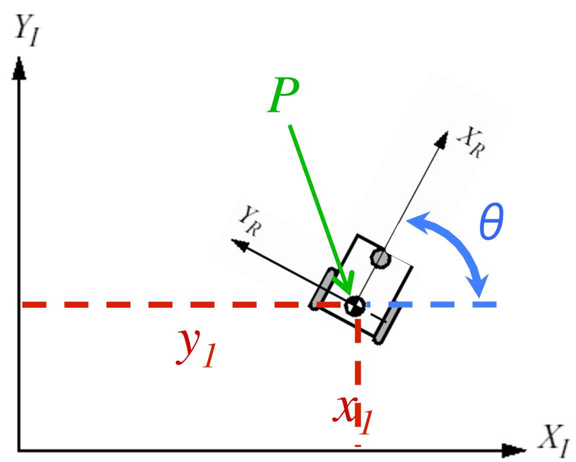
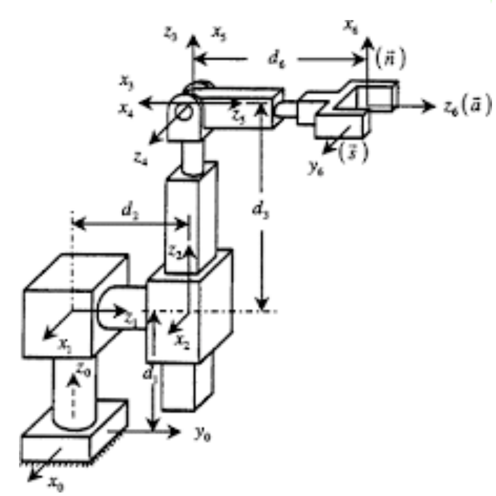
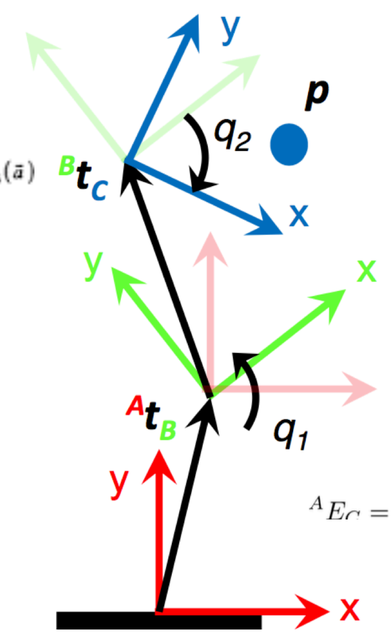
# Kinematics

## *Manipulator Kinematics*






***Many slides adapted from:***
*Siegwart, Nourbakhsh and Scaramuzza, Autonomous Mobile Robots*
*Renata Melamud, An Introduction to Robot Kinematics, CMU*
*Rick Parent, Computer Animation, Ohio State*
*Steve Rotenberg, Computer Animation, UCSD*

# Bookkeeping



- Homework
  - Resolution, Kinematics & IK, Course Progress
  - Due Thursday night
- Quiz 3: Manipulation, Grasping, Kinematics
  - Due tomorrow night (not Thursday)
- Assignment 3:
  - Build hardware!
- Today:
  - Transformations, affines
  - Chasles' theorem

# Assignment 3



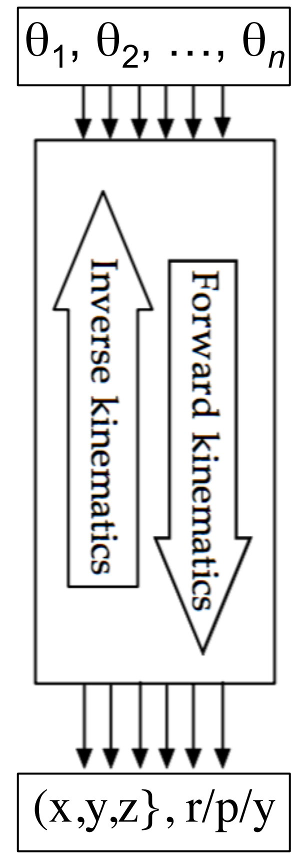
- Build LED circuit
  - Breadboard-based building
  - Parts in lab / class Thursday
- Build motor
  - Very simple conceptually
- Some of this will be in-class workshop
  - 12th November
  - Due on 13th

# Forward & Inverse



- Forward:
  - Inputs: joint angles
  - Outputs: coordinates of end-effector
- Inverse:
  - Inputs: desired coordinates of end-effector
  - Outputs: joint angles
- Inverse kinematics are tricky
  - Multiple solutions
  - No solutions
  - Dead spots

Joint space (robot space – previously $R$)

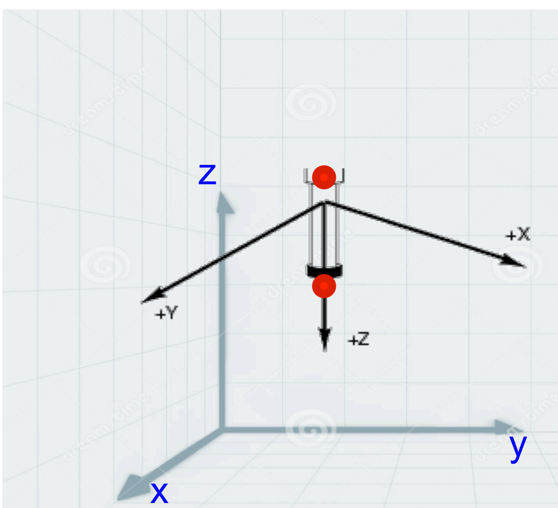



Cartesian space (global space – previously $I$)

# Actual Goal



- Transform between robot and world coordinates
  - Why?
- Transformation of parts (points) of the robot



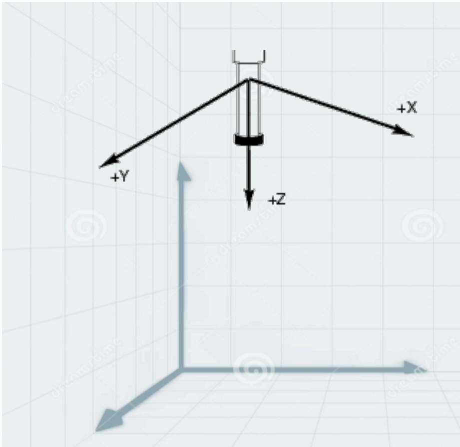

R: {0, 0, 0}
I: {4, 2, 3}

R: {0, 0, -2}
I: {4, 2, 5}

# Actual Goal



- Affine transformation
  - Preserves collinearity (i.e., all points lying on a line initially still lie on a line after transformation)
  - Preserves ratios of distances (e.g., the midpoint of a line segment remains the midpoint after transformation)
- Rigid transform
  - Reflections, translations, rotations
  - Preserves internal relationship of points
    - Distances between every pair of points
    - (Remember, this is not the robot moving!)



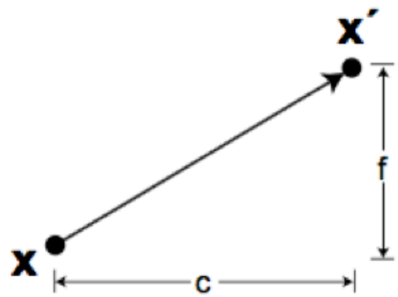
# Affine Transformations



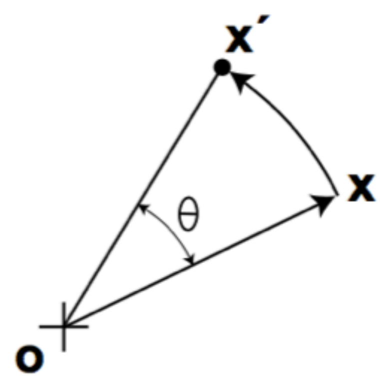
- Affine transformations:
- Given a point x (x,y), transformed x' can be written:

$$x' = \begin{bmatrix} ax + by + c \\ dx + ey + f \end{bmatrix}$$

- Translation
- Rotation
- Scaling
- Shear




$$x' = \begin{bmatrix} x+c \\ y+f \end{bmatrix}$$




$$x' = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

# Homogenous Coordinates



- These can all be done with matrix multiplication

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} ax + by \\ dx + ey \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- But, this is not a linear transform
  - Represent points in space using vectors
  - Transform using 2x2 (or 3x3) matrices

    But:
  - Multiplying a zero vector by a matrix gives you?
    - Another zero vector
  - Can't move the origin!

