Forward and Inverse Kinematics

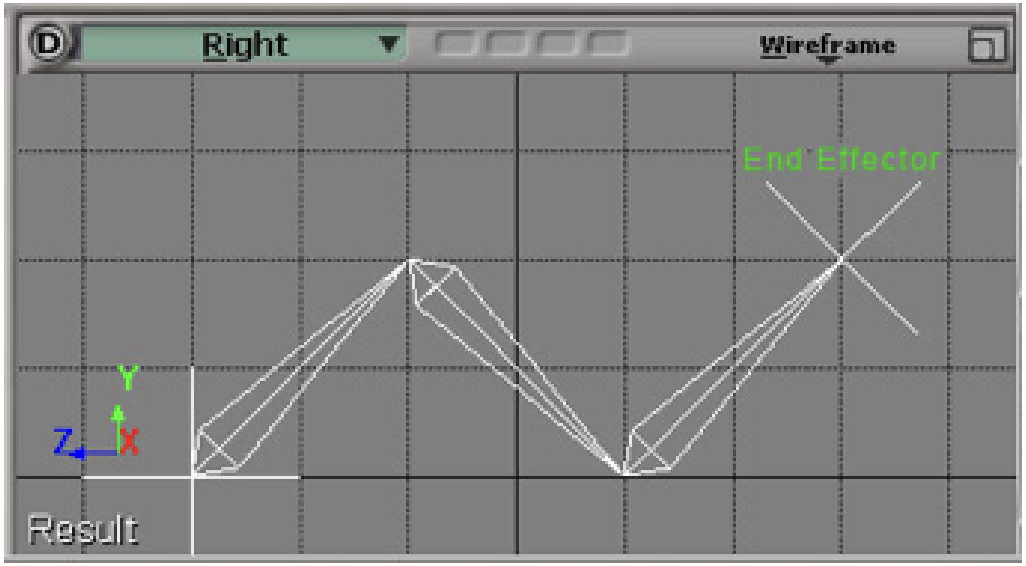
CS475 / 675, Fall 2016

Siddhartha Chaudhuri

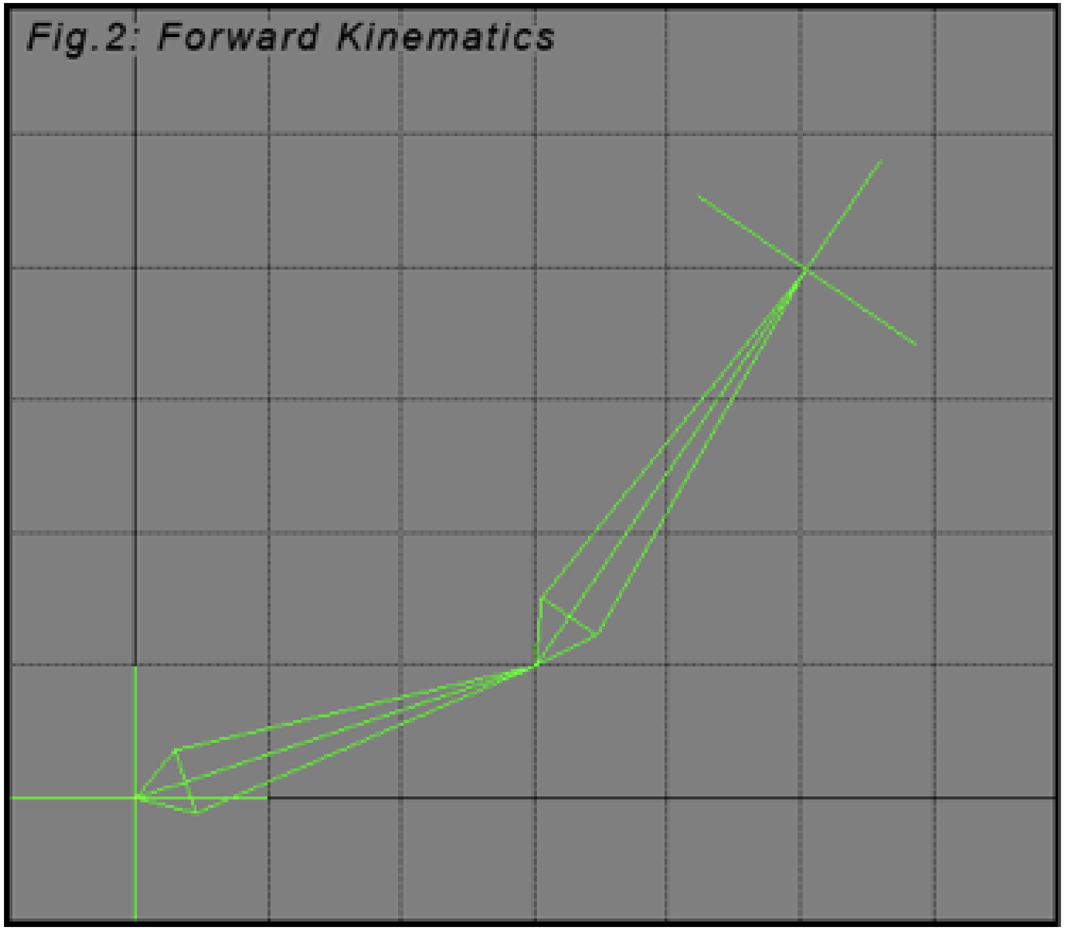
Slides from Tiffany Barnes, Bill Baxter, Serafim Batzoglou and Jean-Claude Latombe

Forward Kinematics

- Incrementally manipulating each component of a flexible, jointed object to achieve an overall desired pose
- Mathematically: find the position of the **end effector**, given the **angles** of the joints and the **length** of each articulated segment

