Computer Graphics
Forward/Inverse kinematics

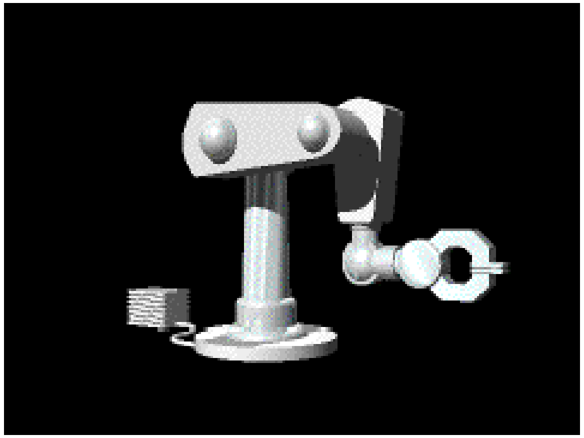
Outline

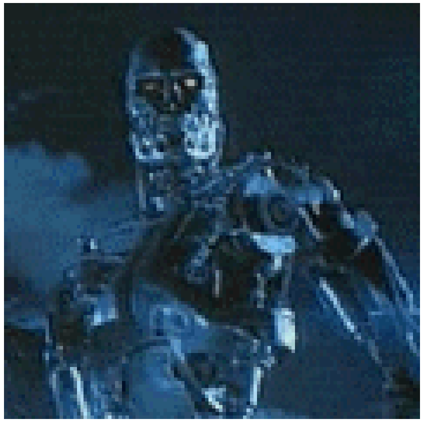
Animation basics:

- – Forward kinematics
- – Inverse kinematics

Kinematics

The study of movement without the consideration of the masses or forces that bring about the motion





Animation

- Robot arm animation (click here)

- Pixar lamp animation (click here)

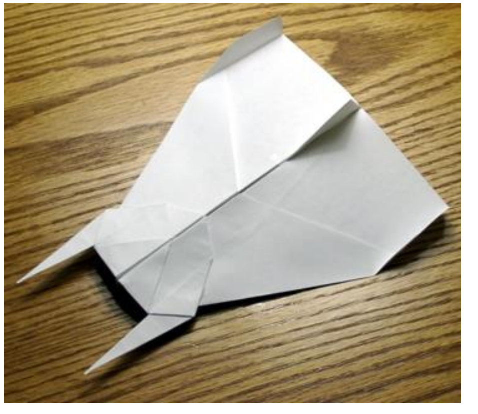
Degrees of Freedom (Dofs)

The set of independent displacements that specify an object's pose

Degrees of Freedom (Dofs)

The set of independent displacements that specify an object's pose

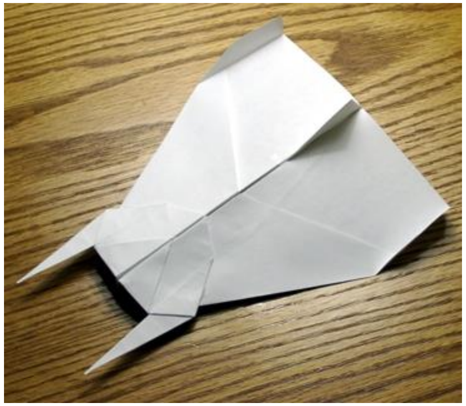
- How many degrees of freedom when flying?



Degrees of Freedom (Dofs)

The set of independent displacements that specify an object's pose

- How many degrees of freedom when flying?



- So the kinematics of this airplane permit movement anywhere in three dimensions
- Six
  - x, y, and z positions
  - roll, pitch, and yaw

Configuration Space vs. Work Space

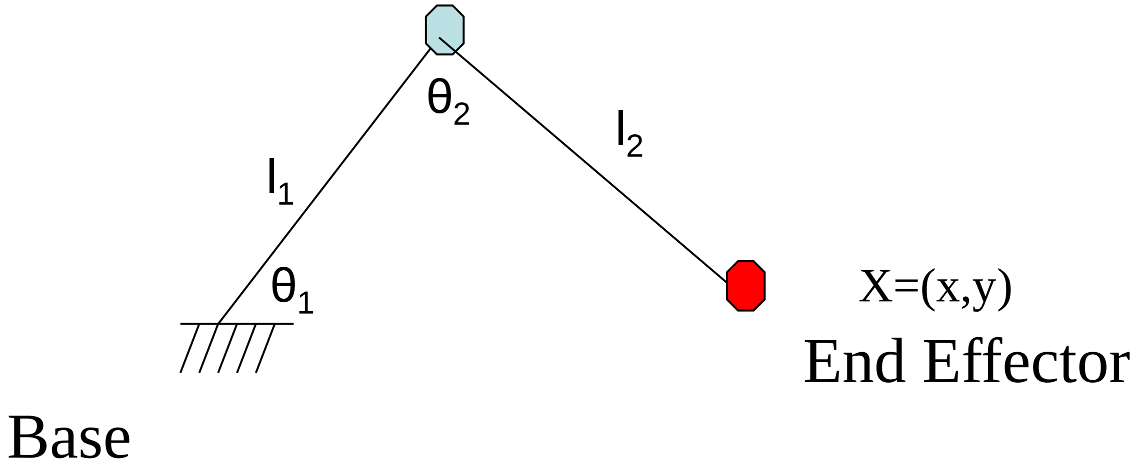
- Configuration space
  - The space that defines the possible object configurations
  - Degrees of Freedom
    - The number of parameters that are necessary and sufficient to define position in configuration
- Work space
  - The space in which the object exists
  - Dimensionality
    - R^3 for most things, R^2 for planar arms

Forward vs. Inverse Kinematics

- Forward Kinematics
  - Compute configuration (pose) given individual DOF values
- Inverse Kinematics
  - Compute individual DOF values that result in specified end effector's position

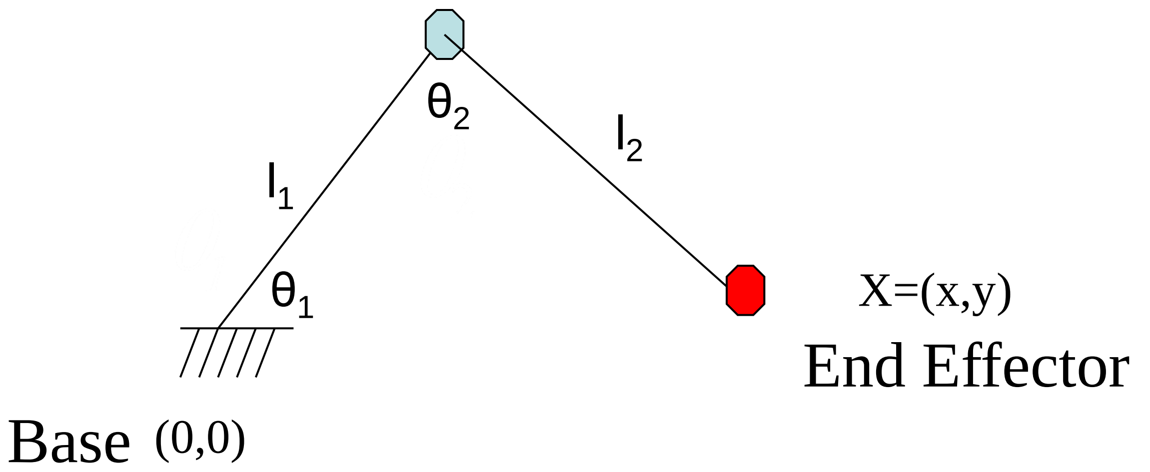
Example: Two-link Structure

- Two links connected by rotational joints



Example: Two-link Structure

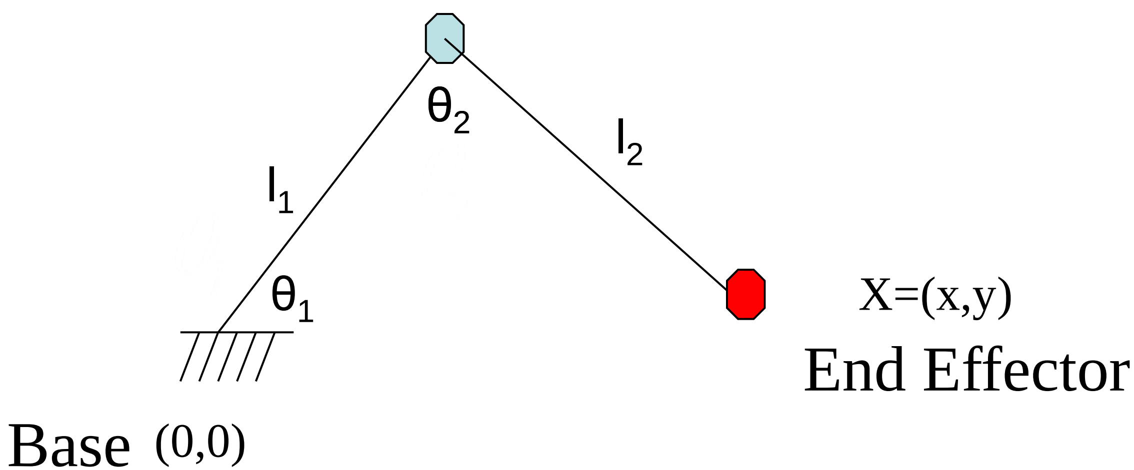
- Animator specifies the joint angles: $\theta_1 \theta_2$
- Computer finds the position of end-effector: x



$$\mathbf{x=f(}\theta_1\mathbf{,}\ \theta_2\mathbf{)}$$

Example: Two-link Structure

- Animator specifies the joint angles: $\theta_1\theta_2$
- Computer finds the position of end-effector: x