# Homogenous Coordinates



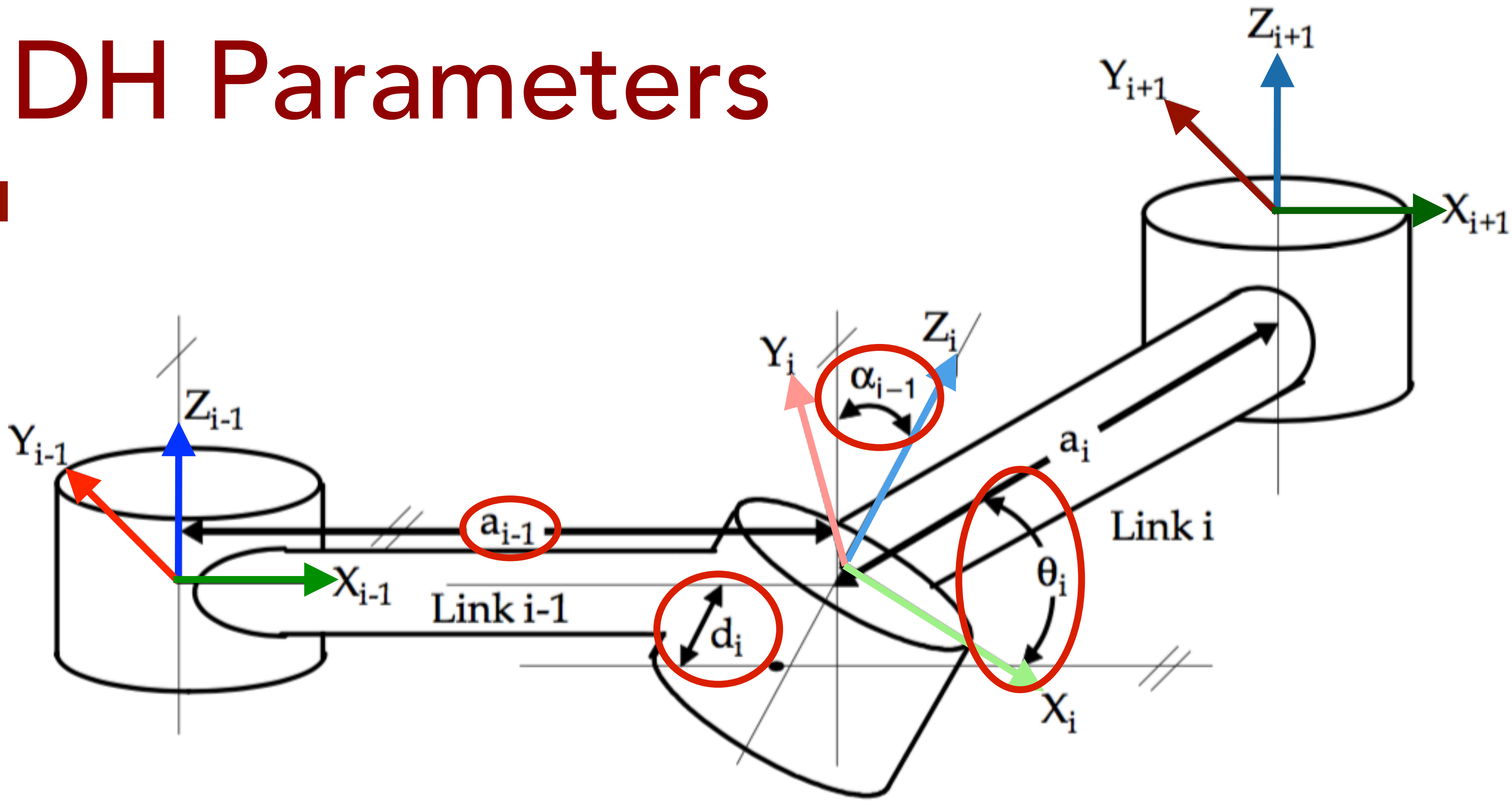
- So we need homogenous coordinates
- Add identity column/row

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Translation becomes $\begin{bmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix}$

# DH Parameters






$a_{i-1}$ : link length – distance $Z_{i-1}$ and $Z_i$ along $X_i$

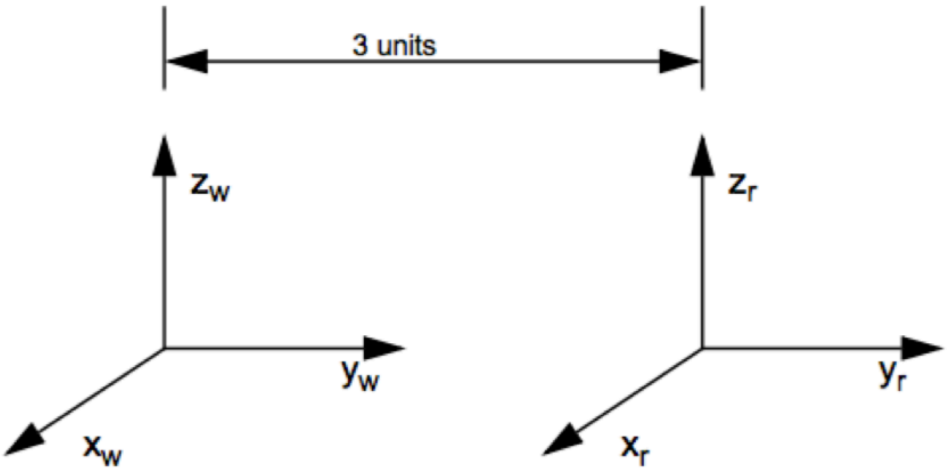
$\alpha_{i-1}$ : link twist – angle $Z_{i-1}$ and $Z_i$ around $X_i$

$d_i$ : link offset – distance $X_{i-1}$ to $X_i$ along $Z_i$

$\theta_2$ : joint angle – angle $X_{i-1}$ and $X_i$ around $Z_i$

# Translation






$$\xi_I = \begin{bmatrix} x_I \\ y_I \\ z_I \\ \theta \end{bmatrix} \qquad \xi_R = \begin{bmatrix} x_R \\ y_R \\ z_R \\ \theta \end{bmatrix}$$

Origin of R in I:

$$\begin{bmatrix} 0 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$

In 3D:

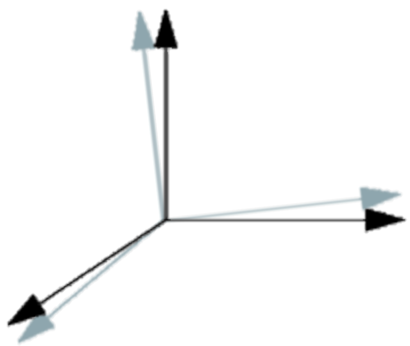
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Generally:

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

# Rotation





$$\xi_I = \begin{bmatrix} x \\ y \\ z \\ \theta_I \end{bmatrix} \qquad \xi_R = \begin{bmatrix} x \\ y \\ z \\ \theta_R \end{bmatrix}$$

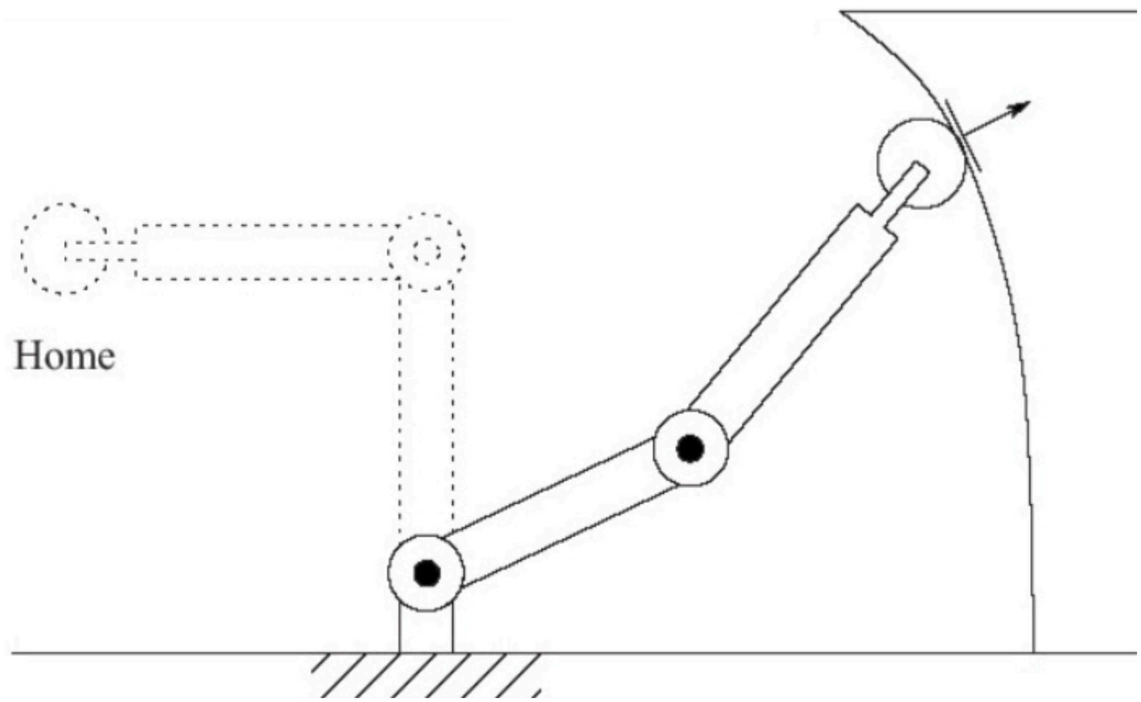
Generally:

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

*Introduction to Homogeneous Transformations & Robot Kinematics*
Jennifer Kay 2005

# Example: Rotation in Plane






$$x = a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2)$$
$$y = a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2)$$
$$a_i = \text{ the length of } i\text{th link}$$

# Transformation i to i-1 (2)



$a_{i-1}$ : distance $Z_{i-1}$ and $Z_i$ along $X_i$
$\alpha_{i-1}$ : angle $Z_{i-1}$ and $Z_i$ around $X_i$

screw displacement:

$$[X_i] = \text{Trans}_{X_i}(a_{i,i+1})\,\text{Rot}_{X_i}(\alpha_{i,i+1})$$

$d_i$ : distance $X_{i-1}$ to $X_i$ along $Z_i$
$\theta_2$ : angle $X_{i-1}$ and $X_i$ around $Z_i$

screw displacement:

$$[Z_i] = \text{Trans}_{Z_i}(d_i)\,\text{Rot}_{Z_i}(\theta_i)$$

- Coordinate transformation:

$$^{i-1}T_i = [Z_i][X_i] = \text{Trans}_{Z_i}(d_i)\,\text{Rot}_{Z_i}(\theta_i)\,\text{Trans}_{X_i}(a_{i,i+1})\,\text{Rot}_{X_i}(\alpha_{i,i+1}),$$

# Transformation i to i-1 (3)



$$\text{Trans}_{Z_i}(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{Z_i}(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Trans}_{X_i}(a_{i,i+1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i,i+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{Rot}_{X_i}(\alpha_{i,i+1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i,i+1} & -\sin\alpha_{i,i+1} & 0 \\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Transformation in DH:

$$^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_{i,i+1} & \sin\theta_i\sin\alpha_{i,i+1} & a_{i,i+1}\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_{i,i+1} & -\cos\theta_i\sin\alpha_{i,i+1} & a_{i,i+1}\sin\theta_i \\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

# Inverse Kinematics



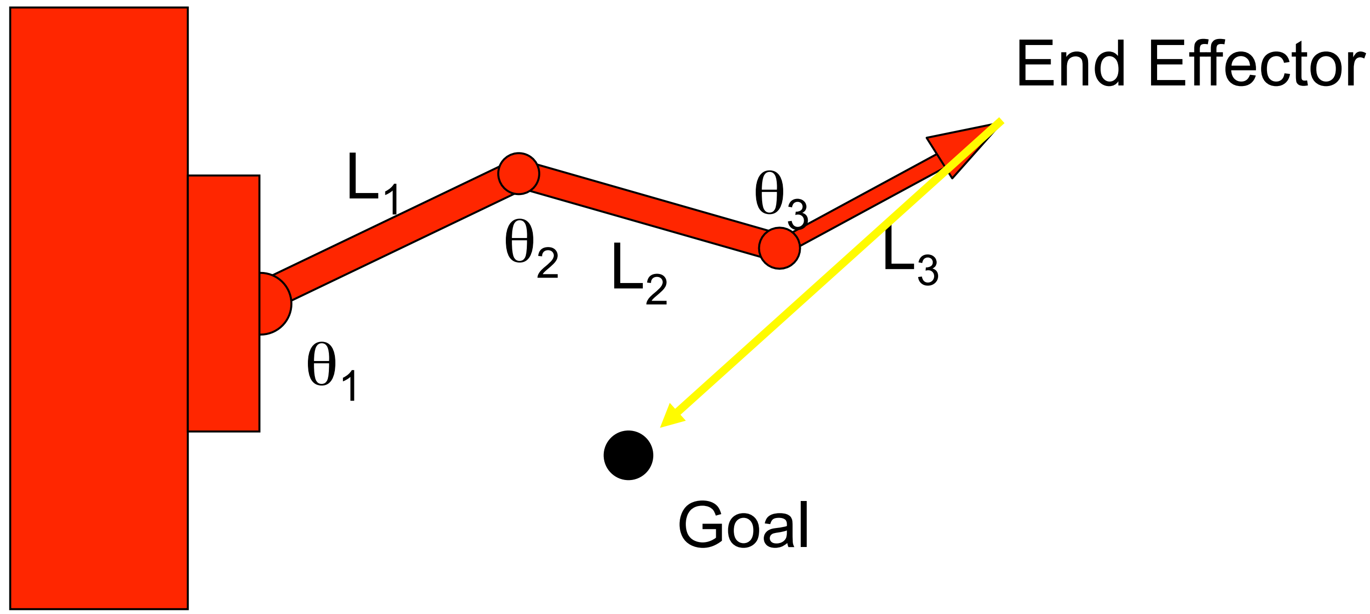
1. Set **goal configuration** of **end effector**
2. calculate interior **joint angles**

- Compute the vector of joint DOFs that will cause the end effector to reach some desired goal state
- In other words, it is the inverse previous problem

- Analytic approach
  - Directly calculate joint angles in configuration that satisfies goal
- Numeric approach
  - At each time slice, determine joint movements that take you in direction of goal position (and orientation)

# Inverse Kinematics





# Inverse Kinematics



- Underconstrained
  - Fewer constraints than DoFs
  - Many solutions
- Overconstrained
  - Too many constraints
  - No solution
- Reachable workspace
  - Volume the end effector can reach
- Dextrous workspace
  - Volume end effector can reach in any orientation

# Analytic



Given arm configuration (L1, L2, …)

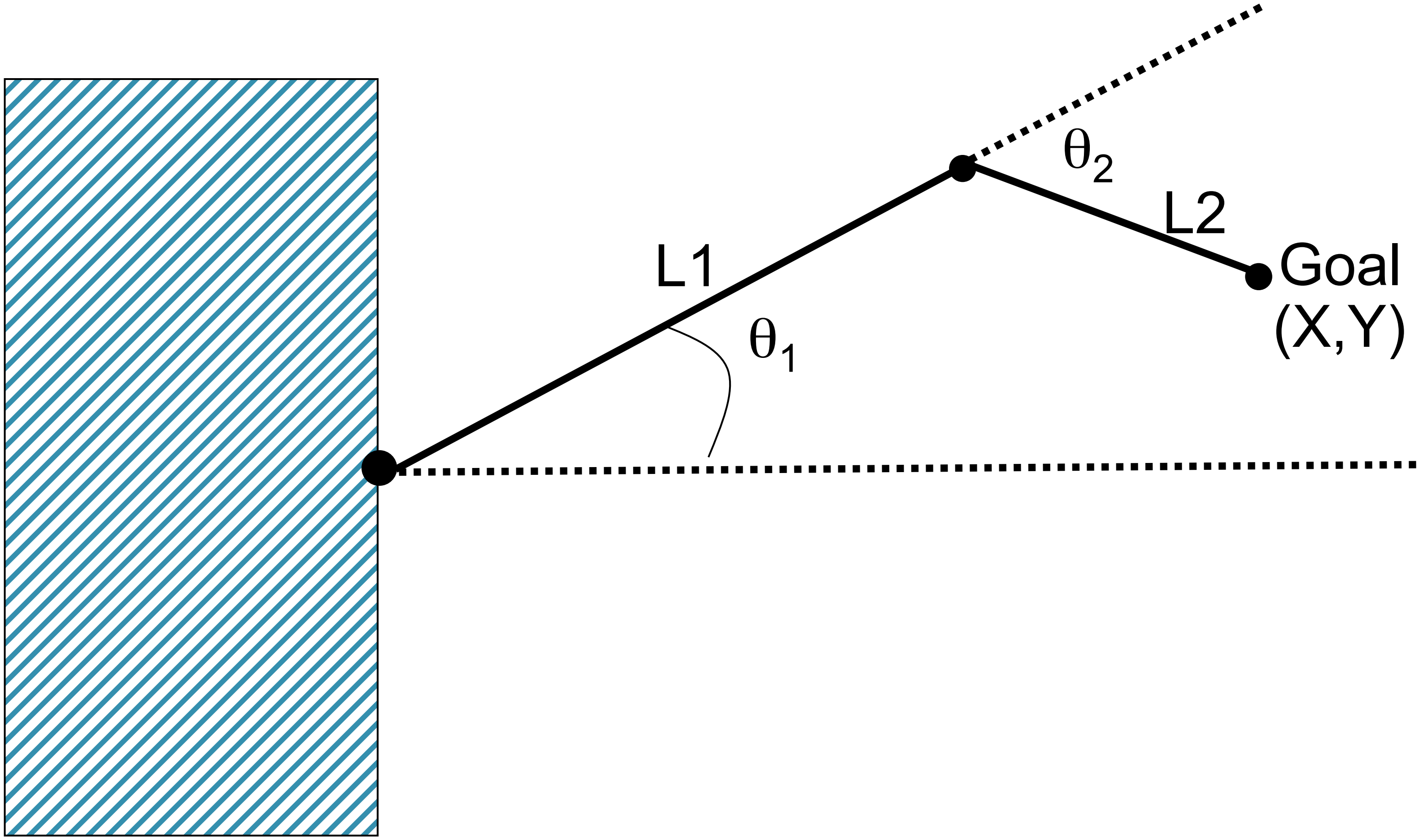
Given desired goal position (and orientation) of end effector: [x,y,z, $\psi 1, \psi 2, \psi 3$]

