Mathematics for Inverse Kinematics

15-464: Technical Animation

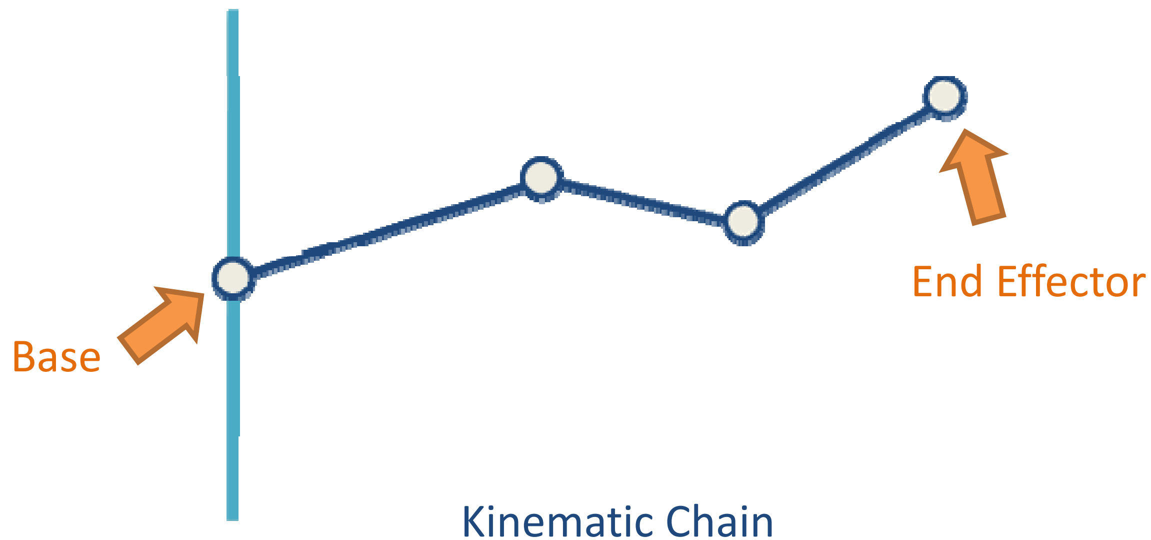
Ming Yao

Overview

- Kinematics
- Forward Kinematics and Inverse Kinematics
- Jabobian
- Pseudoinverse of the Jacobian
- Assignment 2

Vocabulary of Kinematics

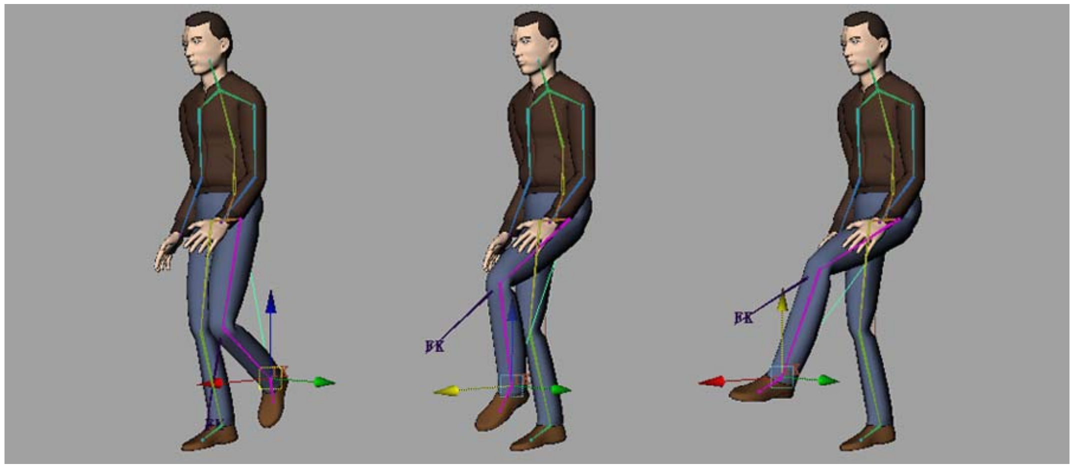
- Kinematics is the study of how things move, it describes the motion of a hierarchical skeleton structure.
- Base and End Effector.



