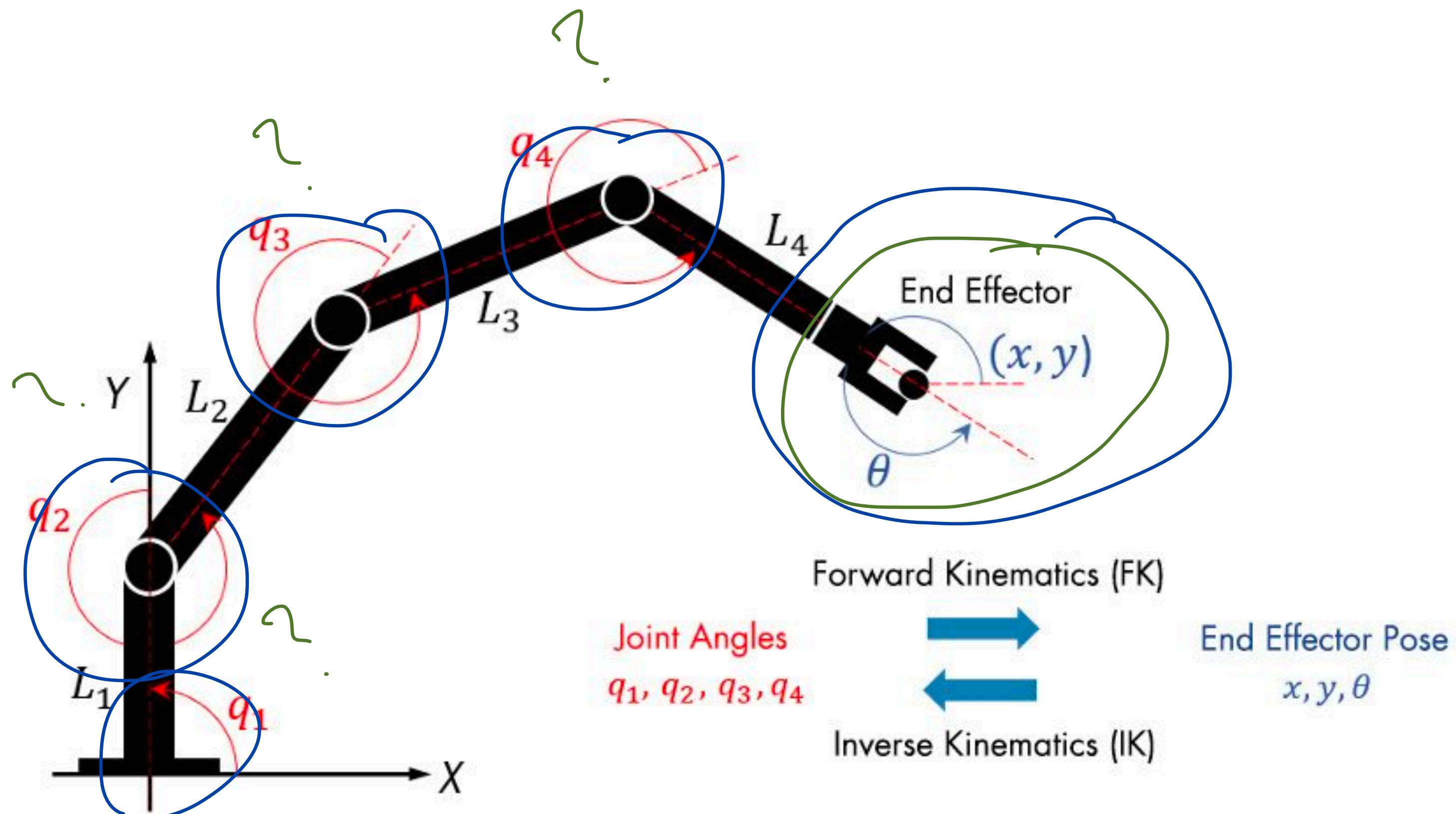
## [Spring 2023]

# Inverse Kinematics (Part 2)

March 7, 2023

Lerrel Pinto

# FK vs. IK



# Solving IK for simple mechanisms

Two linked planar robot

$$x_{des} = l_1 c_1 + l_2 c_{12}$$

$$y_{des} = l_1 s_1 + l_2 s_{12}$$

$$\rightarrow \tilde{x}_{des} = l_1 c_1 + l_2 c_1 \cancel{c_2} - l_2 s_1 \cancel{s_2}$$

$$\tilde{x}_{des} = (l_1 + l_2 c_1) \cancel{c_1} - l_2 s_2 \cancel{s_1} \quad \text{--- ③.}$$

$$\tilde{y}_{des} = l_1 s_1 + l_2 s_1 \cancel{c_2} + l_2 s_2 \cancel{c_1}$$

$$\tilde{y}_{des} = (l_1 + l_2 s_1) \cancel{s_1} + l_2 s_2 \cancel{c_1} \quad \text{--- ④}$$

$$\begin{bmatrix} l_1 + l_2 c_2 & -l_2 s_2 \\ l_2 s_2 & l_1 + l_2 s_2 \end{bmatrix} \begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = \begin{bmatrix} \tilde{x}_{des} \\ \tilde{y}_{des} \end{bmatrix} \rightarrow \begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = A^{-1} \begin{bmatrix} \tilde{x}_{des} \\ \tilde{y}_{des} \end{bmatrix}$$

$$q_1 = \text{atan2}(c_1, s_1)$$

Trigonometric identities:

$$\cos(A+B) = \cos A \cos B - \sin A \sin B$$

$$\sin(A+B) = \sin A \cos B + \cos A \sin B$$

Two linked planar robot

$$(x_{des}, y_{des})$$

$$x_{des} = l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \quad \text{--- ①}$$

$$y_{des} = l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \quad \text{--- ②}$$

$$\begin{aligned} x_{des}^2 + y_{des}^2 &= l_1^2 \left( \frac{s_1^2 + c_1^2}{=1} \right) + l_2^2 \left( \frac{c_{12}^2 + s_{12}^2}{=1} \right) + 2 l_1 l_2 c_1 c_{12} + 2 l_1 l_2 s_1 s_{12} \\ &= l_1^2 + l_2^2 + 2 l_1 l_2 (c_1 c_{12} + s_1 s_{12}) \end{aligned}$$

$$\Rightarrow \cos q_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

$$\sin q_2 = \pm \sqrt{1 - \cos^2 q_2}$$

$$q_2 = \text{atan2}(\cos q_2, \sin q_2)$$

Trigonometric identities:

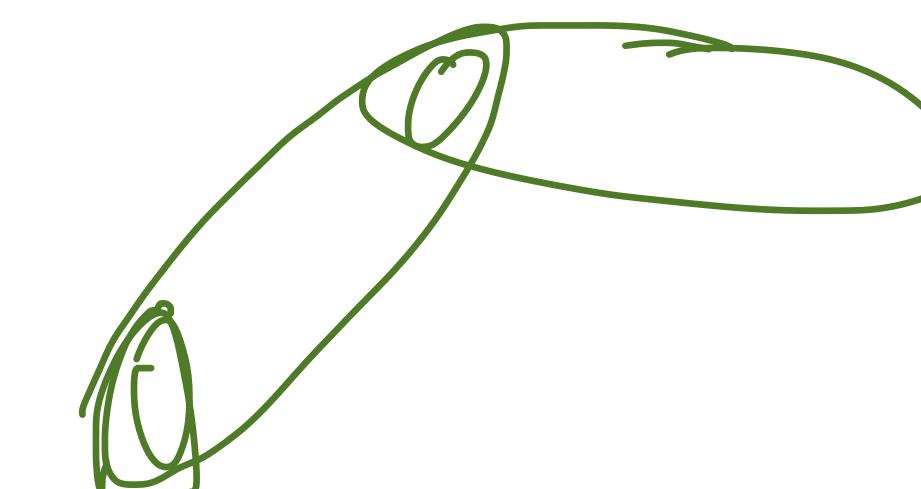
$$\cos q_1 \rightarrow c_1$$

$$\sin q_1 \rightarrow s_1$$

$$\cos(q_1 + q_2) \rightarrow c_{12}$$

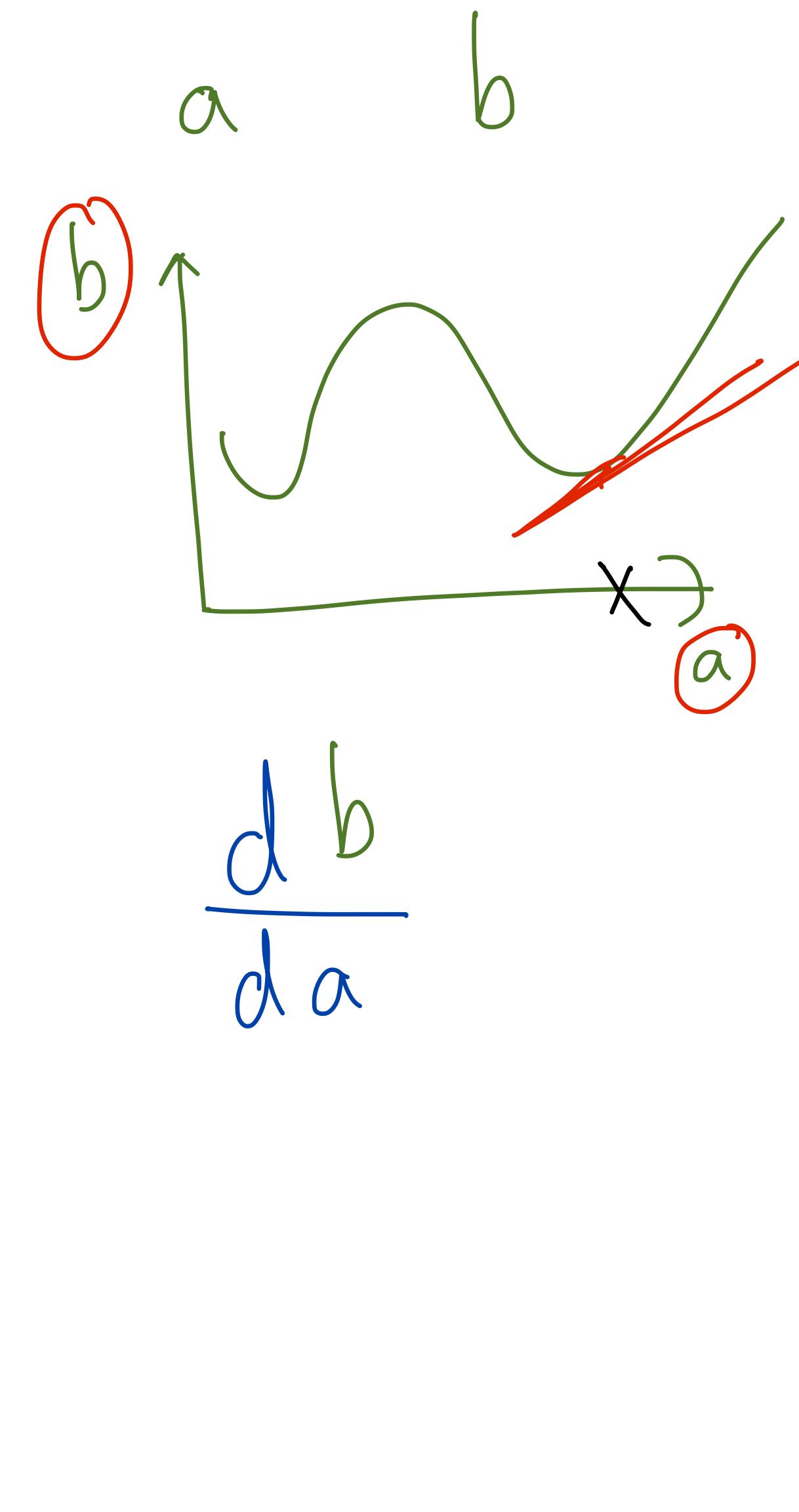
$$\sin(q_1 + q_2) \rightarrow s_{12}$$

$$\begin{aligned} \cos(A+B) &= \cos A \cos B - \sin A \sin B \\ \sin(A+B) &= \sin A \cos B + \cos A \sin B \end{aligned}$$



# Solution 2b: Numerical Methods (Gradient Descent)

# New Concept – The Jacobian



$\vec{b} = [b_1, b_2, \dots, b_n]$

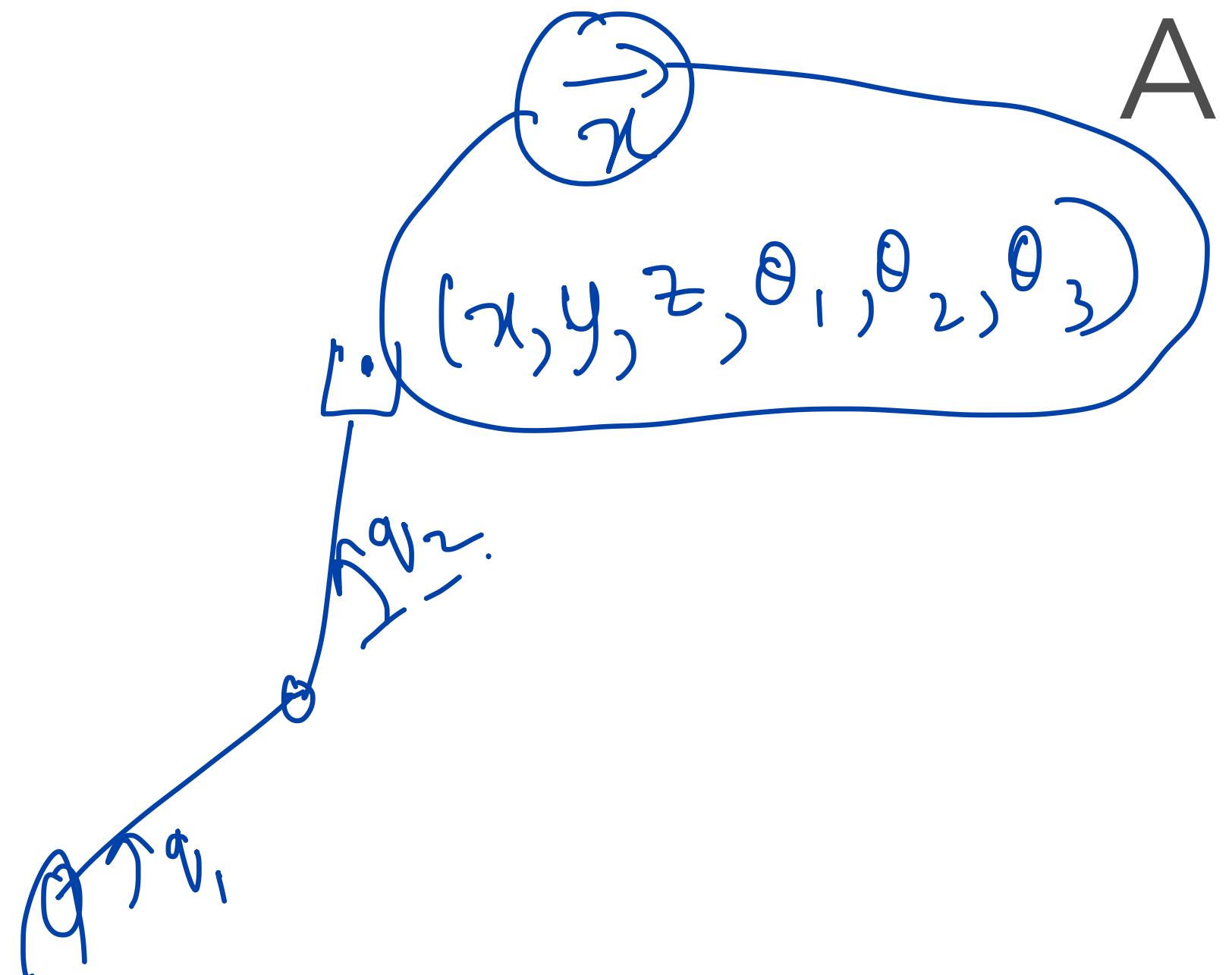
$$\frac{d \vec{b}}{d \vec{a}} = \begin{bmatrix} \frac{d b_1}{d a} \\ \frac{d b_2}{d a} \\ \vdots \\ \frac{d b_n}{d a} \end{bmatrix}$$