Analytically compute goal configuration ($\theta 1, \theta 2$)

Interpolate pose vector from initial to goal

# Analytic Inverse Kinematics

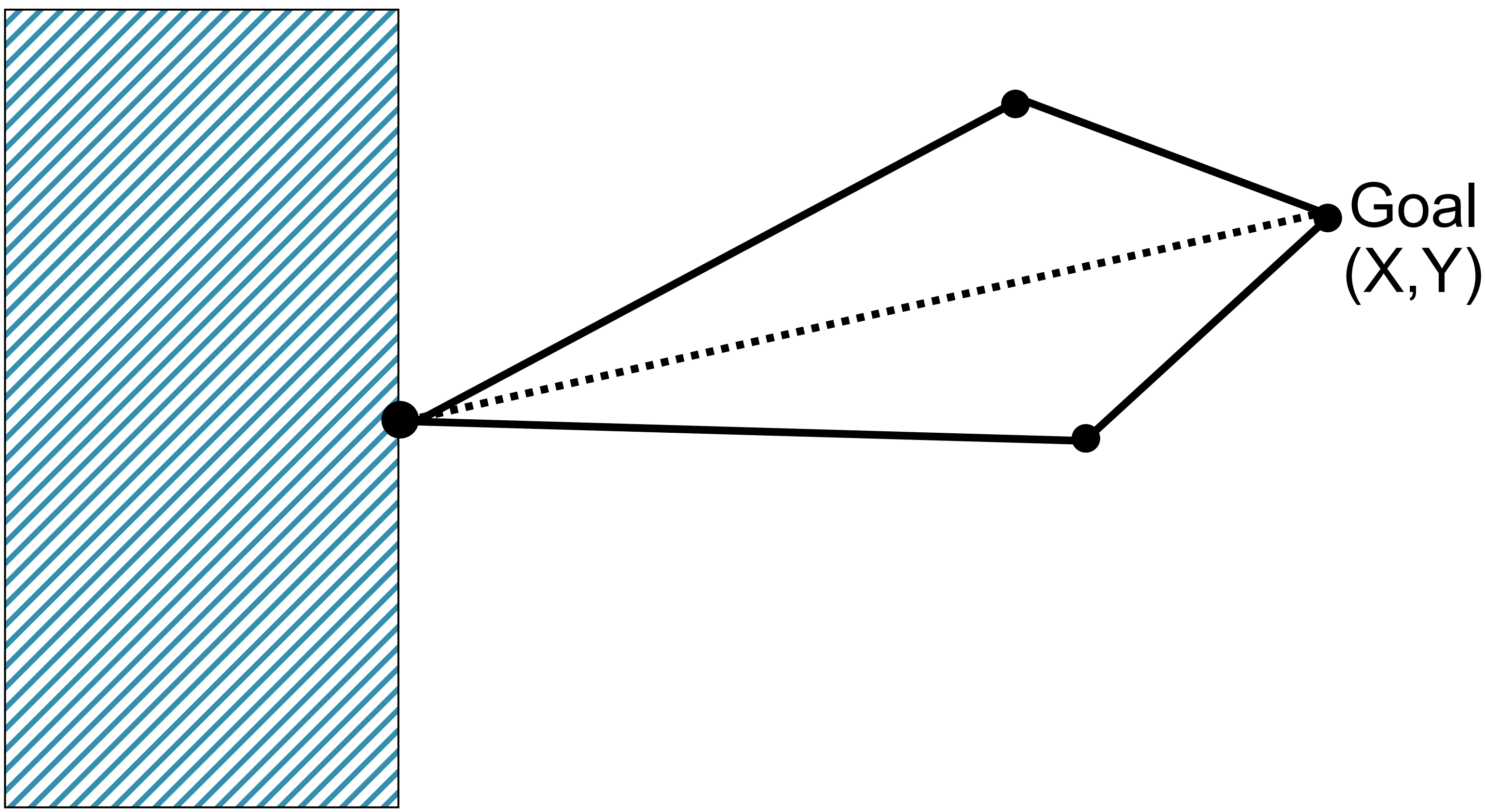




# Analytic Inverse Kinematics

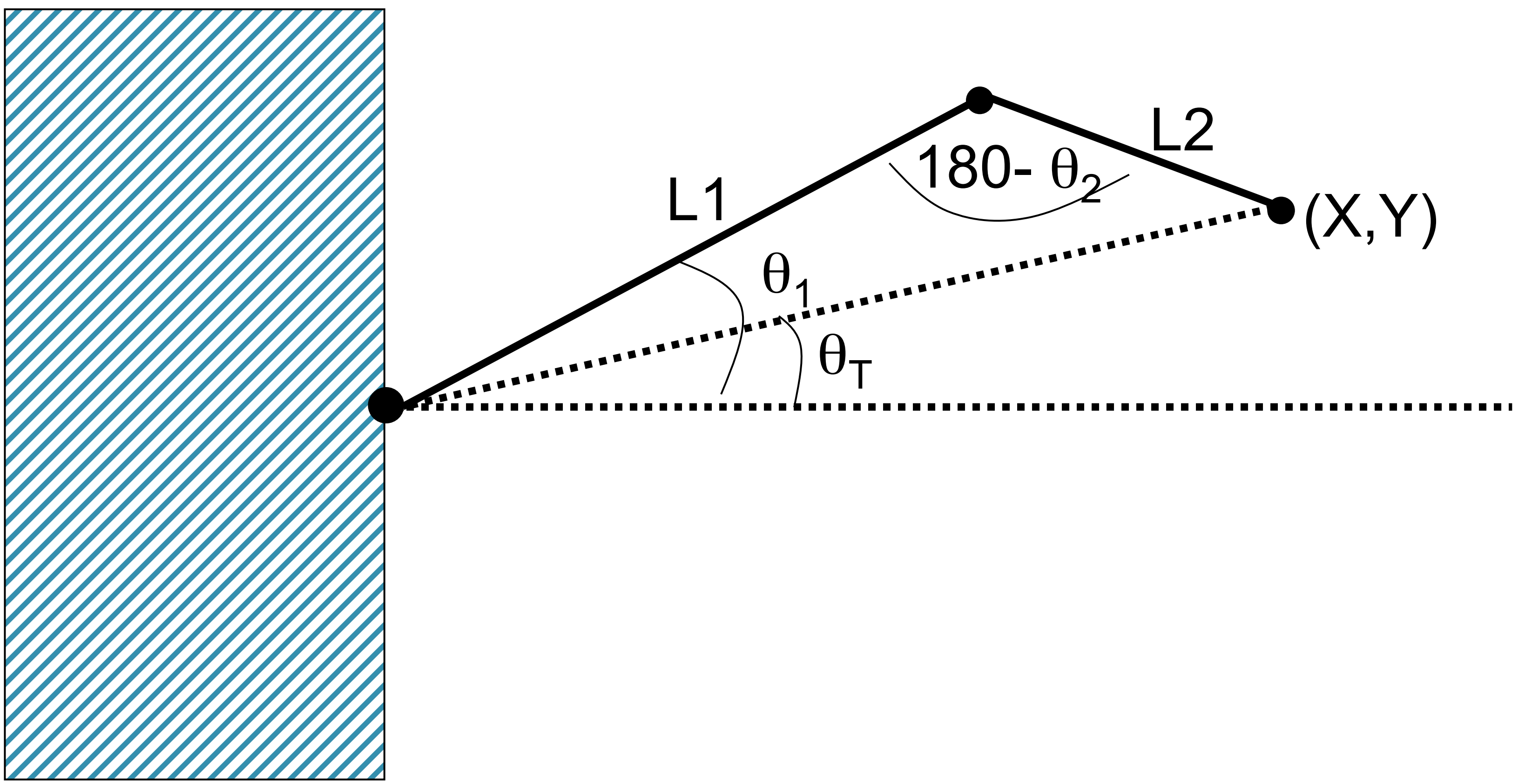


Multiple solutions



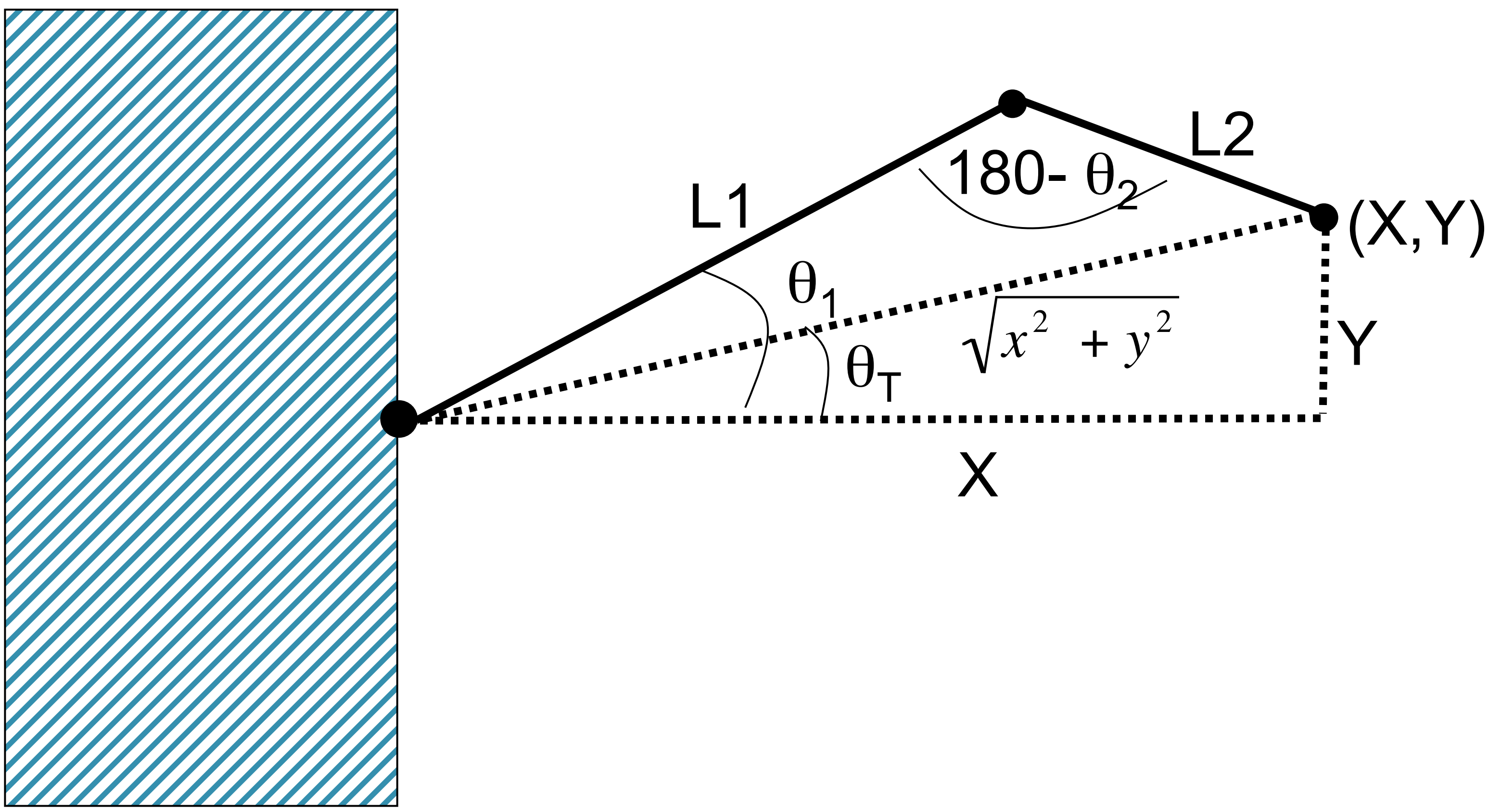
# Analytic Inverse Kinematics





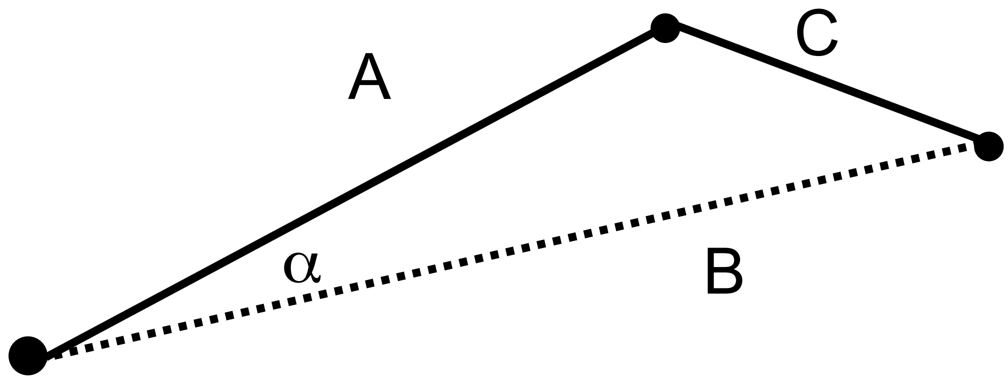
# Analytic Inverse Kinematics





# Law of Cosines

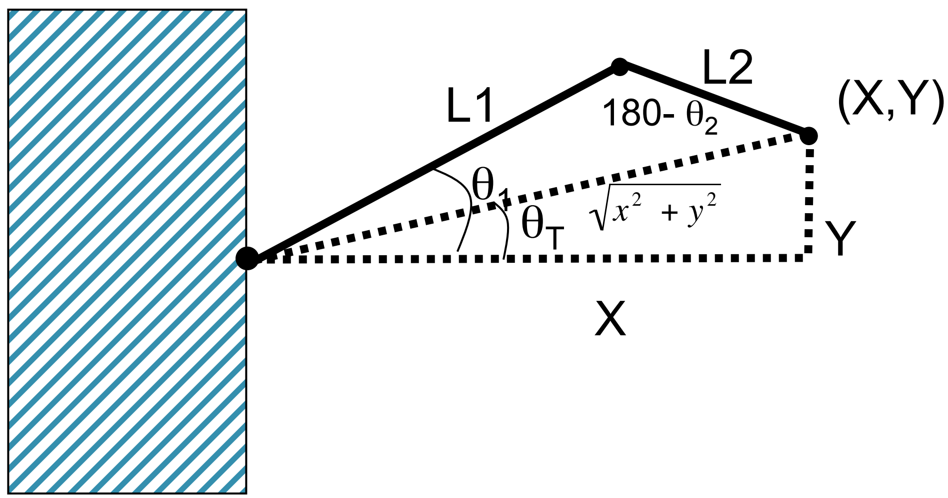





$$\cos(\alpha) = \frac{A^2 + B^2 - C^2}{2AB}$$