Kinematic Chain

FK vs. IK

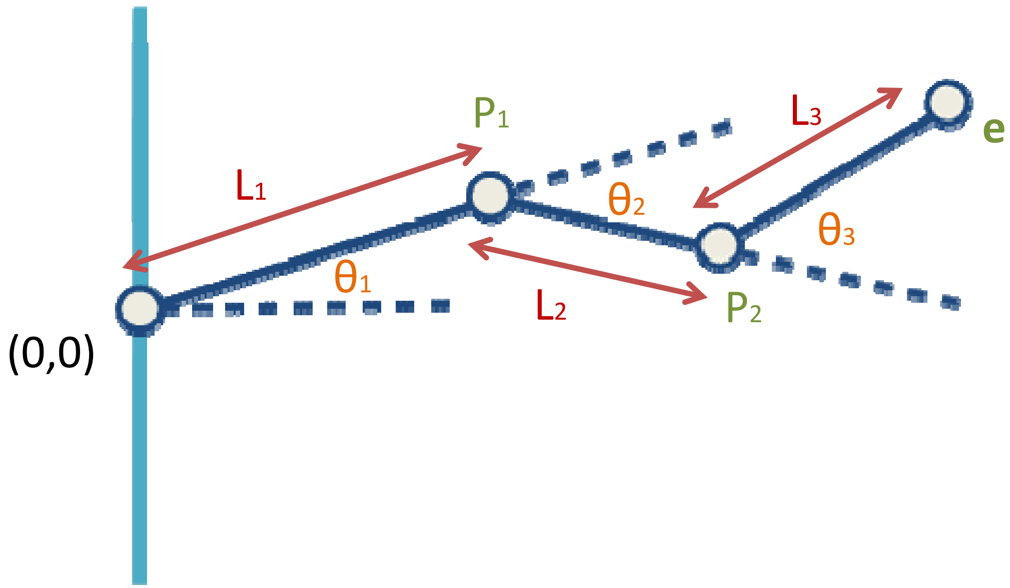
Forward Kinematics



Inverse Kinematics

Forward Kinematics

- The process of computing world space geometric description based on joint DOF values.



Forward Kinematics

- We have joint DOF values:

$$\boldsymbol{\theta} = [\theta_1 \ \theta_2 \ \cdots \ \theta_M]$$

- We want the end effector description in world space (N=3 in our case):

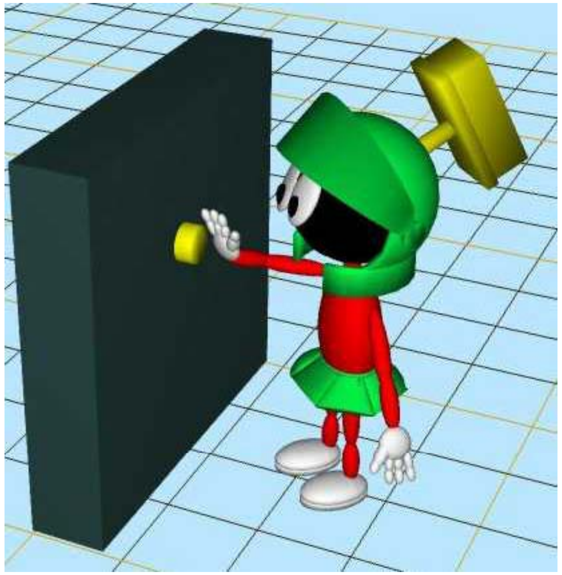
$$\mathbf{e} = [e_1 \ e_2 \ \cdots \ e_N]$$

- FK gives us:

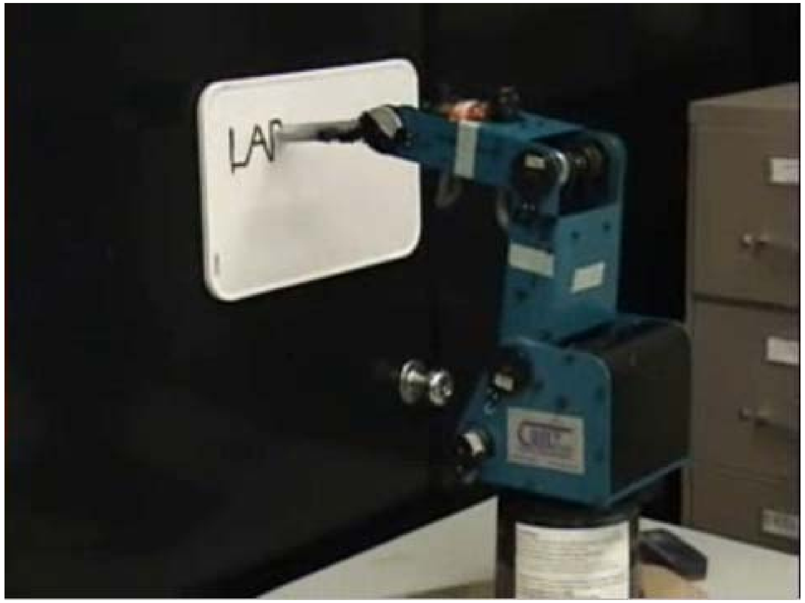
$$\mathbf{e} = \mathbf{f}(\boldsymbol{\theta})$$

But Sometimes We Want the Opposite

- We want to know how the upper joints of the hierarchy would rotate if we want the end effector to reach some goal.



- Animations

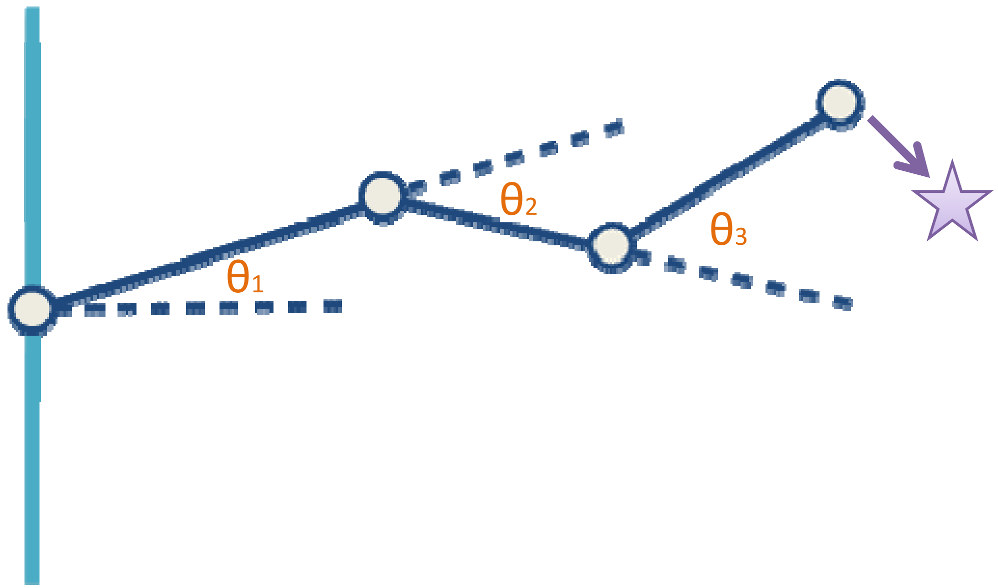


- Robotics

Images are from google and youtube

Inverse Kinematics

- The goal of inverse kinematics is to compute the vector of joint DOFs that will cause the end effector to reach some desired goal state



Inverse Kinematics

- We have:

$$\mathbf{e} = [e_1\ e_2\ \cdots\ e_N]$$

- And we want:

$$\boldsymbol{\theta} = [\theta_1\ \theta_2\ \cdots\ \theta_M]$$

- We need:

$$\boldsymbol{\theta} = \mathbf{f}^{-1}(\mathbf{e})$$

Inverse Kinematics Issues

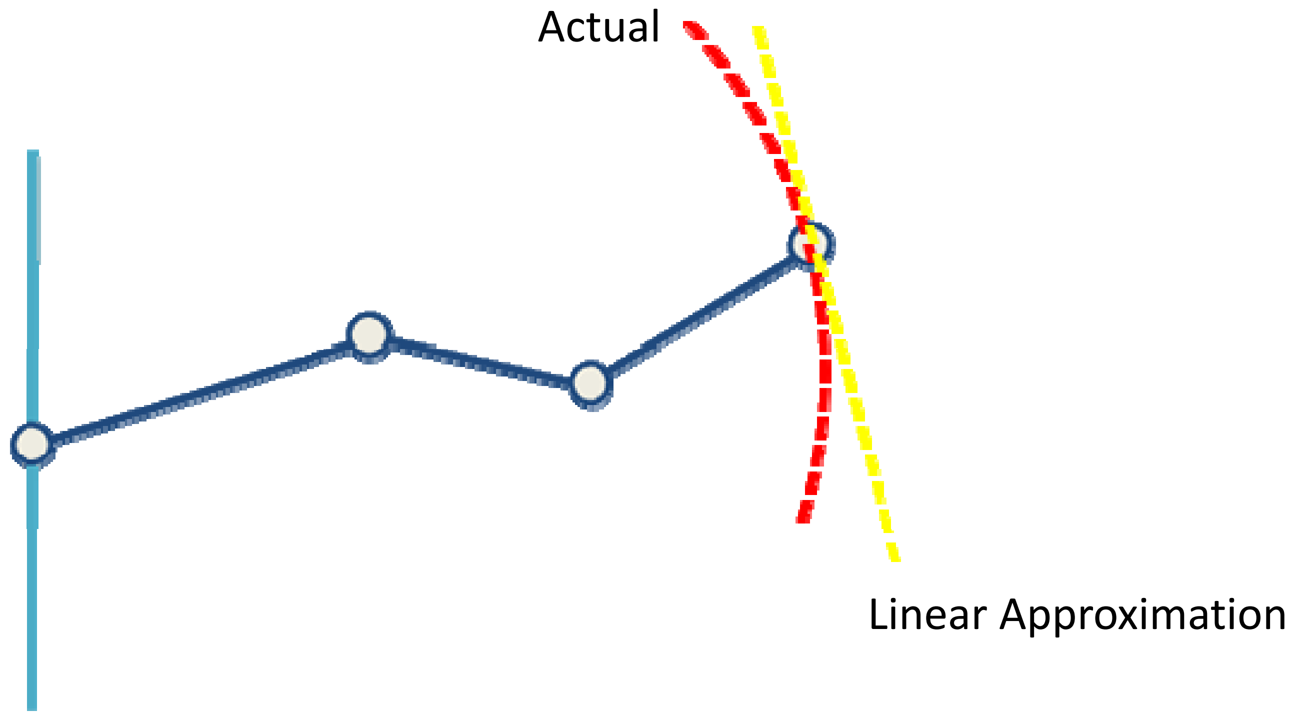
- While FK is relatively easy to evaluate.
- IK is more challenging: several possible solutions, or sometimes maybe no solutions.
- Require Complex and Expensive computations to find a solution.

IK Solutions

- Jacobian
- Cyclic Coordinate Descent (CCD)
- Required to implement in Assignment 2

The Jacobian

- What is Jacobian? A linear approximation to f()



The Jacobian

- Matrix of partial derivatives of entire system.
- Defines how the end effector **e** changes relative to instantaneous changes in the system.

$$J = \frac{d\mathbf{e}}{d\boldsymbol{\theta}} \qquad d\mathbf{e} = J\,d\boldsymbol{\theta}$$

$$\mathbf{e} = [e_x \; e_y \; e_z]^T$$

$$\boldsymbol{\theta} = [\theta_1 \; \theta_2 \; \cdots \; \theta_M]^T$$

The Jacobian

$$J = \begin{bmatrix} \frac{\partial e_x}{\partial \theta_1} & \frac{\partial e_x}{\partial \theta_2} & \cdots & \frac{\partial e_x}{\partial \theta_M} \\ \frac{\partial e_y}{\partial \theta_1} & \frac{\partial e_y}{\partial \theta_2} & \cdots & \frac{\partial e_y}{\partial \theta_M} \\ \frac{\partial e_z}{\partial \theta_1} & \frac{\partial e_z}{\partial \theta_2} & \cdots & \frac{\partial e_z}{\partial \theta_M} \end{bmatrix}$$

The Jacobian

- Recall that

$$\boldsymbol{\theta} = \mathbf{f}^{-1}(\mathbf{e})$$

$$d\mathbf{e} = J d\boldsymbol{\theta}$$

$$\Rightarrow$$

$$d\boldsymbol{\theta} = J^{-1} d\mathbf{e}$$

Problems

- How to compute J?