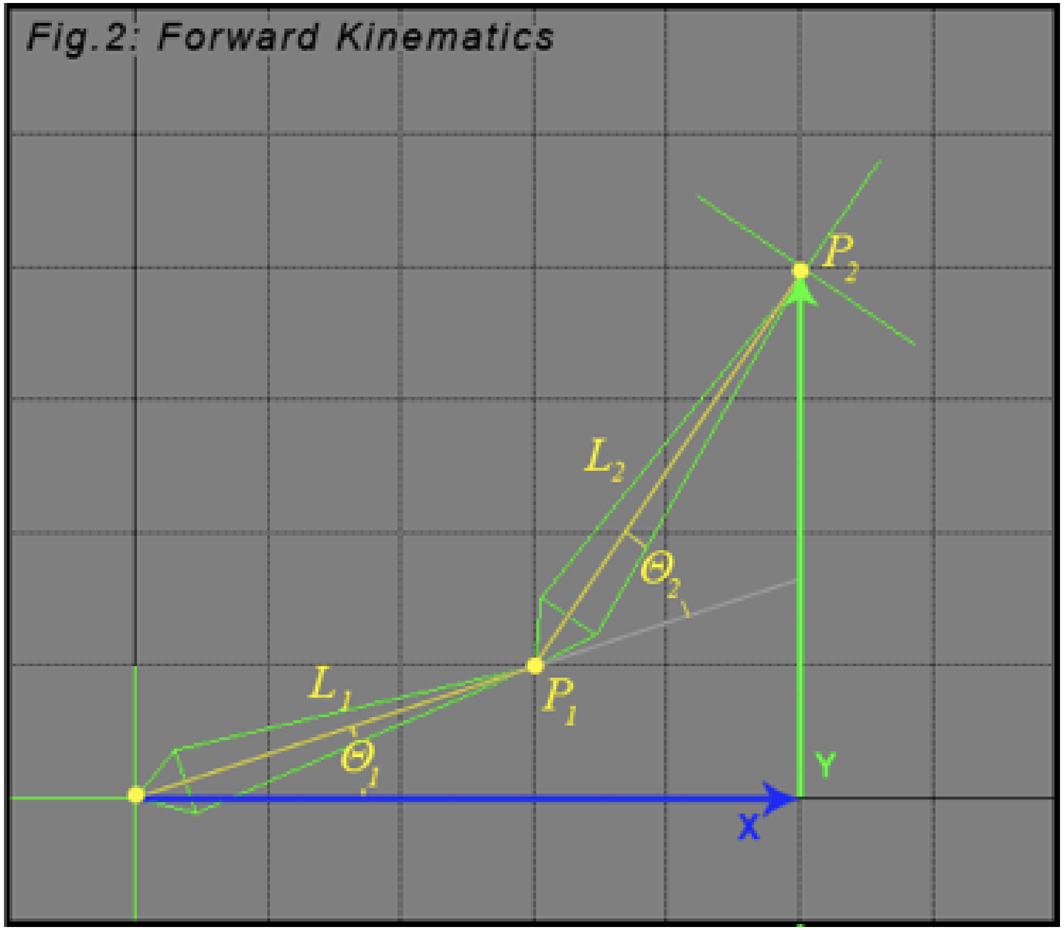


Forward Kinematics Equation



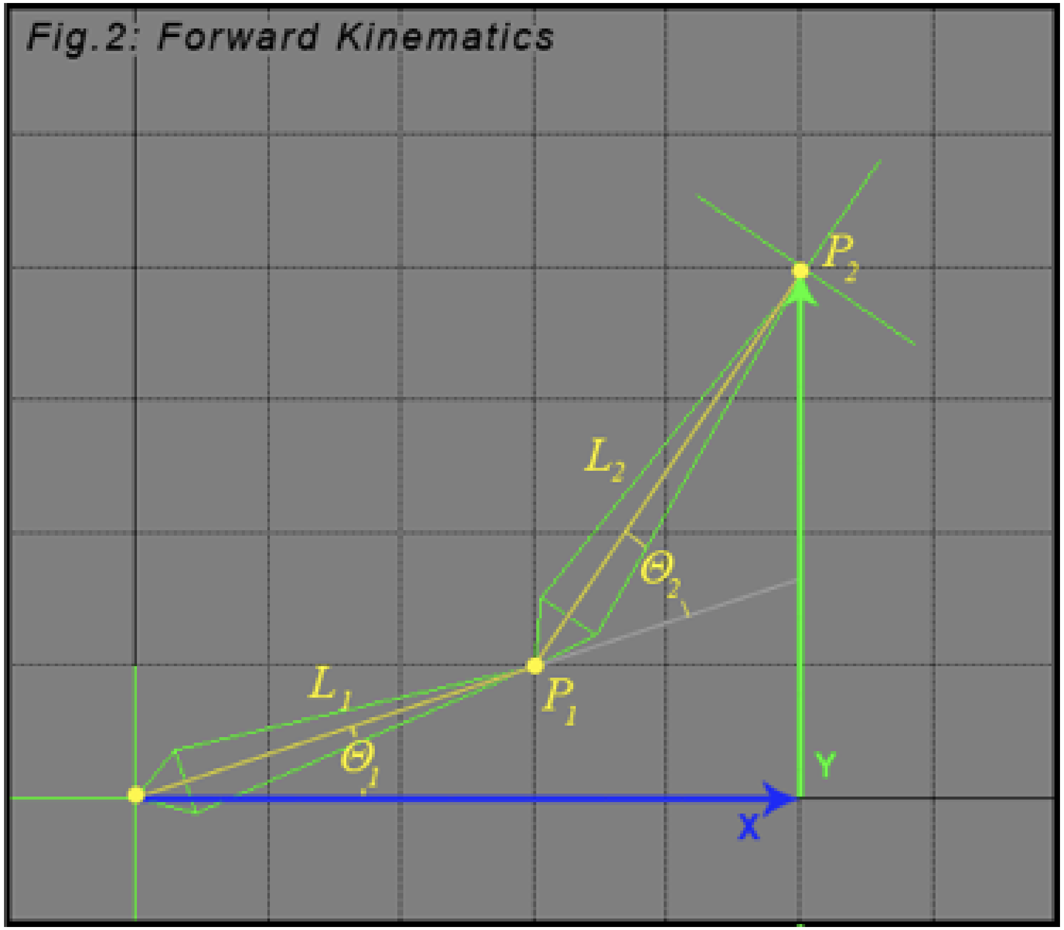
Forward Kinematics Equation

$$Px_1 = L_1 \times \cos(\Theta_1)$$
$$Py_1 = L_1 \times \sin(\Theta_1)$$



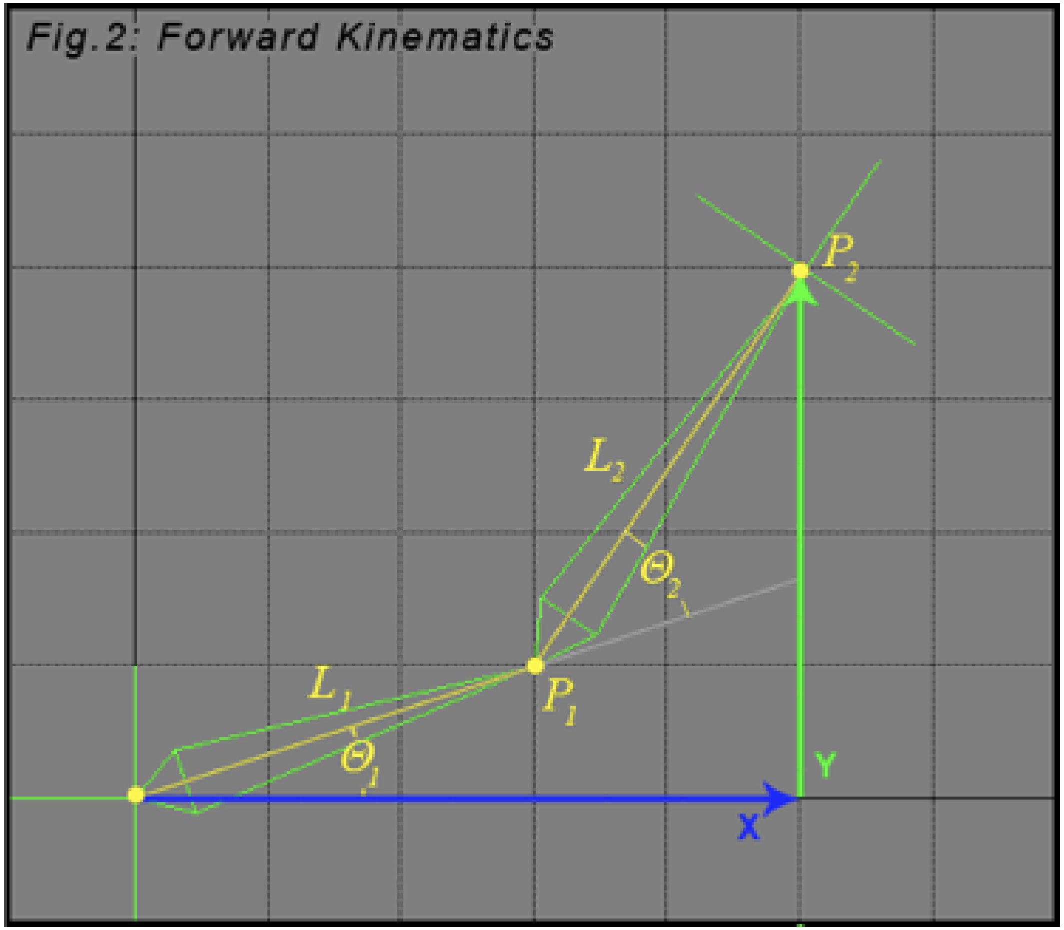
Forward Kinematics Equation

$$Px_2 = Px_1 + L_2 \times \cos(\Theta_1 + \Theta_2)$$
$$Py_2 = Py_1 + L_2 \times \sin(\Theta_1 + \Theta_2)$$



Forward Kinematics Equation

$$Px_2 = L_1 \times \cos(\Theta_1) + L_2 \times \cos(\Theta_1 + \Theta_2)$$
$$Py_2 = L_1 \times \sin(\Theta_1) + L_2 \times \sin(\Theta_1 + \Theta_2)$$

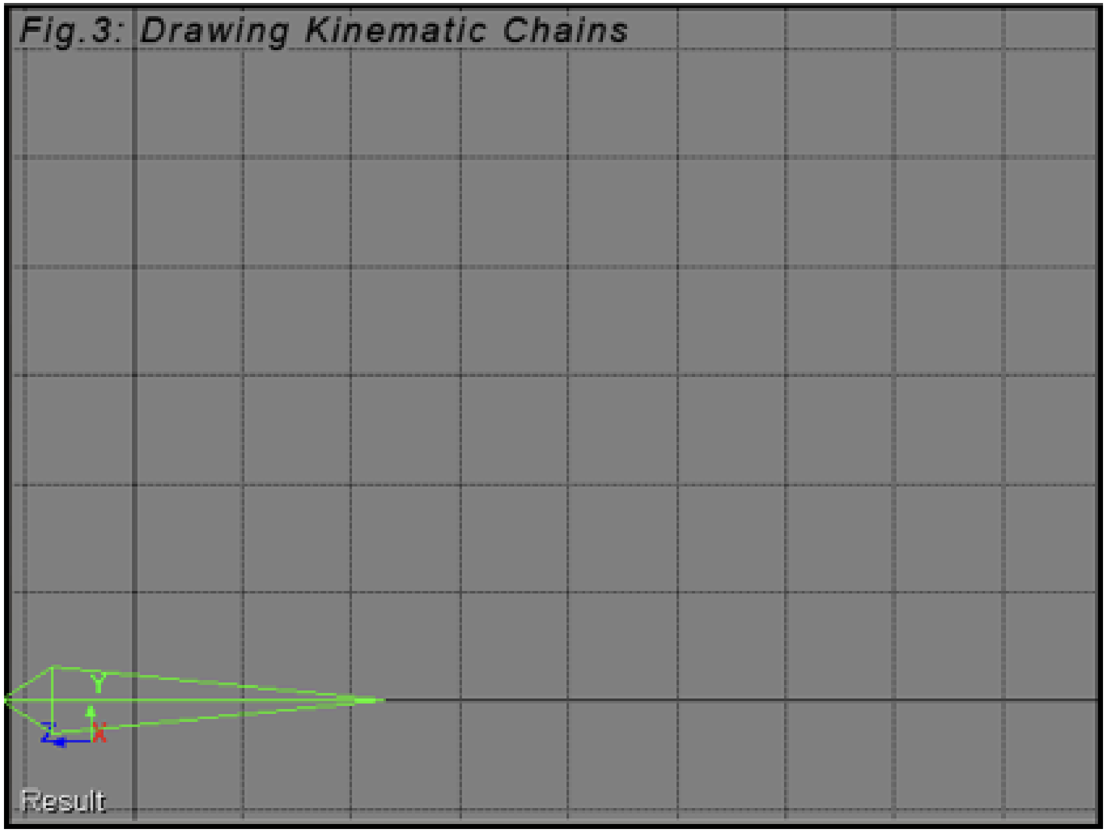


Drawing Kinematic Chains

- Links should be drawn from the outermost link (nearest the end effector) to the innermost (nearest the root)
- The positioning of each link requires translations and rotations from preceding links

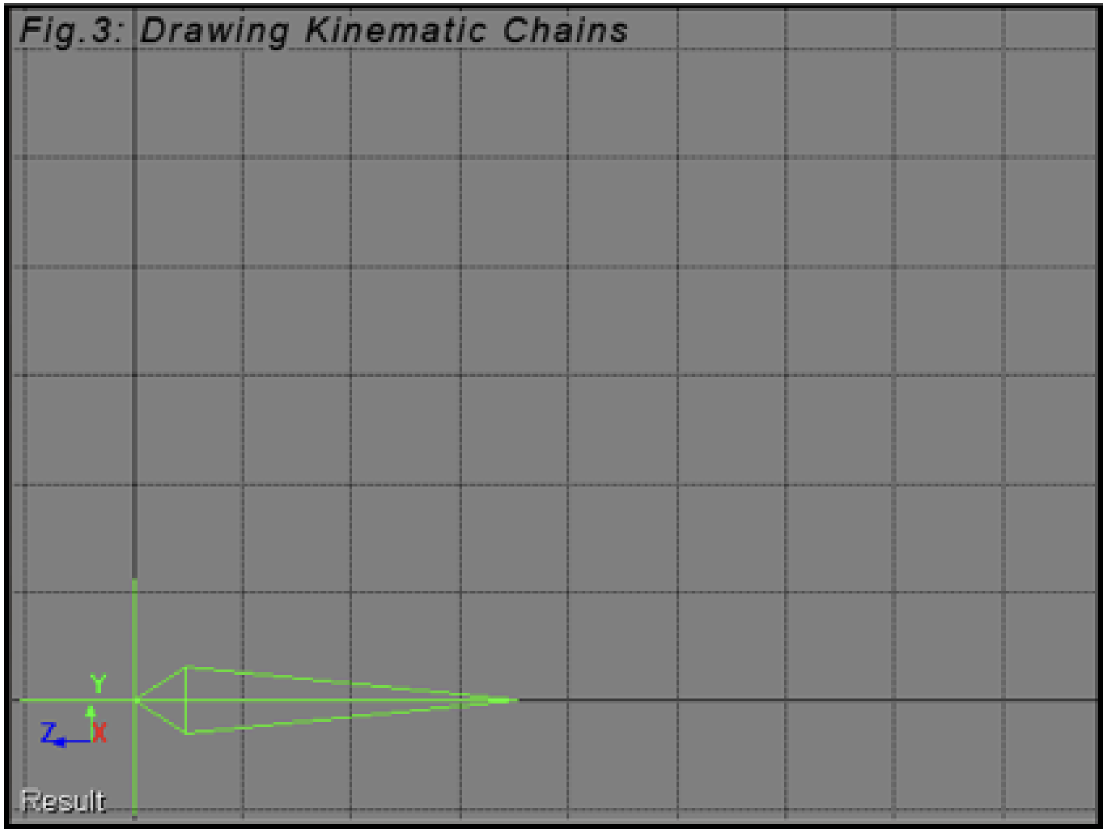
Drawing Kinematic Chains

- Starting with the end effector's object:



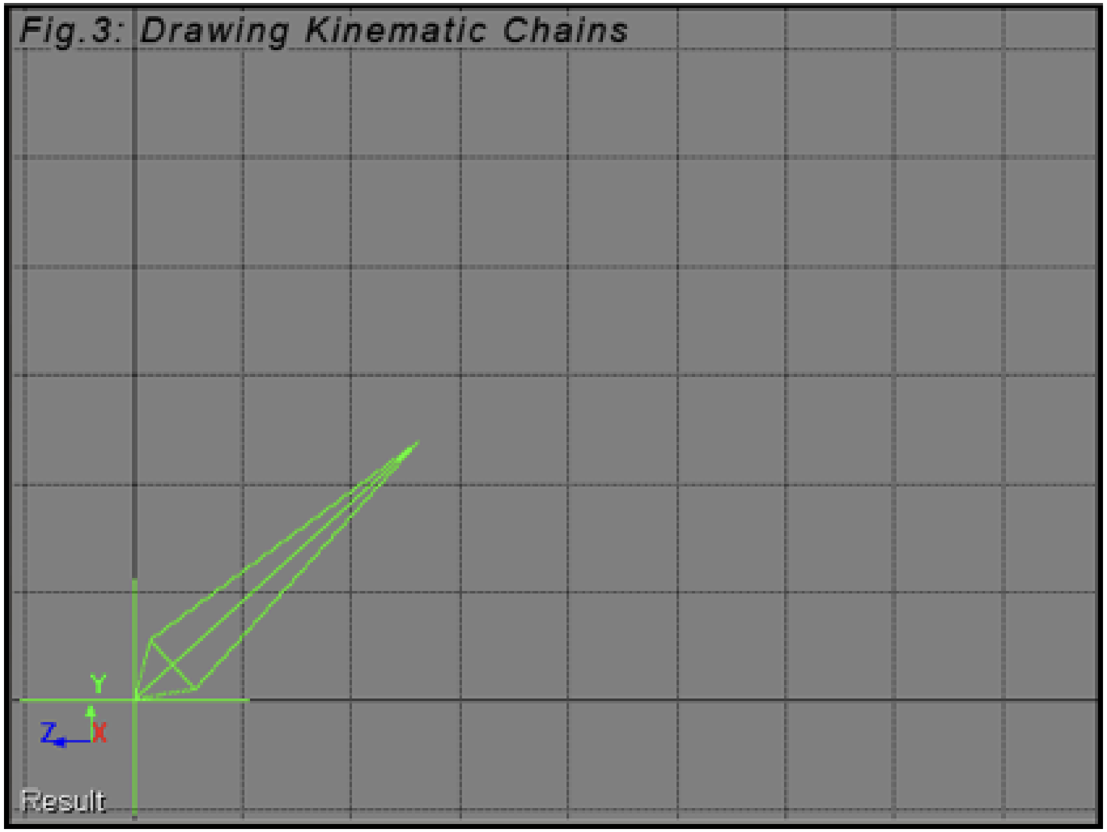
Drawing Kinematic Chains

- Starting with the end effector's object:
  - Translate base to origin



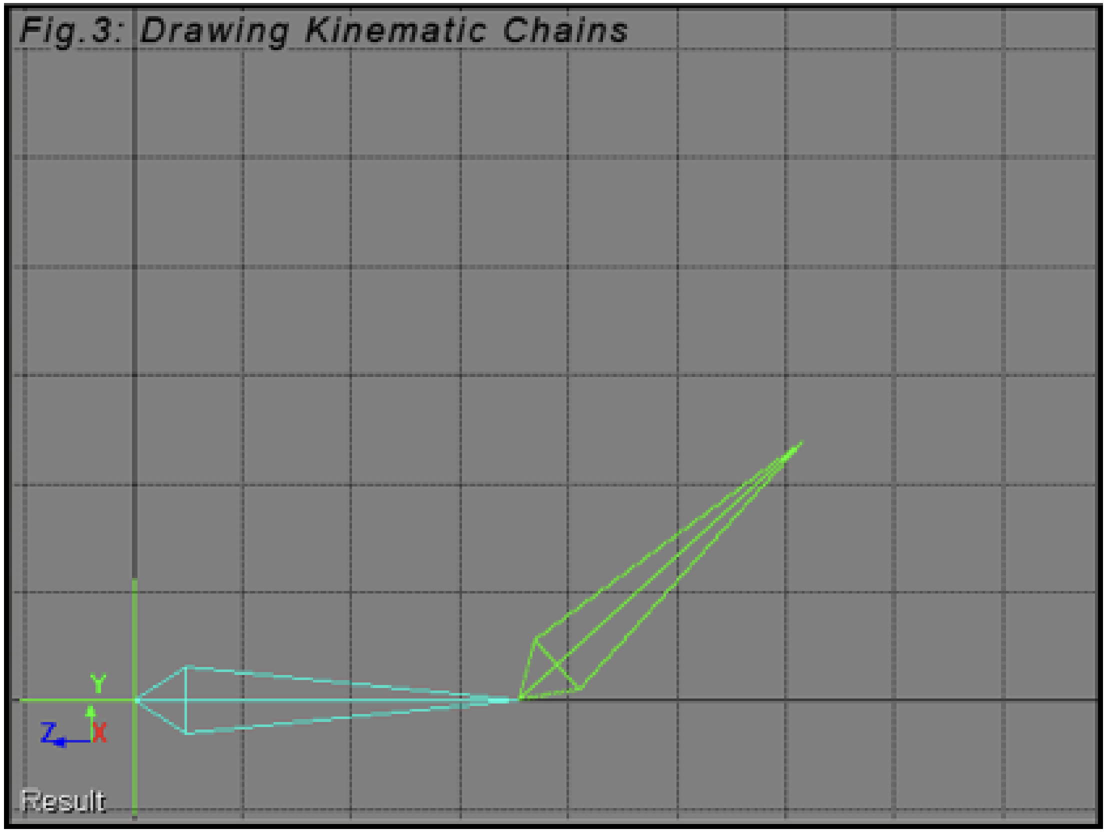
Drawing Kinematic Chains

- Starting with the end effector's object:
  - Translate base to origin
  - Rotate by angle



