

# CS 4610/5335 – Lecture 5

## Inverse kinematics

Lawson L.S. Wong  
Northeastern University  
2/2/22

Material adapted from:

1. Robert Platt, CS 4610/5335
2. Peter Corke, Robotics, Vision and Control

# Recap: Forward kinematics

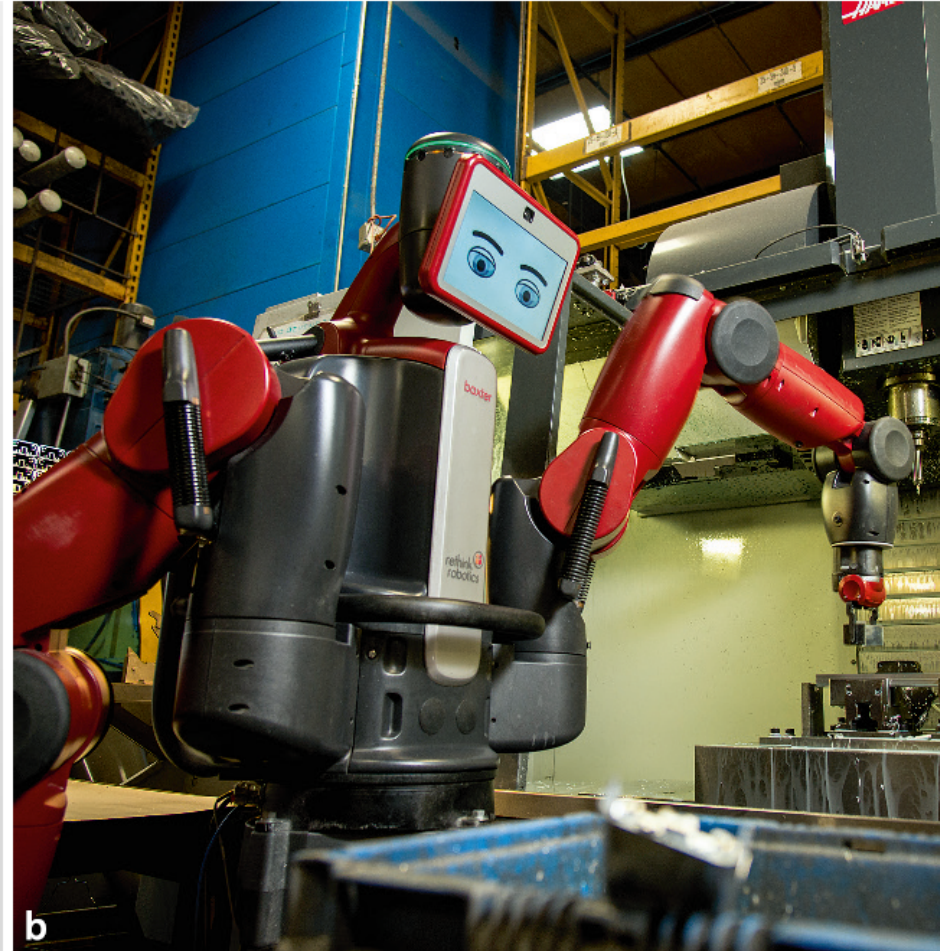
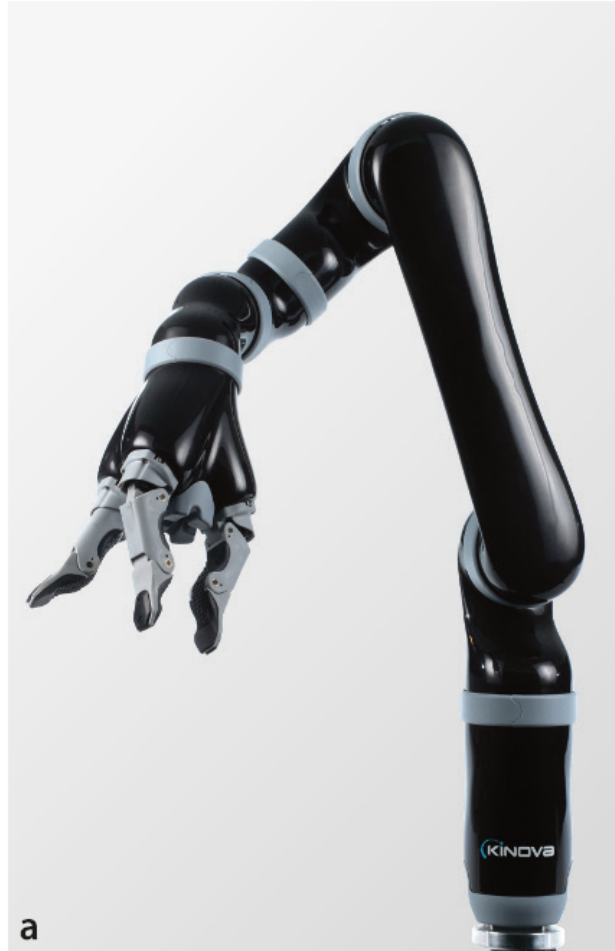


Fig. 7.1.

**a** Mico 6-joint robot with 3-fingered hand (courtesy of Kinova Robotics). **b** Baxter 2-armed robotic cower, each arm has 7 joints (courtesy of Rethink Robotics)

Where is the end-effector with respect to the base frame?

# Recap: Forward kinematics

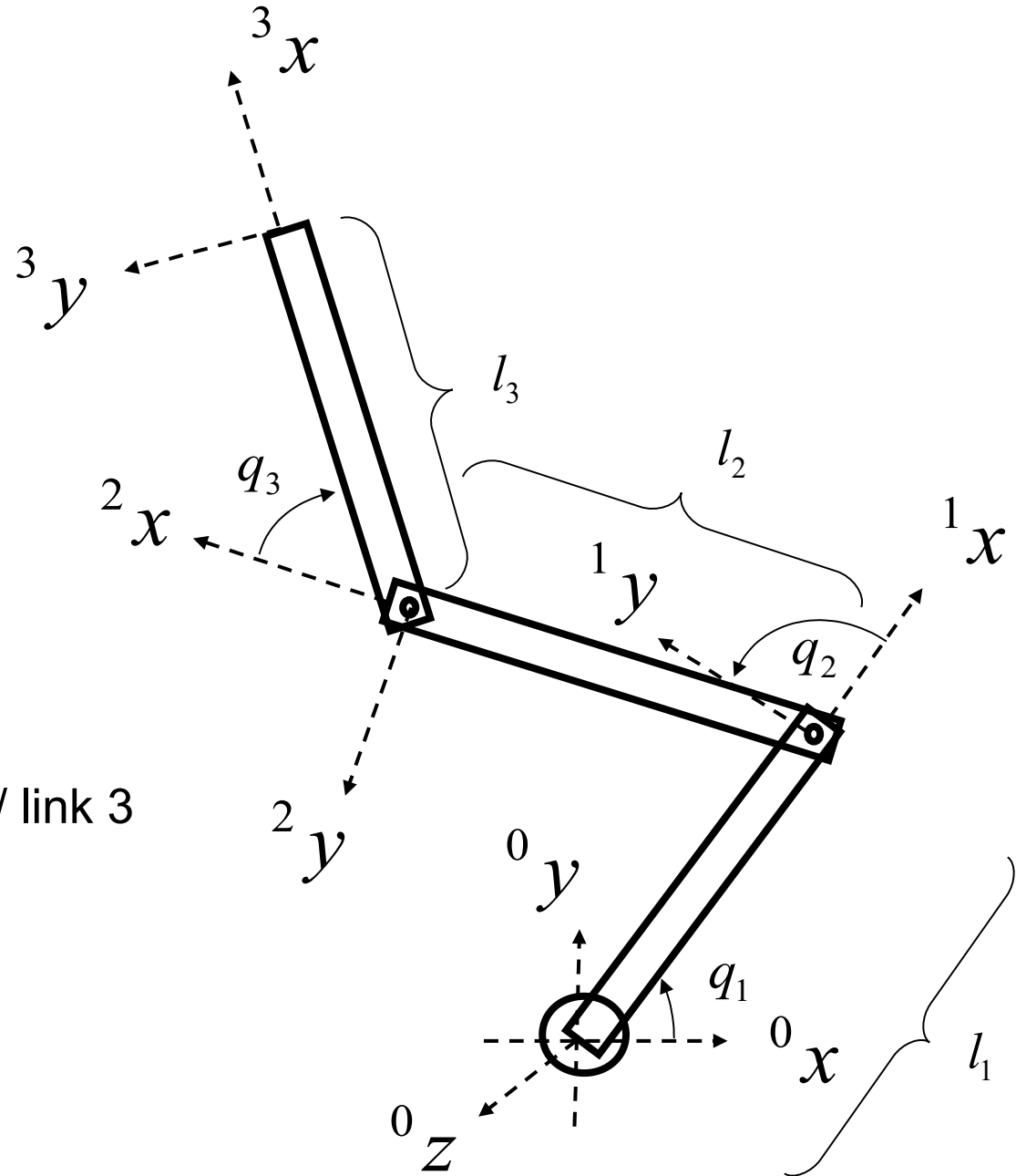
Base to end-effector transformation

$${}^0T_3 = {}^0T_1 {}^1T_2 {}^2T_3$$

Transform associated w/ link 1

Transform associated w/ link 2

Transform associated w/ link 3



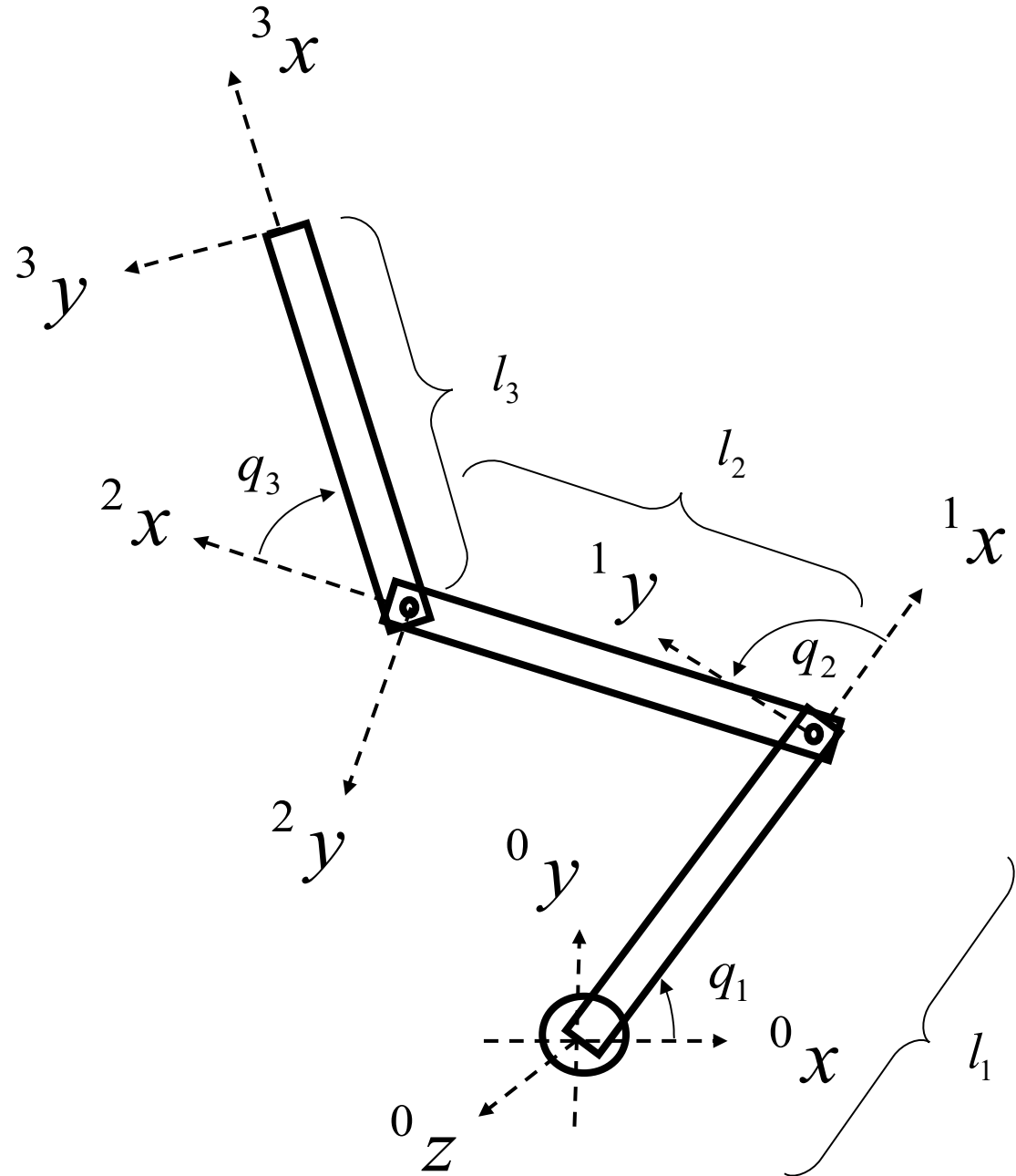
# Recap: Forward kinematics

Where is the end-effector with respect to the base?

$${}^0T_3 = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

More abstractly:

$$\text{EE pose} = \text{FK}(\mathbf{q})$$



# Outline

## Inverse kinematics (IK)

- Closed-form solution
- Jacobian matrix
- Numerical solution

# Recap: Forward kinematics vs. inverse kinematics

Where is the end-effector with respect to the base?