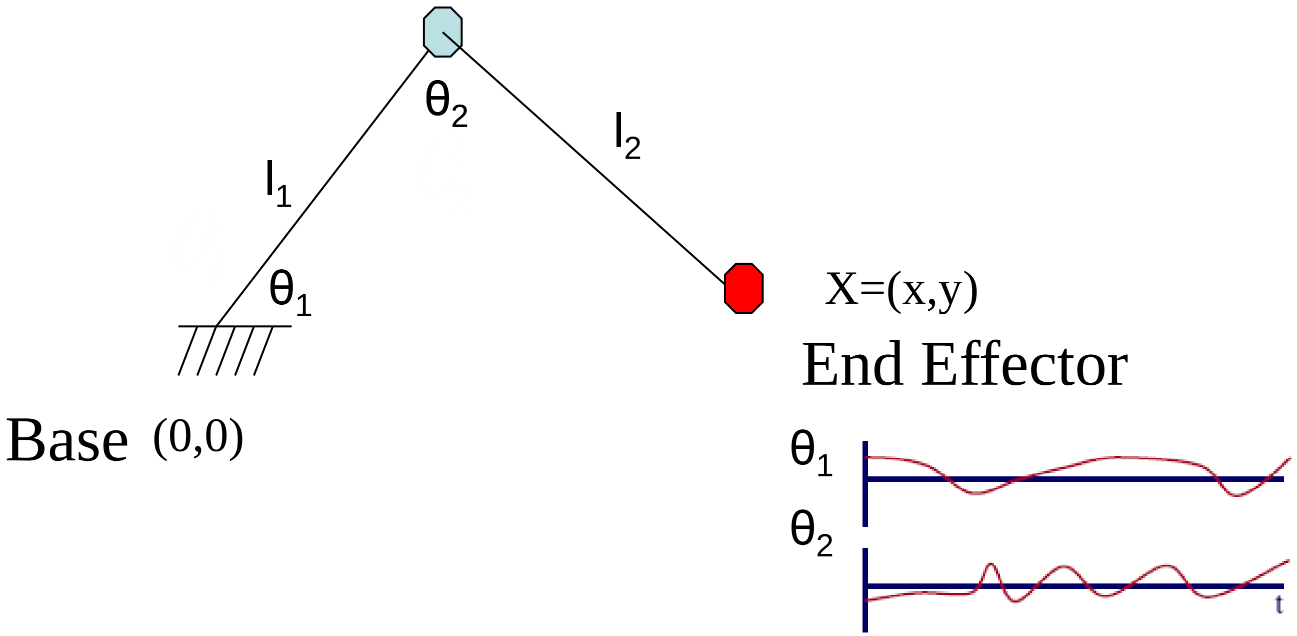


$$x = (l_1\cos\theta_1 + l_2\cos(\theta_2 + \theta_2)$$

$$y = l_1\sin\theta_1 + l_2\sin(\theta_2 + \theta_2))$$

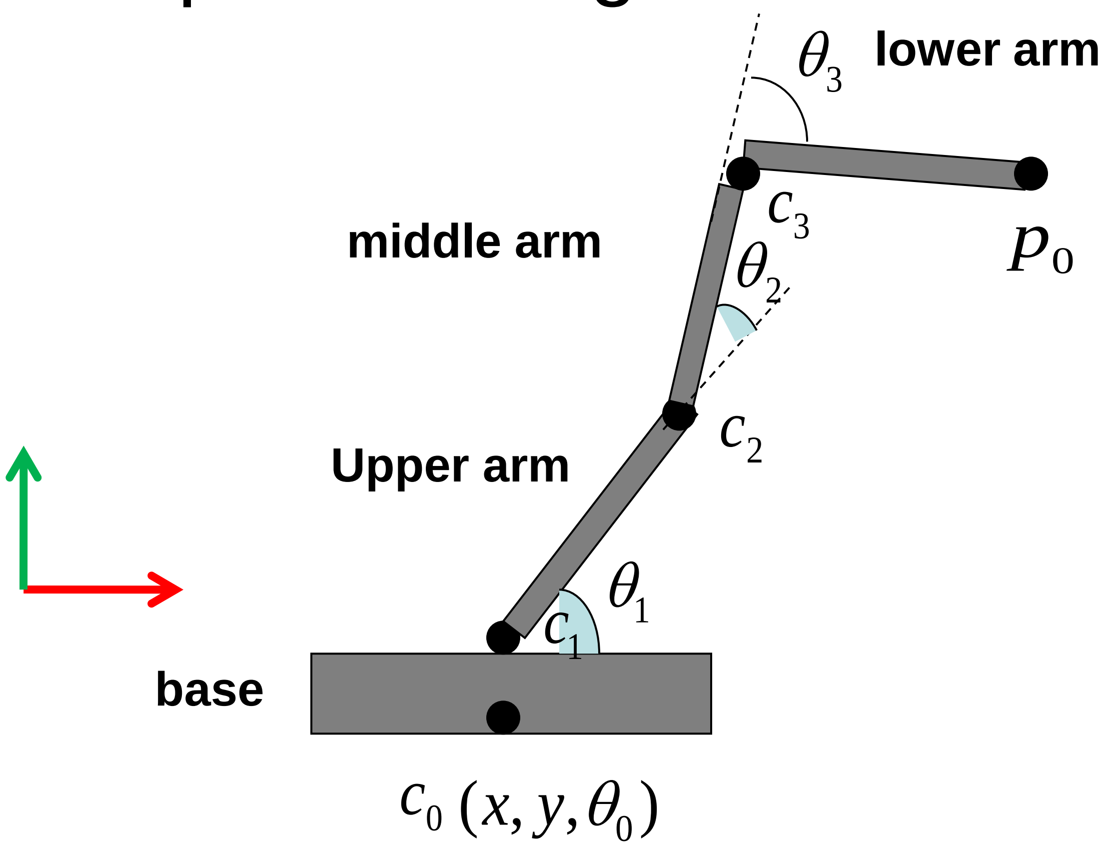
Forward Kinematics

- Create an animation by specifying the joint angle trajectories



Forward Kinematics

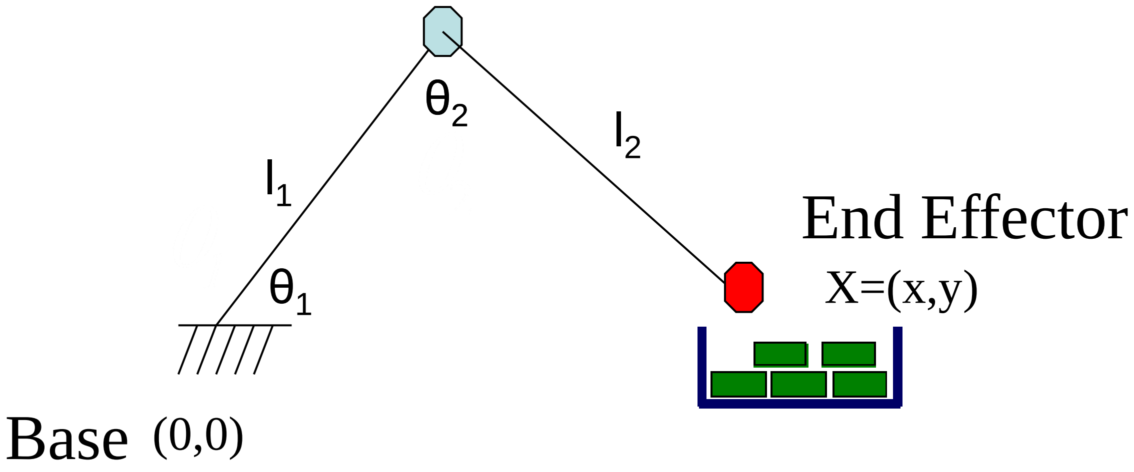
- A 2D lamp with 6 degrees of freedom



$$f(x, y, \theta_0, \theta_1, \theta_2, \theta_3) = T(x, y)R(\theta_0)T(0, l_0)R(\theta_1)T(l_1, 0)R(\theta_2)T(l_2, 0)R(\theta_3)p_0$$

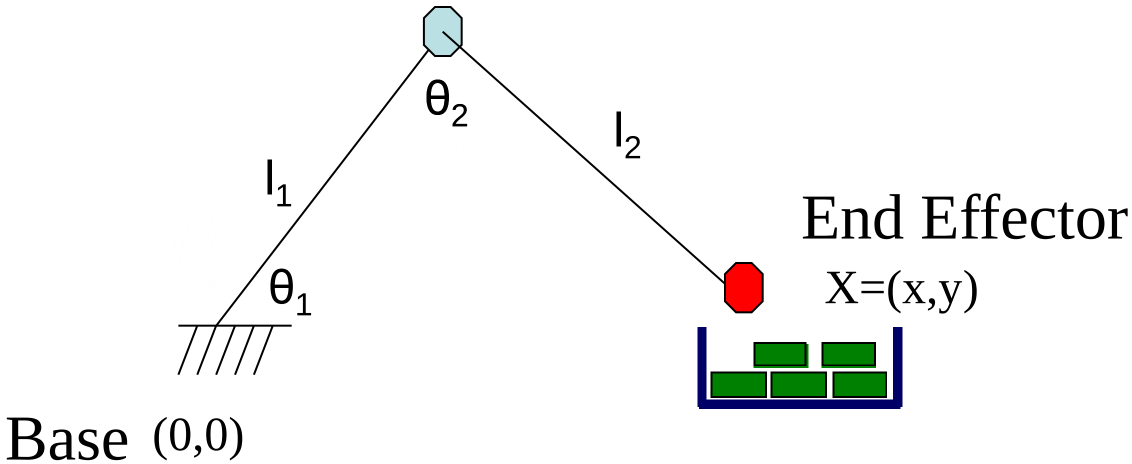
Inverse Kinematics

- What if an animator specifies position of end-effector?



Inverse Kinematics

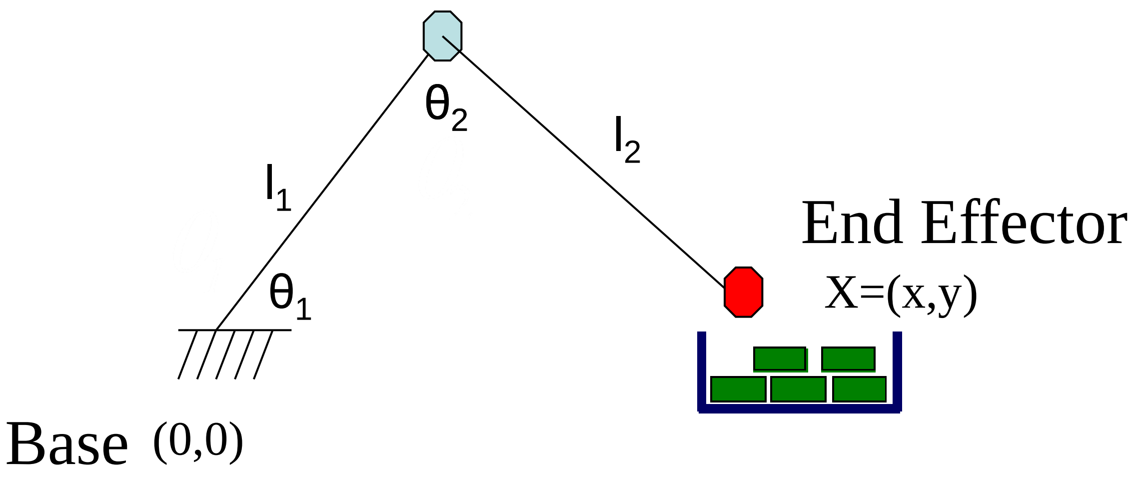
- Animator specifies the position of end-effector: x
- Computer finds joint angles: $\theta_1 \theta_2$



$$(\theta_1, \theta_2) = f^{-1}(x)$$

Inverse Kinematics

- Animator specifies the position of end-effector: x
- Computer finds joint angles: $\theta_1 \theta_2$



$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + x^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right)$$

$$\Theta_1 = \frac{-(l_2 \sin(\Theta_2)x + (l_1 + l_2 \cos(\Theta_2))y}{(l_2 \sin(\Theta_2))y + (l_1 + l_2 \cos(\Theta_2))x}$$

Why Inverse Kinematics?