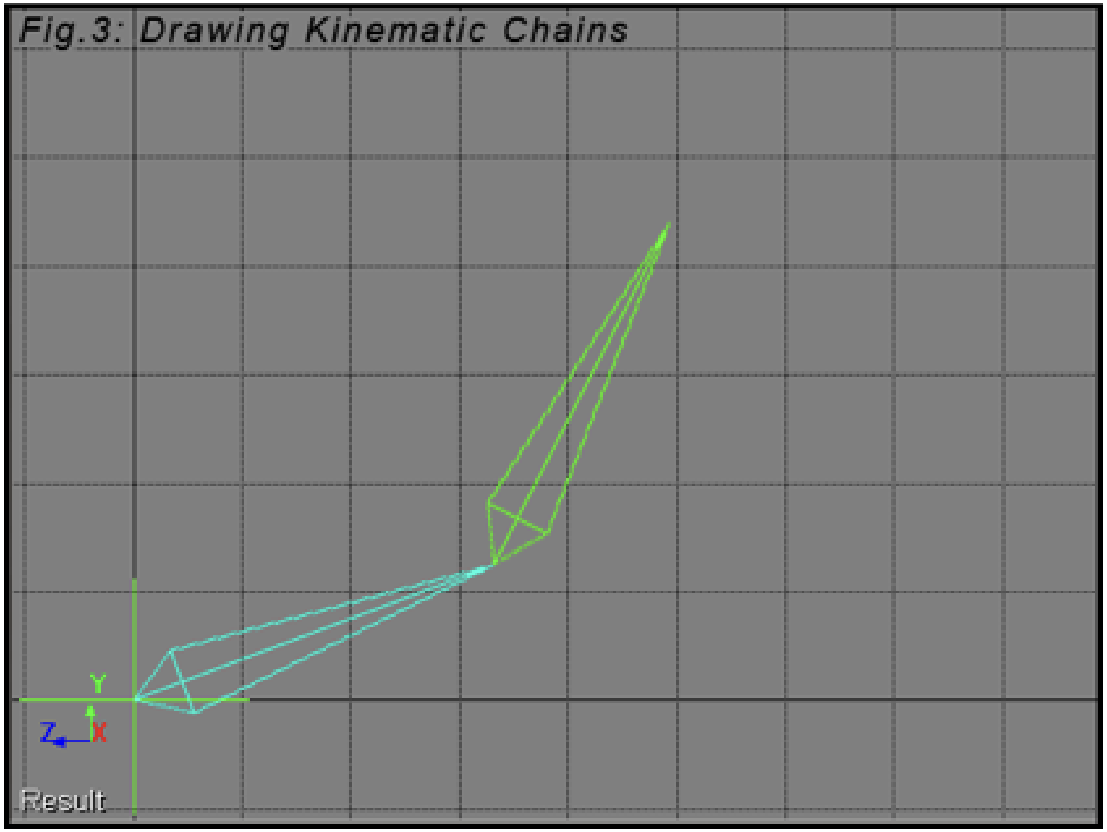
Drawing Kinematic Chains

- Starting with the end effector's object:
  - Now translate by length of next link…



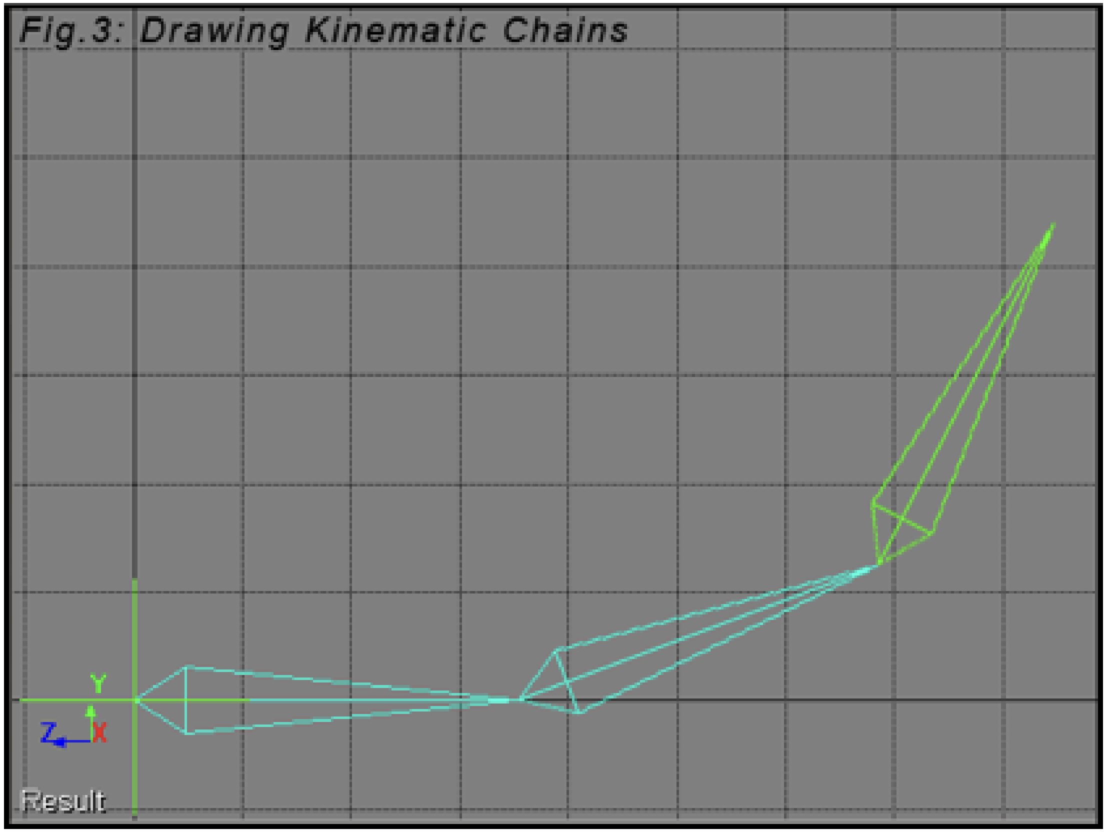
Drawing Kinematic Chains

- Starting with the end effector's object:
  - Now translate by length of next link...
  - ... and rotate the entire chain by the angle of that link



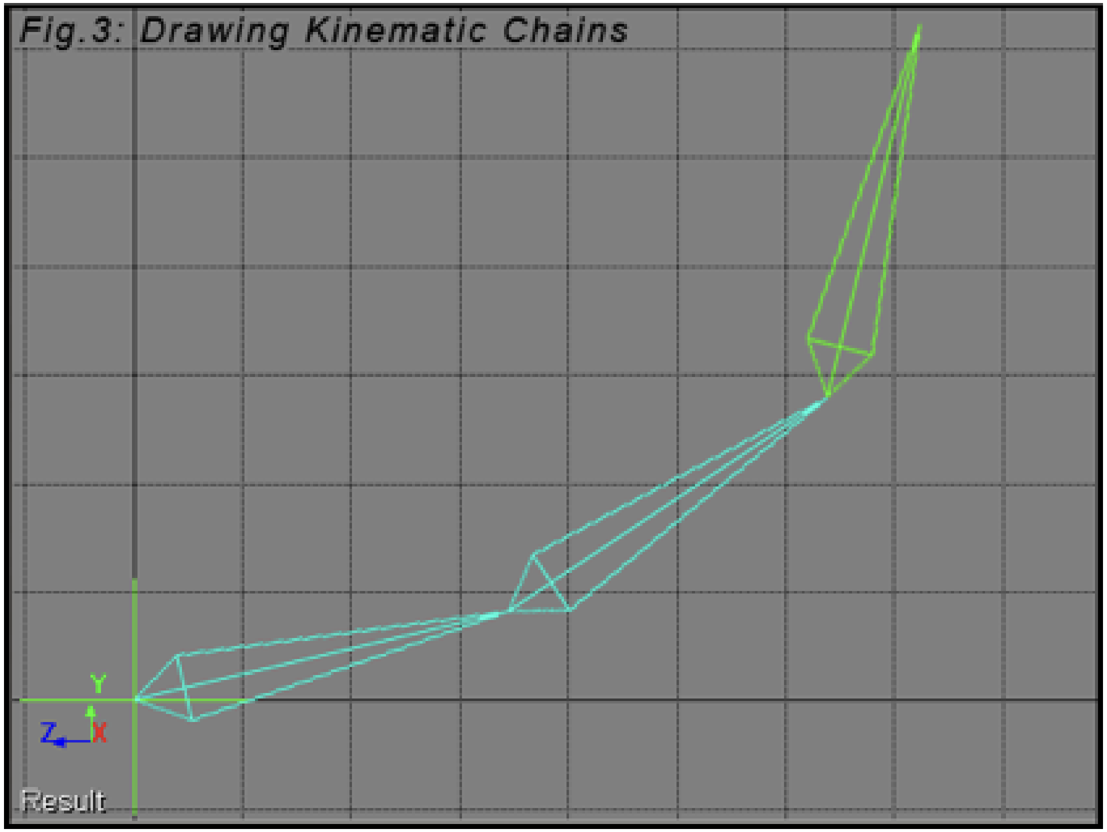
Drawing Kinematic Chains

- Starting with the end effector's object:
  - Translate again by the length of the next link in the chain…



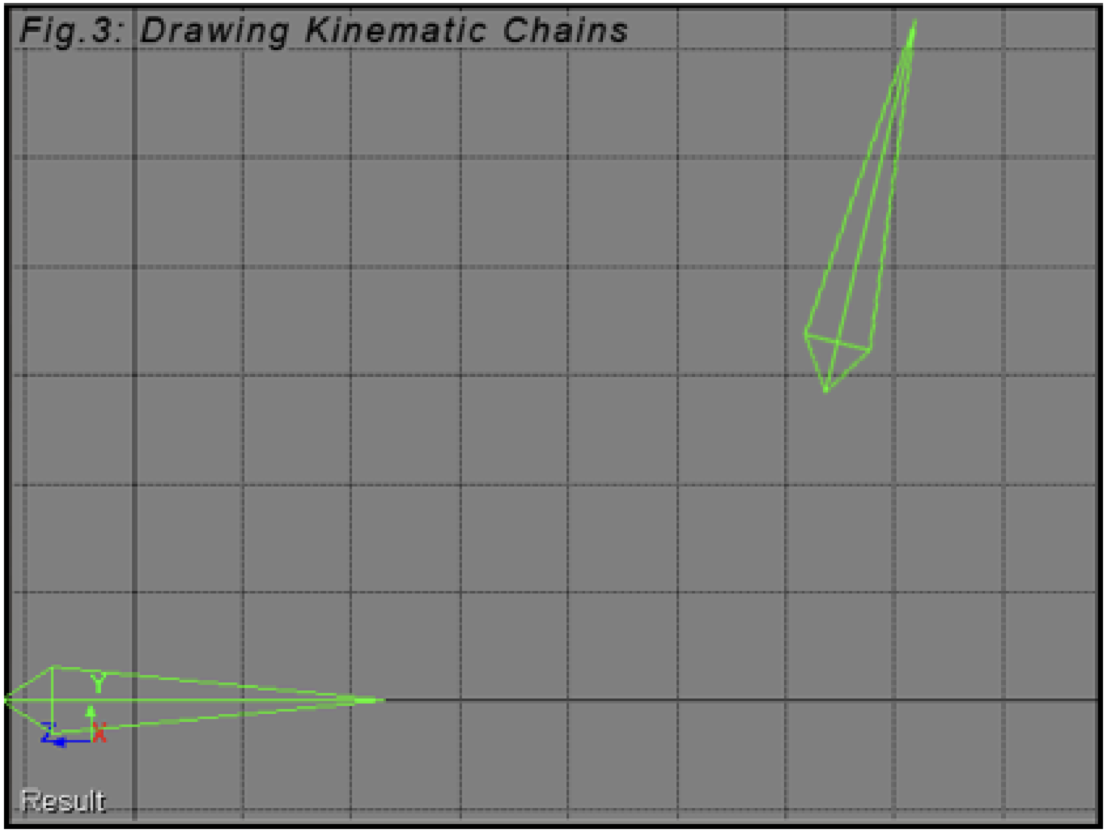
Drawing Kinematic Chains

- Starting with the end effector's object:
  - Translate again by the length of the next link in the chain…
  - … and rotate the entire chain by the angle of that link