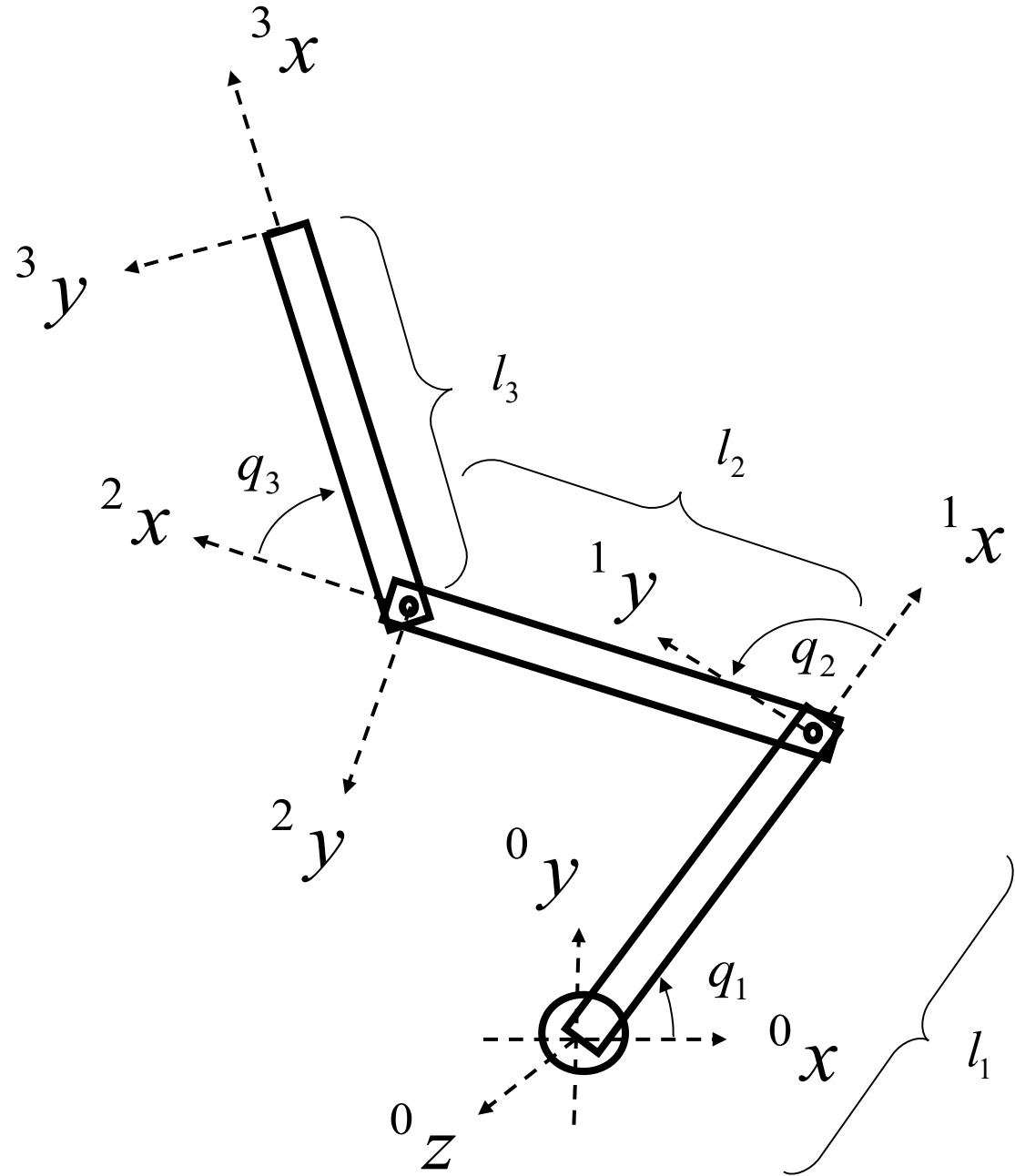
$${}^0T_3 = \begin{pmatrix} c_{123} & -s_{123} & 0 & l_1 c_1 + l_2 c_{12} + l_3 c_{123} \\ s_{123} & c_{123} & 0 & l_1 s_1 + l_2 s_{12} + l_3 s_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

More abstractly:

$$\text{EE pose} = \text{FK}(\mathbf{q})$$

Inverse kinematics:

$$\mathbf{q} = \mathcal{K}^{-1}(\xi)$$



# Recap: Inverse kinematics

$$\mathbf{q} = \mathcal{K}^{-1}(\xi)$$

Unlike FK, inverse problems are typically harder

There can be no solutions, 1 solution,  
or multiple solutions (possibly infinite)

Two solution approaches:

- Closed-form (analytical) solution
- Numerical (iterative) solution

# Inverse kinematics

$$\mathbf{q} = \mathcal{K}^{-1}(\xi)$$

Unlike FK, inverse problems are typically harder

There can be no solutions, 1 solution,  
or multiple solutions (possibly infinite)

Two solution approaches:

- Closed-form (analytical) solution
  - Solve FK equations for the joint angles exactly (see Ex1 Q5) – can be hard to derive!
  - Typically faster / can be computed offline



# Closed-form IK

$$\mathbf{q} = \mathcal{K}^{-1}(\xi)$$

There is no general analytical solution for IK

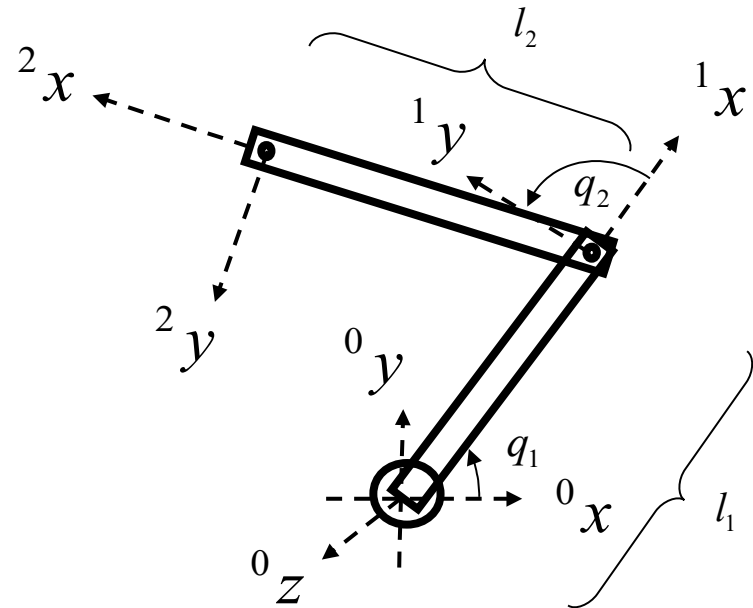
All analytical IK solutions are specific to a robot,  
based on geometric intuition about the robot

# Closed-form IK

$$\mathbf{q} = \mathcal{K}^{-1}(\xi)$$

There is no general analytical solution for IK

All analytical IK solutions are specific to a robot,  
based on geometric intuition about the robot

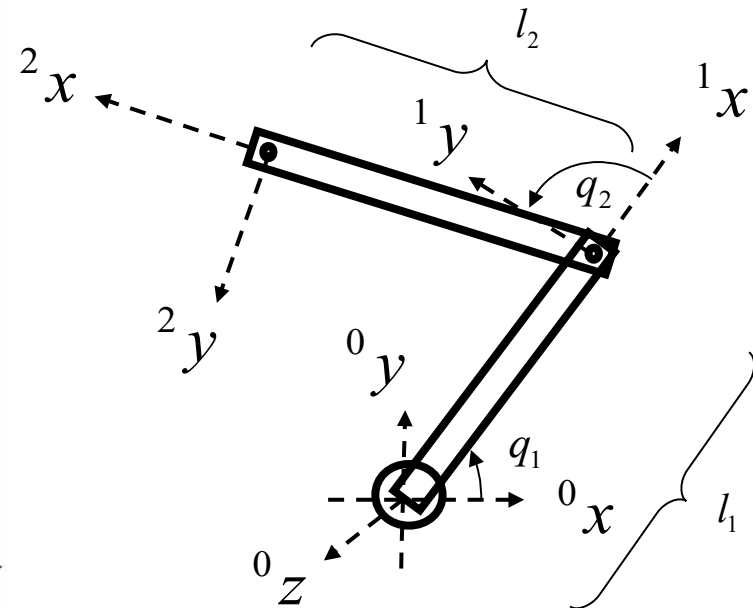
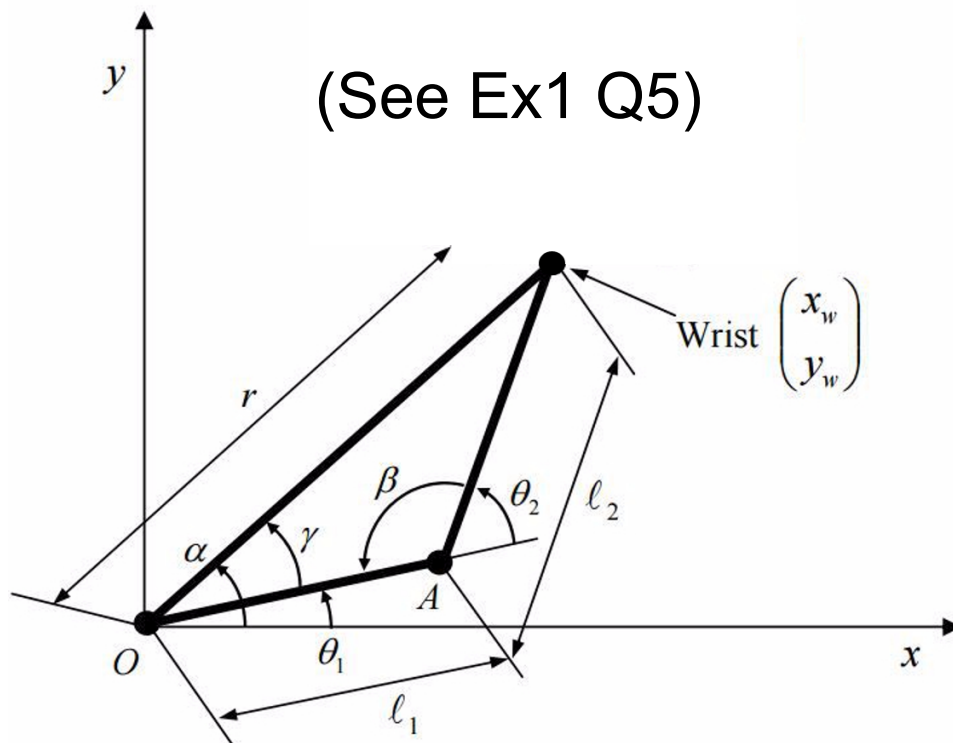


# Closed-form IK

$$\mathbf{q} = \mathcal{K}^{-1}(\xi)$$

There is no general analytical solution for IK

All analytical IK solutions are specific to a robot,  
based on geometric intuition about the robot