- Motion capture
- Basic tools in character animation
  - key frame generation
  - animation control
  - interactive manipulation
- Computer vision (video-based mocap)
- Robotics
- Bioinfomatics (Protein Inverse Kinematics)
- Etc.

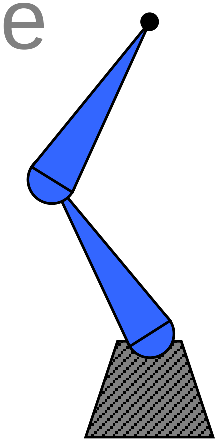
Inverse Kinematics

- Given end effector's positions, compute required joint angles

- In simple case, analytic solution exists
  - Use trig, geometry, and algebra to solve

Inverse Kinematics

- Analytical solution only works for a fairly simple structure



- Numerical/iterative solution needed for a complex structure

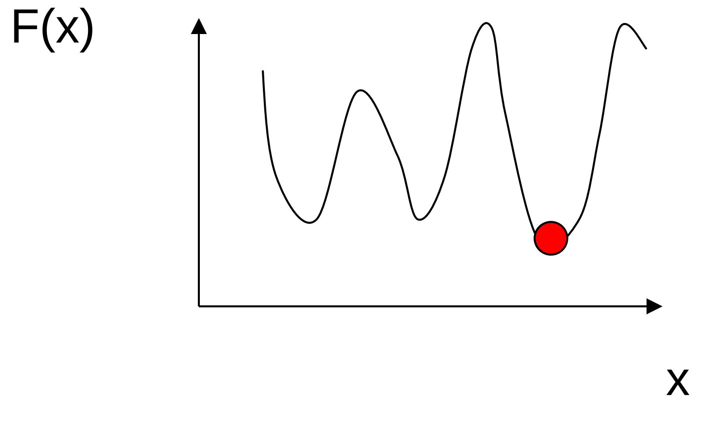
Numerical Approaches

- Inverse kinematics can be formulated as an optimization problem

Function Optimization

- Finding the minimum for nonlinear functions

$$\text{Given } F:\ \mathbb{R}^n \mapsto \mathbb{R}. \text{ Find}$$
$$\mathbf{x}^+ \;=\; \mathrm{argmin}_{\mathbf{x}}\{F(\mathbf{x})\}$$



Optimization As a Tool

- How to use optimization to solve the following linear system?

$$3x + 2y = 5;$$

$$4x - 5y = 6$$

Optimization As a Tool

- How to use optimization to solve the following linear system?

$$3x + 2y = 5;$$

$$4x - 5y = 6$$

Define a function:

$$F(x,y) = (3x+2y-5)^2+(4x-5y-6)^2$$

Optimization As a Tool

- How to use optimization to solve the following linear system?

$3x + 2y = 5;$

$4x - 5y = 6$

Could be nonlinear equations!

Define a function:

$$F(x,y) = (3x+2y-5)^2+(4x-5y-6)^2$$

Finding the minimum for F(x,y) :

$$x^+,y^+= \mathrm{argmin}_{x,y}\ (3x+2y-5)^2+(4x-5y-6)^2$$

Optimization As a Tool

- How to use optimization to solve the following linear system?

$f(x, y) = 0;$

$g(x,y) = 0$

Could be nonlinear equations!

Define a function:

$$F(x,y) = f(x,y)^2+g(x,y)^2$$

Finding the minimum for F(x,y) :

$$x^+,y^+= \mathrm{argmin}_{x,y}\ f(x,y)^2+g(x,y)^2$$

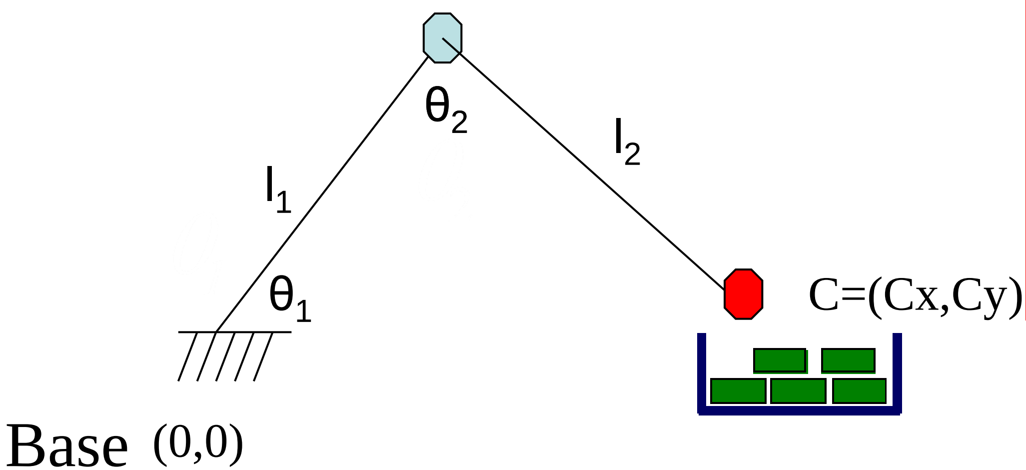
Formulation

- So how to convert the IK process into an optimization function?

$$\arg\min_{\theta} \boxed{F(\theta)}$$

Find the pose satisfying:

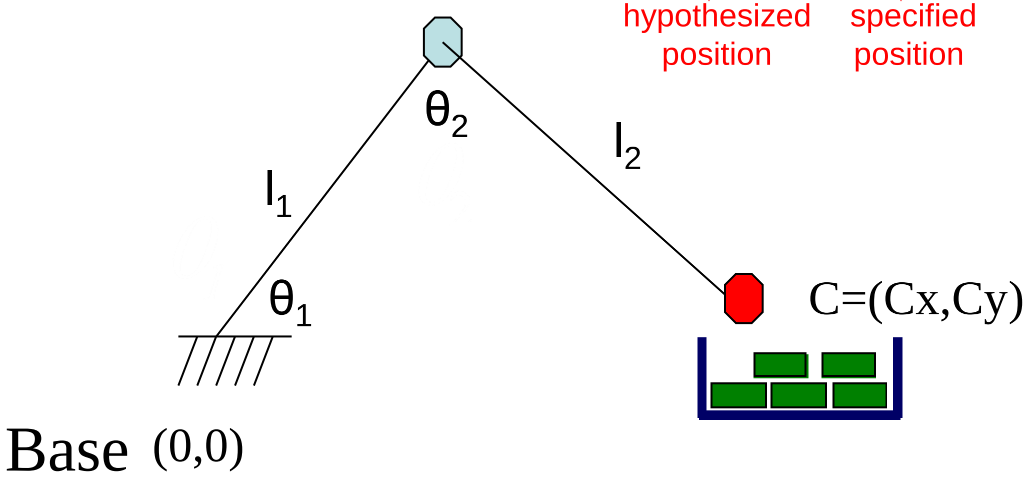
$$f_i(\theta) = C_i$$



Iterative Approaches

Find the joint angles θ that minimizes the distance between the hypothesized character position and user specified position

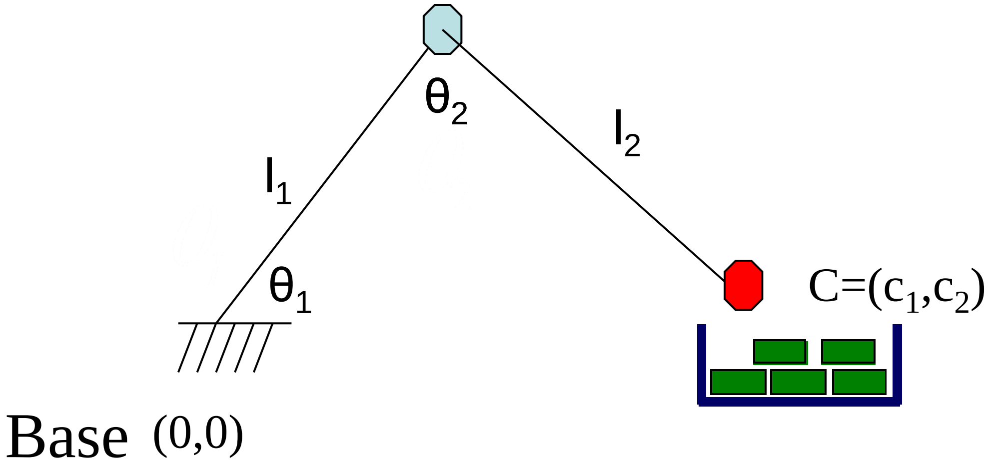
$$\arg\min_{\theta} \sum_i \| [\underbrace{f_i(\theta)}_{\text{hypothesized position}} - \underbrace{C_i}_{\text{specified position}}] \|^2$$



Iterative Approaches

Find the joint angles θ that minimizes the distance between the hypothesized character position and user specified position

$$\arg\min_{\theta_1,\theta_2} (l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) - c_1)^2 + (l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2) - c_2)^2$$



Iterative Approaches

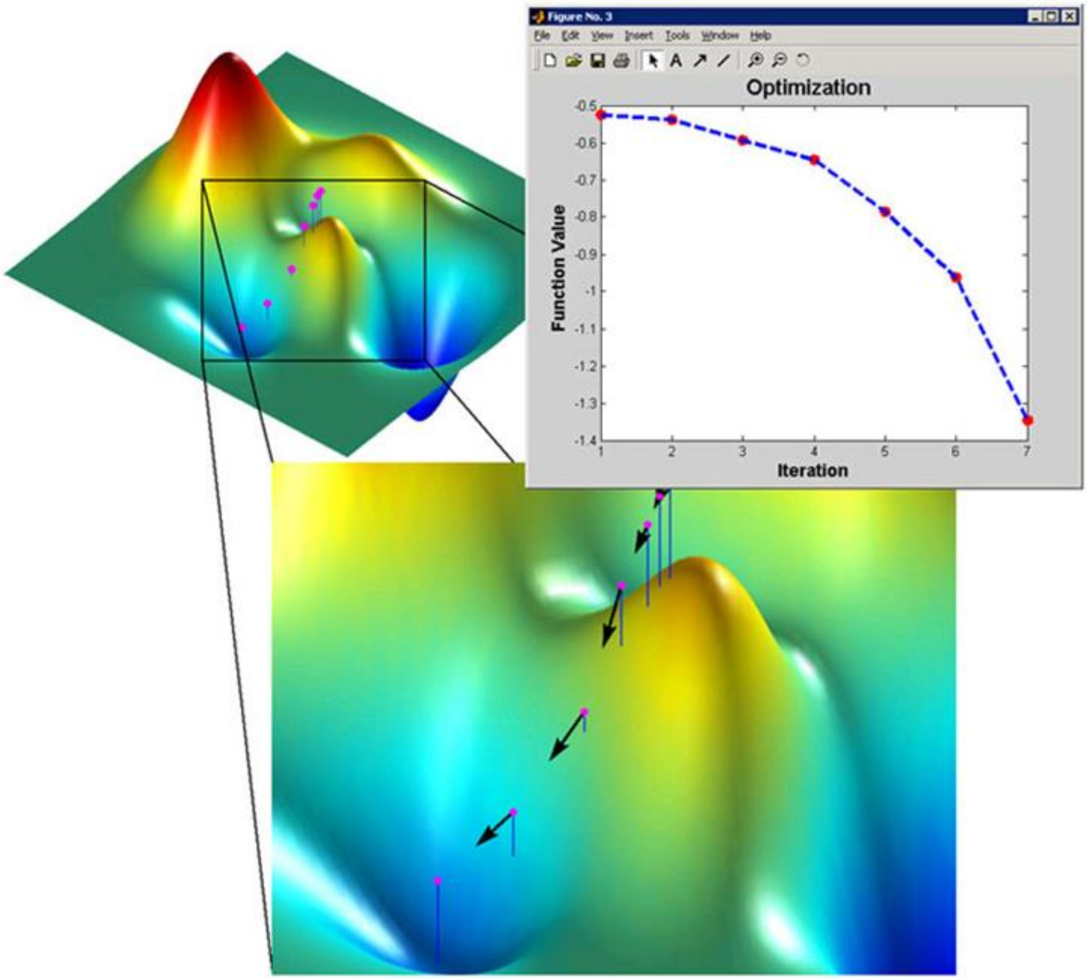
Mathematically, we can formulate this as an optimization problem:

$$\arg\min_{\theta} \sum_i \left[f_i(\theta) - c_i \right]^2$$

The above problem can be solved by many nonlinear optimization algorithms:

- Steepest descent
- Gauss-newton
- Levenberg-marquardt, etc

Gradient-based Optimization



Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles until the solution converges:

$$\theta^{k+1} = \theta^k + \Delta\theta^k$$

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles until the solution converges:

$$\theta^{k+1} = \theta^k + \boxed{\Delta\theta^k}$$

How can we decide the amount of update?

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles: $\theta^{k+1} = \theta^k + \Delta\theta^k$

$$\arg\min_{\Delta\theta} \sum_i \left[f_i(\theta^k + \Delta\theta^k) - c_i \right]^2$$

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles: $\theta^{k+1} = \theta^k + \Delta\theta^k$

$$\arg\min_{\Delta\theta} \sum_i \left[f_i(\boxed{\theta^k} + \Delta\theta^k) - \boxed{c_i} \right]^2$$

Known!

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles: $\theta^{k+1} = \theta^k + \Delta\theta^k$

$$\arg\min_{\Delta\theta^k} \sum_i \left[f_i(\theta^k + \Delta\theta^k) - c_i \right]^2$$

Taylor series expansion

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles: $\theta^{k+1} = \theta^k + \Delta\theta^k$

$$\arg\min_{\Delta\theta^k} \sum_i \left[f_i(\theta^k + \Delta\theta^k) - c_i \right]^2$$

Taylor series expansion

$$= \arg\min_{\Delta\theta^k} \sum_i \left[f_i(\theta^k) + \frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - c_i \right]^2$$

rearrange

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles: $\theta^{k+1} = \theta^k + \Delta\theta^k$

$$\arg\min_{\Delta\theta^k} \sum_i \left[f_i(\theta^k + \Delta\theta^k) - c_i \right]^2$$

Taylor series expansion: linear approximation

$$= \arg\min_{\Delta\theta^k} \sum_i \left[f_i(\theta^k) + \frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - c_i \right]^2$$

rearrange

Can you solve this optimization problem?

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles: $\theta^{k+1} = \theta^k + \Delta\theta^k$

$$\arg\min_{\Delta\theta^k} \sum_i \left[f_i(\theta^k + \Delta\theta^k) - c_i \right]^2$$

Taylor series expansion

$$= \arg\min_{\Delta\theta^k} \sum_i \left[f_i(\theta^k) + \frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - c_i \right]^2$$

rearrange

$$= \arg\min_{\Delta\theta^k} \sum_i \left[\frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - (c_i - f_i(\theta^k)) \right]^2$$

This is a quadratic function of $\Delta\theta$

Gauss-Newton Approach

- Optimizing an quadratic function is easy

$$\arg\min_{\Delta\theta^k} \sum_i \left[\frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - (c_i - f_i(\theta^k)) \right]^2$$

- It has an optimal value when the gradient is zero

$$\begin{vmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_N} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots & \frac{\partial f_2}{\partial \theta_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial \theta_N} & \frac{\partial f_M}{\partial \theta_2} & \cdots & \frac{\partial f_M}{\partial \theta_N} \end{vmatrix} \cdot \begin{vmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_N \end{vmatrix} = \begin{vmatrix} c_1 - f_i(\theta_1, \ldots, \theta_N) \\ c_2 - f_2(\theta_1, \ldots, \theta_N) \\ \vdots \\ c_N - f_i(\theta_1, \ldots, \theta_N) \end{vmatrix}$$

Gauss-Newton Approach

- Optimizing an quadratic function is easy

$$\arg\min_{\Delta\theta} \sum_i \left[\frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - (c_i - f_i(\theta^k)) \right]^2$$

- It has an optimal value when the gradient is zero

$$\underbrace{\begin{vmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_N} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots & \frac{\partial f_2}{\partial \theta_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial \theta_N} & \frac{\partial f_M}{\partial \theta_2} & \cdots & \frac{\partial f_M}{\partial \theta_N} \end{vmatrix}}_{J} \cdot \underbrace{\begin{vmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_N \end{vmatrix}}_{\Delta\theta} = \underbrace{\begin{vmatrix} c_1 - f_i(\theta_1, \ldots, \theta_N) \\ c_2 - f_2(\theta_1, \ldots, \theta_N) \\ \vdots \\ c_N - f_i(\theta_1, \ldots, \theta_N) \end{vmatrix}}_{b}$$

Linear equation!

Gauss-Newton Approach

- Optimizing an quadratic function is easy

$$\arg\min_{\Delta\theta} \sum_i \left[\frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - (c_i - f_i(\theta^k)) \right]^2$$

- It has an optimal value when the gradient is zero

$$\underbrace{\begin{vmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_N} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots & \frac{\partial f_2}{\partial \theta_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial \theta_N} & \frac{\partial f_M}{\partial \theta_2} & \cdots & \frac{\partial f_M}{\partial \theta_N} \end{vmatrix}}_{J} \cdot \underbrace{\begin{vmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_N \end{vmatrix}}_{\Delta\theta} = \underbrace{\begin{vmatrix} c_1 - f_i(\theta_1, \ldots, \theta_N) \\ c_2 - f_2(\theta_1, \ldots, \theta_N) \\ \vdots \\ c_N - f_i(\theta_1, \ldots, \theta_N) \end{vmatrix}}_{b}$$

Gauss-Newton Approach

- Optimizing an quadratic function is easy

$$\arg\min_{\Delta\theta} \sum_i \left[\frac{\partial f_i}{\partial \theta^k} \Delta\theta^k - (c_i - f_i(\theta^k)) \right]^2$$

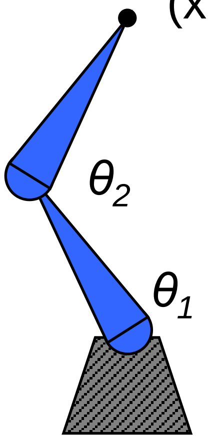
- It has an optimal value when the gradient is zero

$$\underbrace{\begin{vmatrix} \frac{\partial f_1}{\partial \theta_1} & \frac{\partial f_1}{\partial \theta_2} & \cdots & \frac{\partial f_1}{\partial \theta_N} \\ \frac{\partial f_2}{\partial \theta_1} & \frac{\partial f_2}{\partial \theta_2} & \cdots & \frac{\partial f_2}{\partial \theta_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial \theta_N} & \frac{\partial f_M}{\partial \theta_2} & \cdots & \frac{\partial f_M}{\partial \theta_N} \end{vmatrix}}_{J} \cdot \underbrace{\begin{vmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_N \end{vmatrix}}_{\Delta\theta} = \underbrace{\begin{vmatrix} c_1 - f_i(\theta_1, \ldots, \theta_N) \\ c_2 - f_2(\theta_1, \ldots, \theta_N) \\ \vdots \\ c_N - f_i(\theta_1, \ldots, \theta_N) \end{vmatrix}}_{b}$$

$$\Delta\theta = (J^T J)^{-1} J^T b$$

Jacobian Matrix

- Jacobian maps the velocity in joint angle space to velocities in Cartesian space

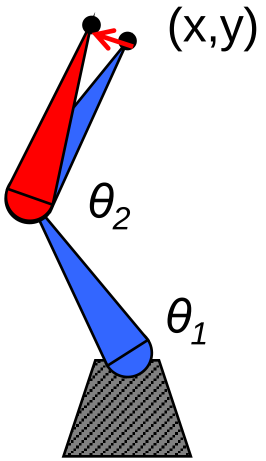


A small change of θ_1 and θ_2 results in how much change of end-effector position *(x,y)*

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{vmatrix} \dfrac{\partial f_1}{\partial \theta_1} & \dfrac{\partial f_1}{\partial \theta_2} \\ \dfrac{\partial f_2}{\partial \theta_1} & \dfrac{\partial f_1}{\partial \theta_2} \end{vmatrix} \begin{pmatrix} \Delta \theta_1 \\ \Delta \theta_2 \end{pmatrix}$$

Jacobian Matrix

- Jacobian maps the velocity in joint angle space to velocities in Cartesian space