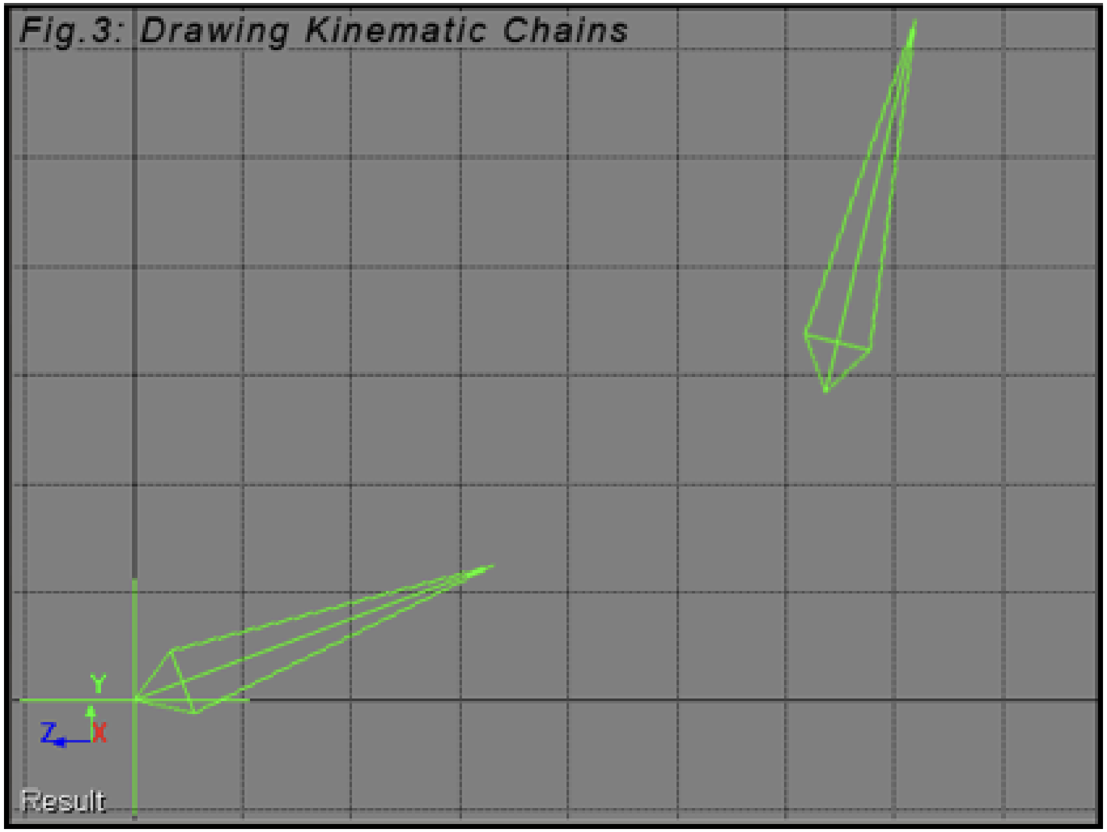
Drawing Kinematic Chains

- Starting with the next link's object:



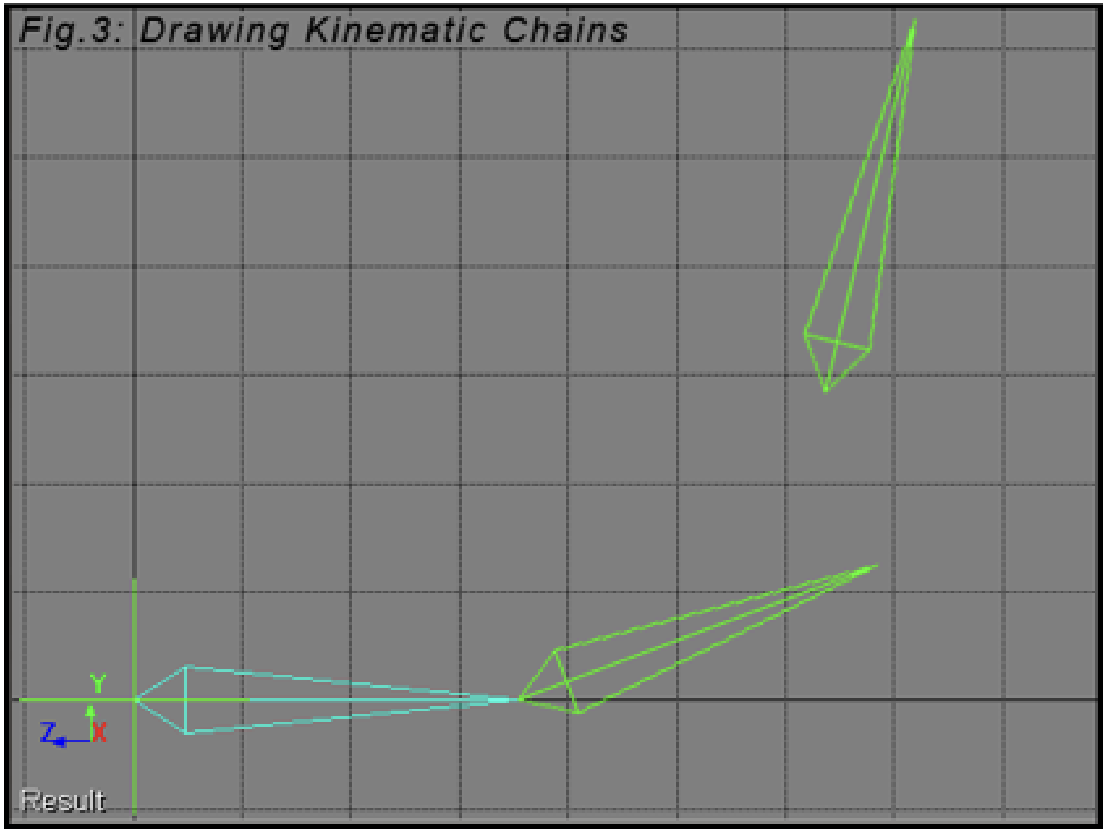
Drawing Kinematic Chains

- Starting with the next link's object:
  - Translate the object's base to the origin...
  - ... and rotate by the object's angle



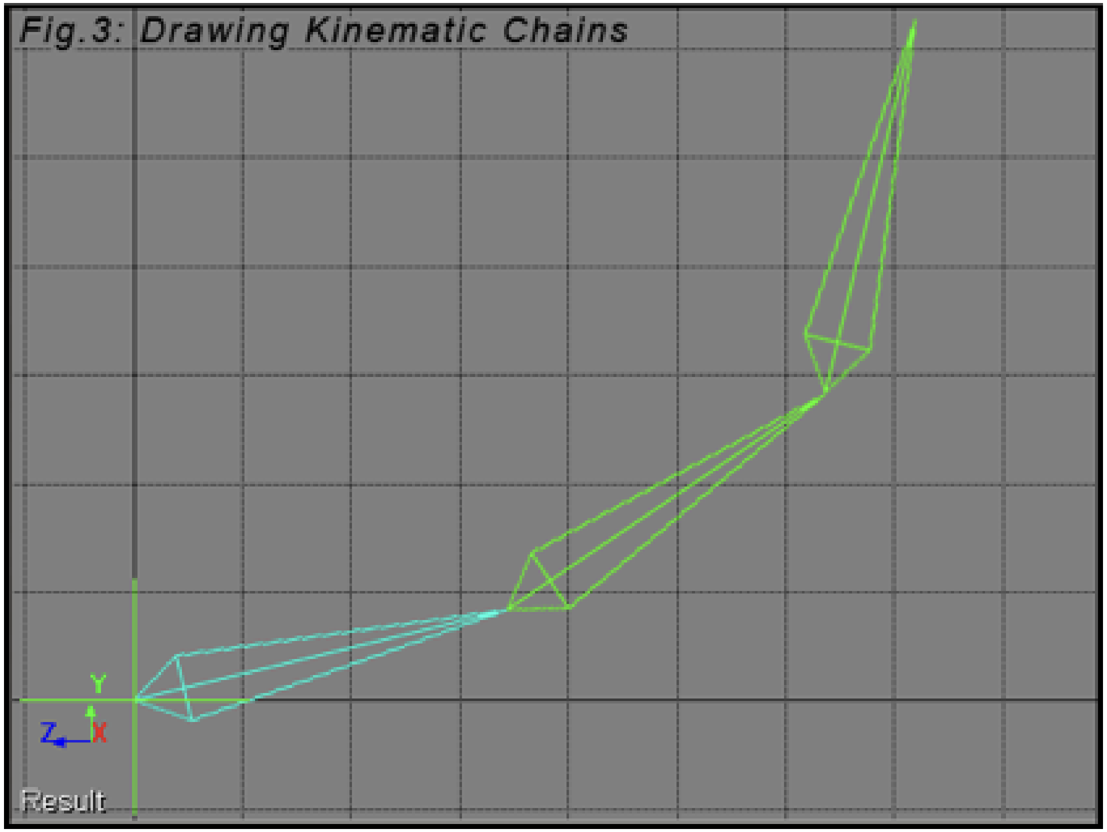
Drawing Kinematic Chains

- Starting with the next link's object:
    - Translate by the next object's length...



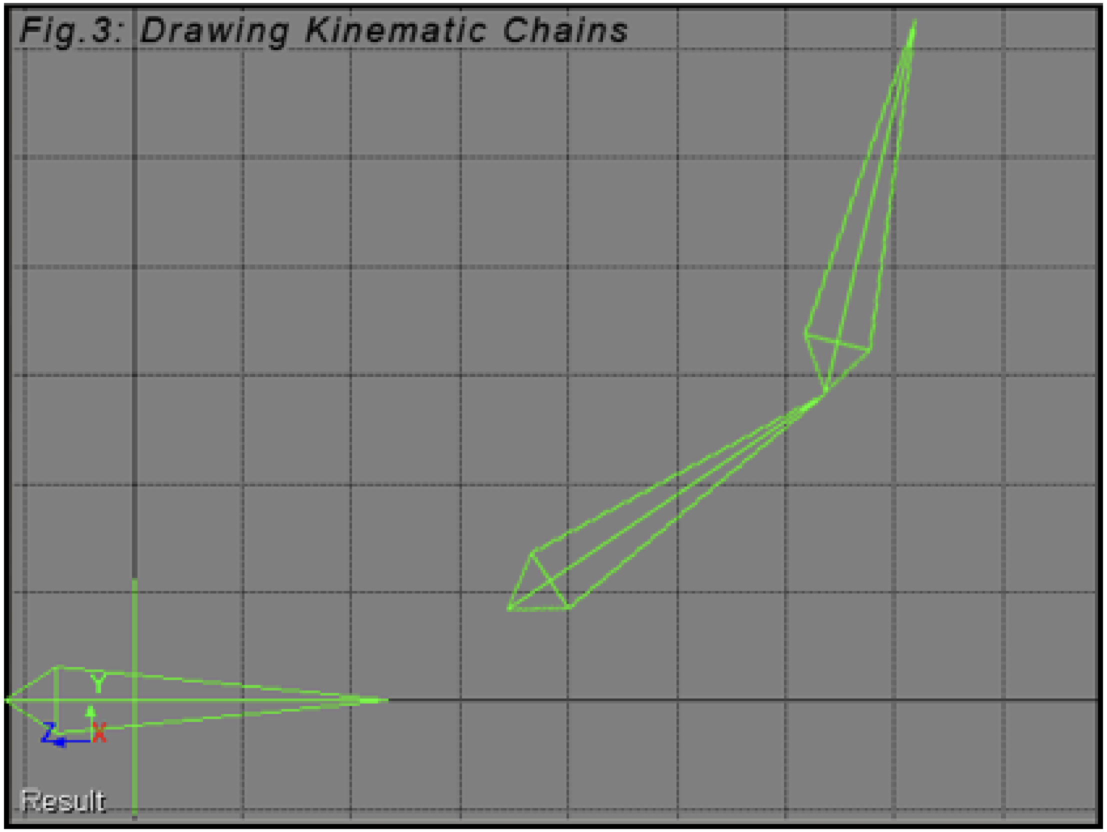
Drawing Kinematic Chains

- Starting with the next link's object:
  - Translate by the next object's length...
  - ... and rotate the entire chain by that object's angle