# Closed-form IK: Example (sketch)

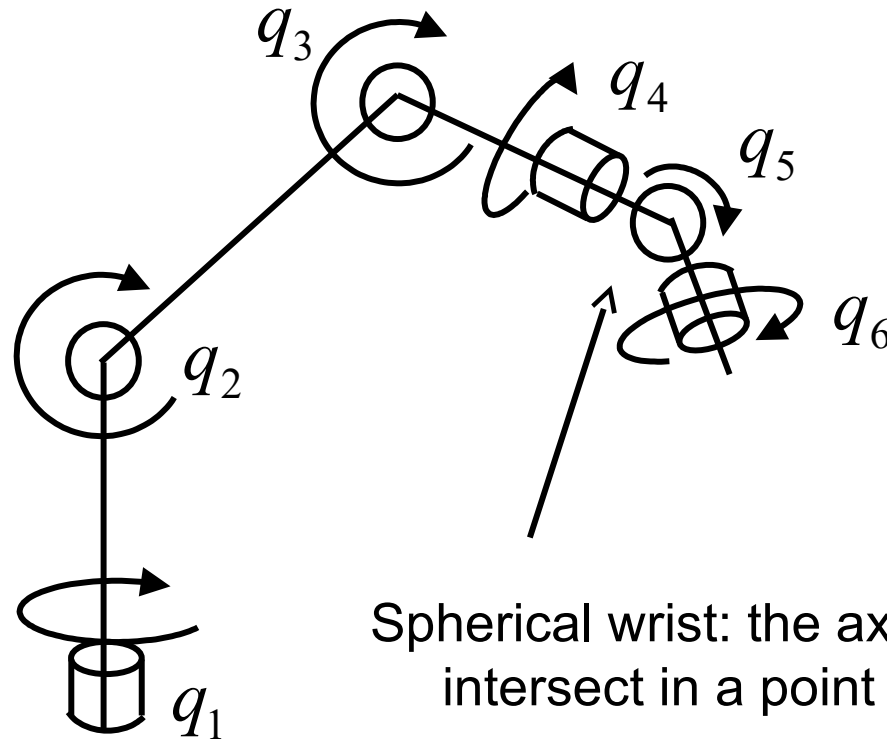
The **Puma 560 robot** (Programmable Universal Manipulator for Assembly) released in 1978 was the first modern industrial robot and became enormously popular. It featured an anthropomorphic design, electric motors and a spherical wrist – the archetype of all that followed. It can be seen in the Smithsonian Museum of American History, Washington DC.

The Puma 560 catalyzed robotics research in the 1980s and it was a very common laboratory robot. Today it is obsolete and rare but in homage to its important role in robotics research we use it here. For our purposes the advantages of this robot are that it has been well studied and its parameters are very well known – it has been described as the “white rat” of robotics research.

Most modern 6-axis industrial robots are very similar in structure and can be accommodated simply by changing the Denavit-Hartenberg parameters. The Toolbox has kinematic models for a number of common industrial robots from manufacturers such as Rethink, Kinova, Motoman, Fanuc and ABB. (Puma photo courtesy Oussama Khatib)



# Closed-form IK: Example (sketch)



Spherical wrist: the axes of the last three joints intersect in a point



# Closed-form IK: Example (sketch)

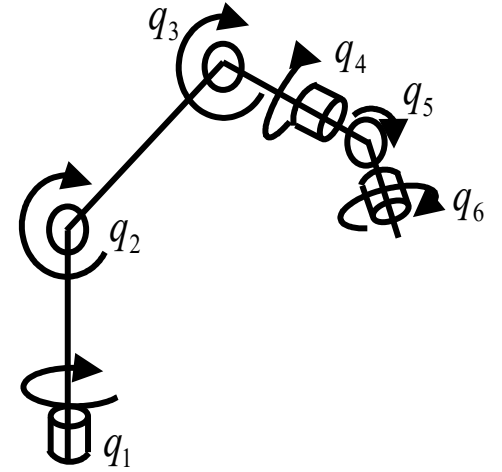
Problem:

Given: desired transform,

$$T_{eff} = \begin{pmatrix} R_{eff} & d_{eff} \\ 0 & 1 \end{pmatrix}$$

Find:

$$q = (q_1 \quad q_2 \quad q_3 \quad q_4 \quad \cdots \quad q_n)$$

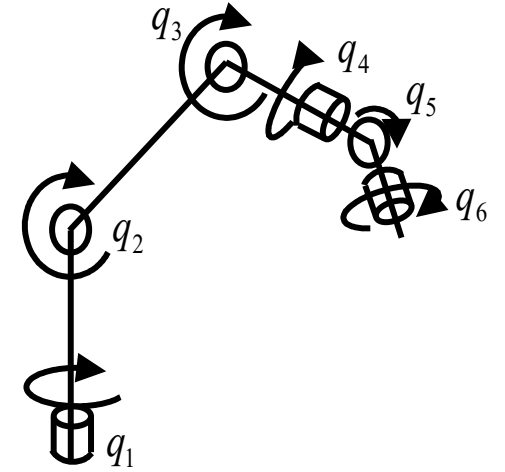


# Closed-form IK: Example (sketch)

Problem:

Given: desired transform,  $T_{eff} = \begin{pmatrix} R_{eff} & d_{eff} \\ 0 & 1 \end{pmatrix}$

Find:  $q = (q_1 \quad q_2 \quad q_3 \quad q_4 \quad \cdots \quad q_n)$



Observations:

The desired transform (pose) encodes six *degrees of freedom*  
(this info can be represented by six numbers)

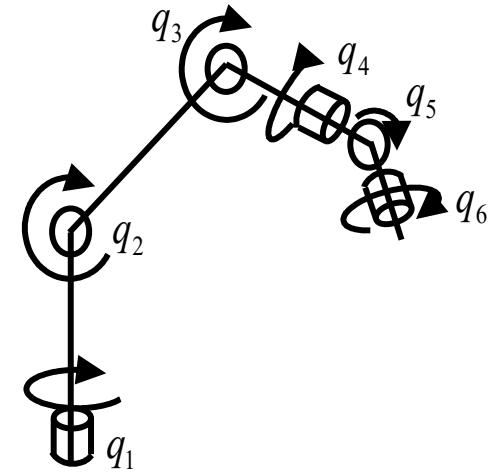
Since we only have six joints at our disposal,  
there is no manifold of *redundant* solutions.

# Closed-form IK: Example (sketch)

Problem:

Given: desired transform,  $T_{eff} = \begin{pmatrix} R_{eff} & d_{eff} \\ 0 & 1 \end{pmatrix}$

Find:  $q = (q_1 \quad q_2 \quad q_3 \quad q_4 \quad \cdots \quad q_n)$



Observations:

The desired transform (pose) encodes six *degrees of freedom*  
(this info can be represented by six numbers)

Since we only have six joints at our disposal,  
there is no manifold of *redundant* solutions.

For this manipulator, the problem can be decomposed into  
a position component (the first three joints) and  
an orientation component (the last three joints)

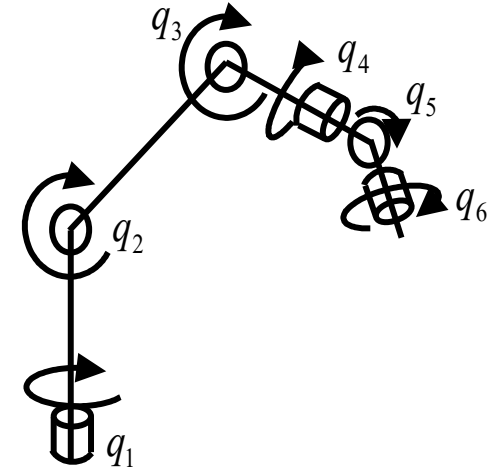
The first three joints tell you what the position of the spherical wrist



# Closed-form IK: Example (sketch)

Solution:

First, find the desired position of the spherical wrist:



Since it's a spherical wrist, the last three joints can be thought of as rotating about a point.

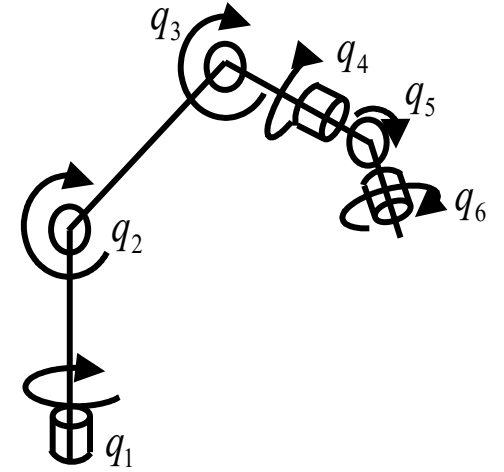
A constant transform exists that goes from the last wrist joint out to the end effector (sometimes this is called the “tool” transform):  ${}^{sw}T_{eff}$

Solve for the position of the wrist:

$${}^bT_{sw} = {}^bT_{eff} {}^{sw}T_{eff}^{-1}$$

# Closed-form IK: Example (sketch)

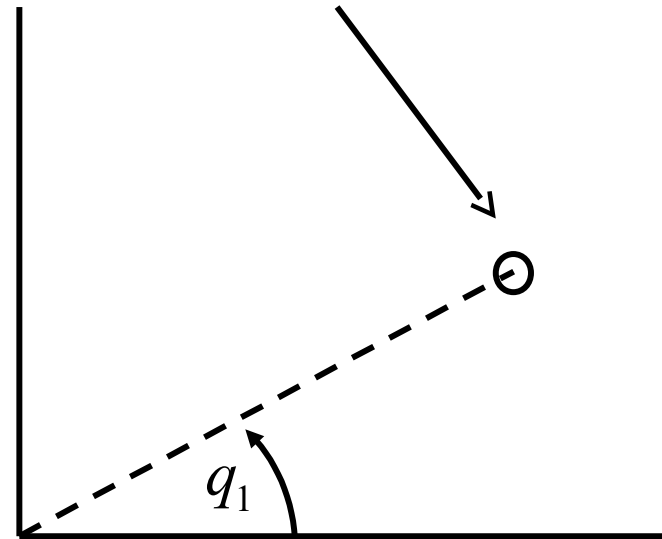
Next, solve for the first three joints:



First, solve for  $q_1$   
(look down from above)

$$q_1 = a \tan 2(x_g, y_g)$$

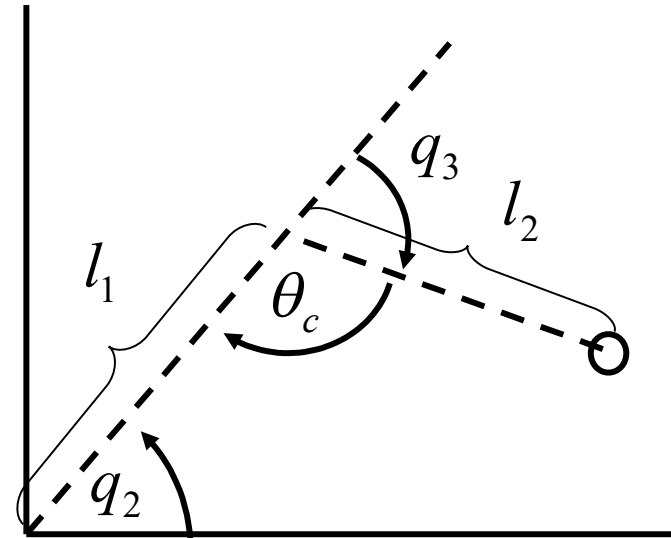
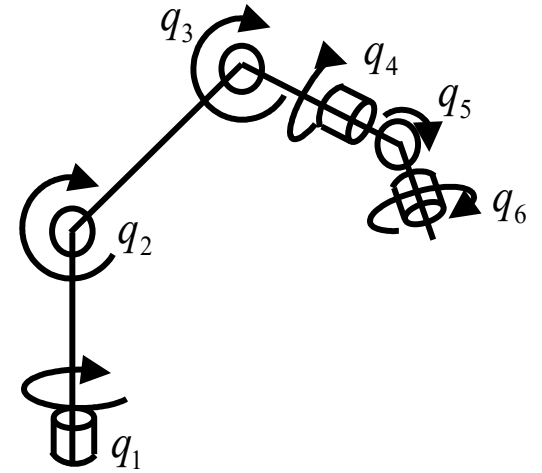
Goal position in horizontal plane



# Closed-form IK: Example (sketch)

Next, solve for the first three joints:

Next, solve for  $q_3$   
(look at the manipulator  
orthogonal to the plane of the first two links)