$\vec{a} = [a_1, a_2, \dots, a_m]$

$$\frac{d \vec{b}}{d \vec{a}} = \begin{bmatrix} \frac{\partial b_1}{\partial a_1} & \frac{\partial b_1}{\partial a_2} & \dots & \dots & \frac{\partial b_1}{\partial a_m} \\ \frac{\partial b_2}{\partial a_1} & \frac{\partial b_2}{\partial a_2} & \dots & \dots & \frac{\partial b_2}{\partial a_m} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \frac{\partial b_n}{\partial a_1} & \frac{\partial b_n}{\partial a_2} & \dots & \dots & \frac{\partial b_n}{\partial a_m} \end{bmatrix}_{n \times m}$$

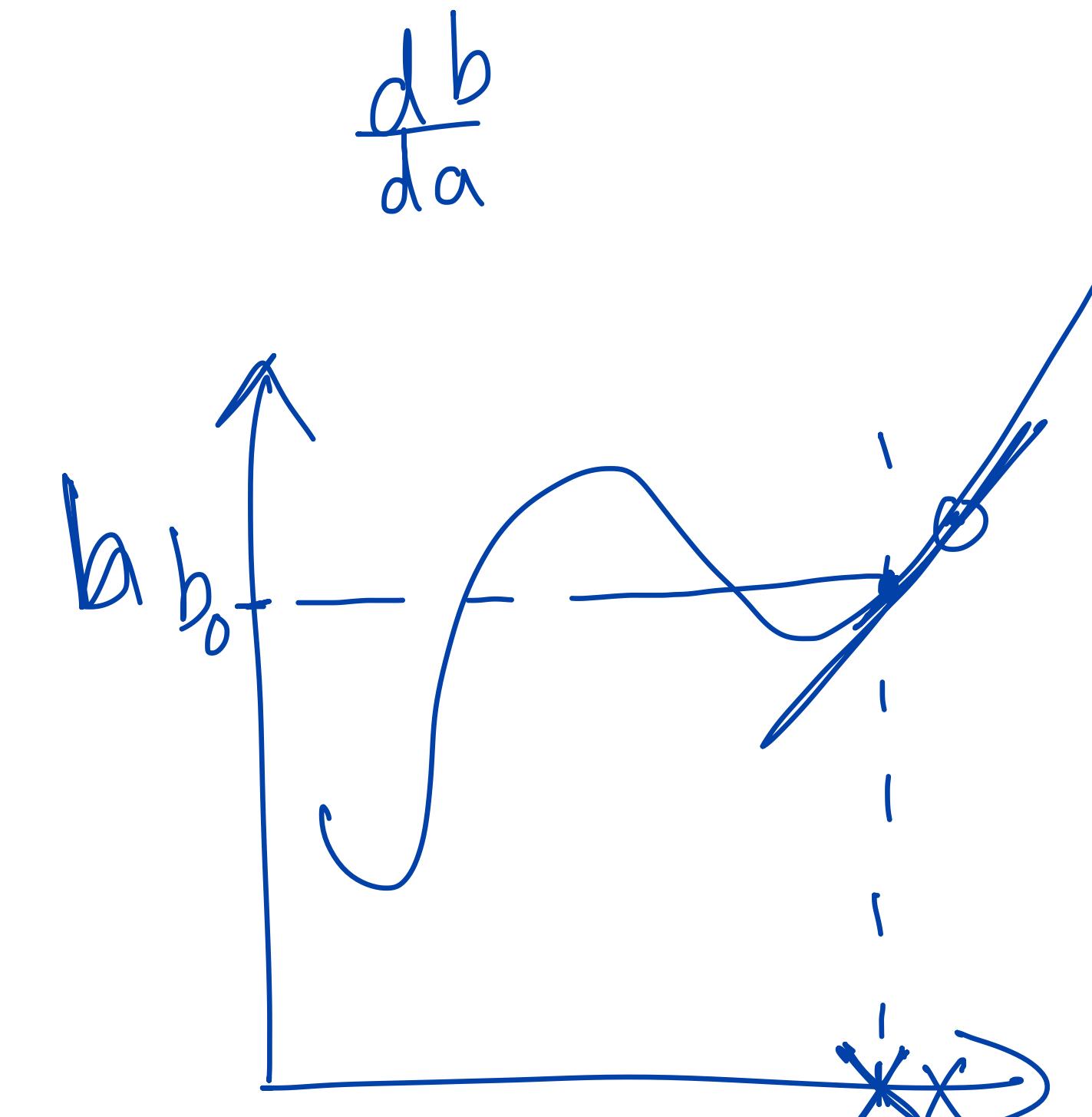
$$= \underline{\underline{J(\vec{b}, \vec{a})}}$$

# A robot's Jacobian



$$J(\vec{x}, \vec{q})$$

$$\frac{d\vec{x}}{d\vec{q}}$$

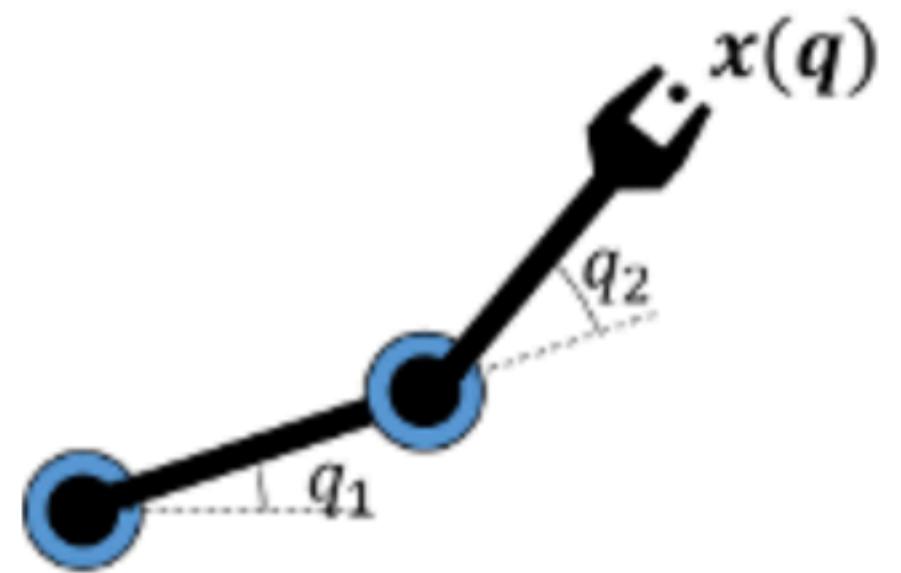


one D variable

$$b(a_0 + \Delta a) = b_0 + \frac{db}{da} \cdot \Delta a$$

$x \rightarrow n - D$  var  
 $q \rightarrow m - D$  var

# A robot's Jacobian



$$\vec{q} = \begin{bmatrix} q_1, q_2 \end{bmatrix}$$
$$\vec{x} = \begin{bmatrix} a, b \end{bmatrix}$$

Diagram illustrating the mapping from joint angles  $q_1$  and  $q_2$  to Cartesian coordinates  $a$  and  $b$ .

Image credits: Kris Hauser.