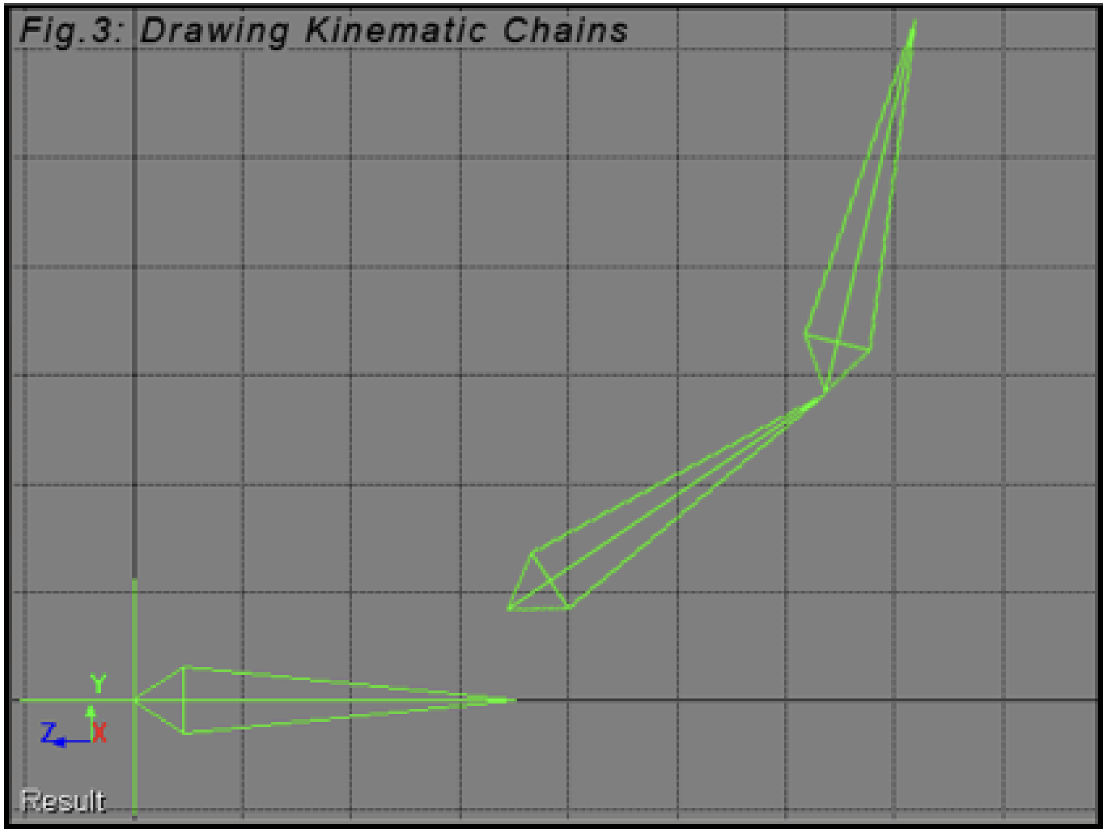
Drawing Kinematic Chains

- So on, and so forth...
    - Place the next link



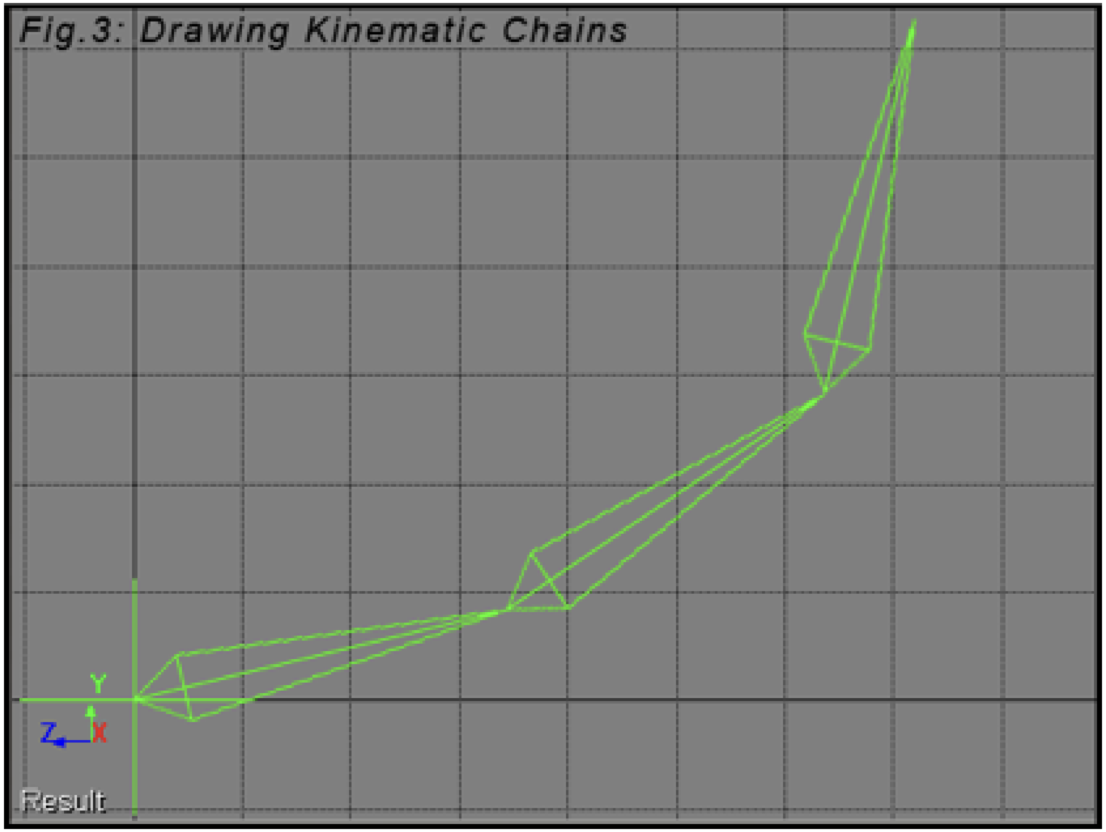
Drawing Kinematic Chains

- So on, and so forth…
  - Place the next link
  - Translate



Drawing Kinematic Chains

- So on, and so forth, until the chain is complete
  - Place the next link
  - Translate
  - Rotate



Inverse Kinematics

- Now that we know what Forward Kinematics (FK) is, what is Inverse Kinematics (IK)?

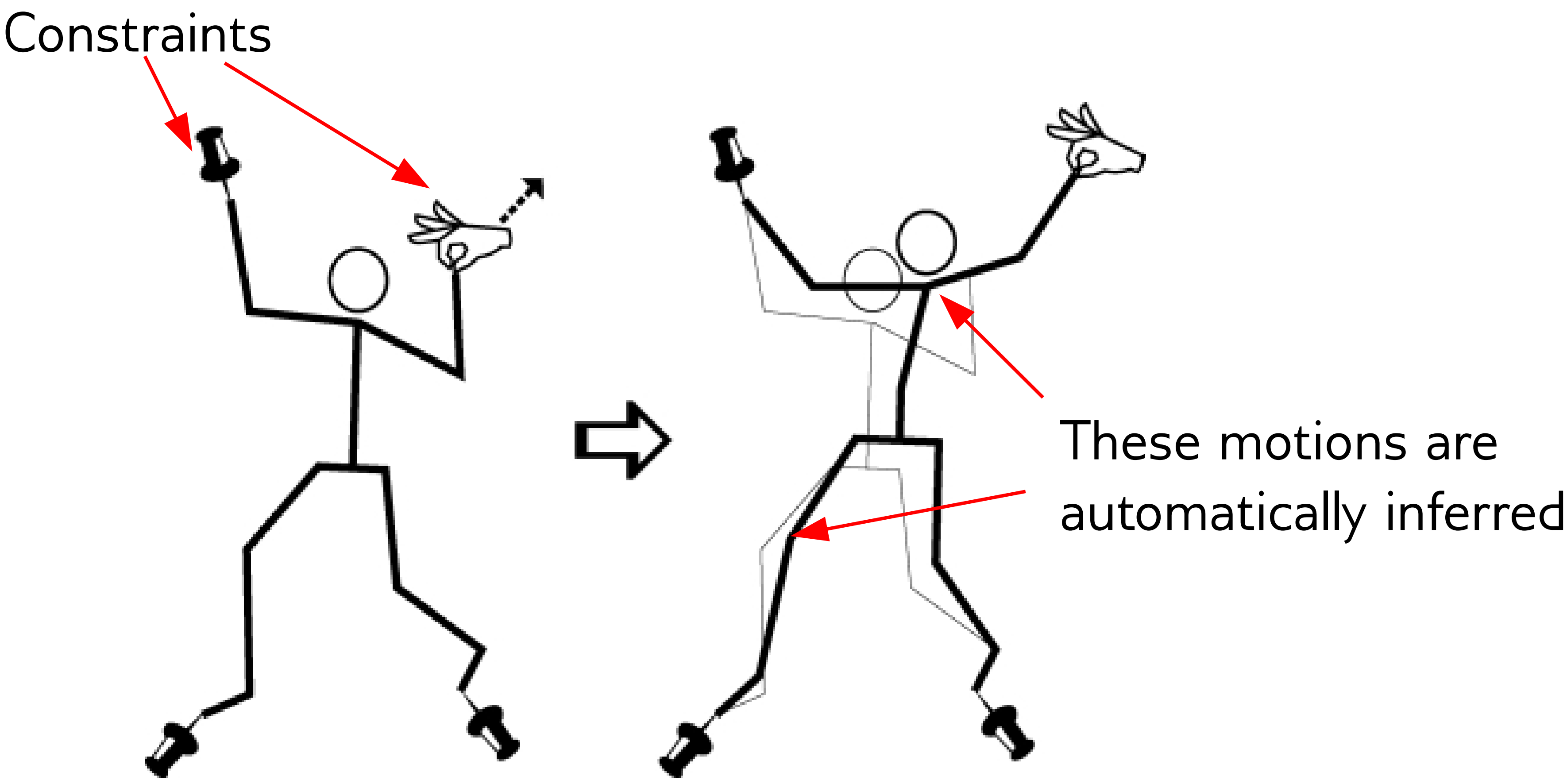
Inverse Kinematics

- Now that we know what Forward Kinematics (FK) is, what is Inverse Kinematics (IK)?

- **Inverse Kinematics:** Mathematically determining the positions and angles of joints in a flexible, jointed object, given the position and orientation of some subset of the joints (typically the end effectors)

Inverse Kinematics

- Now that we know what Forward Kinematics (FK) is, what is Inverse Kinematics (IK)?



"UTPoser", Yamane and Nakamura, 2003

Inverse Kinematics

- What is IK used for?
  - Originally used in industrial robotics for assembly plants
    - To get the robot to weld this point, how do I have to position all the links in its arm?
  - In computer graphics, IK is typically used for character animation
    - Animator manipulates a few handles (e.g. hands, feet)
    - The system infers the pose of the rest of the skeleton

Types of IK solutions

- Closed form/analytical solution
  - Calculate sequence of joint angles from the root to the effector, allowing us to determine if a solution is even possible
  - By parametrizing the solution, we can get a range of feasible configurations
  - For a two link chain the solution is (almost) unique:

$$\Theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - L_1^2 - L_2^2}{2 L_1 L_2}\right)$$

$$\Theta_1 = \tan^{-1}\left(\frac{-L_2 \sin\Theta_2 x + (L_1 + L_2 \cos\Theta_2) y}{L_2 \sin\Theta_2 y + (L_1 + L_2 \cos\Theta_2) x}\right)$$

Types of IK solutions

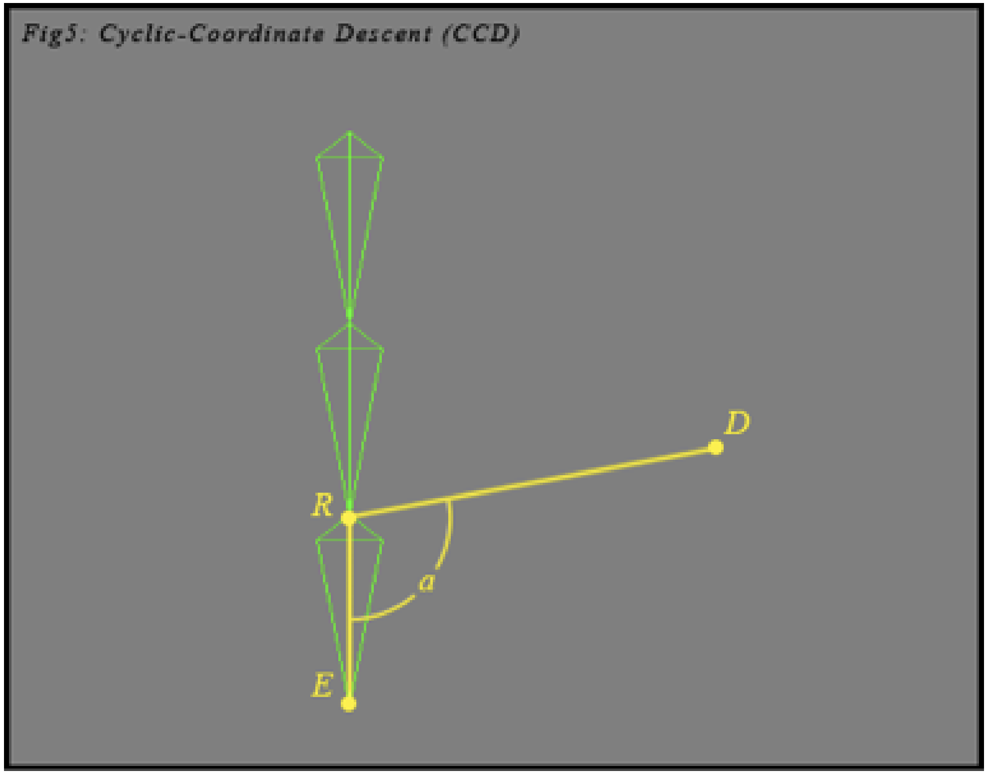
- Closed form/analytical solution
  - Relatively simple solution for smaller problems
  - However, as the chain increases in length, each new element adds new degrees of freedom, and the problem quickly becomes complex

Types of IK solutions

- Numerical solutions
  - Suitable for complex linkages
  - Iteratively improve solution, progressing towards goal configuration
  - **Cyclic Coordinate Descent** (this class)
  - **Inverse Jacobian Method** (next class)