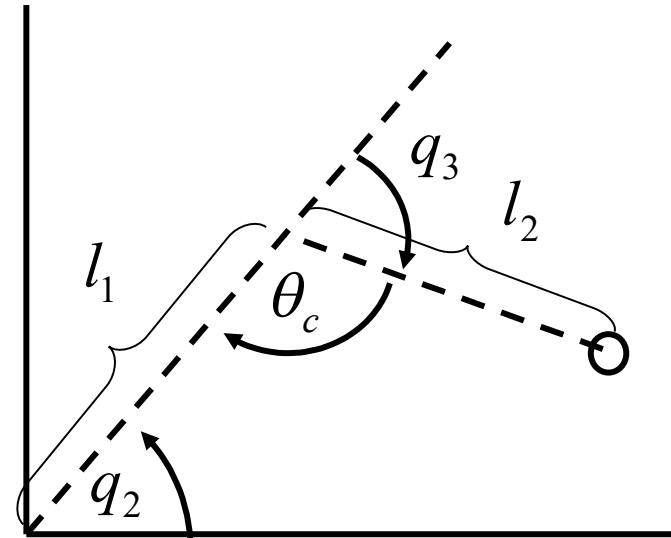
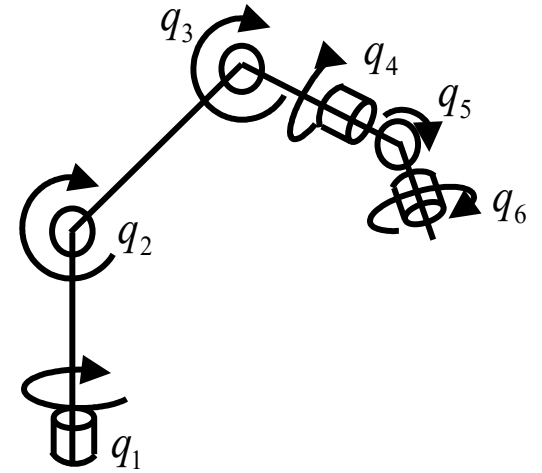


# Closed-form IK: Example (sketch)

Next, solve for the first three joints:

Next, solve for  $q_3$   
(look at the manipulator  
orthogonal to the plane of the first two links)

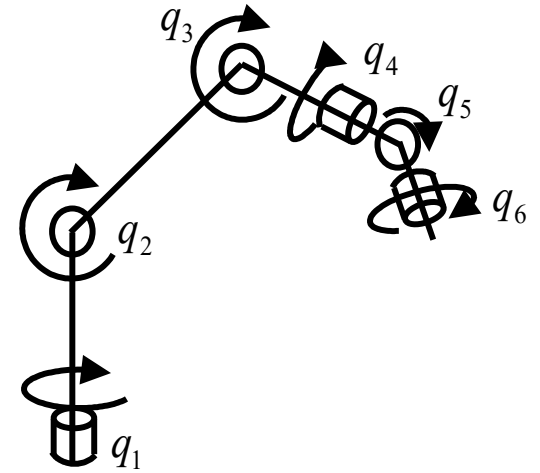
Seem familiar?



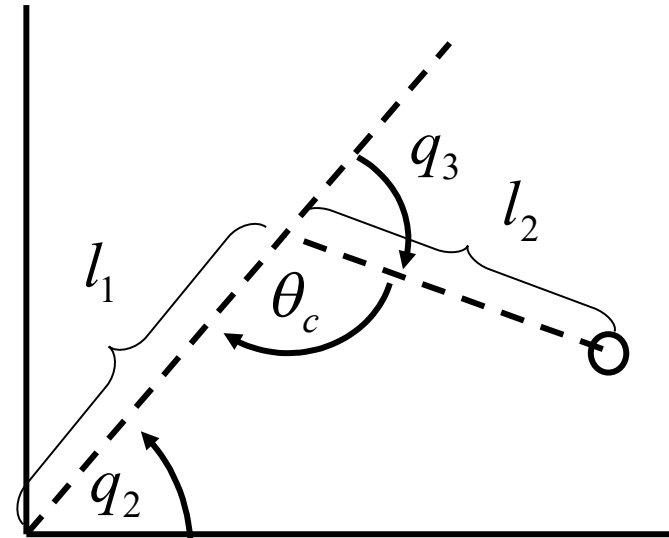
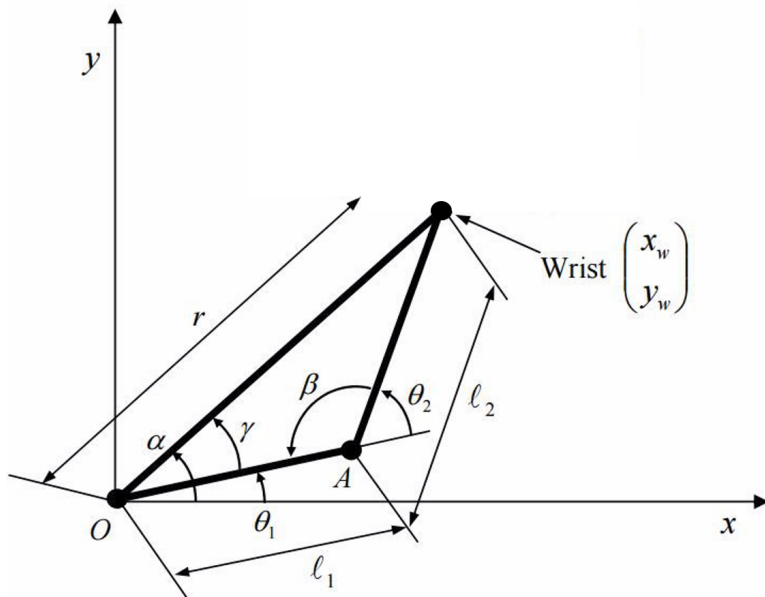
# Closed-form IK: Example (sketch)

Next, solve for the first three joints:

Next, solve for  $q_3$   
(look at the manipulator  
orthogonal to the plane of the first two links)



Seem familiar? This was the 2-D case!

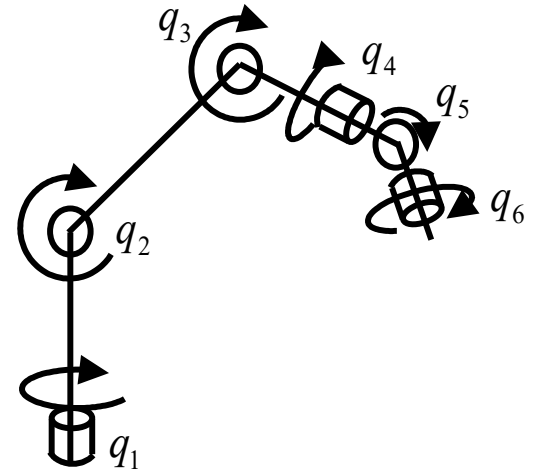


# Closed-form IK: Example (sketch)

Finally, the last three joints completely specify the orientation of the end-effector.

Note that the last three joints look just like ZYZ Euler angles

Determination of the joint angles is easy – just calculate the ZYZ Euler angles corresponding to the desired orientation

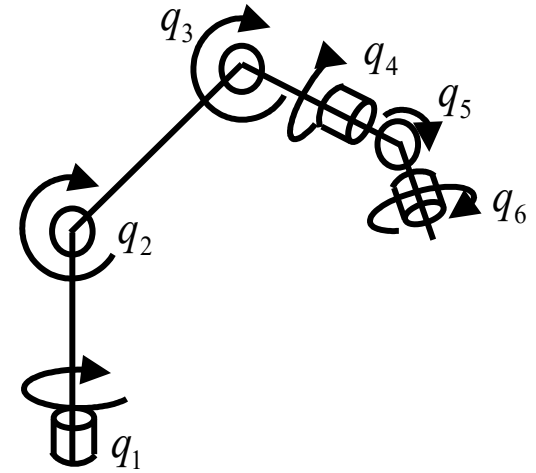


# Closed-form IK: Example (sketch)

Finally, the last three joints completely specify the orientation of the end-effector.

Note that the last three joints look just like ZYZ Euler angles

Determination of the joint angles is easy – just calculate the ZYZ Euler angles corresponding to the desired orientation

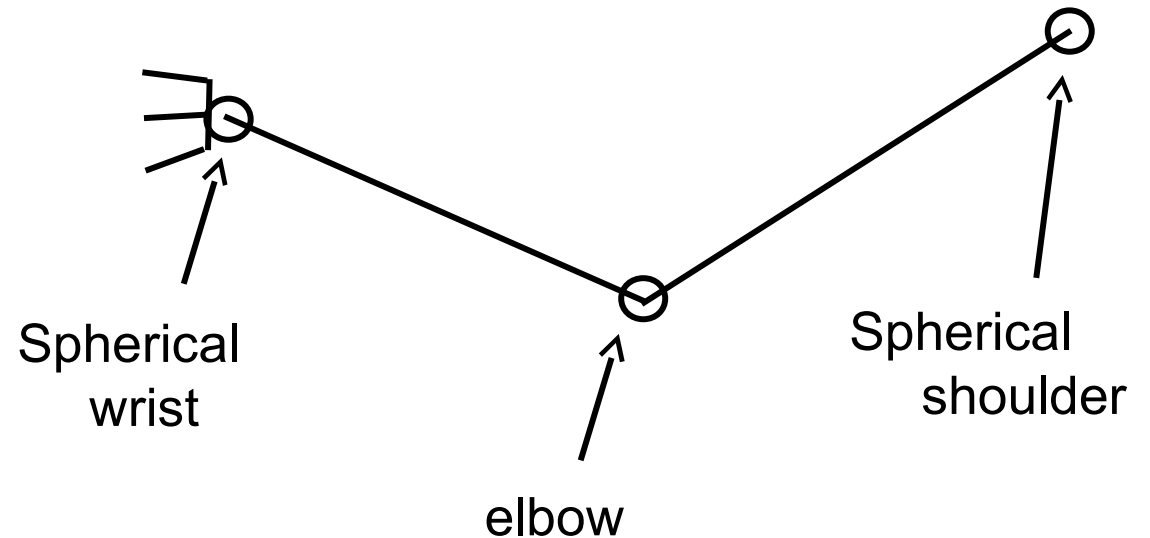


$$R_{zyz}(\phi, \theta, \psi) = \begin{pmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{pmatrix}$$

# Closed-form IK: Humanoid arm

You can do similar types of things for a humanoid (7-DOF) arm.

Since this is a redundant arm, there are a manifold of solutions...

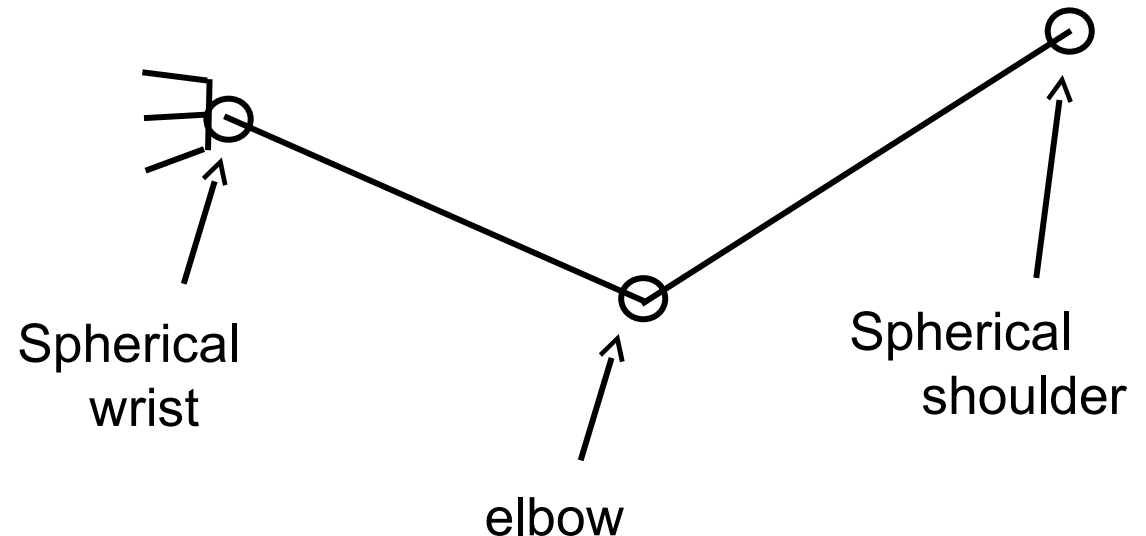




# Closed-form IK: Humanoid arm

You can do similar types of things for a humanoid (7-DOF) arm.