# Analytic Inverse Kinematics






$$\cos(\theta_T) = \frac{X}{\sqrt{X^2 + Y^2}}$$

$$\theta_T = \cos^{-1}\left(\frac{X}{\sqrt{X^2 + Y^2}}\right)$$

$$\cos(\theta_1 - \theta_T) = \frac{L_1^{\ 2} + X^2 + Y^2 - L_2^{\ 2}}{2L_1\sqrt{X^2 + Y^2}}$$

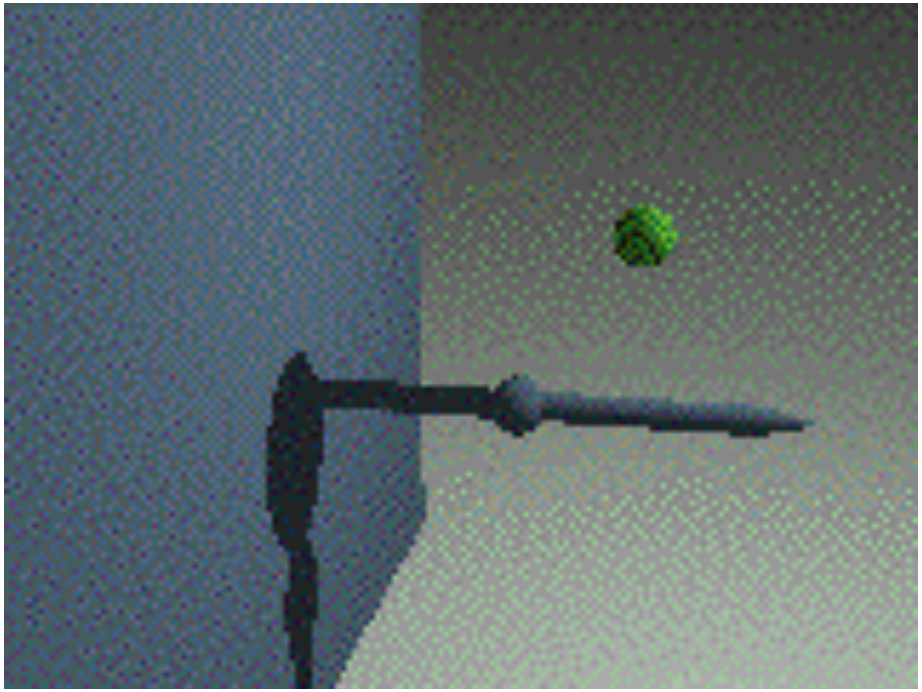
$$\cos(180 - \theta_2) = \frac{L_1^{\ 2} + L_2^{\ 2} - \left(X^2 + Y^2\right)}{2L_1L_2}$$

$$\theta_1 = \cos^{-1}\left(\frac{L_1^{\ 2} + X^2 + Y^2 - L_2^{\ 2}}{2L_1\sqrt{X^2 + Y^2}}\right) + \theta_T$$

$$\theta_2 = 180 - \cos^{-1}\left(\frac{L_1^{\ 2} + L_2^{\ 2} - \left(X^2 + Y^2\right)}{2L_1L_2}\right)$$