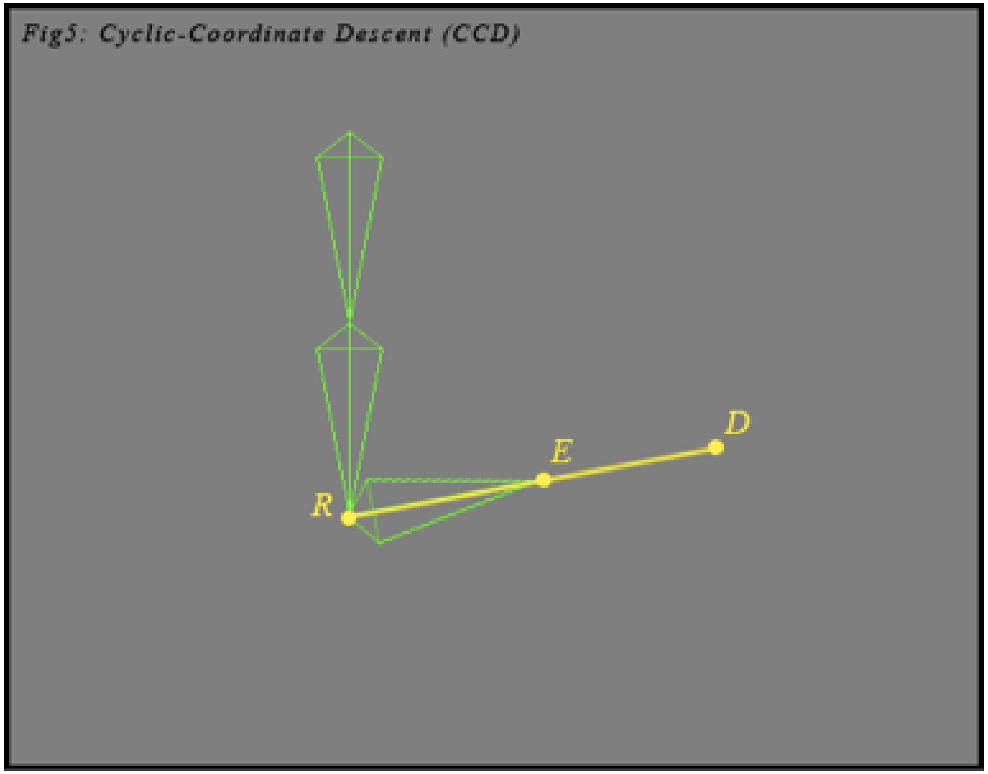
Cyclic Coordinate Descent

- Start with a vector from the root of our effector R to the current endpoint E
- Draw a vector from R to the desired endpoint D
- The inverse cosine of the dot product gives the angle between the vectors: $\cos a = \vec{RD} \cdot \vec{RE}$



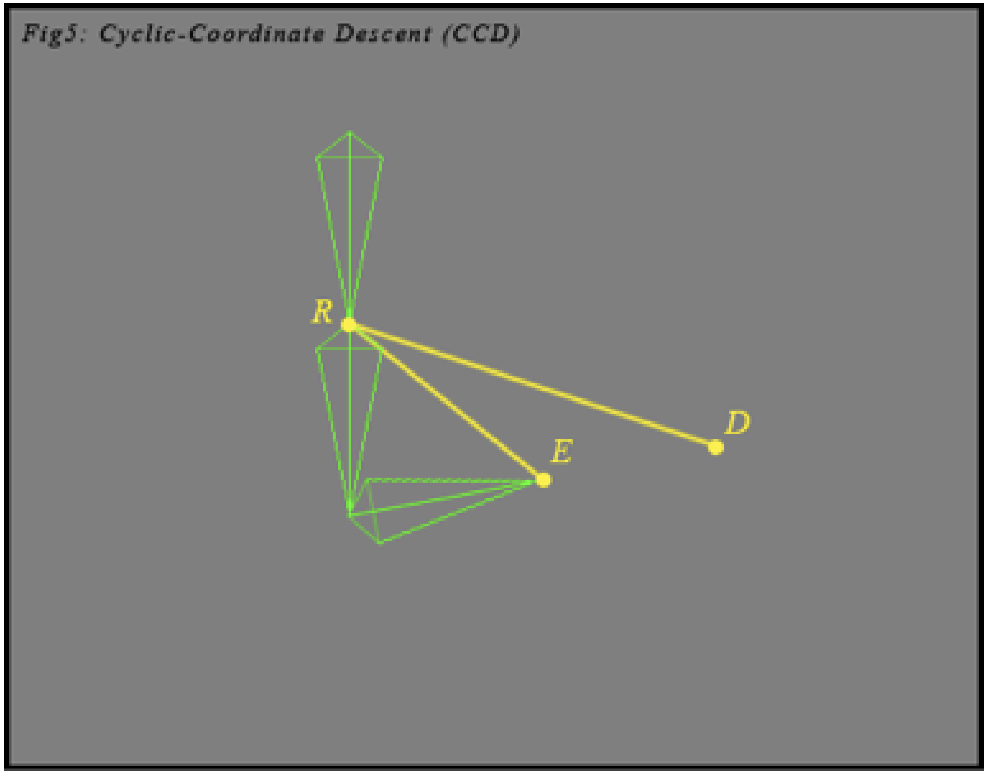
Cyclic Coordinate Descent

- Rotate our link so that RE falls on RD



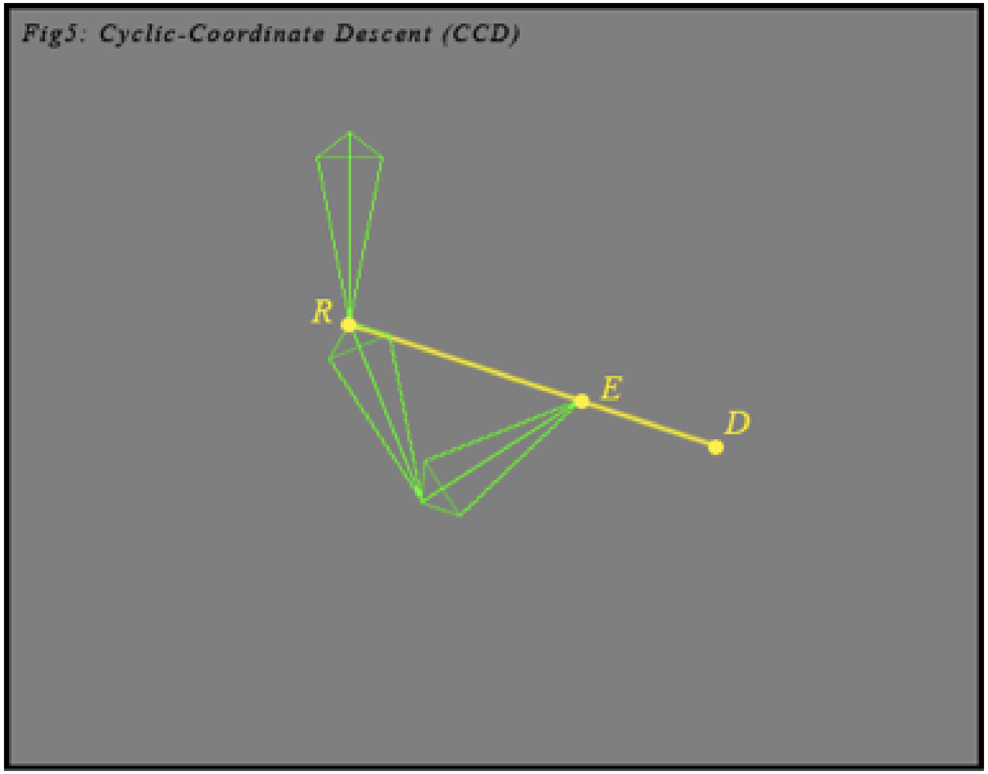
Cyclic Coordinate Descent

- Move one link up the chain and repeat the process



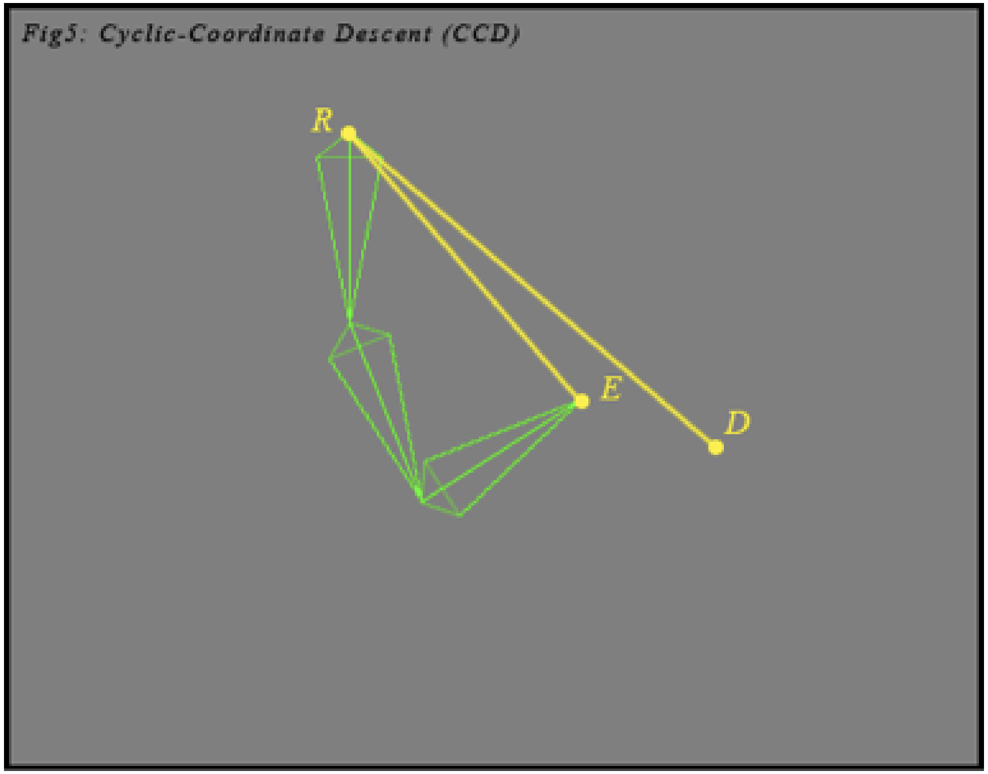
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



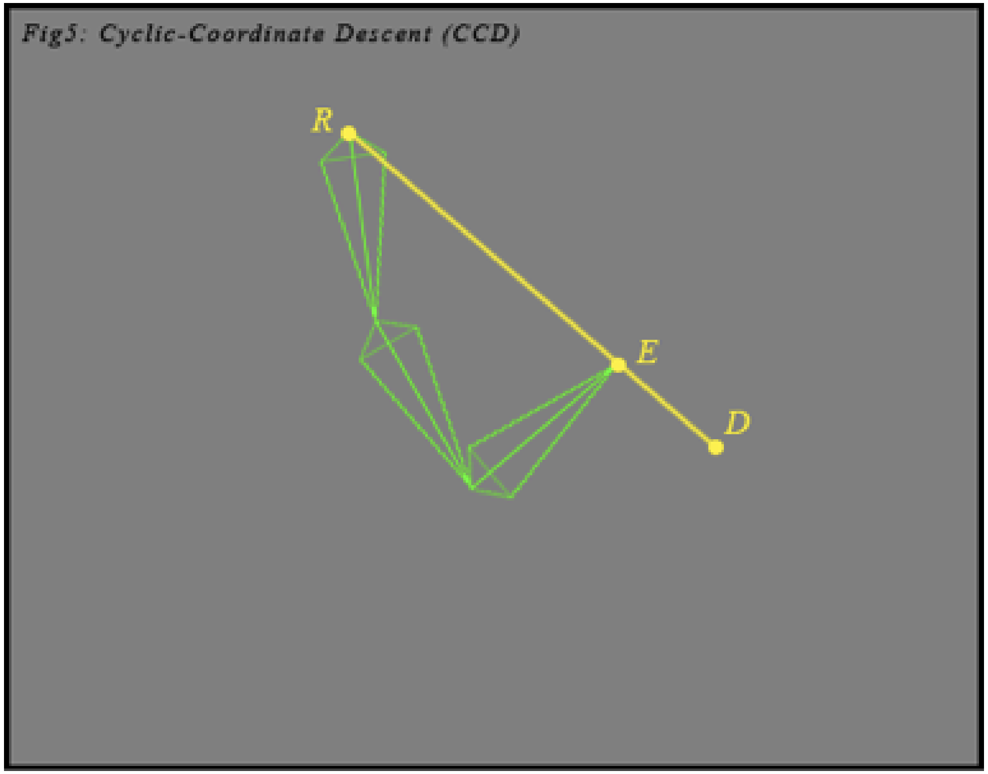
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