Since this is a redundant arm, there are a manifold of solutions...



General strategy:

Solve for elbow angle

Solve for a set of shoulder angles  
that places the wrist in the correct position

Solve for the wrist angles

# Outline

## Inverse kinematics (IK)

- ✓ Closed-form solution
  - Jacobian matrix
  - Numerical solution

# Jacobian matrix

A Jacobian is the matrix equivalent of the derivative – the derivative of a vector-valued function of a vector with respect to a vector. If  $\mathbf{y} = \mathbf{f}(\mathbf{x})$  and  $\mathbf{x} \in \mathbb{R}^n$  and  $\mathbf{y} \in \mathbb{R}^m$  then the Jacobian is the  $m \times n$  matrix

$$J = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \dots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$

The Jacobian is named after Carl Jacobi, and more details are given in Appendix E.

# Jacobian matrix

A Jacobian is the matrix equivalent of the derivative – the derivative of a vector-valued function of a vector with respect to a vector. If  $\mathbf{y} = \mathbf{f}(\mathbf{x})$  and  $\mathbf{x} \in \mathbb{R}^n$  and  $\mathbf{y} \in \mathbb{R}^m$  then the Jacobian is the  $m \times n$  matrix

$$J = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \dots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \dots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$

The Jacobian is named after Carl Jacobi, and more details are given in Appendix E.

Apply this to the forward kinematics map  $K(q)$  (written as  $\phi(q)$  on next slide)

# Jacobian matrix

Given the kinematic map  $y = \phi(q)$  and its Jacobian  $J(q) = \frac{\partial}{\partial q} \phi(q)$ , we have:

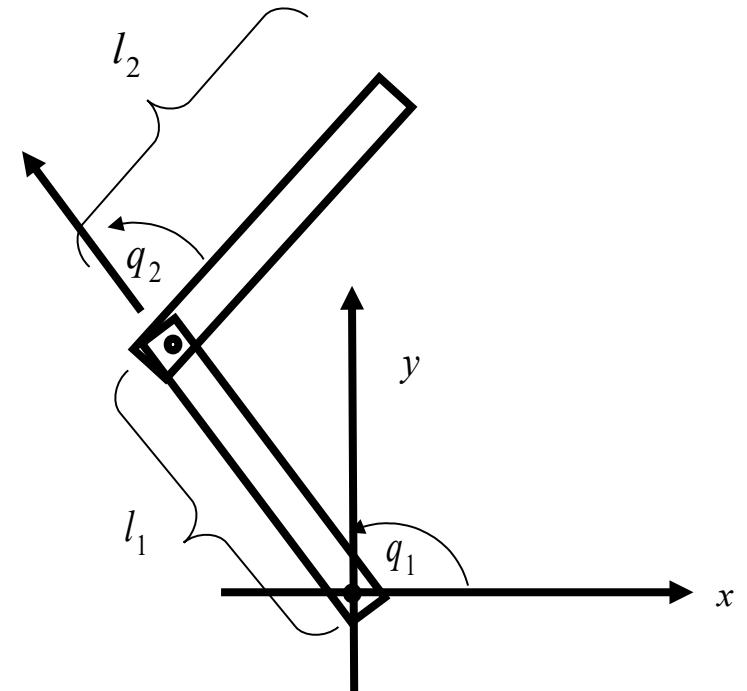
$$\delta y = J(q) \delta q$$

$$J(q) = \frac{\partial}{\partial q} \phi(q) = \begin{pmatrix} \frac{\partial \phi_1(q)}{\partial q_1} & \frac{\partial \phi_1(q)}{\partial q_2} & \cdots & \frac{\partial \phi_1(q)}{\partial q_n} \\ \frac{\partial \phi_2(q)}{\partial q_1} & \frac{\partial \phi_2(q)}{\partial q_2} & \cdots & \frac{\partial \phi_2(q)}{\partial q_n} \\ \vdots & & & \vdots \\ \frac{\partial \phi_d(q)}{\partial q_1} & \frac{\partial \phi_d(q)}{\partial q_2} & \cdots & \frac{\partial \phi_d(q)}{\partial q_n} \end{pmatrix} \in \mathbb{R}^{d \times n}$$

# Jacobian matrix: Example

Forward kinematics (EE pose)  
of the two-link manipulator:

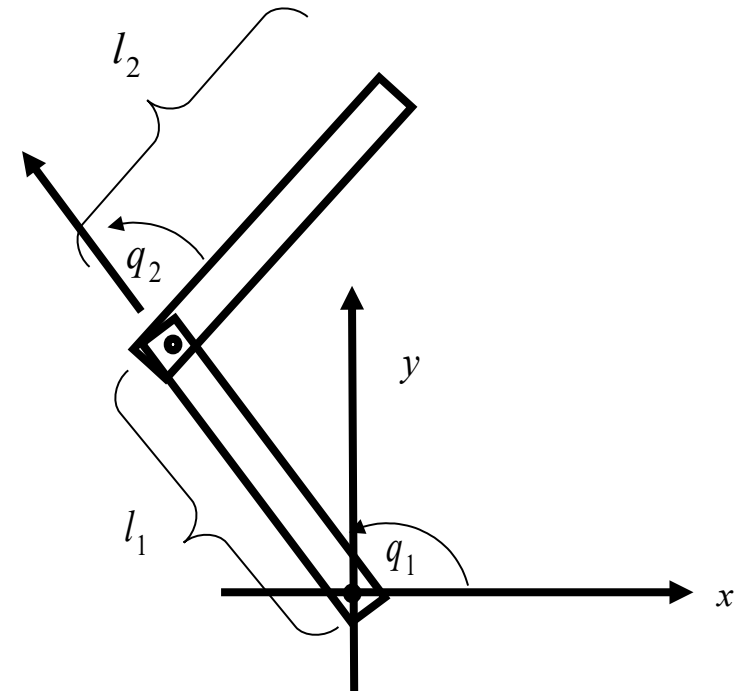
$$\vec{x} = \begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{bmatrix}$$



# Jacobian matrix: Example

Forward kinematics (EE pose)  
of the two-link manipulator:

$$\vec{x} = \begin{bmatrix} l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) \\ l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) \end{bmatrix}$$



Velocity Jacobian  $J_v(q)$ :

$$\frac{dx}{dq} = \begin{pmatrix} -l_1 \sin(q_1) - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) \\ l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) & -l_2 \cos(q_1 + q_2) \end{pmatrix}$$

# Jacobian matrix and velocity

$$\frac{d\mathbf{p}}{d\mathbf{q}} = \mathbf{J}(\mathbf{q})$$

$$d\mathbf{p} = \mathbf{J}(\mathbf{q})d\mathbf{q}$$



# Jacobian matrix and velocity

$$\frac{d\mathbf{p}}{d\mathbf{q}} = J(\mathbf{q})$$

$$d\mathbf{p} = J(\mathbf{q})d\mathbf{q}$$

$$\frac{d\mathbf{p}}{dt} = J(\mathbf{q})\frac{d\mathbf{q}}{dt}$$

$$\dot{\mathbf{p}} = J(\mathbf{q})\dot{\mathbf{q}}$$

# Jacobian matrix and velocity

$$\frac{d\mathbf{p}}{d\mathbf{q}} = J(\mathbf{q})$$

$$d\mathbf{p} = J(\mathbf{q})d\mathbf{q}$$

$$\frac{d\mathbf{p}}{dt} = J(\mathbf{q})\frac{d\mathbf{q}}{dt}$$

$$\dot{\mathbf{p}} = J(\mathbf{q})\dot{\mathbf{q}}$$

The Jacobian matrix maps velocity from the joint coordinate or configuration space to the end-effector's Cartesian coordinate space and is itself a function of the joint coordinates.

# Jacobian matrix and velocity

More generally we write the forward kinematics in functional form, Eq. 7.4, as

$${}^0\xi = \mathcal{K}(\boldsymbol{q})$$

and taking the derivative we write

$${}^0\boldsymbol{v} = {}^0J(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{8.2}$$

# Jacobian matrix and velocity

More generally we write the forward kinematics in functional form, Eq. 7.4, as

$${}^0\xi = \mathcal{K}(\mathbf{q})$$

and taking the derivative we write

$${}^0\boldsymbol{\nu} = {}^0J(\mathbf{q})\dot{\mathbf{q}} \tag{8.2}$$

where  ${}^0\boldsymbol{\nu} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z) \in \mathbb{R}^6$  is the spatial velocity, as discussed in Sect. 3.1.1, of the end-effector in the world frame and comprises translational and rotational velocity components.

# Jacobian matrix and velocity

More generally we write the forward kinematics in functional form, Eq. 7.4, as

$${}^0\xi = \mathcal{K}(\mathbf{q})$$

and taking the derivative we write

$${}^0\boldsymbol{\nu} = {}^0J(\mathbf{q})\dot{\mathbf{q}} \quad (8.2)$$

where  ${}^0\boldsymbol{\nu} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z) \in \mathbb{R}^6$  is the spatial velocity, as discussed in Sect. 3.1.1, of the end-effector in the world frame and comprises translational and rotational velocity components.

Also known as the twist  
(concatenation of linear and angular velocity)

# Jacobian matrix and velocity

More generally we write the forward kinematics in functional form, Eq. 7.4, as

$${}^0\xi = \mathcal{K}(\mathbf{q})$$

and taking the derivative we write

$${}^0\boldsymbol{\nu} = {}^0J(\mathbf{q})\dot{\mathbf{q}} \quad (8.2)$$

where  ${}^0\boldsymbol{\nu} = (v_x, v_y, v_z, \omega_x, \omega_y, \omega_z) \in \mathbb{R}^6$  is the spatial velocity, as discussed in Sect. 3.1.1, of the end-effector in the world frame and comprises translational and rotational velocity components.

Also known as the twist  
(concatenation of linear and angular velocity)

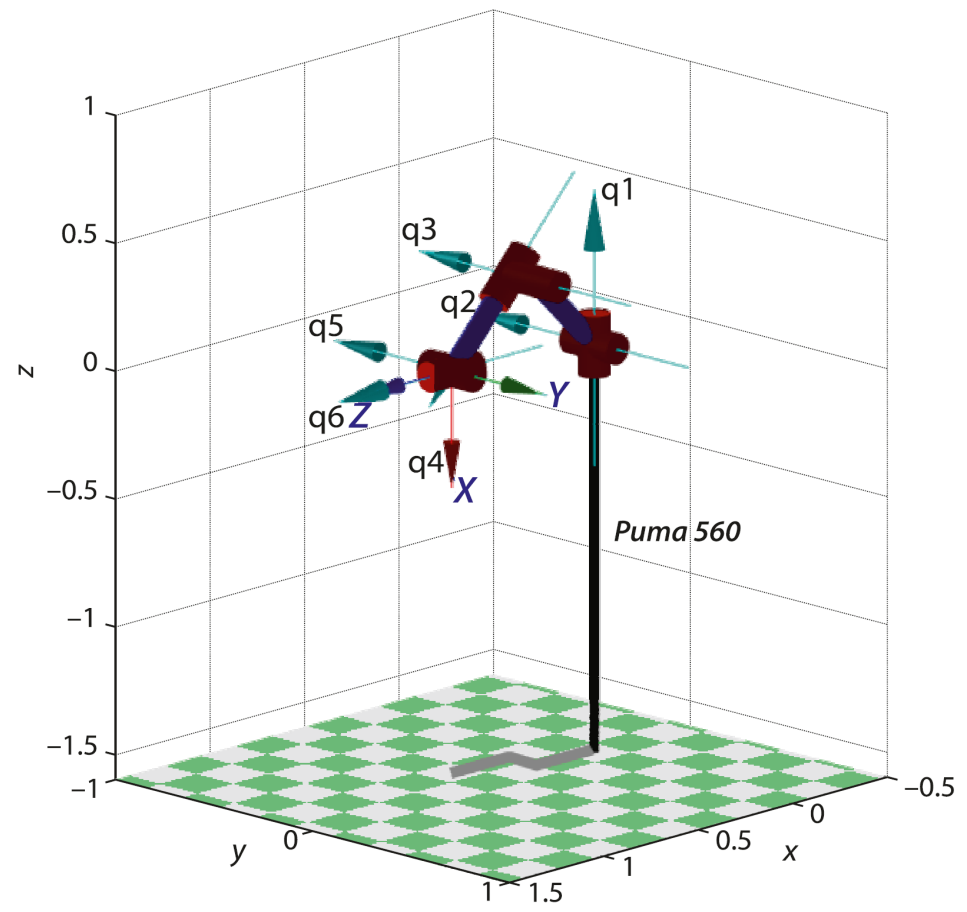
Angular velocity:	Direction = axis of rotation
	Magnitude = velocity of rotation