# A robot's Jacobian

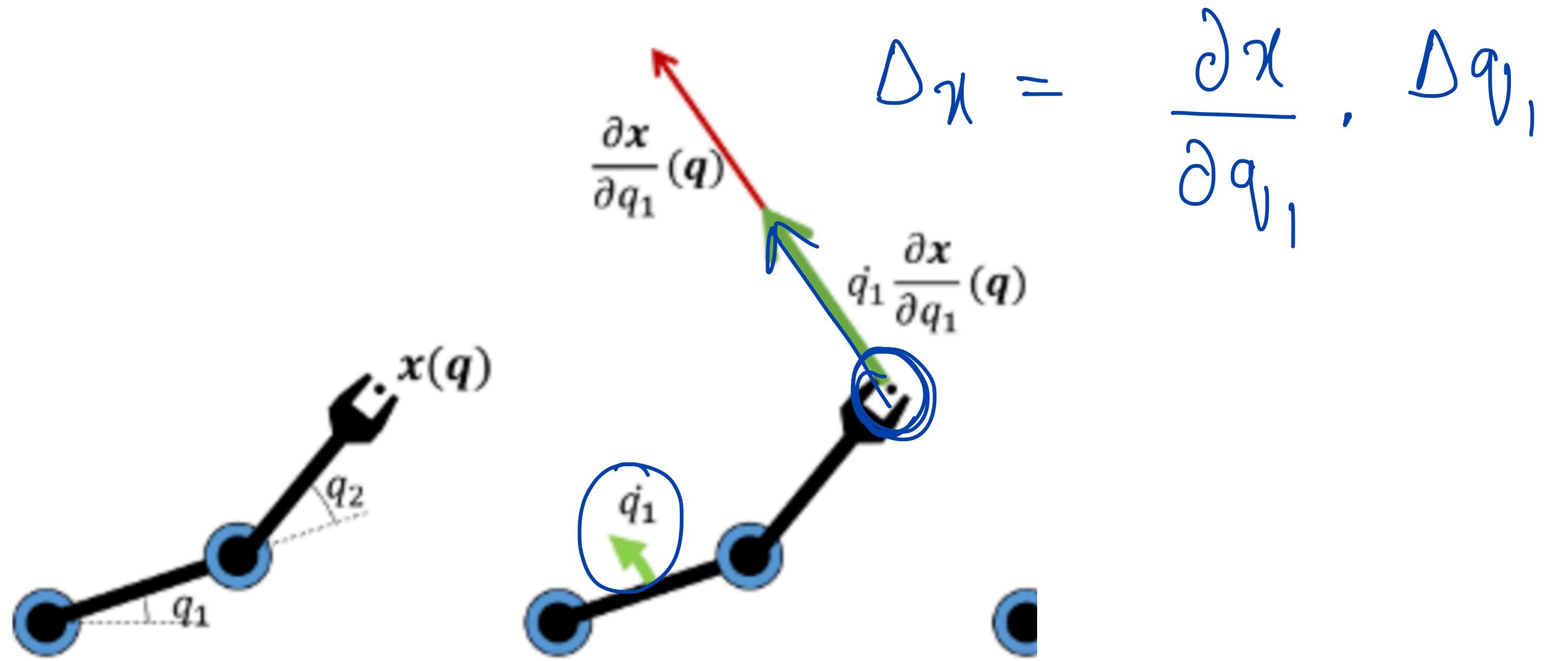


Image credits: Kris Hauser.

# A robot's Jacobian

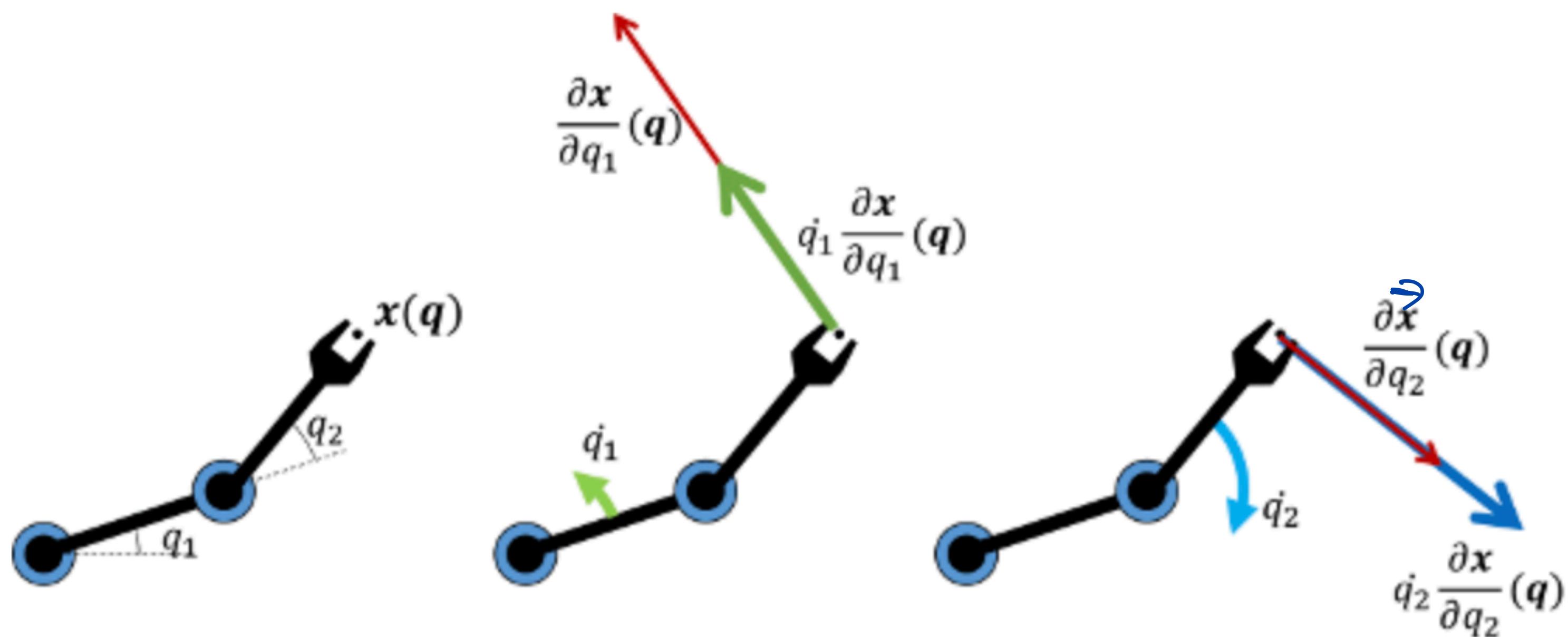
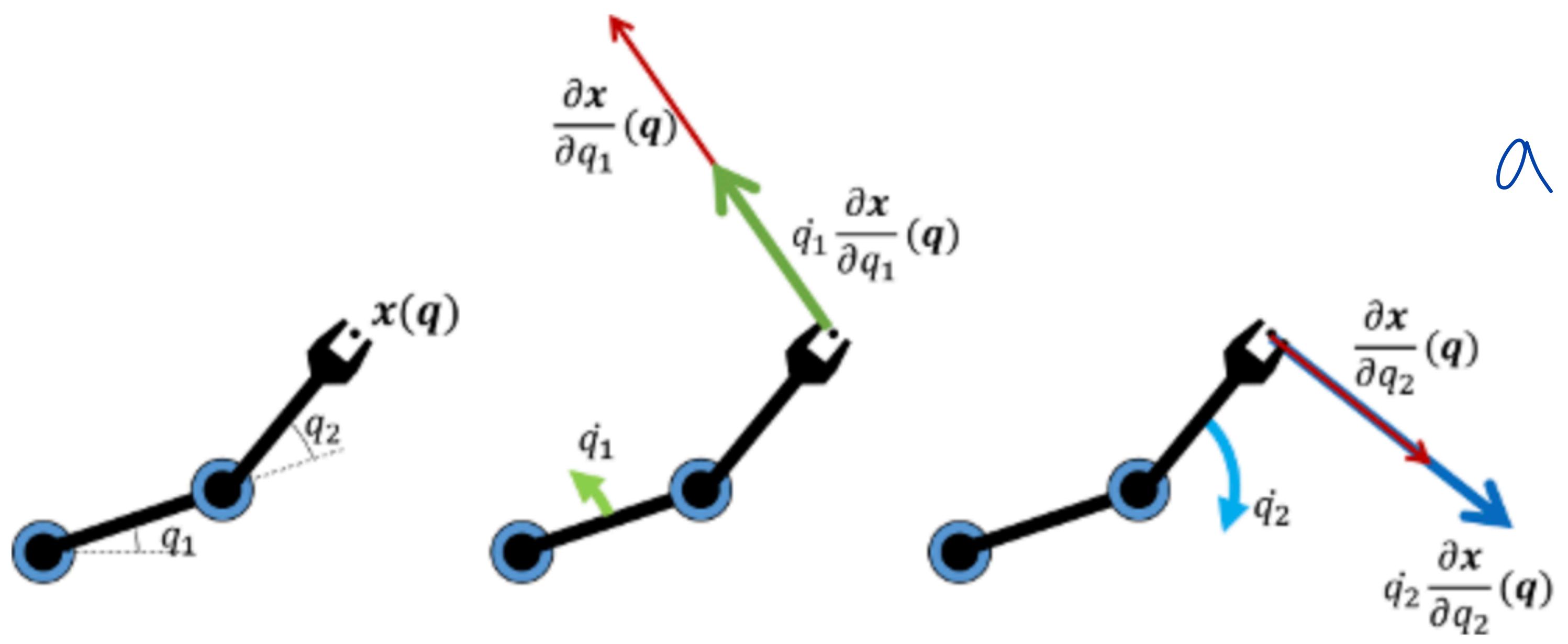