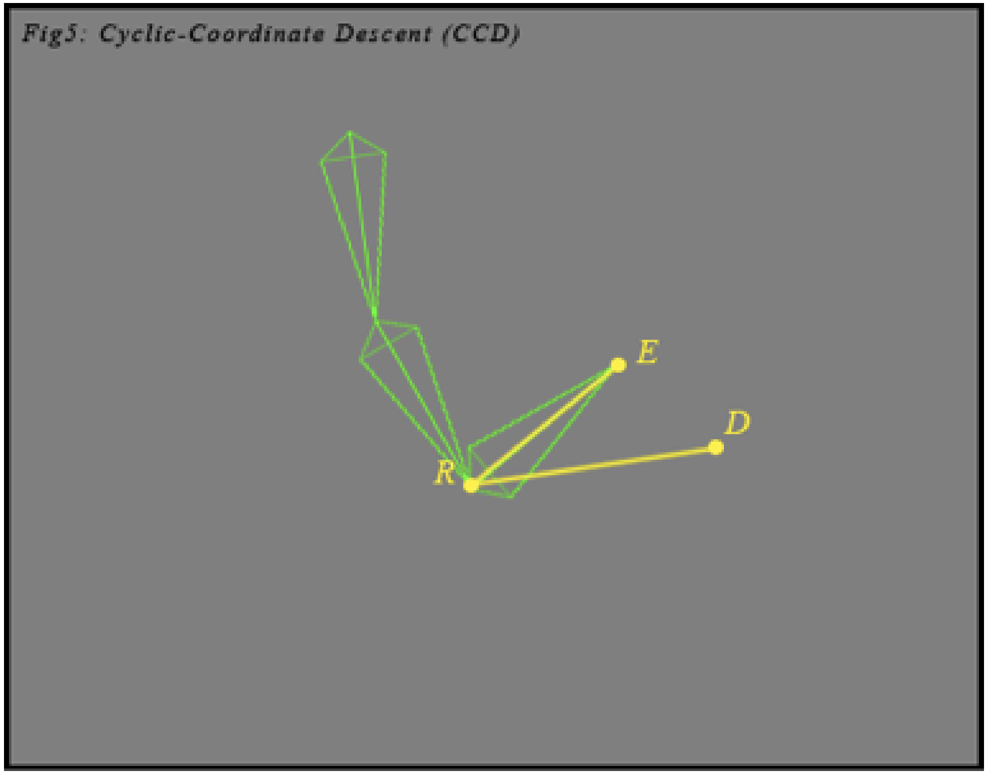
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



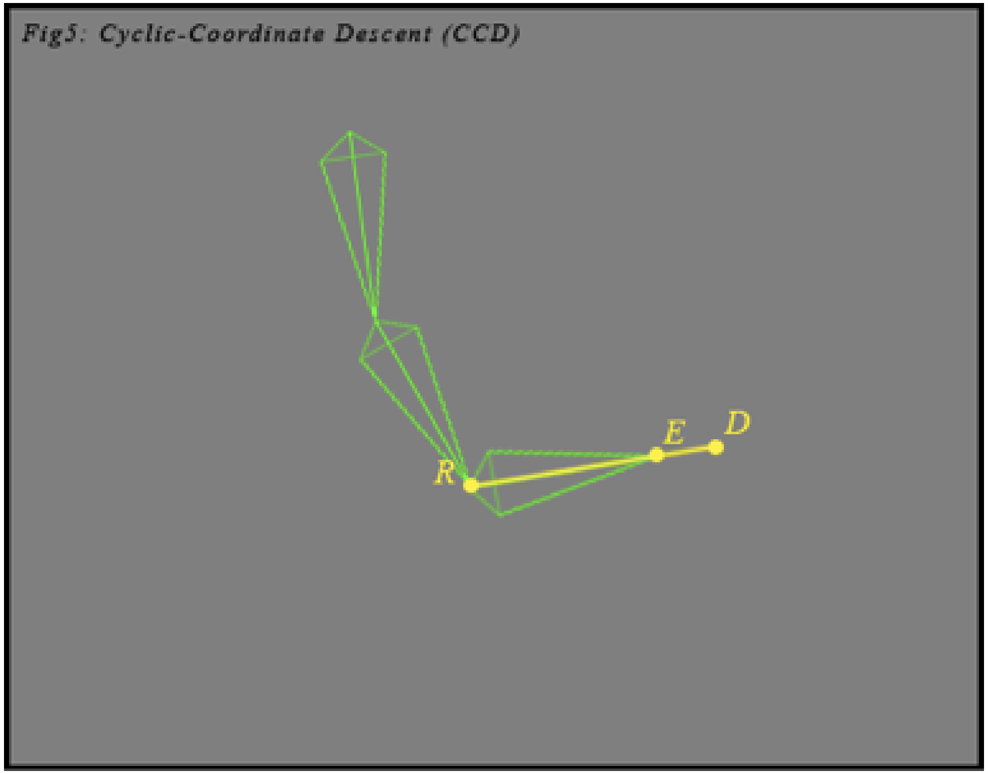
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



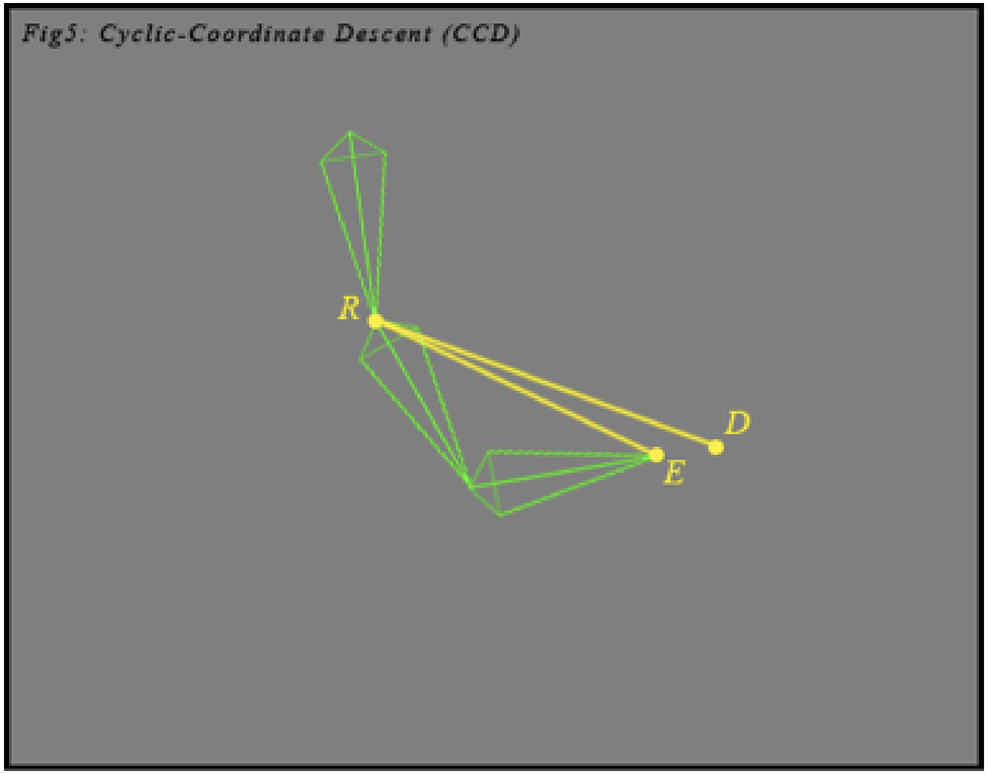
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



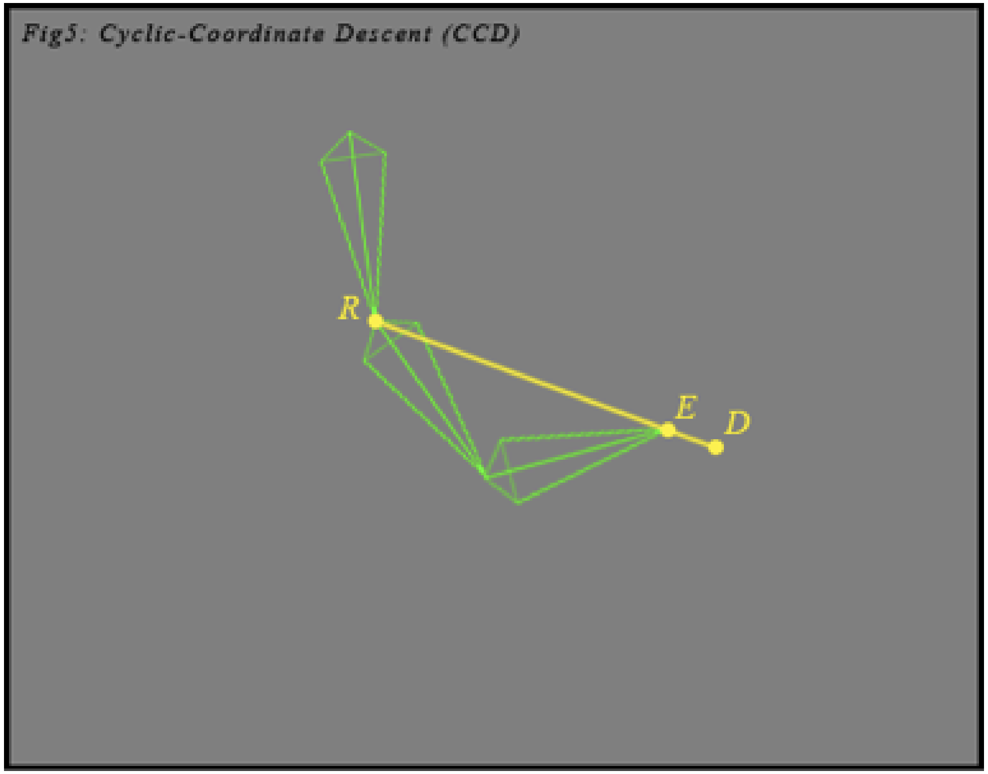
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



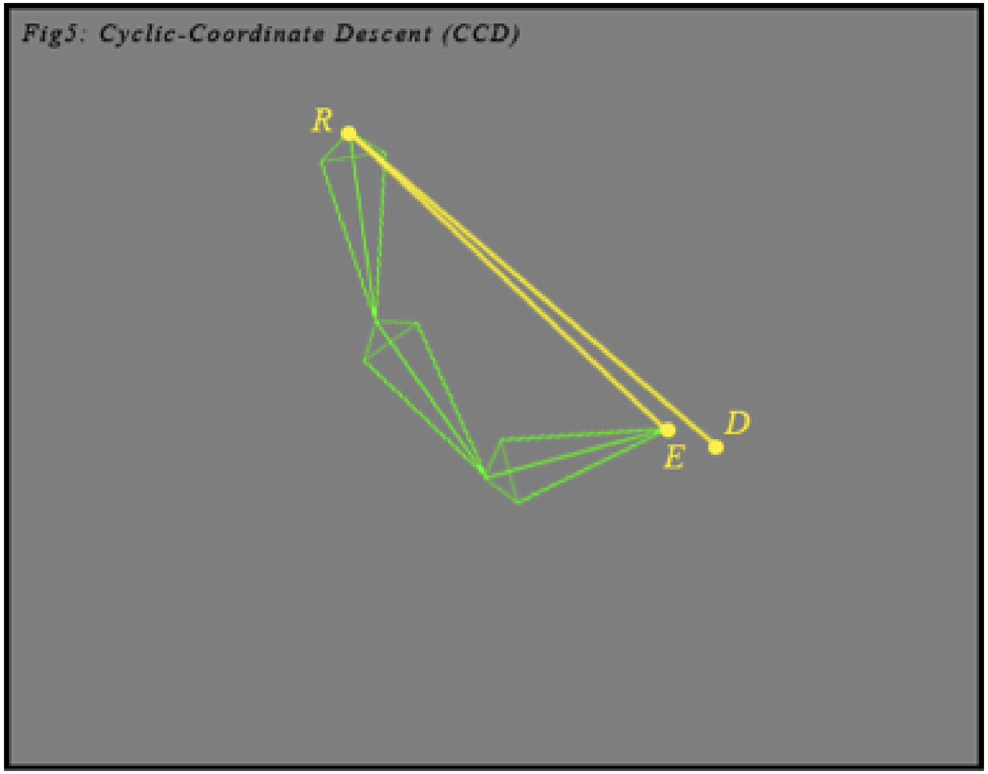
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



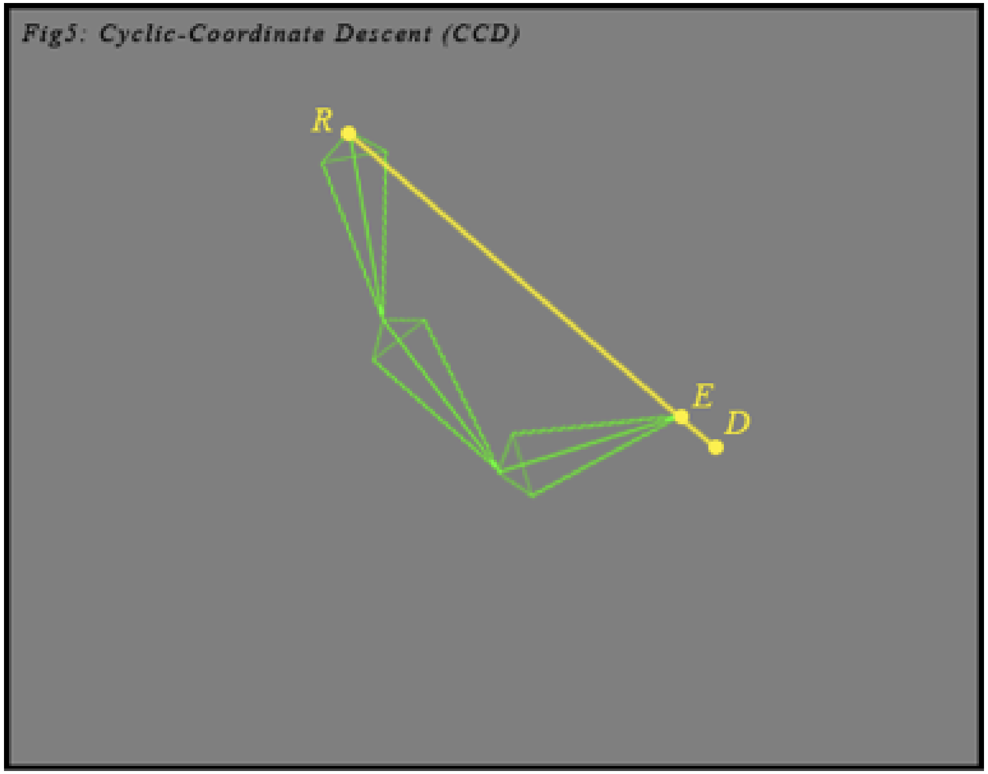
Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



Cyclic Coordinate Descent

- The process is repeated until the root joint is reached. Then, the process begins all over again starting with the end effector, and continues until we are close enough to D for an acceptable solution.