# Jacobian matrix and velocity: Example

```
>> J = p560.jacob0(qn)
```

```
J =
```

0.1501	0.0144	0.3197	0	0	0
0.5963	0.0000	0.0000	0	0	0
0	0.5963	0.2910	0	0	0
0	-0.0000	-0.0000	0.7071	-0.0000	-0.0000
0	-1.0000	-1.0000	-0.0000	-1.0000	-0.0000
1.0000	0.0000	0.0000	-0.7071	0.0000	-1.0000



# Jacobian matrix transformation

$${}^E \boldsymbol{v} = {}^E J_0 \left( {}^E \boldsymbol{\xi}_0 \right) {}^0 J(\boldsymbol{q}) \dot{\boldsymbol{q}} = \underbrace{\begin{pmatrix} {}^E \boldsymbol{R}_0 & \boldsymbol{0}_{3 \times 3} \\ \boldsymbol{0}_{3 \times 3} & {}^E \boldsymbol{R}_0 \end{pmatrix}}_{} {}^0 J(\boldsymbol{q}) \dot{\boldsymbol{q}} = {}^E J(\boldsymbol{q}) \dot{\boldsymbol{q}}$$

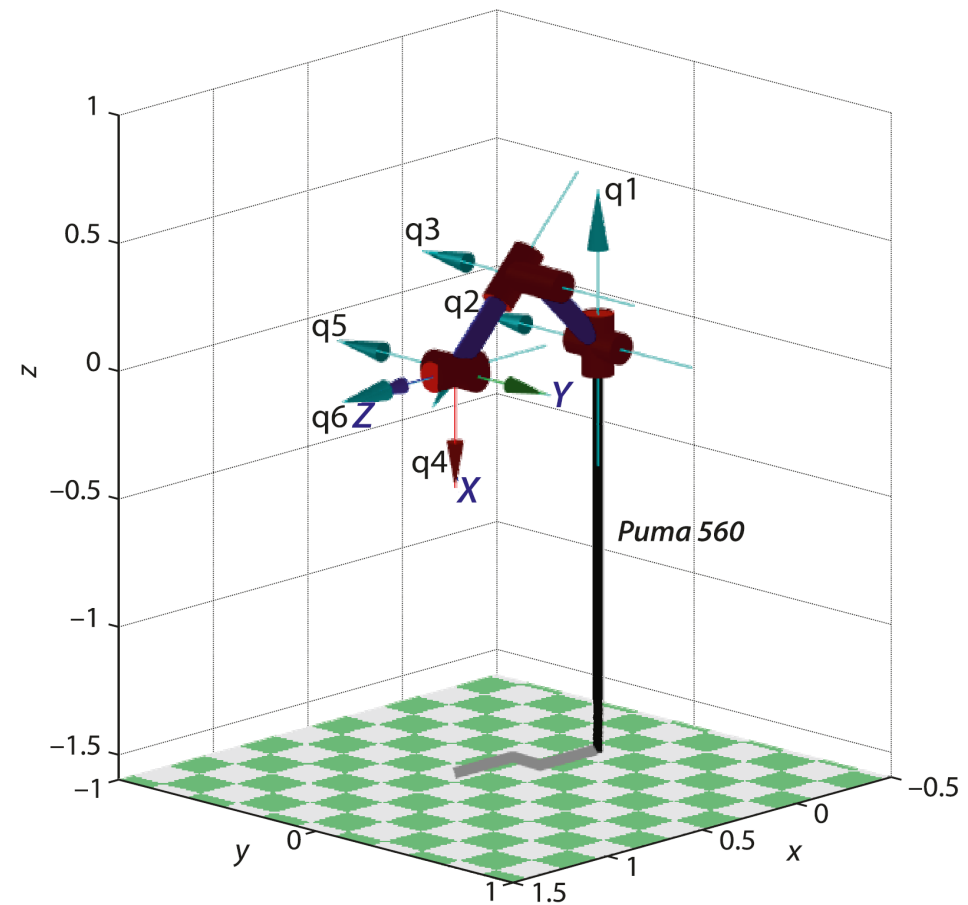


# Jacobian matrix transformation

$${}^E v = {}^E J_0 \left( {}^E \xi_0 \right) {}^0 J(q) \dot{q} = \underbrace{\begin{pmatrix} {}^E R_0 & 0_{3 \times 3} \\ 0_{3 \times 3} & {}^E R_0 \end{pmatrix}}_{} {}^0 J(q) \dot{q} = {}^E J(q) \dot{q}$$

```
>> p560.jacobe(qn)
ans =
-0.0000    -0.5963   -0.2910
 0.5963     0.0000    0.0000
 0.1500     0.0144    0.3197
-1.0000         0         0
-0.0000   -1.0000   -1.0000
-0.0000     0.0000    0.0000
```

(Actually a 6x6 matrix,  
see RVC 8.1.2)



# Outline

## Inverse kinematics (IK)

- ✓ Closed-form solution
- ✓ Jacobian matrix
- Numerical solution

# Inverse kinematics

$$\mathbf{q} = \mathcal{K}^{-1}(\xi)$$

Unlike FK, inverse problems are typically harder

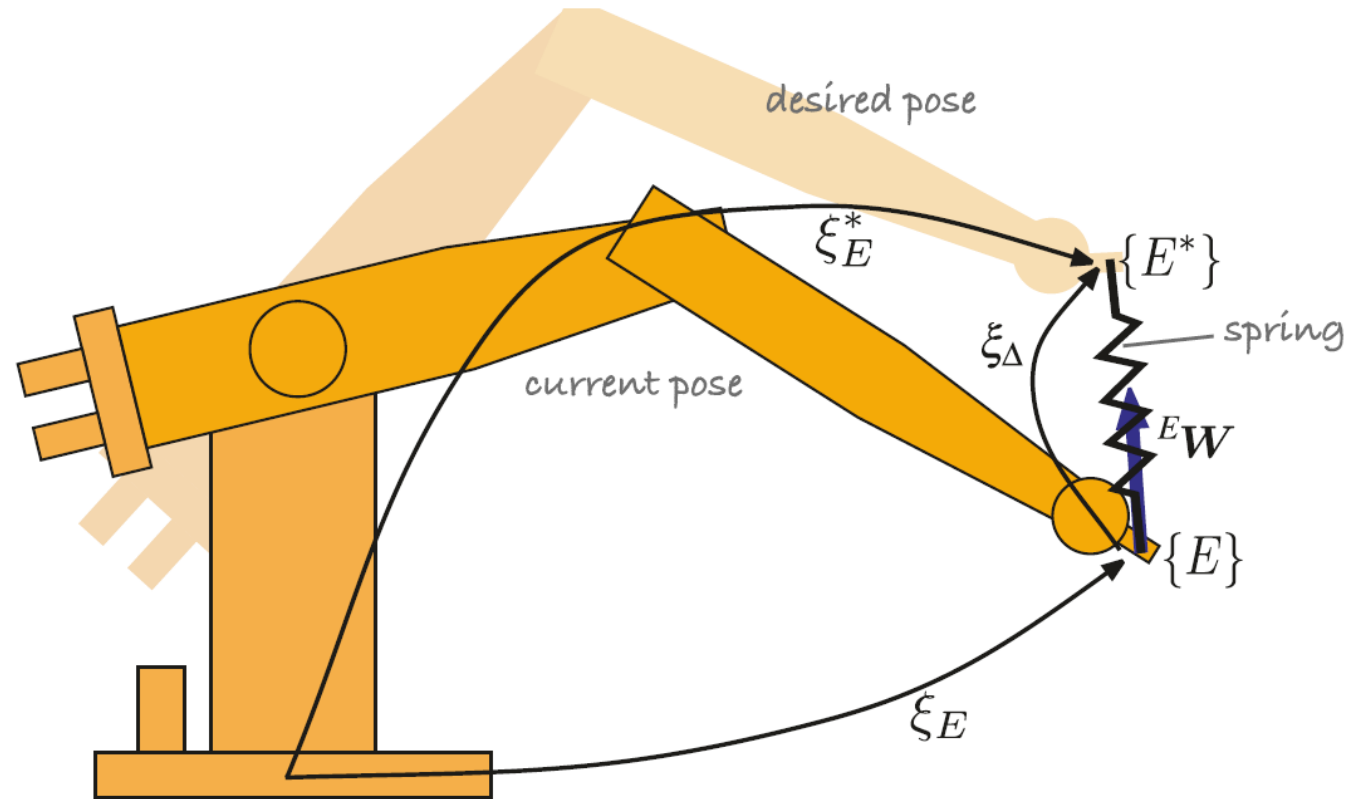
There can be no solutions, 1 solution,  
or multiple solutions (possibly infinite)

Two solution approaches:

- Closed-form (analytical) solution
- Numerical (iterative) solution

To discuss numerical IK,  
we need to first discuss the Jacobian matrix

# Inverse kinematics: Numerical solution



**Fig.8.10.**

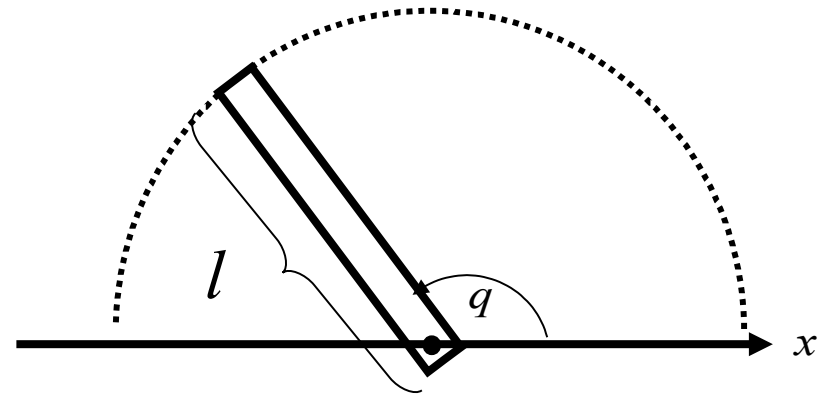
Schematic of the numerical inverse kinematic approach, showing the current  $\xi_E$  and the desired  $\xi_E^*$  manipulator pose

# Example

Consider a one-link arm

As the arm rotates, the end-effector  
sweeps out an arc

Let's assume that we are only interested  
in the  $x$  coordinate...



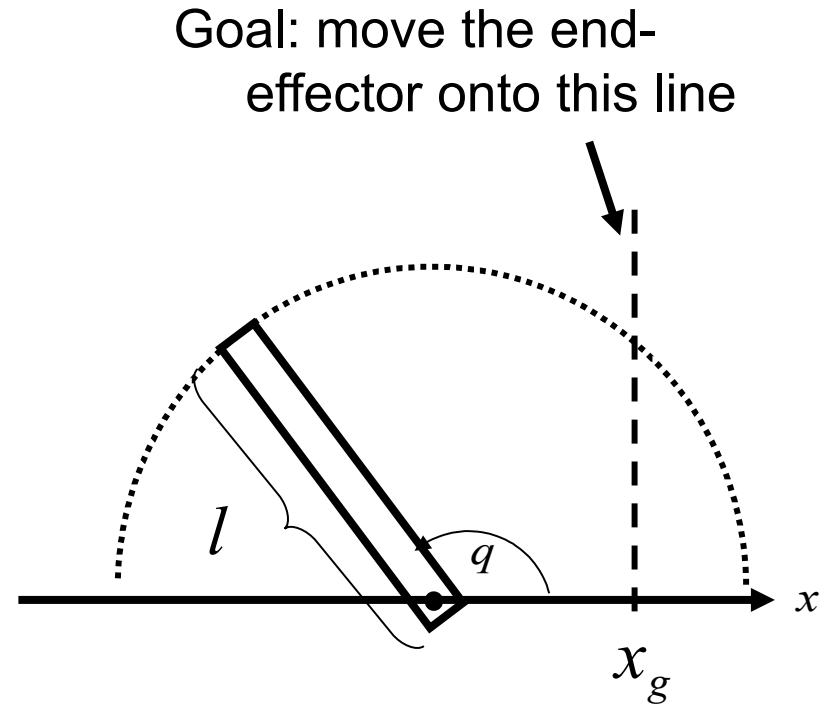
# Example

Consider a one-link arm

As the arm rotates, the end-effector  
sweeps out an arc

Let's assume that we are only interested  
in the  $x$  coordinate...