  Numerically (Required)

  Analytically (Extra Credit)

- How to invert J?

  Pseudoinverse of Jacobian (Required)

  Cheat by using transpose (Too easy, we don't do that)

Computing the Jacobian Numerically

- Let's examine one column of the Jacobian Matirx
$$\frac{\partial \mathbf{e}}{\partial \theta_1} = \left[\frac{\partial e_x}{\partial \theta_1} \; \frac{\partial e_y}{\partial \theta_1} \; \frac{\partial e_z}{\partial \theta_1}\right]^T$$
- We can add a small $\Delta\theta$ to θ_i
- Then we can calculate how the end effector moves: $\Delta\mathbf{e} = \mathbf{e}' - \mathbf{e}$
- Now we have:
$$\frac{\partial \mathbf{e}}{\partial \theta_1} \approx \frac{\Delta \mathbf{e}}{\Delta \theta} = \left[\frac{\Delta e_x}{\Delta \theta} \; \frac{\Delta e_y}{\Delta \theta} \; \frac{\Delta e_z}{\Delta \theta}\right]^T$$

Computing the Jacobian Numerically

$$\frac{\partial \mathbf{e}}{\partial \theta_1} \approx \frac{\Delta \mathbf{e}}{\Delta \theta} = \left[\frac{\Delta e_x}{\Delta \theta} \; \frac{\Delta e_y}{\Delta \theta} \; \frac{\Delta e_z}{\Delta \theta} \right]^T$$

- We can use this method to fill the Jacobian Matirx!

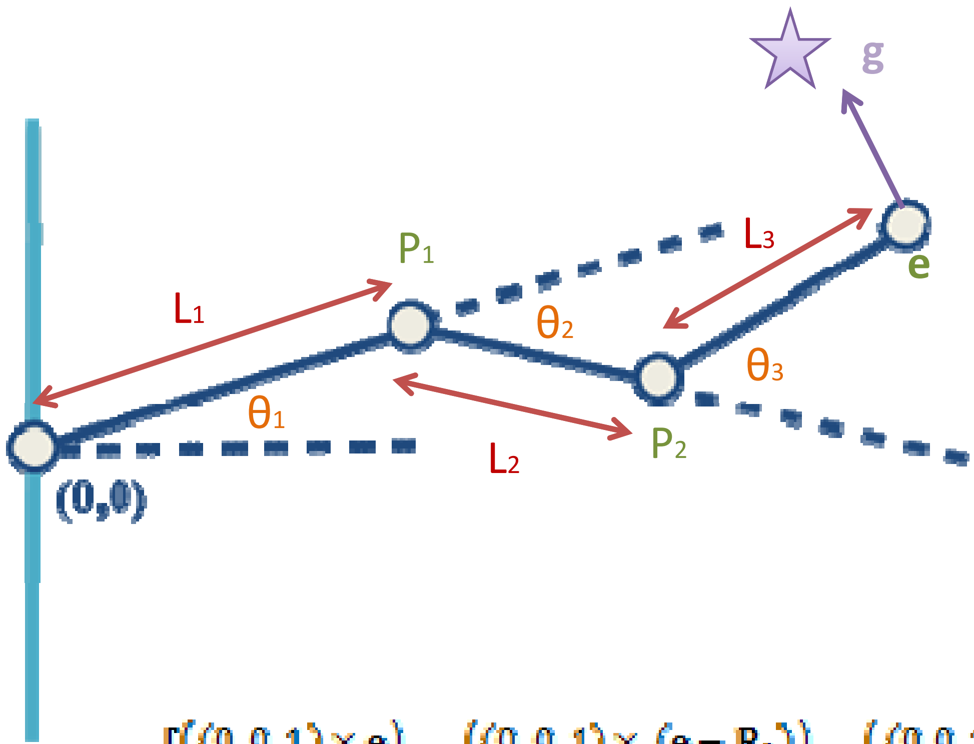
Computing the Jacobian Analytically

- For a rotational joint, the linear change in the end effector is the cross product of the axis of revolution and a vector from the joint to the end effector.