Inverse Jacobian Method

- Q: n-vector of internal parameters (joint angles)
- X: 6-vector defining the endpoint's position and orientation
  - … we can have other target configurations/constraints as well, but we'll currently stick with this for simplicity
- Assume $n \geq 6$
- Forward kinematics: $X = F(Q)$
- $dx_i = \frac{\partial f_i(Q)}{d\,q_1} dq_1 + \frac{\partial f_i(Q)}{d\,q_2} dq_2 + \ldots + \frac{\partial f_i(Q)}{d\,q_n} dq_n$
- $\mathrm{d}X = J\,\mathrm{d}Q$

The Jacobian

$$J = \begin{bmatrix} \frac{\partial f_1(Q)}{d q_1} & \frac{\partial f_1(Q)}{d q_2} & \cdots & \frac{\partial f_1(Q)}{d q_n} \\ \frac{\partial f_2(Q)}{d q_1} & \frac{\partial f_2(Q)}{d q_2} & \cdots & \frac{\partial f_2(Q)}{d q_n} \\ \vdots & \vdots & \ddots & \\ \frac{\partial f_6(Q)}{d q_1} & \frac{\partial f_6(Q)}{d q_2} & \cdots & \frac{\partial f_6(Q)}{d q_n} \end{bmatrix}$$

If J is square, $\mathrm{d}X = J\,\mathrm{d}Q$ implies $\mathrm{d}Q = J^{-1}\,\mathrm{d}X$

Case where $n = 6$

- J is a square 6x6 matrix
- **Problem:** Given D (desired endpoint location), find Q such that $D = F(Q)$
- **Solution**, starting at any initial configuration X_0 with known parameters Q_0
  - Interpolate linearly between X_0 and D
    - Produces sequence $X_1, X_2, X_3, \ldots, X_p = \mathrm{D}$
  - For $i = 1, 2, \ldots, p$ do
    - $Q_i = Q_{i-1} + J^{-1}(\mathrm{Q}_{i-1})(X_i - X_{i-1})$
    - Reset X_1 to $F(Q_i)$

Case where $n > 6$

- J is a 6 x n matrix
- Recall: we're trying to solve the linear system

$$\mathrm{d}X = J\,\mathrm{d}Q$$

- for $\mathrm{d}Q$, where $\mathrm{d}X = X_i - X_{i-1}$, and J is the Jacobian evaluated at Q_{i-1}
- The system is under-determined!
  - Too few constraints (6)
  - Too many variables (n)
- We can't take the inverse of J

Non-square linear system

- Assume we're trying to solve $A\mathbf{x} = \mathbf{b}$

  ... but A is not square!

- So we'll add some constraints:
  - If A has more rows than columns (over-determined), find $\mathbf{x}$ that minimizes
    $$\| A\mathbf{x} - \mathbf{b} \|$$
  - If A has more columns than rows (under-determined), find $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$ that minimizes
    $$\| \mathbf{x} \|$$

Enter the pseudoinverse

- If A has more columns than rows (under-determined), find $\mathbf{x}$ satisfying $A\mathbf{x} = \mathbf{b}$ that minimizes $\| \mathbf{x} \|$
- It turns out that this solution is given by

$$\mathbf{x} = A^{+}\,\mathbf{b}$$

where A^{+} is the **pseudoinverse** $A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}$ of A

(we'll assume A is full rank so this pseudoinverse formula works)

Why least-norm solution?

Let $\mathbf{x}^*$ be the pseudoinverse solution $A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}\,\mathbf{b}$

Proof that it is a solution:

- $A\,\mathbf{x}^* = A\,A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}\,\mathbf{b} = (A\,A^{\mathrm{T}})(AA^{\mathrm{T}})^{-1}\,\mathbf{b} = I\,\mathbf{b} = \mathbf{b}$

Why least-norm solution?

Proof that it is least-norm:

- Consider any solution $\mathbf{x}$ of $A\mathbf{x} = \mathbf{b}$
- ... we have $A(\mathbf{x} - \mathbf{x}^*) = \mathbf{b} - \mathbf{b} = 0$
- $$\begin{aligned} (\mathbf{x} - \mathbf{x}^*)^{\mathrm{T}}\mathbf{x}^* &= (\mathbf{x} - \mathbf{x}^*)^{\mathrm{T}} A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}\mathbf{b} \\ &= (A(\mathbf{x} - \mathbf{x}^*))^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}\mathbf{b} \\ &= 0^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}\mathbf{b} \\ &= 0 \end{aligned}$$
- ... so $\mathbf{x} - \mathbf{x}^*$ and $\mathbf{x}^*$ are orthogonal
- Hence, $||\mathbf{x}||^2 = ||\mathbf{x}^* + \mathbf{x} - \mathbf{x}^*||^2 = ||\mathbf{x}^*||^2 + ||\mathbf{x} - \mathbf{x}^*||^2 \geq ||\mathbf{x}^*||^2$

Derivation via Lagrange multipliers

- **Problem:** minimize $\|\mathbf{x}\|^2 = \mathbf{x}^T\mathbf{x}$
  subject to $A\mathbf{x} = \mathbf{b}$
- **Solution:** Introduce Lagrange multiplier $\boldsymbol{\lambda}$

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{x}^T\mathbf{x} + \boldsymbol{\lambda}^T(A\mathbf{x} - \mathbf{b})$$

Hence the minimum of L is also a solution of the linear system

  - At the minimum of L, we have

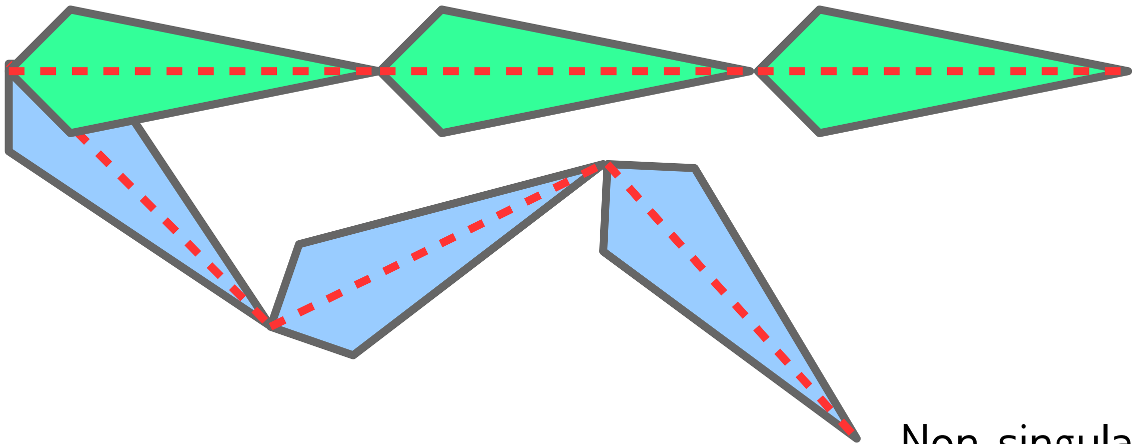
    $\nabla_{\mathbf{x}} L = 2\mathbf{x} + A^T\boldsymbol{\lambda} = 0$ and $\nabla_{\boldsymbol{\lambda}} L = A\mathbf{x} - \mathbf{b} = 0$
  - From first condition, we have $\mathrm{x} = -A^T \boldsymbol{\lambda} / 2$
  - Substituting into second condition, $\boldsymbol{\lambda} = -2(AA^T)^{-1}\mathbf{b}$
  - Hence, $\mathbf{x} = A^T(AA^T)^{-1}\mathbf{b}$

Inverse Jacobian Method

- Need to reach some target configuration D from initial configuration X_0
- **General Algorithm:**
  - Interpolate linearly between X_0 and D
    - Produces sequence $X_1, X_2, X_3, \ldots, X_p = \mathrm{D}$
  - For $i = 1, 2, \ldots, p$ do
    - $Q_i = Q_{i-1} + J^{+}(\mathrm{Q}_{i-1})(X_i - X_{i-1})$
    - Reset X_1 to $F(Q_i)$

Inverse Jacobian Method

- **Disadvantages:**
  - Slow to compute inverse of AA^{T}
  - Instability around singularities (J loses full rank)



- **Improvement in practice:**
  - Use the Jacobian transpose J^{T} instead of J^{+}

Jacobian Transpose Method

- Replace

$$\mathrm{d}Q = J^{+}\,\mathrm{d}X$$

with

$$dQ = J^{\mathrm{T}}\,dX$$

- Why does this work?!!!

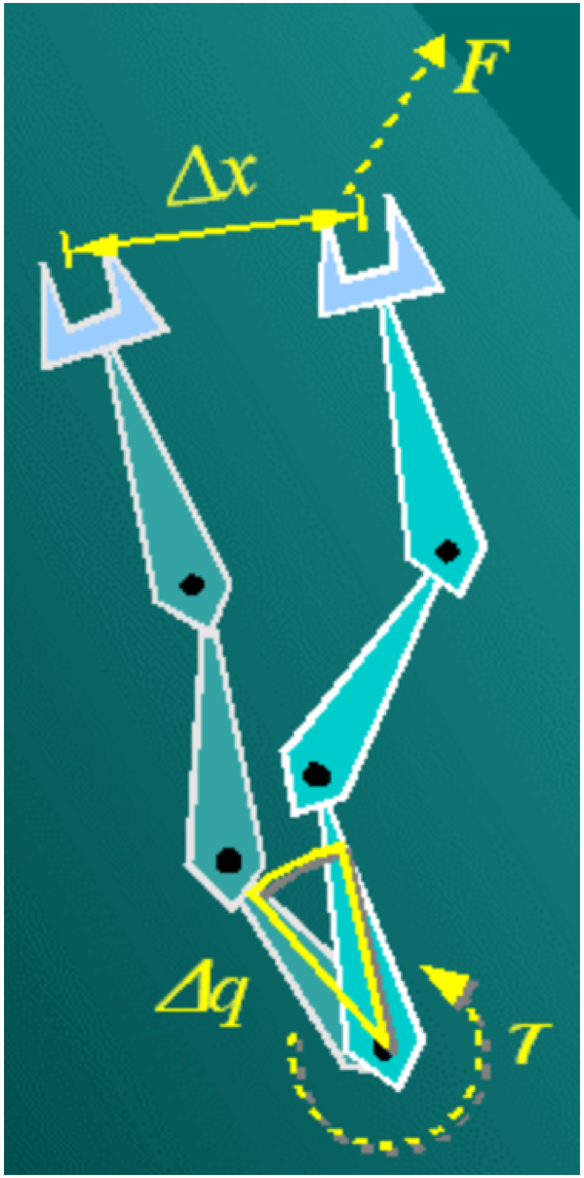
Principle of Virtual Work

- "External work = internal work"
- External work = force * distance
- Internal Work = torque * angle

$$\mathbf{f}^{\mathrm{T}}\, \mathrm{d}X = \boldsymbol{\tau}^{\mathrm{T}}\, \mathrm{d}Q$$

- $\mathrm{d}X = J\, \mathrm{d}Q$ (forward kinematics)
- $\mathbf{f}^{\mathrm{T}} J\, \mathrm{d}Q = \boldsymbol{\tau}^{\mathrm{T}}\, \mathrm{d}Q$ (substituting)
- $\mathbf{f}^{\mathrm{T}} J = \boldsymbol{\tau}^{\mathrm{T}}$ (holds for any dQ)

  i.e. $\boldsymbol{\tau} = J^{\mathrm{T}} \mathbf{f}$



Jacobian Transpose Method

- Virtual work equation:

$$\boldsymbol{\tau} = J^{\mathrm{T}}\mathbf{f}$$

- Compare with:

$$\mathrm{d}Q = J^{\mathrm{T}}\,\mathrm{d}X$$

- We're taking the distance to the goal to be a force that pulls our end effector
- With J^{+}, we had an exact solution to linearized problem
- … now no longer

Jacobian Transpose Method

- $\mathrm{d}Q = J^{\mathrm{T}}\,\mathrm{d}X$ is not exact, but has the right trend
- Throw in a scaling factor h and iterate

$$(\Delta Q)_{i+1} = h\,J^{\mathrm{T}}\,(\Delta X)_i$$

- h can be thought of as a timestep Δt

$$\frac{\Delta Q}{\Delta t} = J^{T}\,\Delta X$$

- So we're just solving the differential equation

$$\frac{dQ}{dt} = J^{T}X = J^{T}F(Q)$$