Image credits: Kris Hauser.

# A robot's Jacobian



$$\begin{bmatrix} \Delta a \\ \Delta b \end{bmatrix} = J \begin{bmatrix} \Delta q_1 \\ \Delta q_2 \end{bmatrix}$$

Diagram illustrating the Jacobian matrix  $J(q) \dot{q}$  relating joint space velocities to Cartesian velocities. The diagram shows a robot arm with joints  $q_1$  and  $q_2$  moving from state  $a$  to state  $b$ . The Jacobian matrix  $J(q) \dot{q}$  is circled in blue.

**Figure 5.** The Jacobian relates joint space velocities to Cartesian velocities. The first column of the Jacobian gives the velocity at which the end effector would move if the first joint moves at unit speed, and the second column gives its velocity if the second joint moves at unit velocity. The linear combination of Jacobian columns, weighted by joint speed, gives the overall Cartesian velocity.

Image credits: Kris Hauser.

# Recall: Gradient Descent Algorithm with multiple params

# Recall: Gradient Descent Algorithm with multiple params

$$\vec{\theta}$$

- Given: cost / loss/ objective function  $f(\vec{\theta})$ . Where  $\vec{\theta} \in \mathbb{R}^d$ .

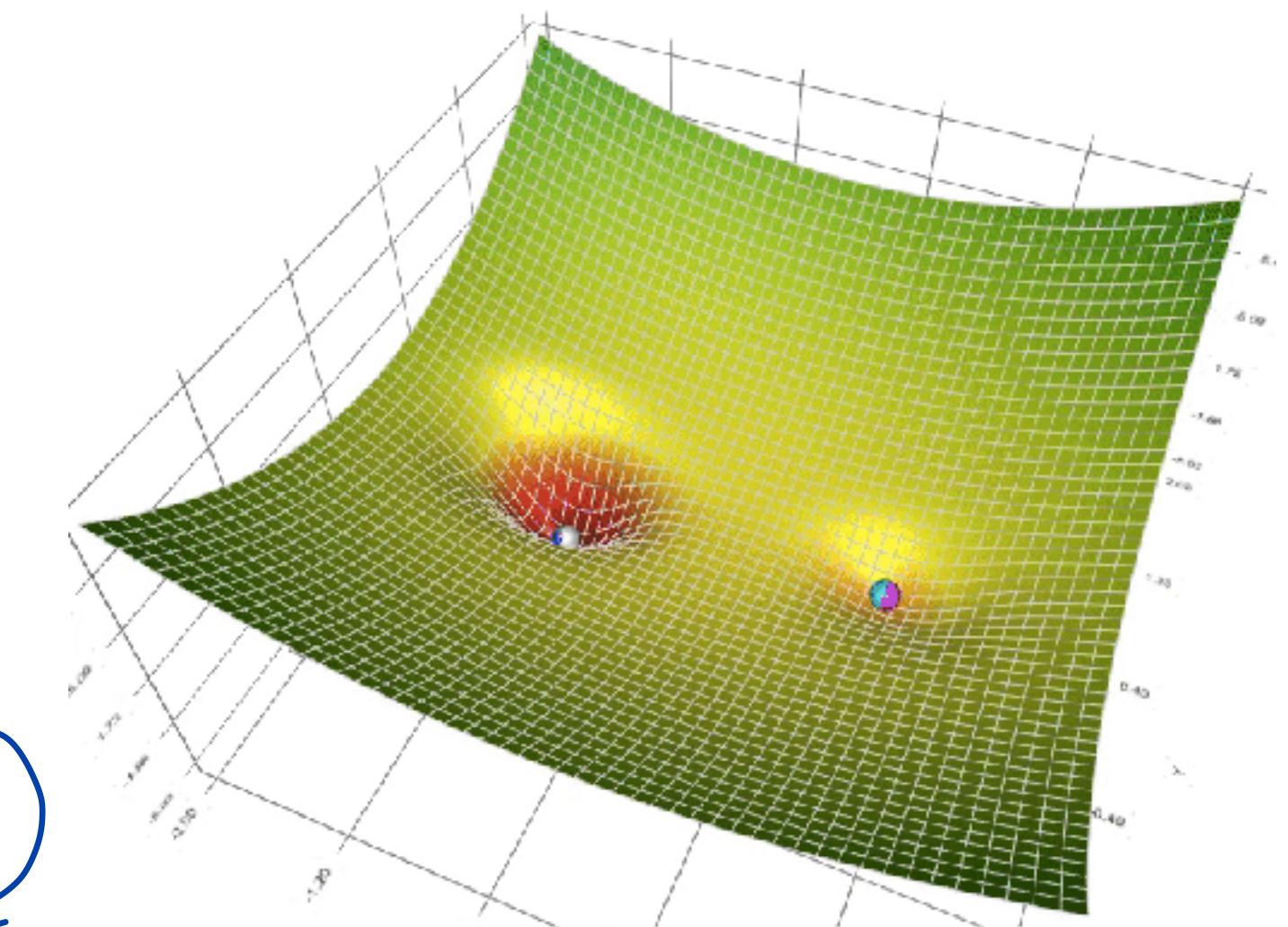
- Goal: find  $\vec{\theta}^*$  such that  $f(\vec{\theta}^*) = \min_{\vec{\theta}} f(\vec{\theta})$ .

→ • Gradient descent solution:

→ • Start from initial guess  $\vec{\theta}^0$  and learning rate  $\alpha$

→ • Update  $\vec{\theta}^{i+1} \leftarrow \vec{\theta}^i - \alpha \nabla f(\vec{\theta})$

→ • Repeat until change in  $\theta$  is small, or maximum number of steps reached.