$$\frac{\partial \mathbf{e}}{\partial \theta_1} = \left[\frac{\partial e_x}{\partial \theta_1} \; \frac{\partial e_y}{\partial \theta_1} \; \frac{\partial e_z}{\partial \theta_1}\right]^T = \left(\mathbf{a}_1' \times (\mathbf{e} - \mathbf{r}_1')\right)$$

- Important to make sure all the coordinate values are in the same coordinate system. (Hard to get right.)

Computing the Jacobian Analytically — A 2D example



$$J = \begin{bmatrix} ((0,0,1)\times e)_x & ((0,0,1)\times(e-P_1))_x & ((0,0,1)\times(e-P_2))_x \\ ((0,0,1)\times e)_y & ((0,0,1)\times(e-P_1))_y & ((0,0,1)\times(e-P_2))_y \\ ((0,0,1)\times e)_z & ((0,0,1)\times(e-P_1))_z & ((0,0,1)\times(e-P_2))_z \end{bmatrix}$$

Inverting the Jacobian

- No guarantee it is invertible
  - Typically not a square matrix.
  - Singularities.
  - Even it’s invertible, as the pose vector changes, the properties of the matrix will change.

Inverting the Jacobian—Pseudo Inverse

- We can try using the pseudo inverse to find a matrix that effectively inverts a non-square matrix:

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

Inverting the Jacobian—Pseudo Inverse

$$d\mathbf{e} = J \cdot d\boldsymbol{\theta}$$

$$J^T \cdot d\mathbf{e} = J^T J \cdot d\boldsymbol{\theta}$$

$$(J^T J)^{-1} J^T \cdot d\mathbf{e} = (J^T J)^{-1} (J^T J) \cdot d\boldsymbol{\theta}$$

$$(J^T J)^{-1} J^T \cdot d\mathbf{e} = d\boldsymbol{\theta}$$

$$J^+ \cdot d\mathbf{e} = d\Delta\boldsymbol{\theta}$$

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

Inverting the Jacobian— Jacobian Transpose

- Another technique is just to use the transpose of the Jacobian matrix.
- The Jacobian is already an approximation to f()—Cheat more
- It is much faster.
- But if you prefers quality over performance, the pseudo inverse method would be better.

Solving IK—Incremental Changes

- FK is nonlinear
- Implies that the Jacobian can only be used as an approximation that is valid near the current configuration
- So we must **Repeat** the process of computing a Jacobian and then taking a small step towards the goal until we get close enough

Solving IK—Algorithm of the Jacobian Method

while ($\mathbf{e}$ is too far from $\mathbf{g}$){

compute the Jacobian matrix J

compute the pseudoinverse of the Jacobian matrix— $\mathbf{J}^{+}$

compute change in joint DOFs: $\Delta\boldsymbol{\theta} = \mathbf{J}^{+} \cdot \Delta\mathbf{e}$

apply the change to DOFs, move a small step of $\boldsymbol{\alpha}\Delta\boldsymbol{\theta}$: $\boldsymbol{\theta} = \boldsymbol{\theta} + \boldsymbol{\alpha}\Delta\boldsymbol{\theta}$

}

Cyclic Coordinate Descent (CCD)

- Much easier than the Jacobian Method
- Read two articles “Oh My God, I inverted Kine!” and “Making Kine More Flexible”
- Will be talked about next time

Assignment 2

- Will be out later today.
- Require to implement the Jacobian Method and CCD.
- Load ASF file into Maya and add IK to the skeleton.
- Start early! More challenging than Assignment 1.

Assignment 1 is Due Tonight

- Handin videos and presentation materials
- Presentation on Thursday!

Questions?

- Thank you and See you Thursday!