Suppose you want to move the end  
effector above a specified point,  $x_g$



# Example

Consider a one-link arm

As the arm rotates, the end-effector sweeps out an arc

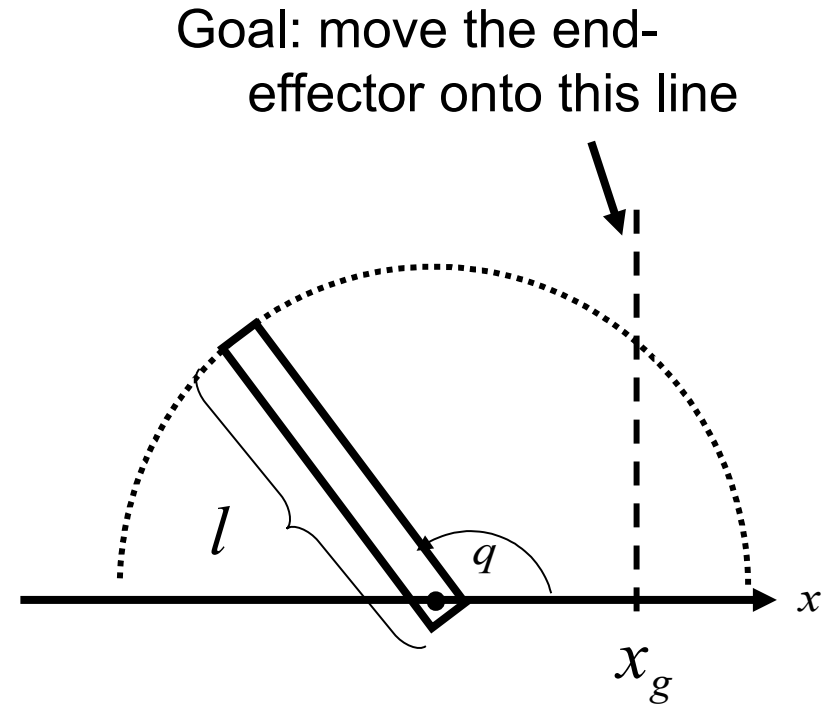
Let's assume that we are only interested in the  $x$  coordinate...

Suppose you want to move the end effector above a specified point,  $x_g$

Closed-form solution:

Forward kinematics:  $x = l \cos(q)$

Solve for IK:  $q_g = \cos^{-1} \left( \frac{x_g}{l} \right)$



# Example

Consider a one-link arm

As the arm rotates, the end-effector  
sweeps out an arc

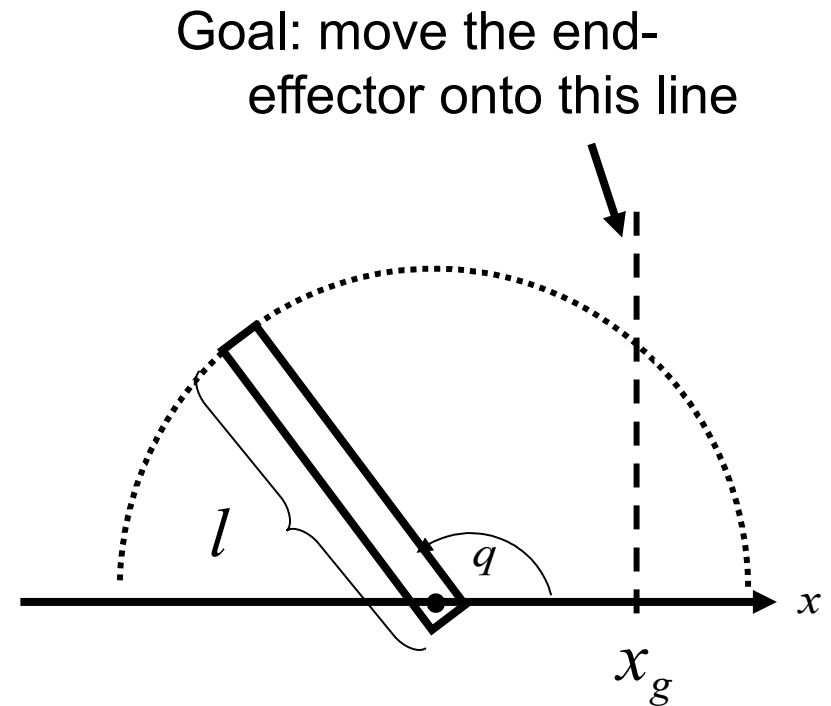
Let's assume that we are only interested  
in the  $x$  coordinate...

Suppose you want to move the end  
effector above a specified point,  $x_g$

Numerical solution:

What  $x$  velocity is desirable?

How does that translate to  $q$ ?





# Example

Consider a one-link arm

As the arm rotates, the end-effector sweeps out an arc

Let's assume that we are only interested in the  $x$  coordinate...

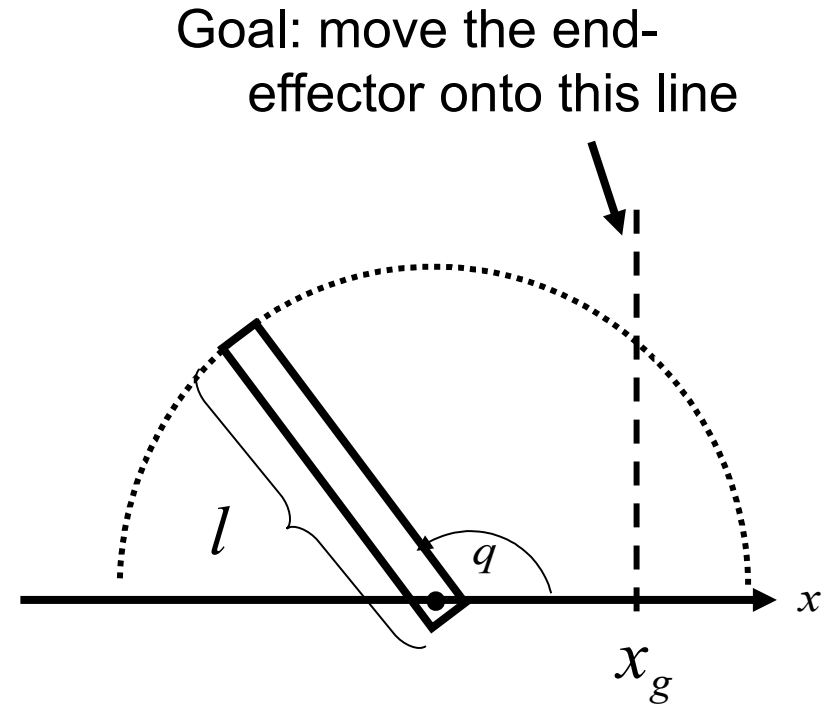
Suppose you want to move the end effector above a specified point,  $x_g$

Numerical solution:

What  $x$  velocity is desirable?

How does that translate to  $q$ ? Use the Jacobian!

$$dp = J(q)dq$$



# Example

Numerical solution:

What  $x$  velocity is desirable?

How does that translate to  $q$ ?

1.

2.  $x_i = l \cos(q_i)$

3.  $\delta x = \alpha (x_g - x_i)$

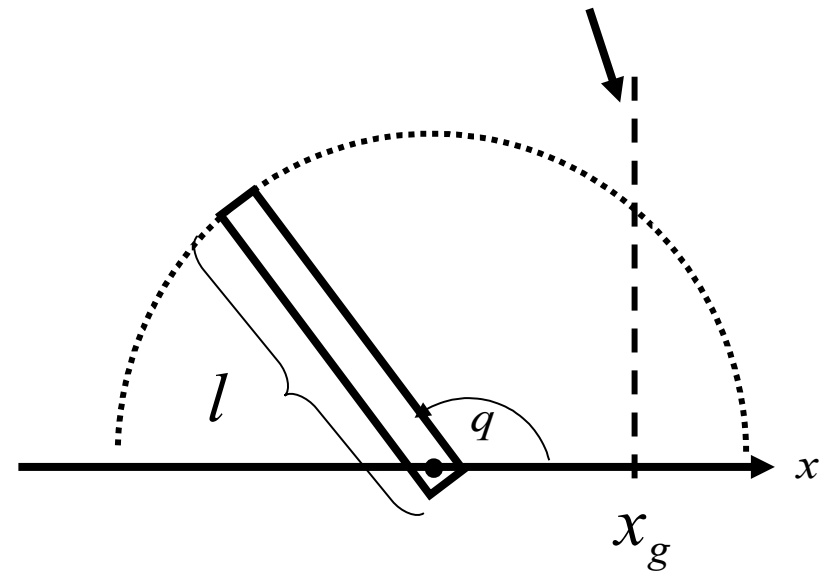
4.

Step size (a constant)

5.

6.

Goal: move the end-effector onto this line



# Example

Numerical solution:

What  $x$  velocity is desirable?

How does that translate to  $q$ ?

1.

2.  $x_i = l \cos(q_i)$

3.  $\delta x = \alpha (x_g - x_i)$

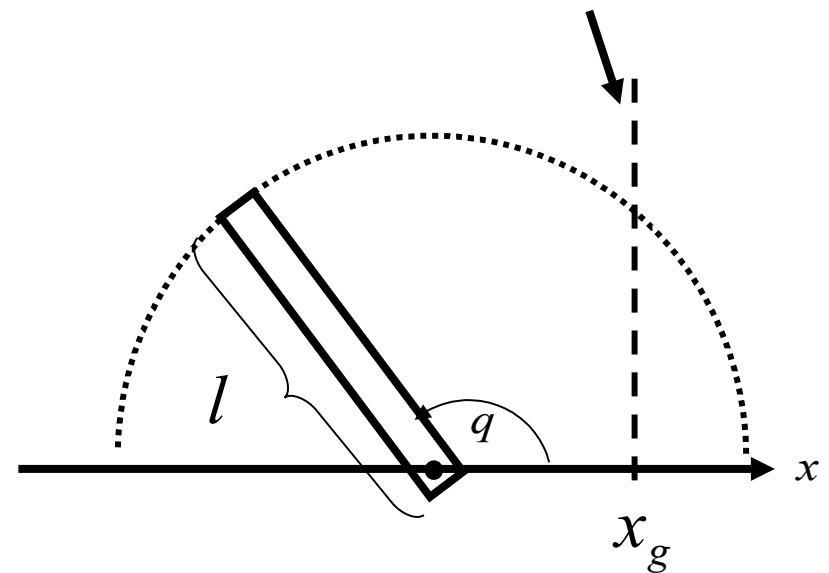
4. 
$$\delta q = \frac{1}{-l \sin(q_i)} \delta x$$

5.

1x1 Jacobian inverse

6.

Goal: move the end-effector onto this line



# Example

Numerical solution:

What  $x$  velocity is desirable?

How does that translate to  $q$ ?

1.

2.  $x_i = l \cos(q_i)$

3.  $\delta x = \alpha(x_g - x_i)$

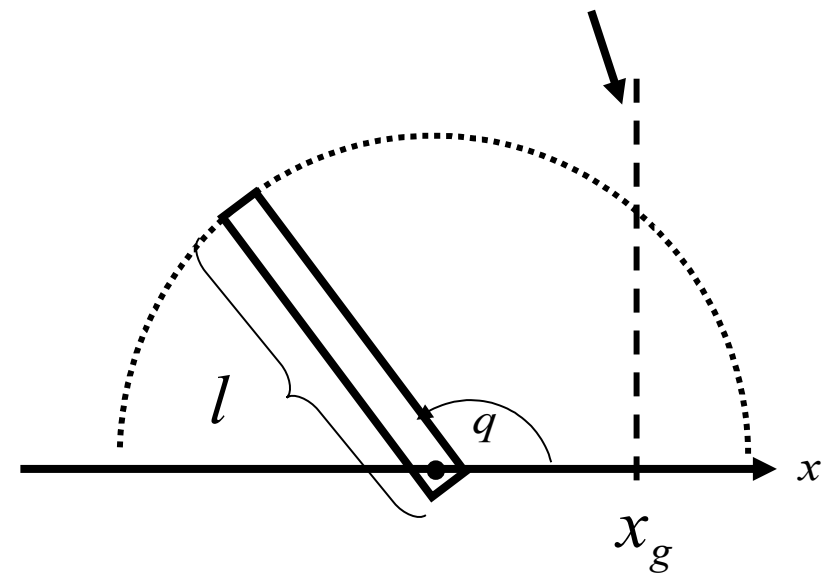
4. 
$$\delta q = \frac{1}{-l \sin(q_i)} \delta x$$

5.