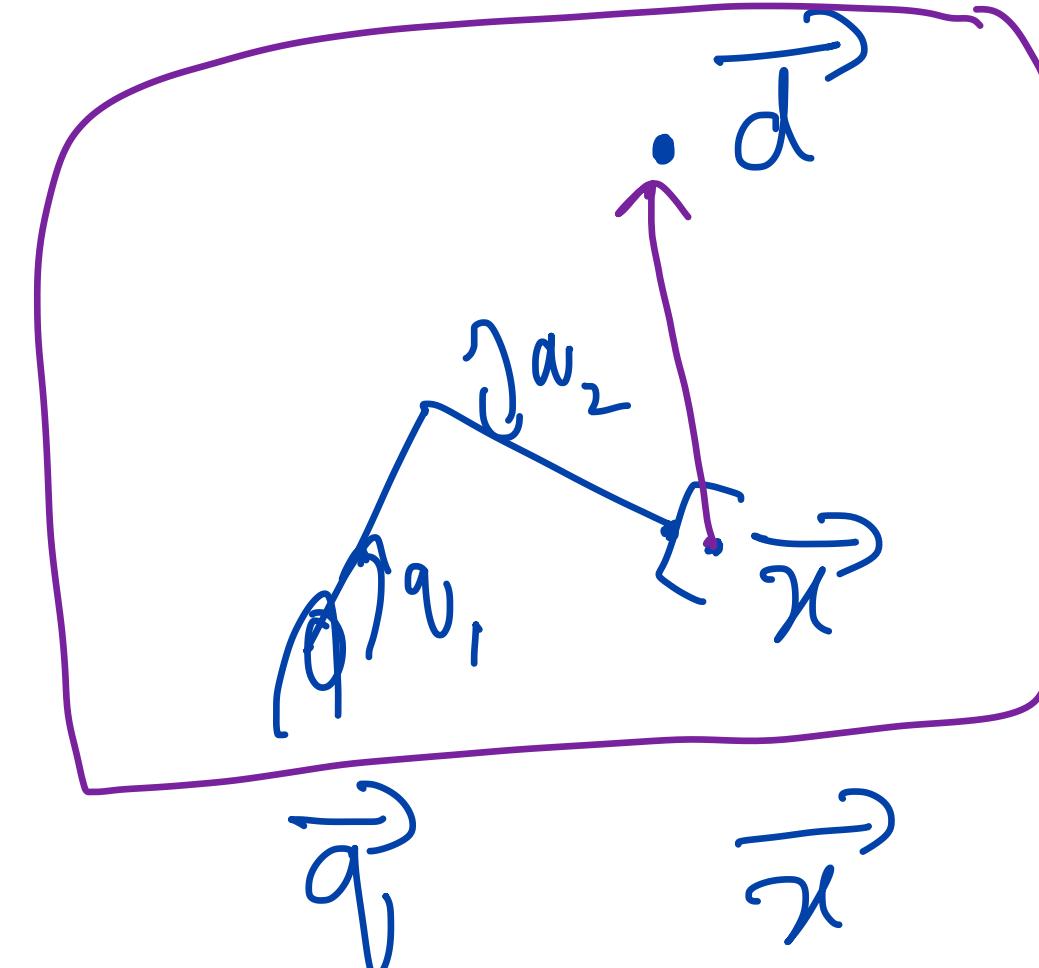


$$\frac{d^2 f(y)}{dy^2} = 2 f(y) \cdot \frac{dy}{dx}$$

$$f(g(x))' = f'(g(x)) g'(x)$$

## Gradient Descent for IK

Vanilla Gradient Descent



$$\vec{x} \leftarrow F.K(\vec{q}) \quad | \quad \vec{x}(\vec{q})$$

$$\min. f(\vec{q}) = \| \vec{x}(\vec{q}) - \vec{d} \|^2 \quad \| \vec{a} \|^2 = \vec{a}^T \vec{a}$$

Objective:

$$\rightarrow \text{Initial} \quad \vec{q}_0$$

$$\rightarrow \text{Update} \quad \vec{q}_1 \leftarrow \vec{q}_0 - \alpha \nabla_{\vec{q}} f(\vec{q})$$

$\rightarrow$  Repeat until convergence

$$\begin{aligned}
 f(\vec{q}) &= (\vec{x}(\vec{q}) - \vec{d})^T (\vec{x}(\vec{q}) - \vec{d}) \\
 \nabla_{\vec{q}} f(\vec{q}) &= 2 \frac{d \vec{x}}{d \vec{q}} (\vec{x}(\vec{q}) - \vec{d}) \\
 &= 2 J^T (\vec{x}(\vec{q}) - \vec{d}) \\
 &= \begin{pmatrix} 2 & \left[ \begin{array}{c|c|c|c} \frac{\partial x_1}{\partial q_1} & \frac{\partial x_2}{\partial q_1} & \cdots & \frac{\partial x_n}{\partial q_1} \\ \hline \frac{\partial x_1}{\partial q_m} & \frac{\partial x_2}{\partial q_m} & \cdots & \frac{\partial x_n}{\partial q_m} \end{array} \right] & \left[ \begin{array}{c} e_1 \\ e_2 \\ \vdots \\ e_n \end{array} \right] \end{pmatrix}
 \end{aligned}$$

$$Ax = b$$

$$x = A^{-1}b$$

# Gradient Descent for IK

Jacobian-pseudoinverse gradient descent

$$J(\vec{x}, \vec{q})$$

$$\Delta x = J \Delta q$$

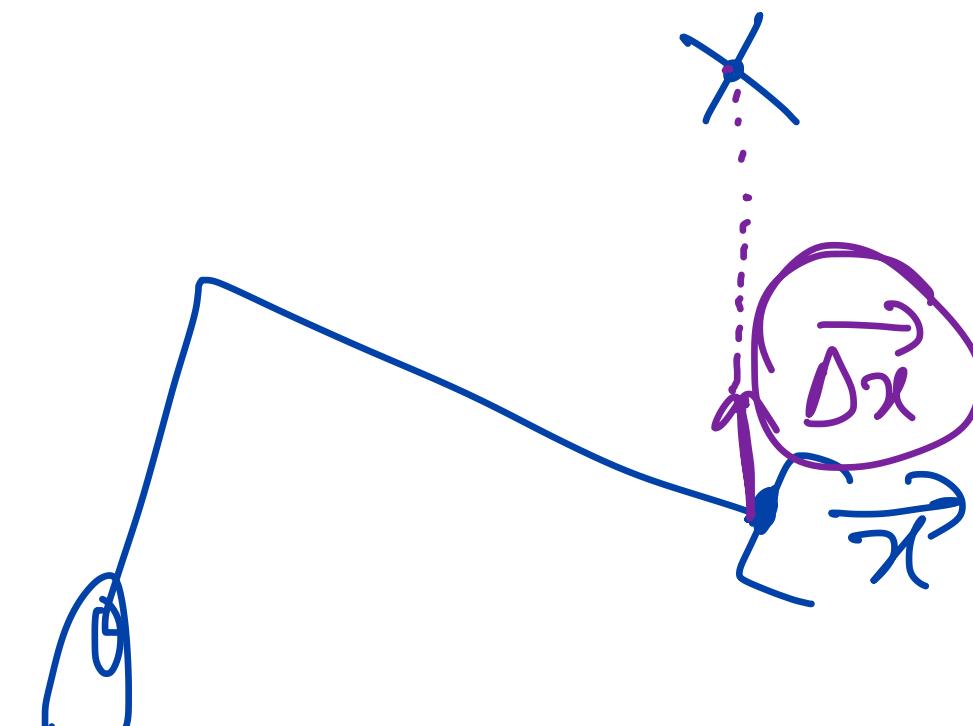
$n \times m$

$$\frac{db}{da}$$

$$\Delta b = \frac{db}{da} \Delta a$$

$$\Delta x_d = J \Delta q_d$$

$$\begin{aligned} \Delta q_d &= J^\top \Delta x_d \\ &= (J^\top J)^{-1} J^\top \Delta x_d \end{aligned}$$



(i) Guess  $\vec{q}_0$

(ii) Compute

$$\rightarrow \vec{x}_0 = F.K(\vec{q}_0)$$

$$\rightarrow (\vec{d} - \vec{x}_0)$$

$$\rightarrow \alpha (\vec{d} - \vec{x}_0)$$

(iii)  $\vec{q}_1 \leftarrow \vec{q}_0 + \alpha (J^\top J)^{-1} J^\top (\vec{d} - \vec{x}_0)$

# How do I do this practically?

- Step 1: Figure out the Forward Kinematics

# How do I do this practically?

- Step 1: Figure out the Forward Kinematics
- Step 2: Differentiate it!
  - Option 1: Do it by hand.
  - Option 2: Do it numerically.