# Analytic Inverse Kinematics





# Iterative Inverse Kinematics



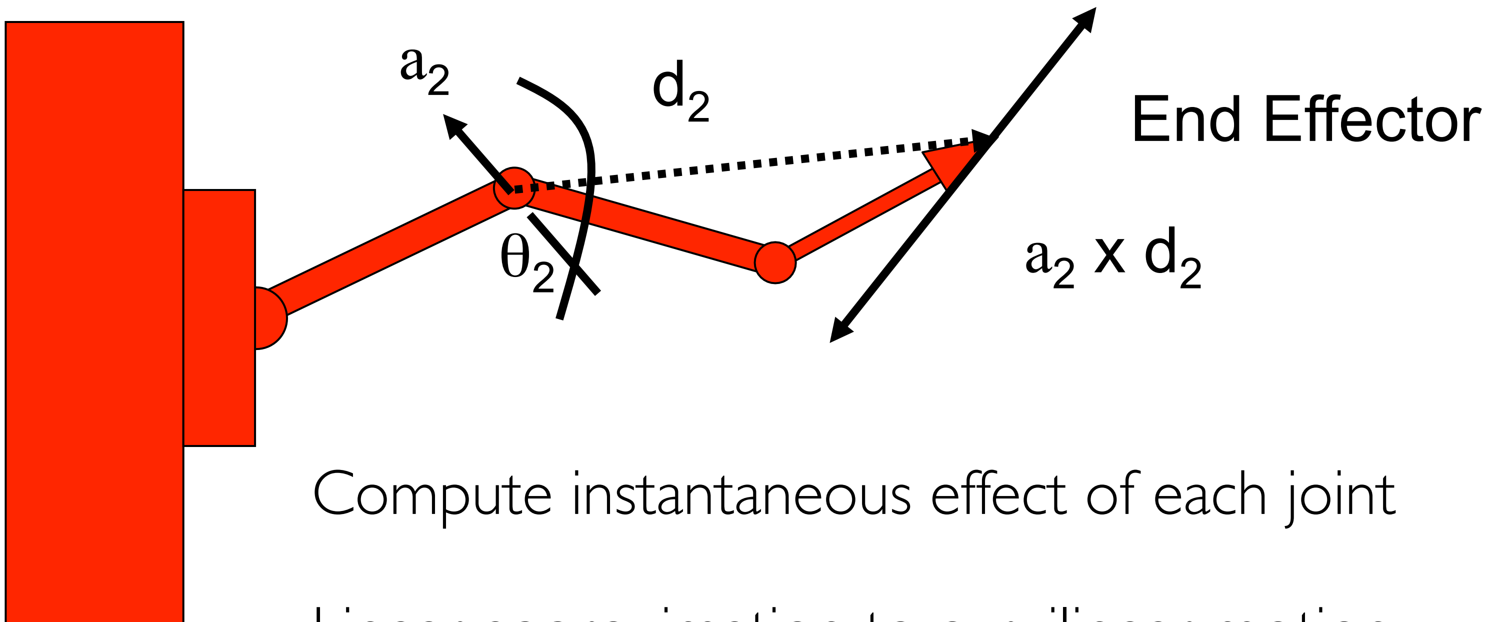
When linkage is too complex for analytic methods

At each time step, determine changes to joint angles that take the end effector toward goal position and orientation