1x1 Jacobian inverse

6.

Goal: move the end-effector onto this line



$$\frac{dx}{dq} = -l \sin(q)$$

$$\delta x = -l \sin(q) \delta q$$

$$\delta q = -\frac{1}{l \sin(q)} \delta x$$

# Example

Numerical solution:

What  $x$  velocity is desirable?

How does that translate to  $q$ ?

1.

2.  $x_i = l \cos(q_i)$

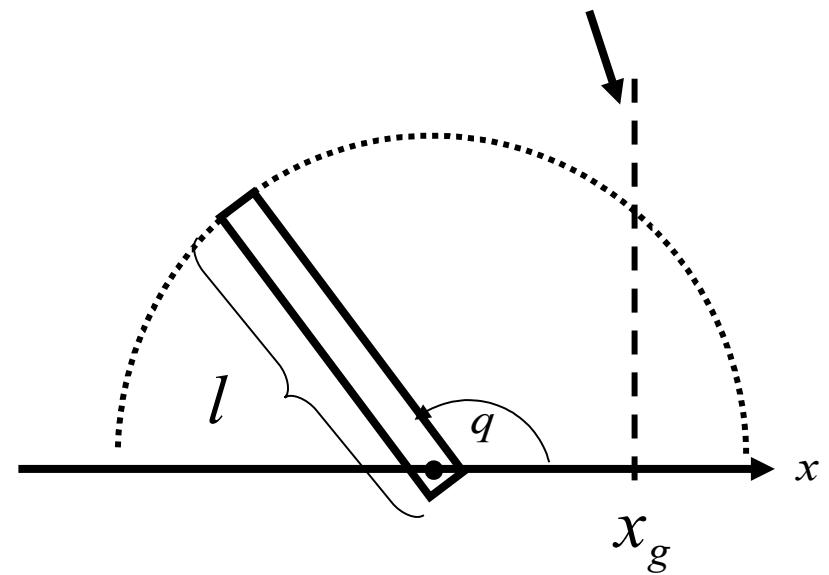
3.  $\delta x = \alpha(x_g - x_i)$

4. 
$$\delta q = \frac{1}{-l \sin(q_i)} \delta x$$

5.  $q_{i+1} = q_i + \delta q$

6.

Goal: move the end-effector onto this line



# Example

Numerical solution:

What  $x$  velocity is desirable?

How does that translate to  $q$ ?

1.  $i = 0, q_0 = \text{arbitrary}$

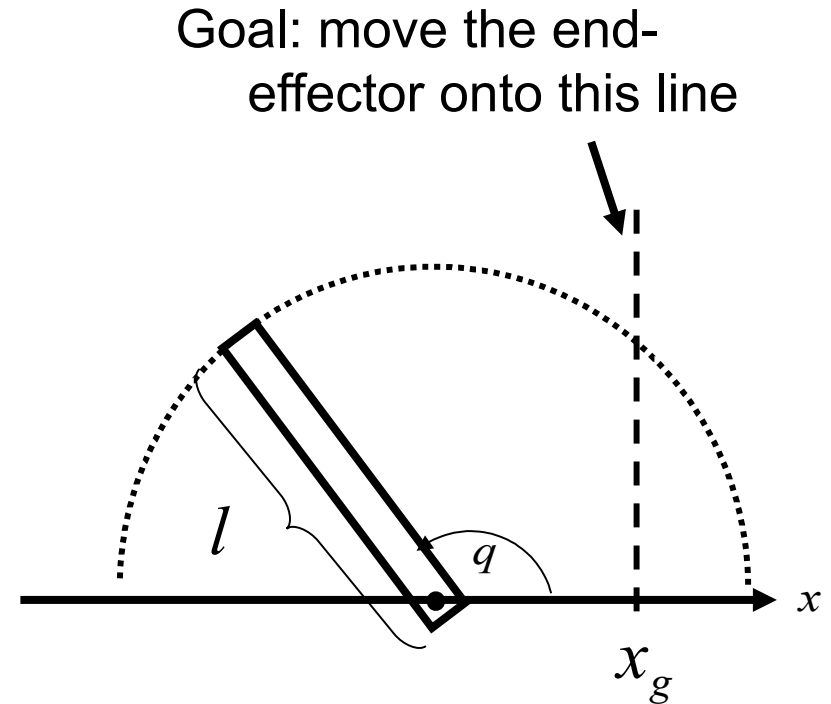
2.  $x_i = l \cos(q_i)$

3.  $\delta x = \alpha(x_g - x_i)$

4. 
$$\delta q = \frac{1}{-l \sin(q_i)} \delta x$$

5.  $q_{i+1} = q_i + \delta q$

6.  $i++$ ; GOTO line 2



1x1 Jacobian inverse

# Numerical IK solution

Input:  $x^*$

Output:  $q^*$

1. repeat until  $dx$  is small:
2. init  $q$  to random joint configuration
3. repeat  $K$  times:
  4.  $x = FK(q)$
  5.  $dx = x^* - x$
  6.  $dq = \text{stepsize} * J^{-1} dx$
  7.  $q = q + dq$
8. return  $q^* = q$

# Numerical IK solution

Input:  $x^*$

Output:  $q^*$

1. repeat until  $dx$  is small:
2. init  $q$  to random joint configuration
3. repeat  $K$  times:
4.      $x = \text{FK}(q)$
5.      $dx = x^* - x$
6.      $dq = \text{stepsize} * J^{-1} dx$
7.      $q = q + dq$
8. return  $q^* = q$

Idea:

$$\dot{q} = J^{-1} \dot{x}$$



# Numerical IK solution

Input:  $x^*$

Output:  $q^*$

1. repeat until  $dx$  is small:
2. init  $q$  to random joint configuration
3. repeat  $K$  times:
4.  $x = \text{FK}(q)$
5.  $dx = x^* - x$
6.  $dq = \text{stepsize} * J^{-1} dx$
7.  $q = q + dq$
8. return  $q^* = q$

Idea:

$$\dot{q} = J^{-1} \dot{x}$$

What if inverse does not exist?

( $J$  non-square or determinant = 0)

Use the pseudoinverse

(MATLAB: `pinv`)

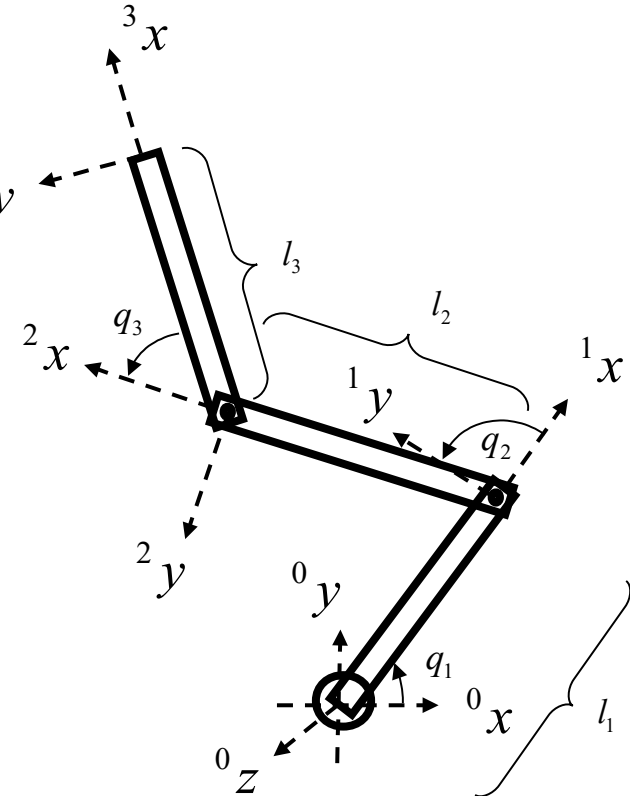
# Non-square Jacobian matrix

Example of a non-square Jacobian matrix:

$$J(q) = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} & -l_1 s_1 - l_2 s_{12} & -l_1 s_1 \\ l_1 c_1 + l_2 c_{12} + l_3 c_{123} & l_1 c_1 + l_2 c_{12} & l_1 c_1 \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = J(q) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix}$$

Two equations of  
three variables



This is an *under-constrained* system of equations:

Multiple solutions –

there are multiple joint angle velocities that realize the same EFF velocity

# Non-square Jacobian matrix

What if inverse does not exist?

(J non-square or determinant = 0)

Use the pseudoinverse (MATLAB: pinv)

# Non-square Jacobian matrix

What if inverse does not exist?

(J non-square or determinant = 0)

Use the pseudoinverse (MATLAB: pinv)

An alternative is to use the pseudo-inverse of the Jacobian  $J^+$  which has the property

$$J^+ J = I$$

just as the inverse does. It is defined as

$$J^+ = (J^T J)^{-1} J^T$$

and is readily computed using the MATLAB® builtin function `pinv`.► The solution

$$\dot{q} = J(q)^+ \nu$$

provides a least-squares solution for which  $\|J\dot{q} - \nu\|$  is smallest.►

# Non-square Jacobian matrix

What if inverse does not exist?

(J non-square or determinant = 0)

Use the pseudoinverse (MATLAB: pinv)

An alternative is to use the pseudo-inverse of the Jacobian  $J^+$  which has the property

$$J^+ J = I$$

just as the inverse does. It is defined as

$$J^+ = (J^T J)^{-1} J^T$$

and is readily computed using the MATLAB® builtin function `pinv`.► The solution


$$\dot{q} = J(q)^+ \nu$$

provides a least-squares solution for which  $\|J\dot{q} - \nu\|$  is smallest.►

(Technically for the over-constrained case, # rows > # cols)

# Non-square Jacobian matrix

The pseudoinverse can be calculated using two different equations depending upon the number of rows and columns:


$$\left\{ \begin{array}{ll} J^{\#} = J^T (J J^T)^{-1} & \text{Underconstrained case} \\ & \text{(if there are more columns than rows } (m < n)) \\ J^{\#} = (J^T J)^{-1} J^T & \text{Overconstrained case} \\ & \text{(if there are more rows than columns } (n < m)) \\ J^{\#} = J^{-1} & \text{If there are} \\ & \text{an equal number of rows and columns } (n = m) \end{array} \right.$$

These equations can only be used if the Jacobian is full rank;  
otherwise, use singular value decomposition (SVD)  
or add small diagonal ( $\text{eps} * I$ ) to  $J J^T$  or  $J^T J$

$$\dot{\mathbf{q}} = (J(\mathbf{q}) + \lambda I)^{-1} \boldsymbol{\nu}$$

where  $\lambda$  is a small constant added to the diagonal

# Numerical IK solution

Input:  $x^*$

Output:  $q^*$

1. repeat until  $dx$  is small:
2. init  $q$  to random joint configuration
3. repeat  $K$  times:
4.  $x = FK(q)$
5.  $dx = x^* - x$
6.  $dq = \text{stepsize} * J^+ dx$
7.  $q = q + dq$
8. return  $q^* = q$

Idea:

$$\dot{q} = J^{-1} \dot{x}$$

What if inverse does not exist?

( $J$  non-square or determinant = 0)

Use the pseudoinverse (MATLAB: `pinv`)

(`pinv` will handle both over-/under-constrained cases)

# Numerical IK solution (alternative)

Input:  $x^*$

Output:  $q^*$

1. repeat until  $dx$  is small:
2. init  $q$  to random joint configuration
3. repeat  $K$  times:
4.  $x = \text{FK}(q)$
5.  $dx = x^* - x$
6.  $dq = \text{stepsize} * J^T dx$
7.  $q = q + dq$
8. return  $q^* = q$

Alternative:

$$\dot{q} = J^T \dot{x}$$

See RVC 8.6.1 (derived based on forces)

- Results in different behavior, but still correct



# Feedback

## Piazza thread: 2/2 Lec 05 Feedback

Please post your answers to the following anonymously.

1. What did you like so far?
2. What was unclear?
3. Any additional feedback / comments?