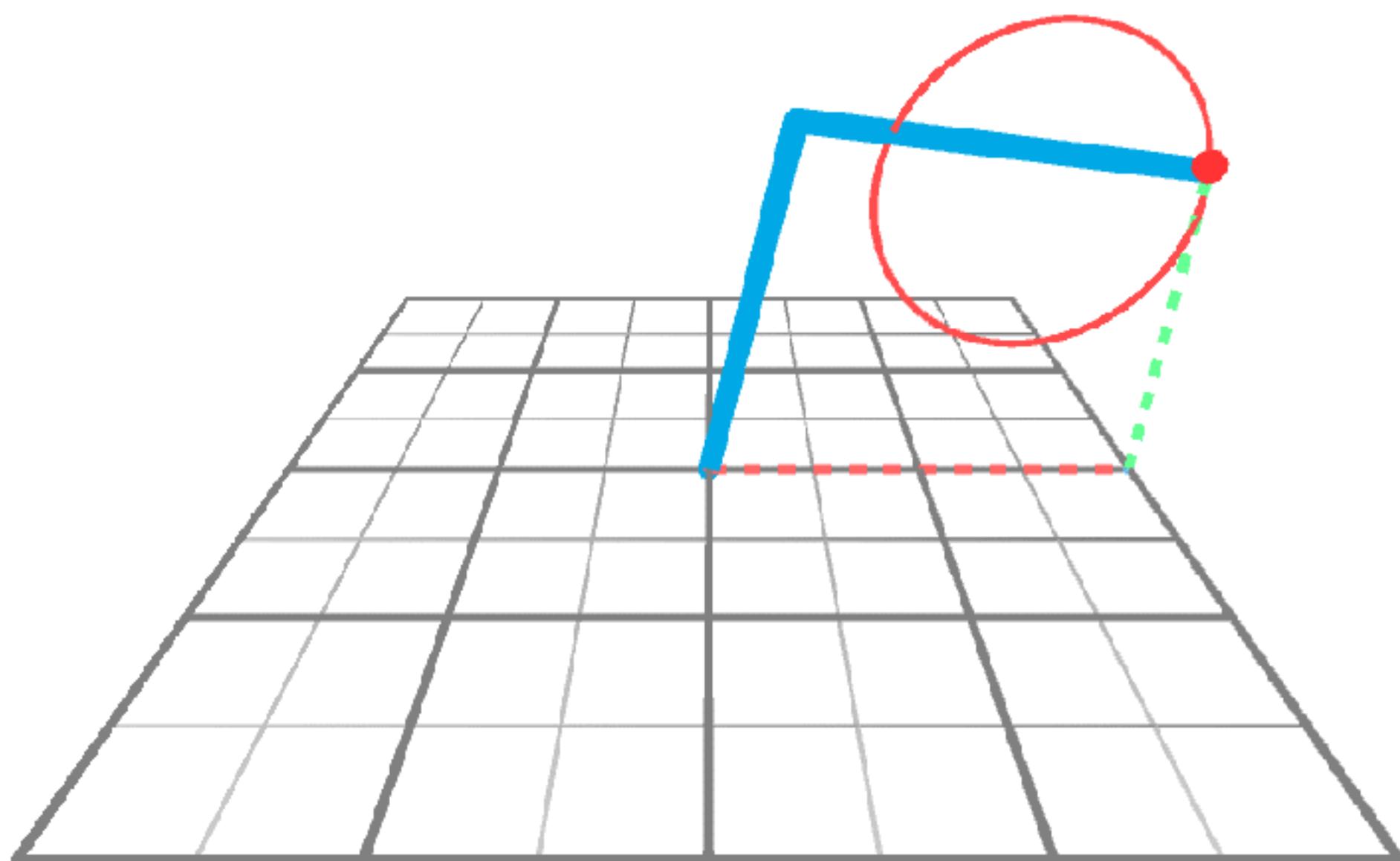
# How do I do this practically?

- Step 1: Figure out the Forward Kinematics
- Step 2: Differentiate it!
  - Option 1: Do it by hand.
  - Option 2: Do it numerically.
- Step 3: Choose learning rate.

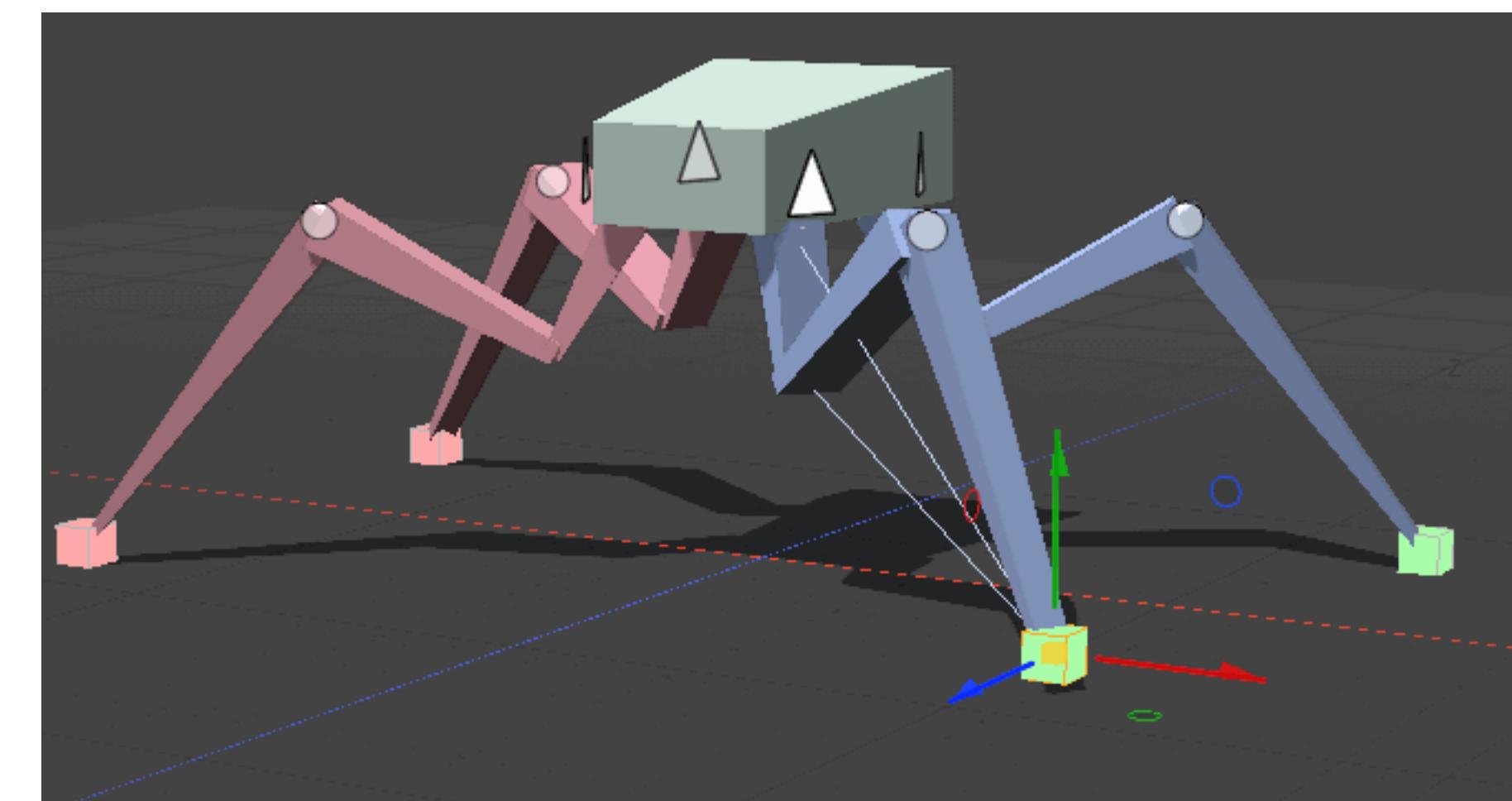
# How do I do this practically?

- Step 1: Figure out the Forward Kinematics
- Step 2: Differentiate it!
  - Option 1: Do it by hand.
  - Option 2: Do it numerically.
- Step 3: Choose learning rate.
- Step 4: Run optimizer.