Need to recompute at each time step

# Inverse Jacobian Method





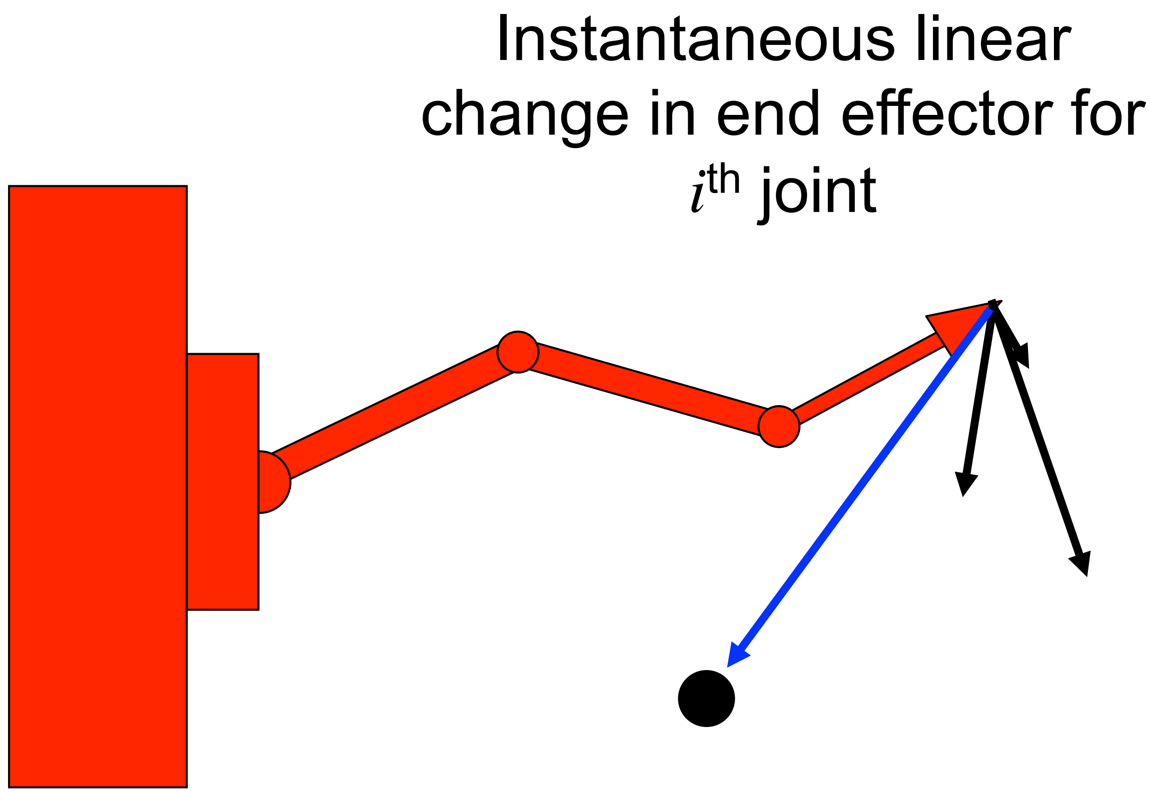

Compute instantaneous effect of each joint

Linear approximation to curvilinear motion

Find linear combination to take end effector **towards** goal position

# Inverse Jacobian Method





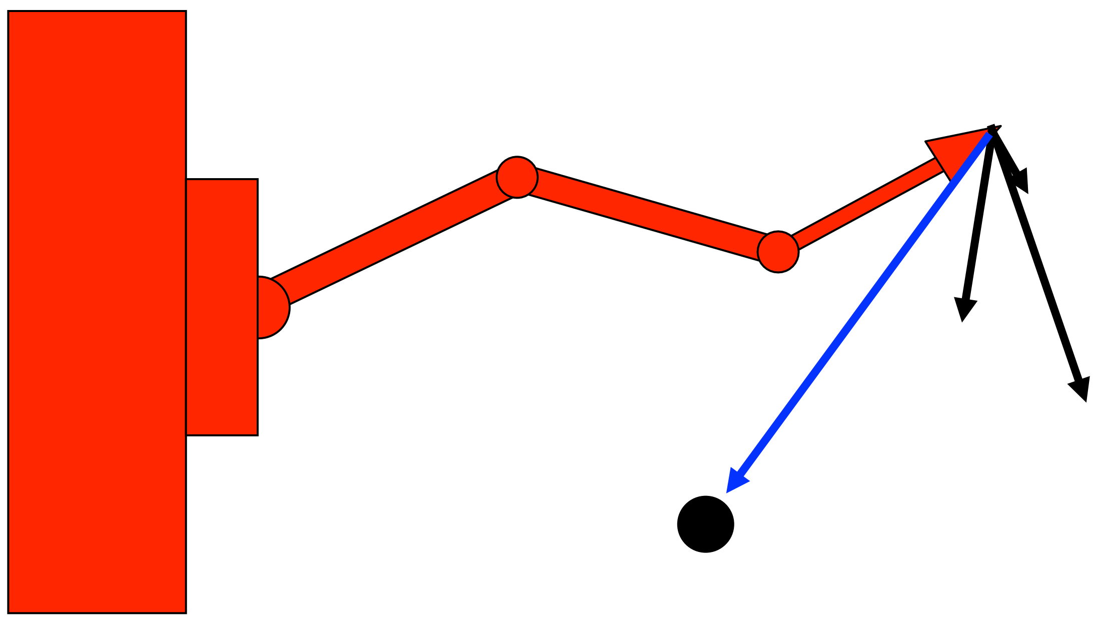
# Inverse Jacobian Method



What is the change in orientation of end effector induced by joint i that has axis of rotation $a_i$ and position $J_i$?

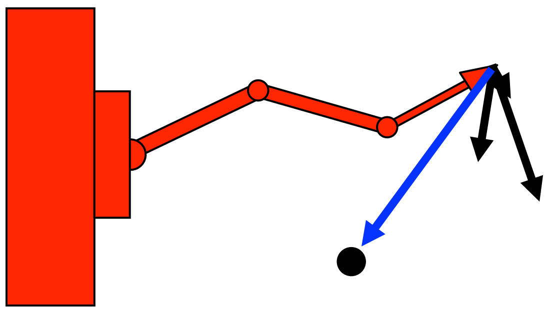
Angular velocity

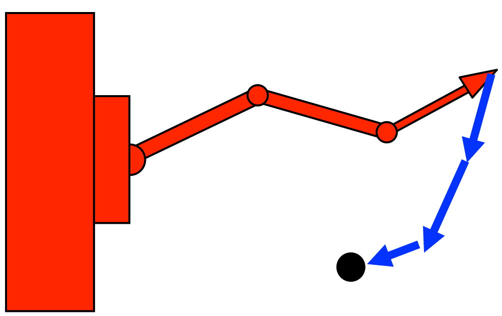
$$\dot{a}_i = \omega_i$$



# Inverse Jacobian Method







Solution only valid for an instantaneous step

Angular affect is really curved, not straight line

Once a step is taken, need to recompute solution

# Inverse Jacobian: Math



Set up equations

$y_i$: state variable
$x_i$ : system parameter
$f_i$ : relate system parameters to state variable

$$y_1 = f_1(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_2 = f_2(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_3 = f_3(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_4 = f_4(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_5 = f_5(x_1, x_2, x_3, x_4, x_5, x_6)$$

$$y_6 = f_6(x_1, x_2, x_3, x_4, x_5, x_6)$$

# Inverse Jacobian: Math



$$y_1 = f_1(x_1, x_2, x_3, x_4, x_5, x_6)$$
$$y_2 = f_2(x_1, x_2, x_3, x_4, x_5, x_6)$$
$$y_3 = f_3(x_1, x_2, x_3, x_4, x_5, x_6)$$
$$y_4 = f_4(x_1, x_2, x_3, x_4, x_5, x_6)$$
$$y_5 = f_5(x_1, x_2, x_3, x_4, x_5, x_6)$$
$$y_6 = f_6(x_1, x_2, x_3, x_4, x_5, x_6)$$

Matrix Form → $Y = F(X)$

# Inverse Jacobian: Math



$$y_i = f_1(x_1, x_2, x_3, x_4, x_5, x_6)$$

Use chain rule to differentiate equations to relate changes in system parameters to changes in state variables

$$dy_i = \frac{\partial f_i}{\partial x_1} dx_1 + \frac{\partial f_i}{\partial x_2} dx_2 + \frac{\partial f_i}{\partial x_3} dx_3 + \frac{\partial f_i}{\partial x_4} dx_4 + \frac{\partial f_i}{\partial x_5} dx_5 + \frac{\partial f_i}{\partial x_6} dx_6$$

# Inverse Jacobian: Math



$$\partial y_i = \frac{\partial f_i}{\partial x_1} dx_1 + \frac{\partial f_i}{\partial x_2} dx_2 + \frac{\partial f_i}{\partial x_3} dx_3 + \frac{\partial f_i}{\partial x_4} dx_4 + \frac{\partial f_i}{\partial x_5} dx_5 + \frac{\partial f_i}{\partial x_6} dx_6$$

Matrix Form

$$Y = F(X)$$

$$dY = \frac{\partial F}{\partial X} dX$$

# Inverse Jacobian: Math



$$dY = \frac{\partial F}{\partial X} dX$$

Change in position (and orientation) of end effector

Change in joint angles

Linear approximation that relates change in joint angle to change in end effector position (and orientation)

# Inverse Jacobian: Math



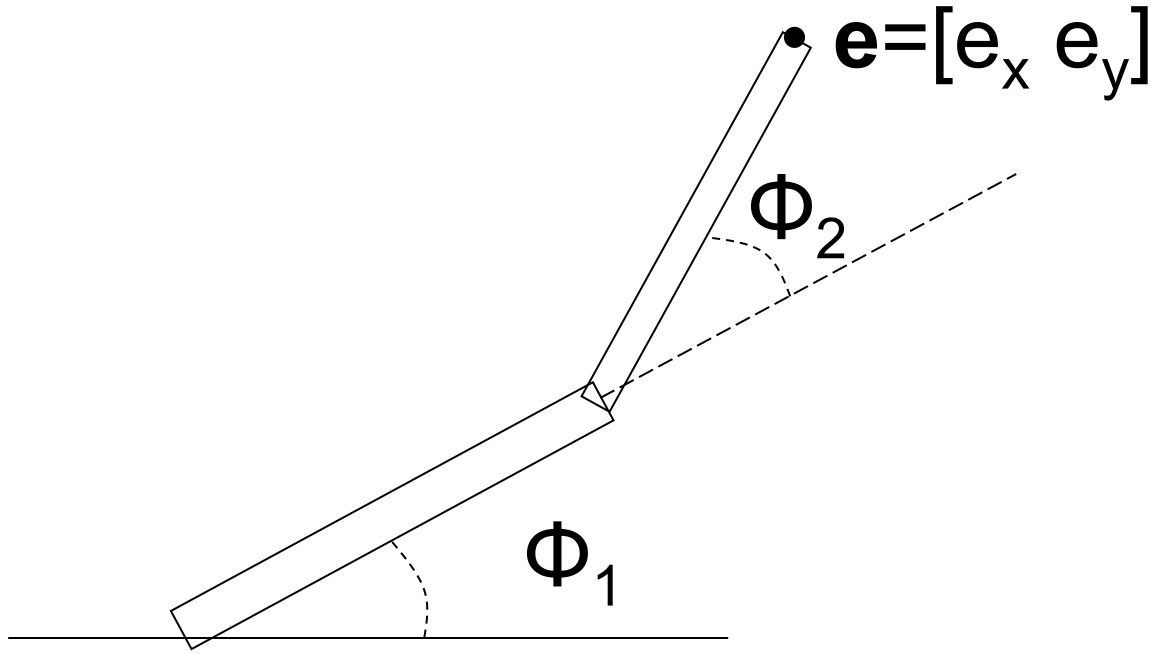
$$dY = \frac{\partial F}{\partial X} dX$$

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \frac{\partial p_x}{\partial \theta_1} & \frac{\partial p_x}{\partial \theta_2} & \cdots & \frac{\partial p_x}{\partial \theta_n} \\ \frac{\partial p_y}{\partial \theta_1} & \frac{\partial p_y}{\partial \theta_2} & \cdots & \frac{\partial p_y}{\partial \theta_n} \\ \frac{\partial p_z}{\partial \theta_1} & \cdots & \cdots & \cdots \\ \frac{\partial a_x}{\partial \theta_1} & \cdots & \cdots & \cdots \\ \frac{\partial a_y}{\partial \theta_1} & \cdots & \cdots & \cdots \\ \frac{\partial a_z}{\partial \theta_1} & \cdots & \cdots & \cdots \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \cdots \\ \dot{\theta}_n \end{bmatrix}$$

# Jacobians



- Let's say we have a simple 2D robot arm with two 1-DOF rotational joints:



# Jacobians



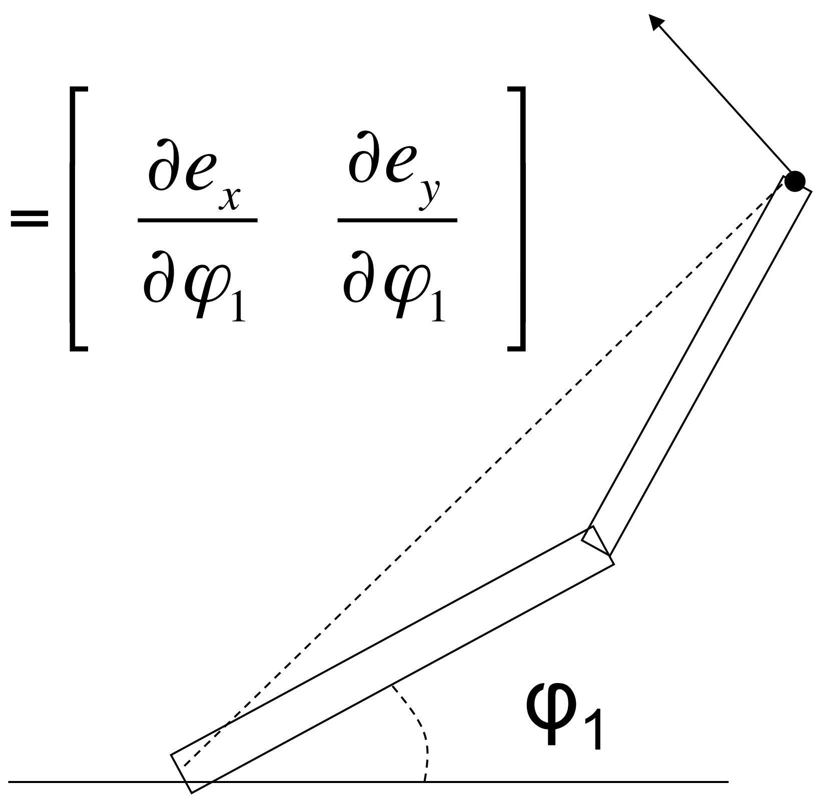
- The Jacobian matrix J(**e**, **Φ**) shows how each component of **e** varies wrt. each joint angle

$$J(\mathbf{e},\mathbf{\Phi}) = \begin{bmatrix} \dfrac{\partial e_x}{\partial \phi_1} & \dfrac{\partial e_x}{\partial \phi_2} \\ \dfrac{\partial e_y}{\partial \phi_1} & \dfrac{\partial e_y}{\partial \phi_2} \end{bmatrix}$$

# Jacobians

- Consider what would happen if we increased $\phi_1$ by a small amount. What would happen to **e** ?

$$\frac{\partial \mathbf{e}}{\partial \varphi_1} = \begin{bmatrix} \dfrac{\partial e_x}{\partial \varphi_1} & \dfrac{\partial e_y}{\partial \varphi_1} \end{bmatrix}$$

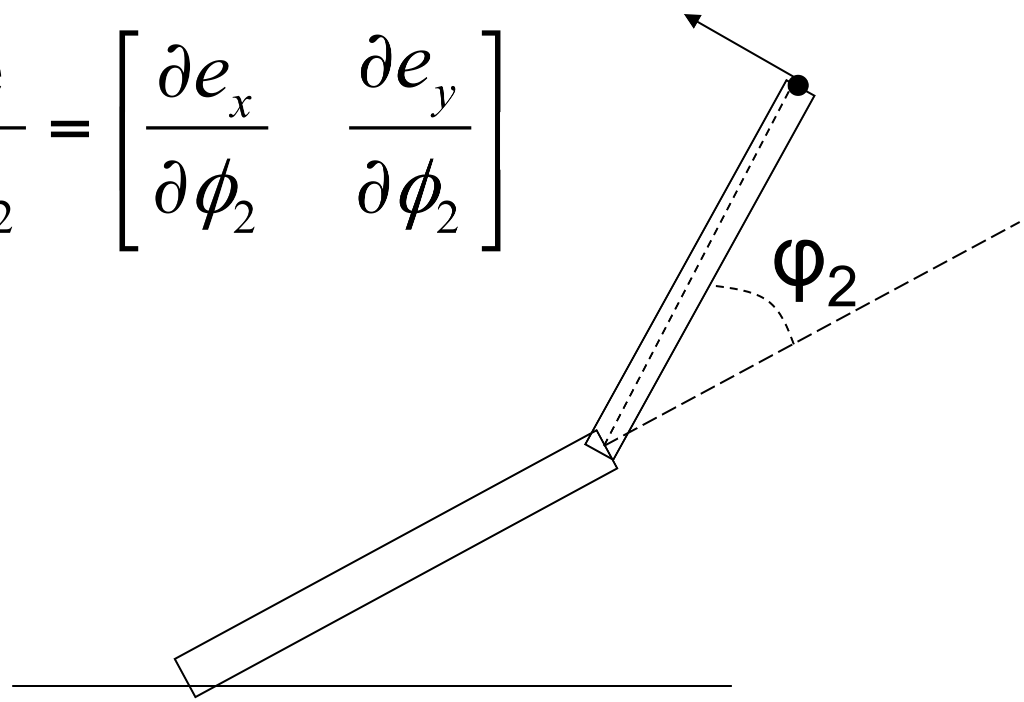


# Jacobians



- What if we increased $\phi_2$ by a small amount?

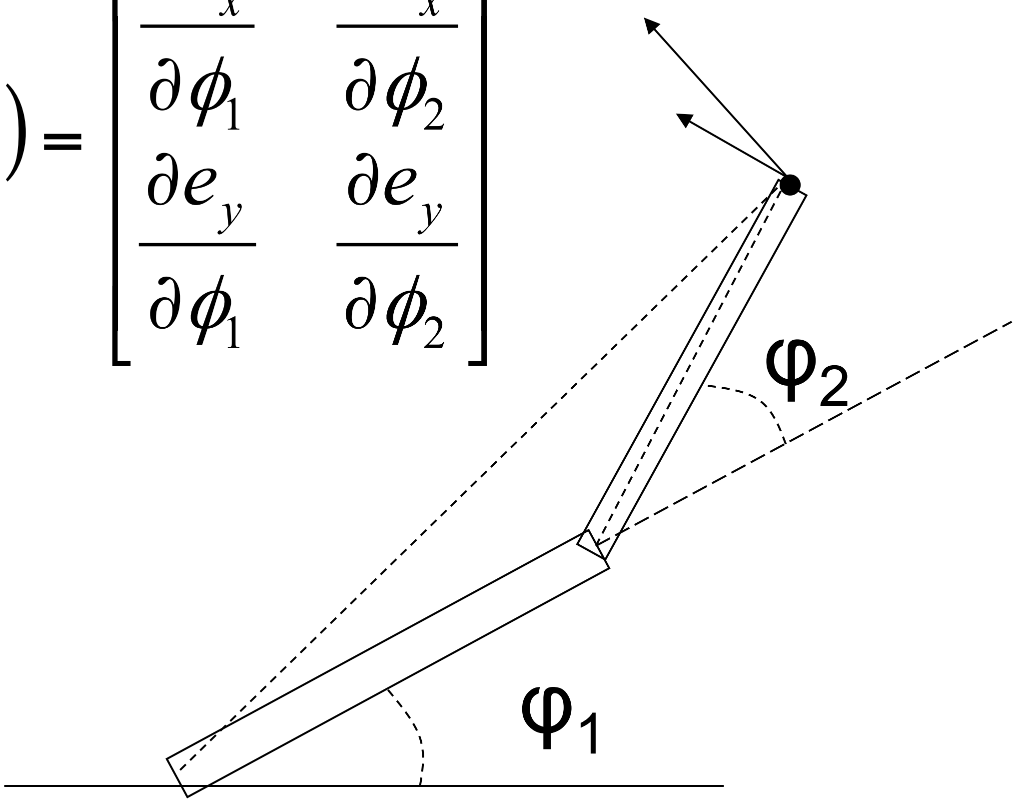
$$\frac{\partial \mathbf{e}}{\partial \phi_2} = \begin{bmatrix} \frac{\partial e_x}{\partial \phi_2} & \frac{\partial e_y}{\partial \phi_2} \end{bmatrix}$$



# Jacobian for a 2D Robot Arm



$$J(\mathbf{e},\mathbf{\Phi}) = \begin{bmatrix} \frac{\partial e_x}{\partial \phi_1} & \frac{\partial e_x}{\partial \phi_2} \\ \frac{\partial e_y}{\partial \phi_1} & \frac{\partial e_y}{\partial \phi_2} \end{bmatrix}$$



# Other Numeric IK



Jacobian transpose

Alternate Jacobian – use goal position

HAL – human arm linkage

Damped Least Squares

CCD