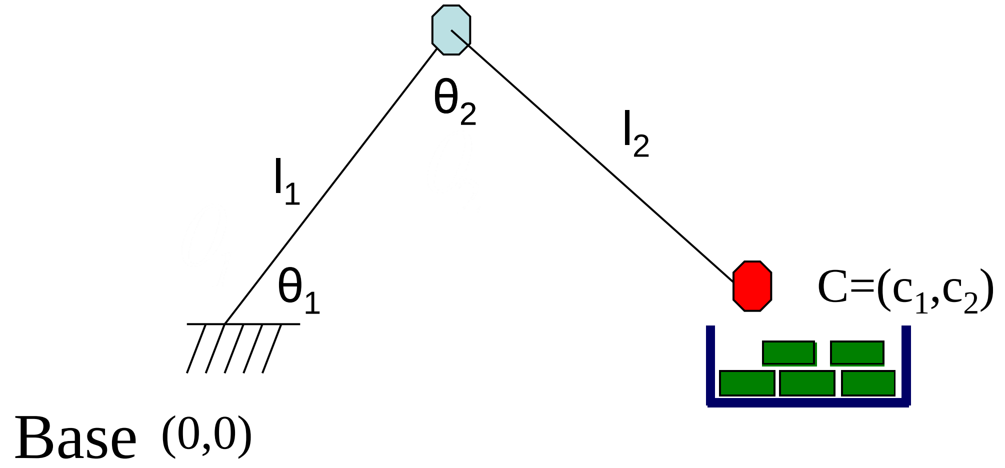
A small change of θ_2 results in how much change of end-effector position *f=(x,y)*

$$\theta_2 \Rightarrow \theta_2 + \Delta\theta_2$$

$$x \Rightarrow x + \Delta x$$

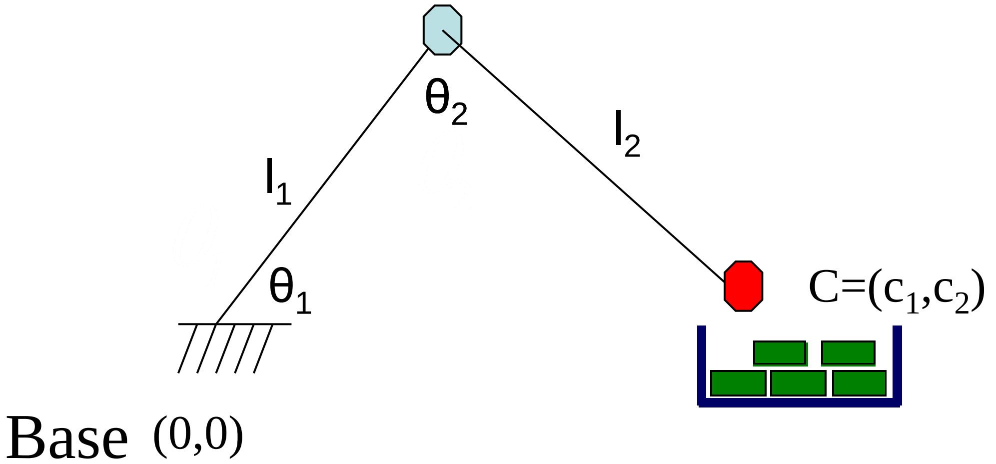
$$y \Rightarrow y + \Delta y$$

Jacobian Matrix: 2D-link Structure



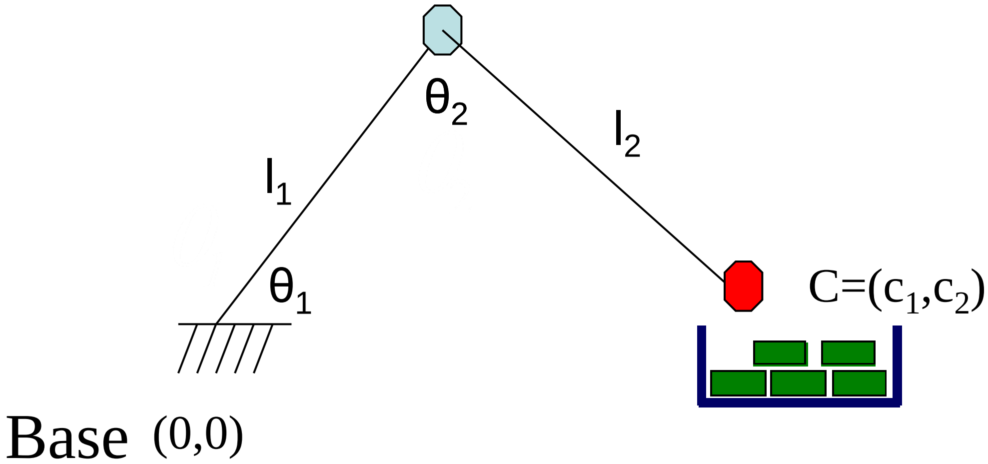
$$\arg\min_{\theta_1,\theta_2} (l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) - c_1)^2 + (l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) - c_2)^2$$

Jacobian Matrix: 2D-link Structure



$$\arg\min_{\theta_1,\theta_2}(\underbrace{l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2)}_{f_1} - c_1)^2 + (\underbrace{l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2)}_{f_2} - c_2)^2$$

Jacobian Matrix: 2D-link Structure

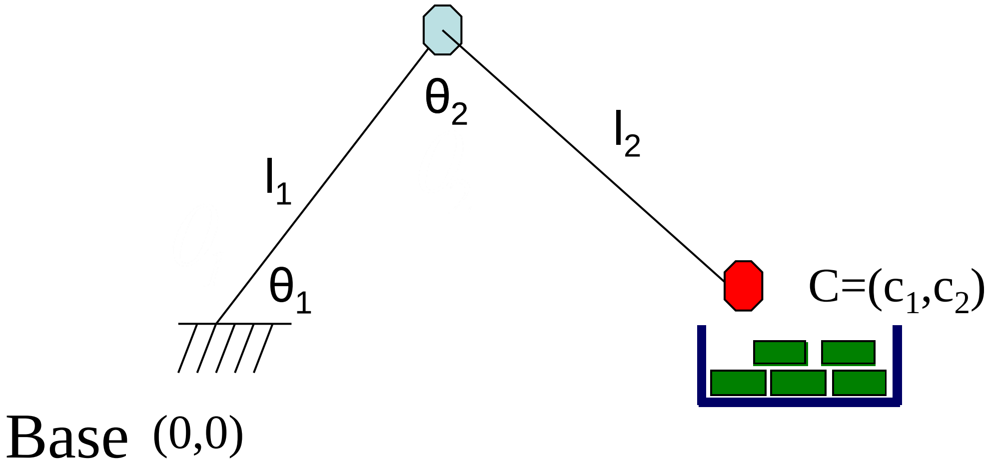


$$\arg\min_{\theta_1,\theta_2} (l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) - c_1)^2 + (l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) - c_2)^2$$

$$f_1 = l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2)$$
$$f_2 = l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2)$$

$$\begin{vmatrix} \frac{\partial f_1}{\partial\theta_1} & \frac{\partial f_1}{\partial\theta_2} & \cdots & \frac{\partial f_1}{\partial\theta_N} \\ \frac{\partial f_2}{\partial\theta_1} & \frac{\partial f_2}{\partial\theta_2} & \cdots & \frac{\partial f_2}{\partial\theta_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_M}{\partial\theta_1} & \frac{\partial f_M}{\partial\theta_2} & \cdots & \frac{\partial f_M}{\partial\theta_N} \end{vmatrix}$$

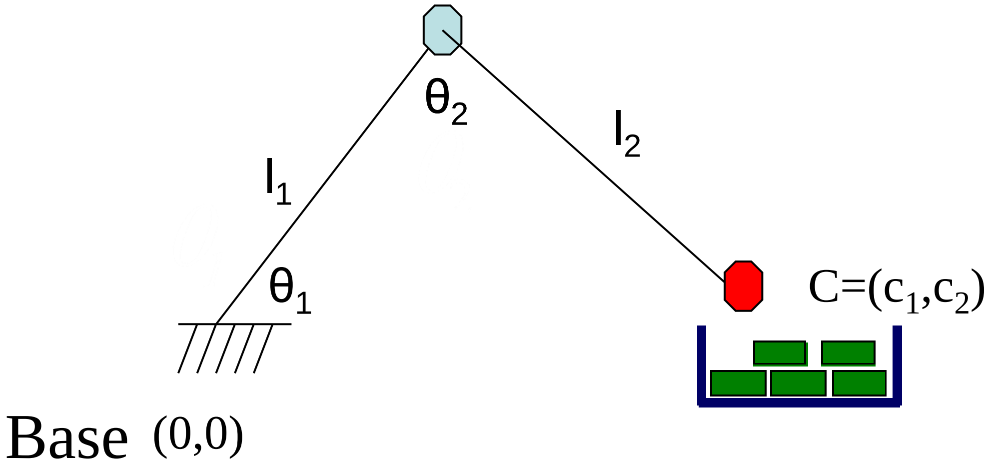
Jacobian Matrix: 2D-link Structure



$$\arg\min_{\theta_1,\theta_2} (l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) - c_1)^2 + (l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2) - c_2)^2$$

$$J = \begin{vmatrix} -l_1 \sin\theta_1 - l_2 \sin(\theta_1+\theta_2) & -l_2 \sin(\theta_1+\theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) & l_2 \cos(\theta_1+\theta_2) \end{vmatrix}$$

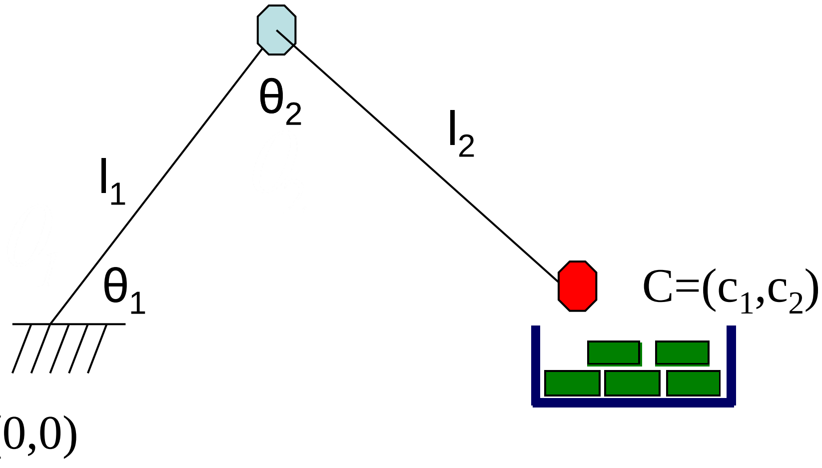
Jacobian Matrix: 2D-link Structure



$$\arg\min_{\theta_1,\theta_2} (l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) - c_1)^2 + (l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2) - c_2)^2$$

$$J = \begin{vmatrix} \boxed{-l_1 \sin\theta_1 - l_2 \sin(\theta_1+\theta_2)} & -l_2 \sin(\theta_1+\theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) & l_2 \cos(\theta_1+\theta_2) \end{vmatrix}$$