# Examples

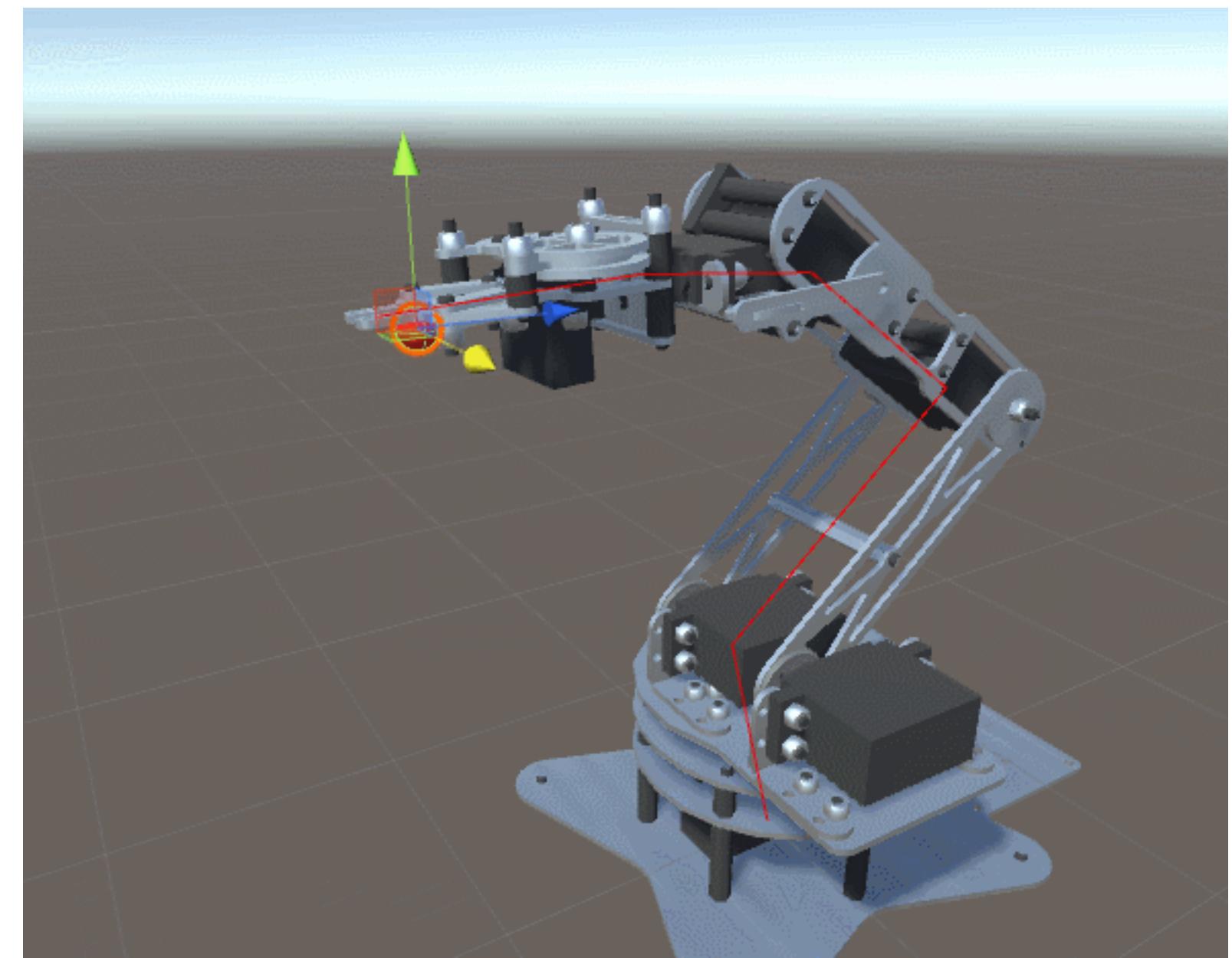
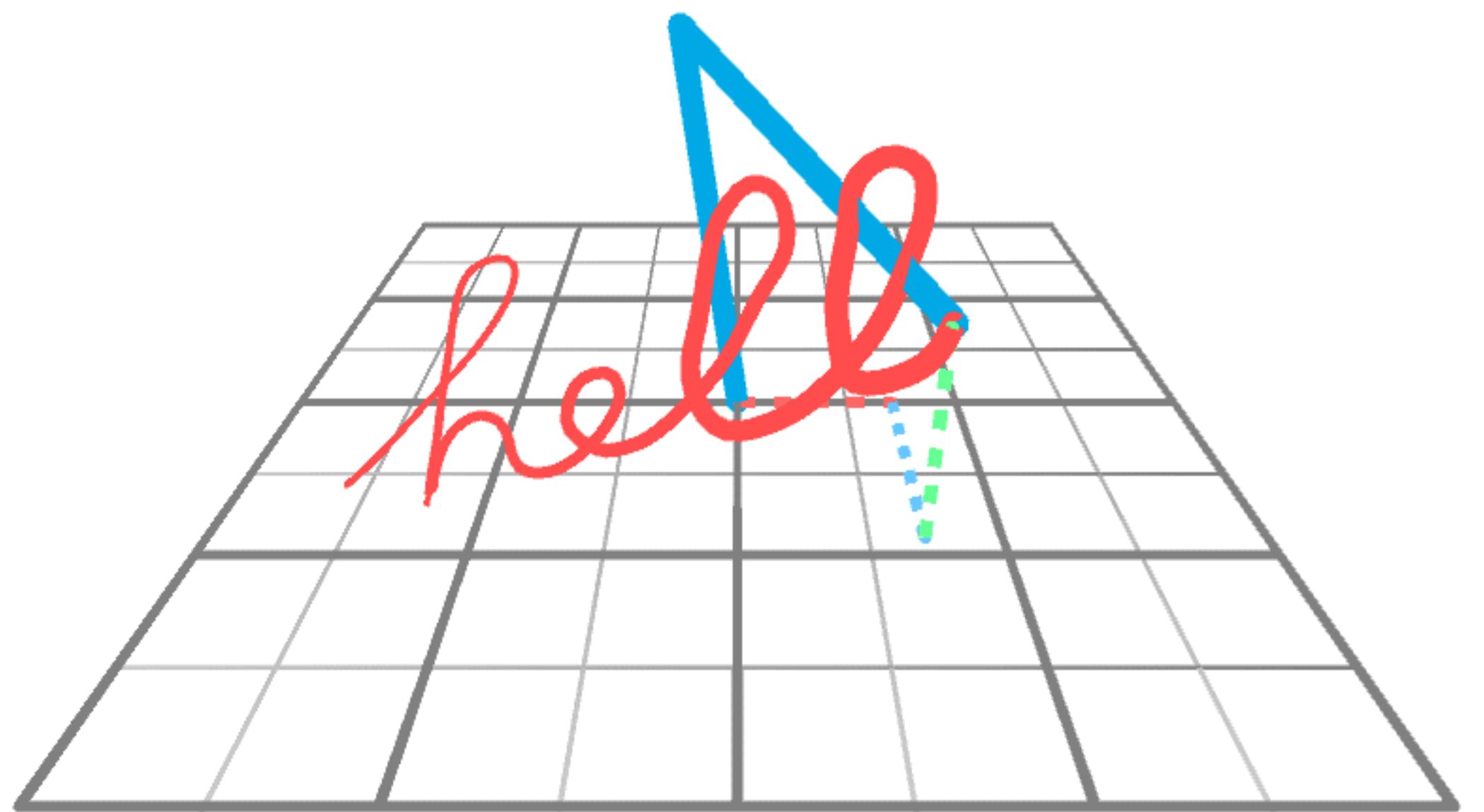


<https://www.alanzucconi.com/>



<https://learn.foundry.com/>

# Examples



<https://www.alanzucconi.com/>

# Additional Reading

- <http://motion.cs.illinois.edu/RoboticSystems/Kinematics.html>
- Textbook: <http://hades.mech.northwestern.edu/images/2/2a/Park-lynch.pdf>