Jacobian Matrix: 2D-link Structure

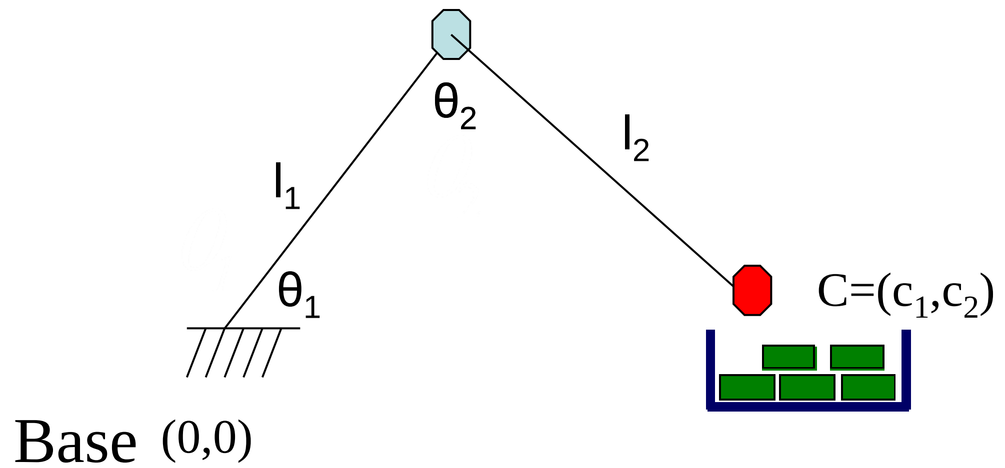


$$\arg\min_{\theta_1,\theta_2} (l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) - c_1)^2 + (l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2) - c_2)^2$$

$$\frac{\partial f_1}{\partial \theta_1} = \frac{\partial (l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2))}{\partial \theta_1}$$

$$J = \begin{vmatrix} \boxed{-l_1 \sin\theta_1 - l_2 \sin(\theta_1+\theta_2)} & -l_2 \sin(\theta_1+\theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2) & l_2 \cos(\theta_1+\theta_2) \end{vmatrix}$$

Jacobian Matrix: 2D-link Structure



$$\arg\min_{\theta_1,\theta_2} (l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) - c_1)^2 + (l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2) - c_2)^2$$

$$\frac{\partial f_2}{\partial \theta_2} = \frac{\partial(l_1\sin\theta_1 + l_2\sin(\theta_1+\theta_2))}{\partial \theta_2}$$

$$J = \begin{vmatrix} -l_1\sin\theta_1 - l_2\sin(\theta_1+\theta_2) & -l_2\sin(\theta_1+\theta_2) \\ l_1\cos\theta_1 + l_2\cos(\theta_1+\theta_2) & \boxed{l_2\cos(\theta_1+\theta_2)} \end{vmatrix}$$

Gauss-Newton Approach

Step 1: initialize the joint angles with θ^0

Step 2: update the joint angles:

$$\theta^{k+1} = \theta^k + \boxed{\lambda}\Delta\theta^k$$

Step size: specified by the user

When to stop

Option #1: When the change of solution from previous iteration to the current one is below a user-specified threshold.

Option #2: When the number of iteration reaches a user-specified threshold.

Option #3: either #1 or #2 is satisfied.

Iterative Approaches for IK

Mathematically, we can formulate this as an optimization problem:

$$\arg\min_{\theta} \sum_i \left[f_i(\theta) - c_i \right]^2$$

The above problem can be solved by many nonlinear optimization algorithms:

- Steepest descent
- Gauss-newton
- Levenberg-marquardt, etc

C/C++ Optimization Library

- levmar : Levenberg-Marquardt nonlinear least squares algorithms in C/C+

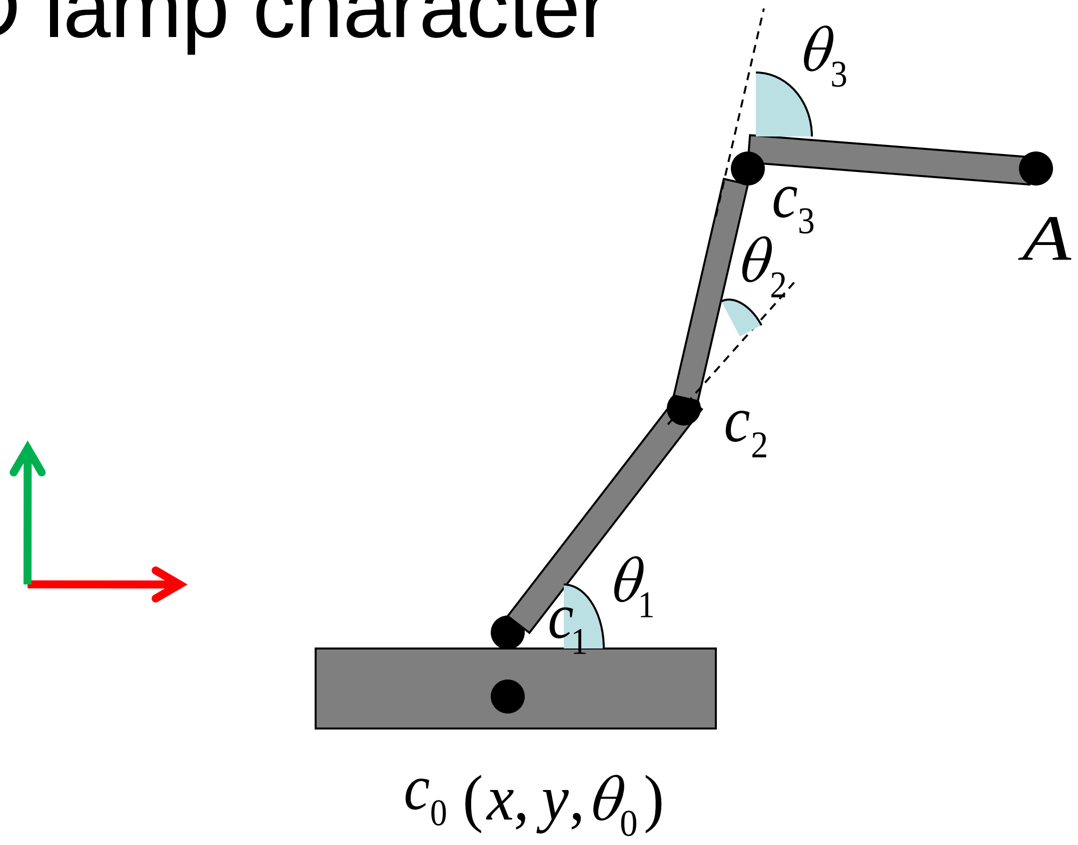
  - Works with/without analytical Jacobian matrix

  - Speeds up the optimization process with analytical Jacobian matrix.

- http://www.ics.forth.gr/~lourakis/levmar/

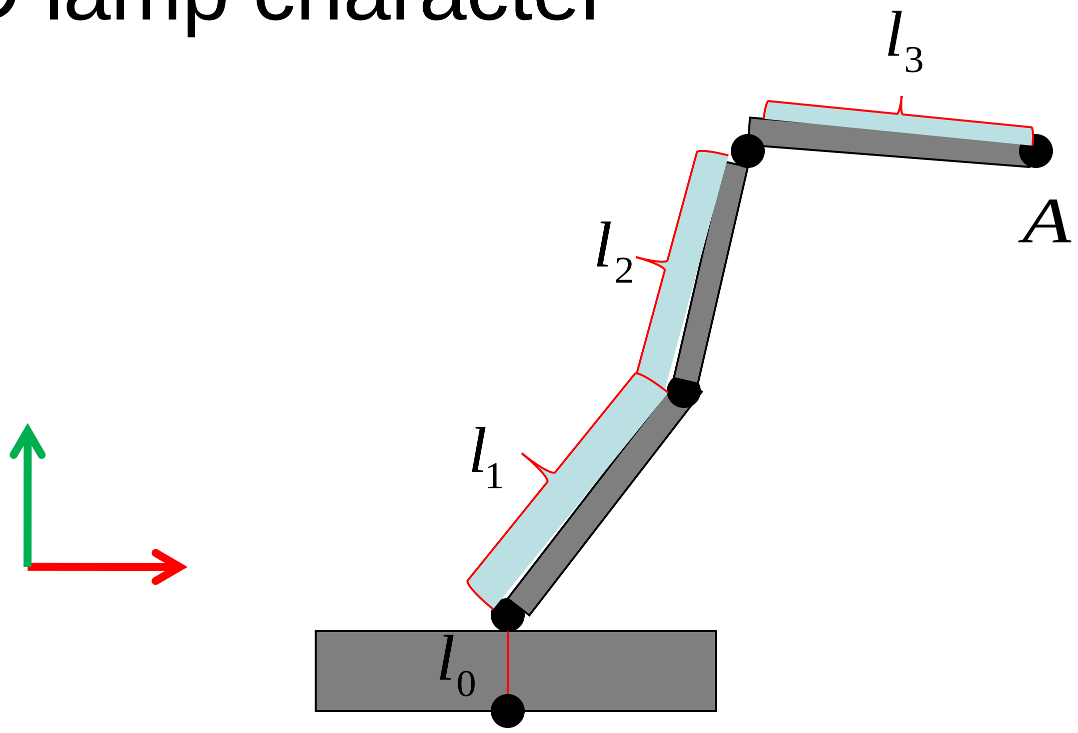
Another Example

- A 2D lamp character



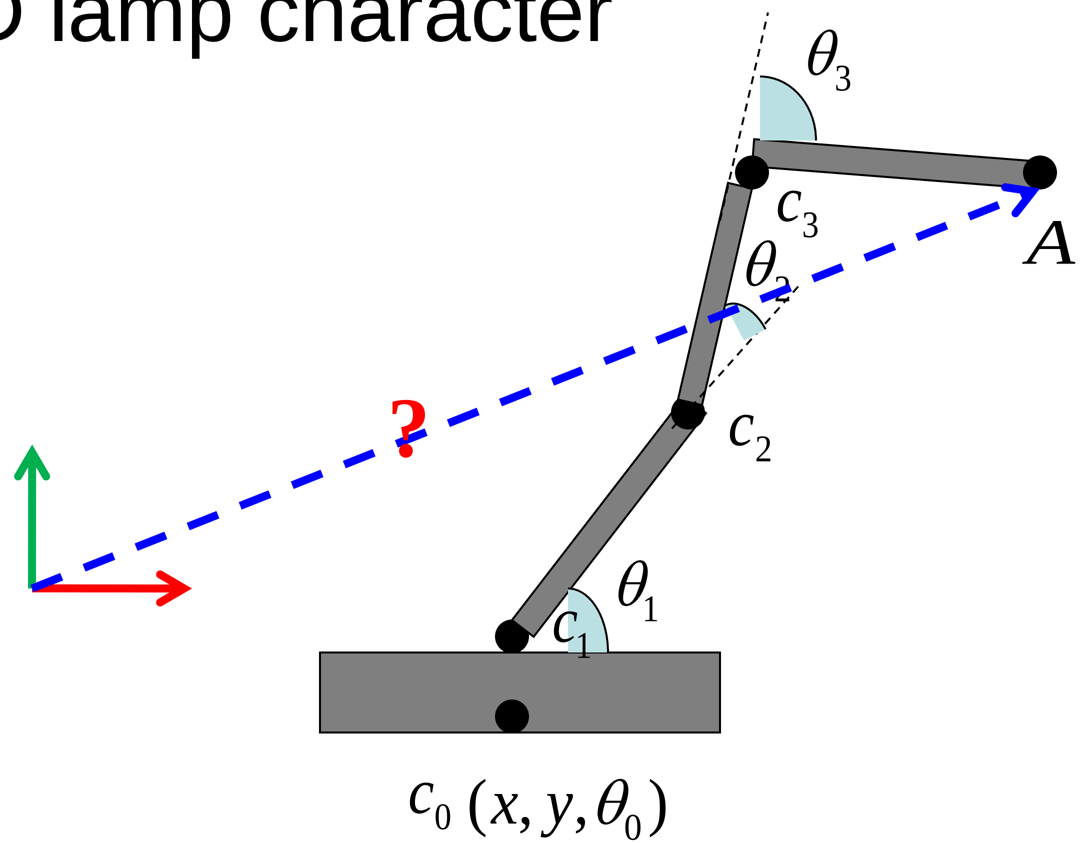
Another Example

- A 2D lamp character



Forward Kinematics

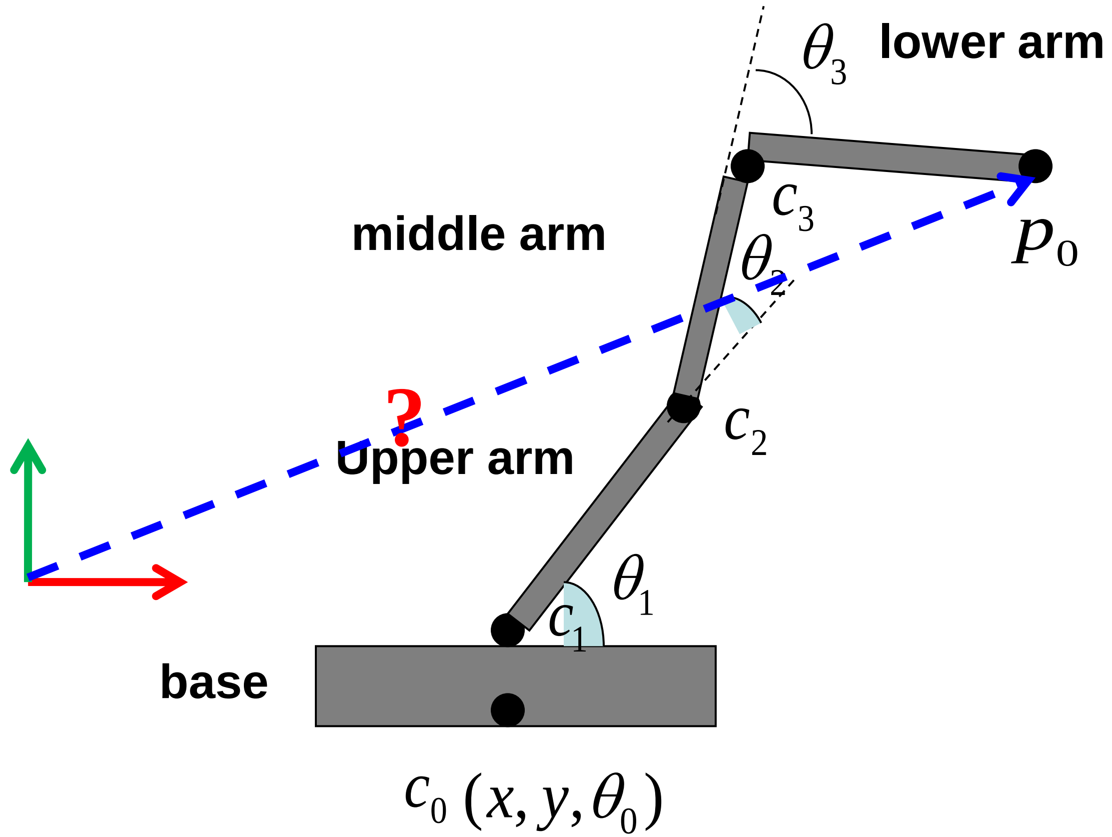
- A 2D lamp character



Given $(x, y, \theta_0, \theta_1, \theta_2, \theta_3)$, how to compute the global position of the point A?

Forward Kinematics

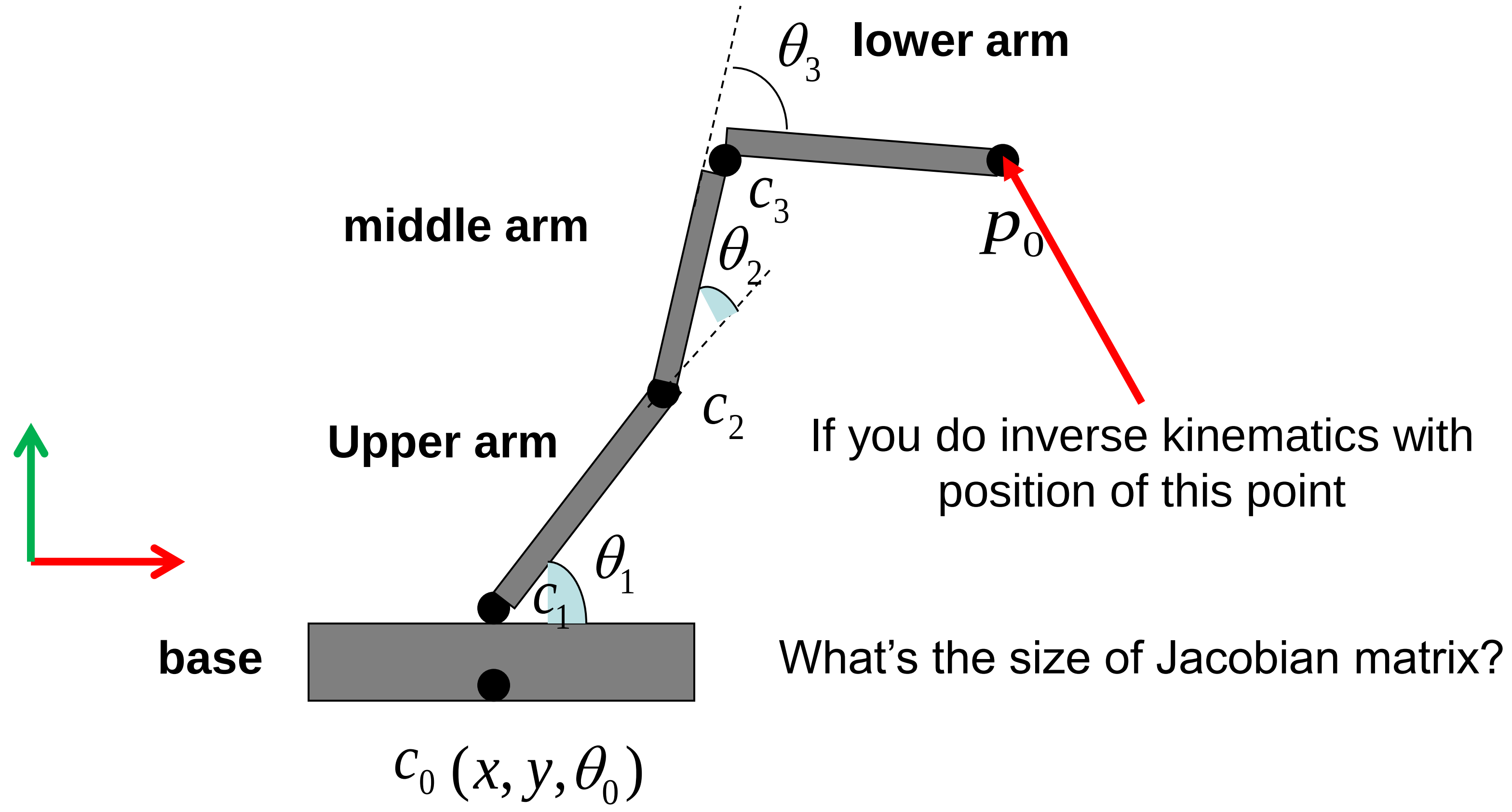
- A 2D lamp with 6 degrees of freedom



$$f(x, y, \theta_0, \theta_1, \theta_2, \theta_3) = T(x, y)R(\theta_0)T(0, l_0)R(\theta_1)T(l_1, 0)R(\theta_2)T(l_2, 0)R(\theta_3)p_0$$

Inverse Kinematics

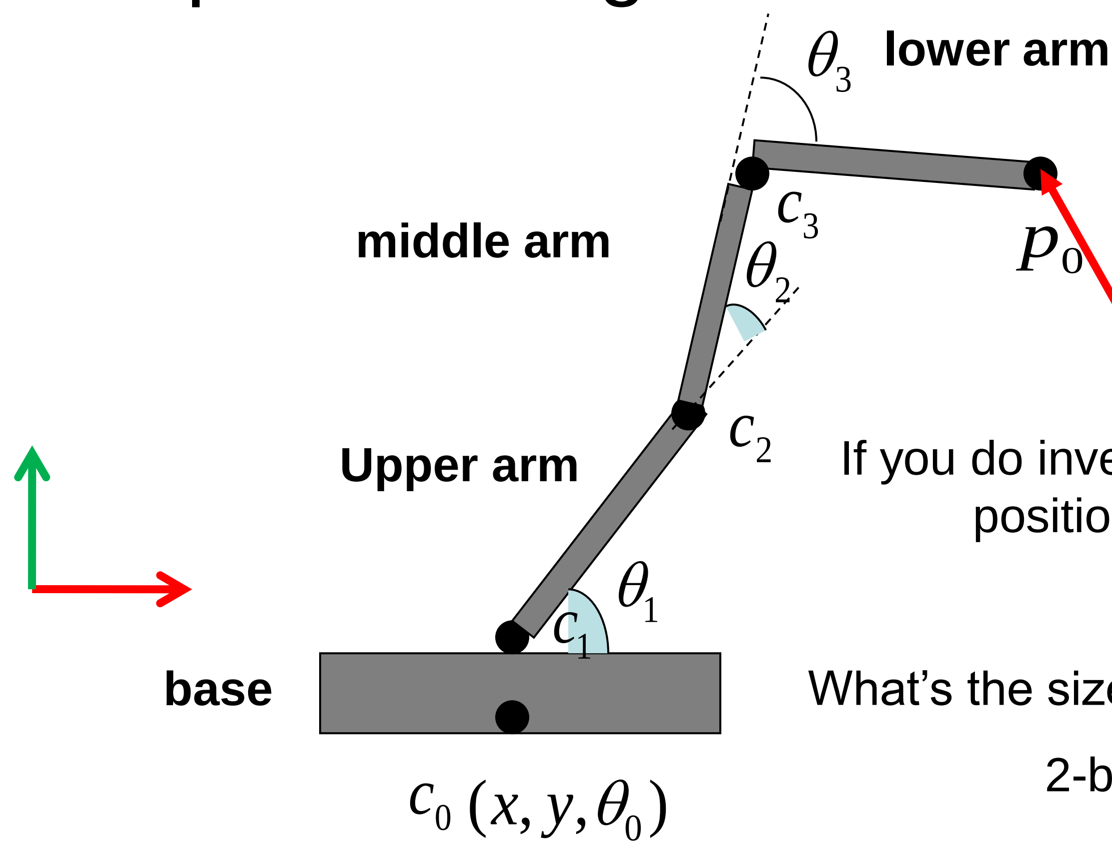
- A 2D lamp with 6 degrees of freedom



$$f(x, y, \theta_0, \theta_1, \theta_2, \theta_3) = T(x, y)R(\theta_0)T(0, l_0)R(\theta_1)T(l_1, 0)R(\theta_2)T(l_2, 0)R(\theta_3)p_0$$

Inverse Kinematics

- A 2D lamp with 6 degrees of freedom



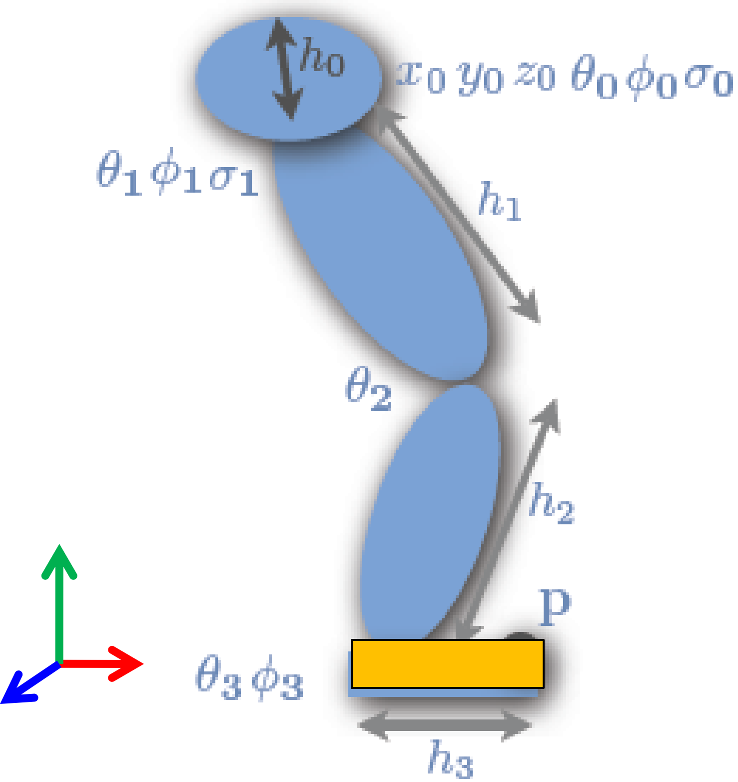
If you do inverse kinematics with position of this point

What's the size of Jacobian matrix?

2-by-6 matrix

$$f(x, y, \theta_0, \theta_1, \theta_2, \theta_3) = T(x, y)R(\theta_0)T(0, l_0)R(\theta_1)T(l_1, 0)R(\theta_2)T(l_2, 0)R(\theta_3)\,p_0$$

Human Characters



A series of transformations on an object can be applied as a series of matrix multiplications

$\mathbf{p}$: position in the global coordinate

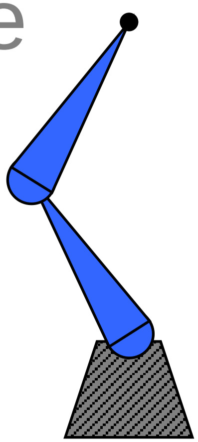
$\mathbf{x}$: position in the local coordinate

$(h_3, 0, 0)$

$$f = \boxed{T(x_0, y_0, z_0) R(\theta_0) R(\varphi_0) R(\delta_0) T_1^0 R(\theta_1) R(\varphi_1) R(\delta_1) T_2^1 R(\theta_2) T_3^2 R(\theta_3) R(\varphi_3)}\, x$$

Inverse Kinematics

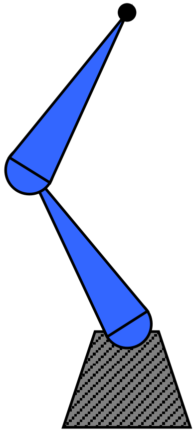
- Analytical solution only works for a fairly simple structure



- Iterative approach needed for a complex structure

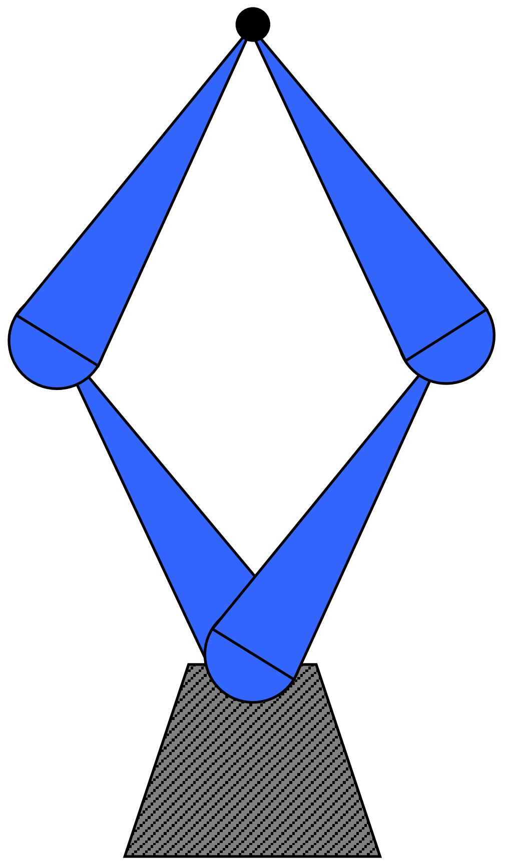
Inverse Kinematics

- Is the solution unique?
- Is there always a good solution?



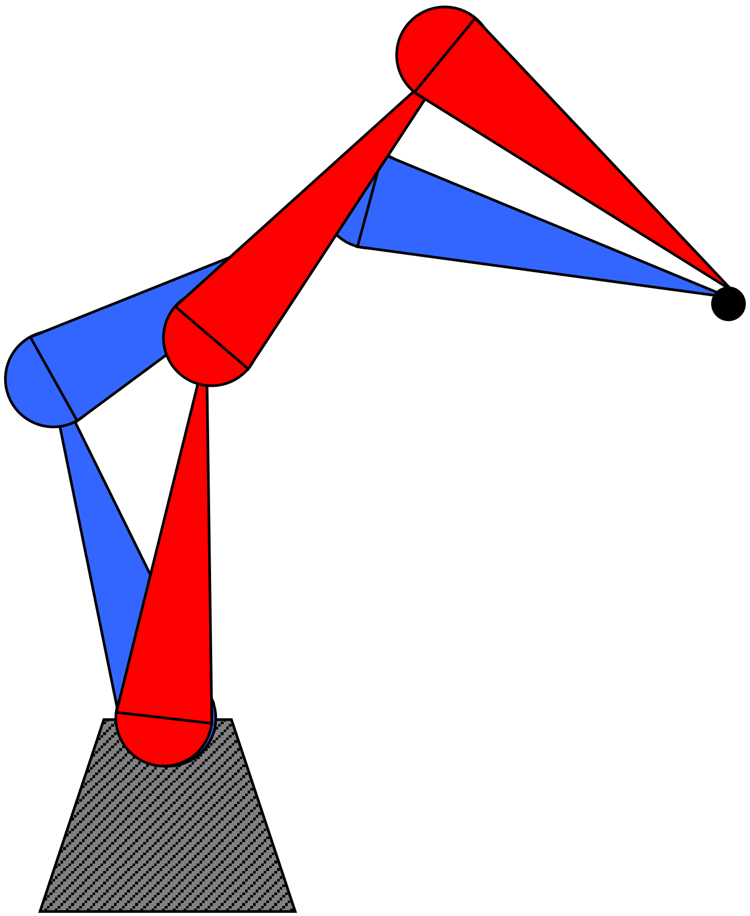
Ambiguity of IK

- Multiple solutions



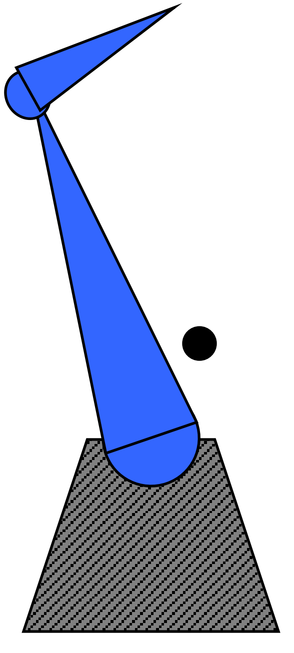
Ambiguity of IK

- Infinite solutions



Failures of IK

- Solution may not exist



Inverse Kinematics

- Generally ill-posed problem when the number of Dofs is higher than the number of constraints

Inverse Kinematics

- Generally ill-posed problem when the number of Dofs is higher than the number of constraints

- Additional objective
  - Minimal Change from a reference pose θ_0

Inverse Kinematics

- Generally ill-posed problem when the number of Dofs is higher than the number of constraints

- Additional objective
  - Minimal Change from a reference pose θ_0

$$\arg\min_{\theta} \sum_i \left\| f_i(\theta) - c_i \right\|^2 + \lambda \left\| \theta - \theta_0 \right\|^2$$

Inverse Kinematics

- Generally ill-posed problem when the number of Dofs is higher than the number of constraints

- Additional objective
  - Minimal Change from a reference pose θ_0

$$\arg\min_{\theta} \boxed{\sum_i \left\| f_i(\theta) - c_i \right\|^2} + \boxed{\lambda \left\| \theta - \theta_0 \right\|^2}$$

Satisfy the constraints

Minimize the difference between the solution and a reference pose

Inverse Kinematics

- Generally ill-posed problem when the number of Dofs is higher than the number of constraints

- Additional objective
  - Minimal Change from a reference pose θ_0
  - Naturalness $g(\theta)$ (particularly for human characters)

Inverse Kinematics

- Generally ill-posed problem when the number of Dofs is higher than the number of constraints

- Additional objective
  - Minimal Change from a reference pose θ_0
  - Naturalness $g(\theta)$ (particularly for human characters)

$$\arg\min_{\theta} \sum_i \left\| f_i(\theta) - c_i \right\|^2 + g(\theta)$$

Inverse Kinematics

- Generally ill-posed problem when the number of Dofs is higher than the number of constraints

- Additional objective
  - Minimal Change from a reference pose θ_0
  - Naturalness $g(\theta)$ (particularly for human characters)

$$\arg\min_{\theta} \boxed{\sum_i \| f_i(\theta) - c_i \|^2} + \boxed{g(\theta)}$$

Satisfy the constraints

How natural is the solution pose?

Interactive Human Character Posing

- Video (click here)

Summary of IK

- Very simple structure allows an analytic solution

- Most of complex articulated figures requires a numerical solution

- May not always get the “right” solution

  - need additional objectives