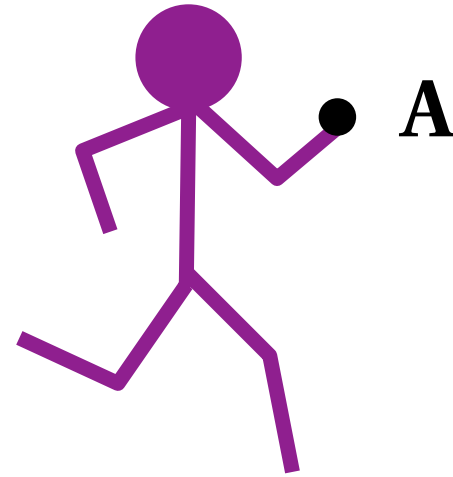
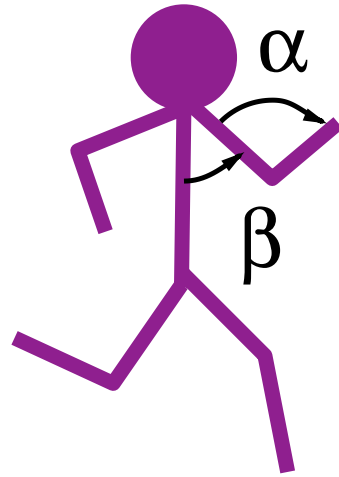


# Kinematics -- the study of motion without regard to the forces that cause it.



**Forward:**  $A = f(\alpha, \beta)$

draw graphics

**Inverse:**  $\alpha, \beta = f^{-1}(A)$

specify fewer degrees of freedom

more intuitive control of dof

contact with the environment

calculate desired joint angles for control

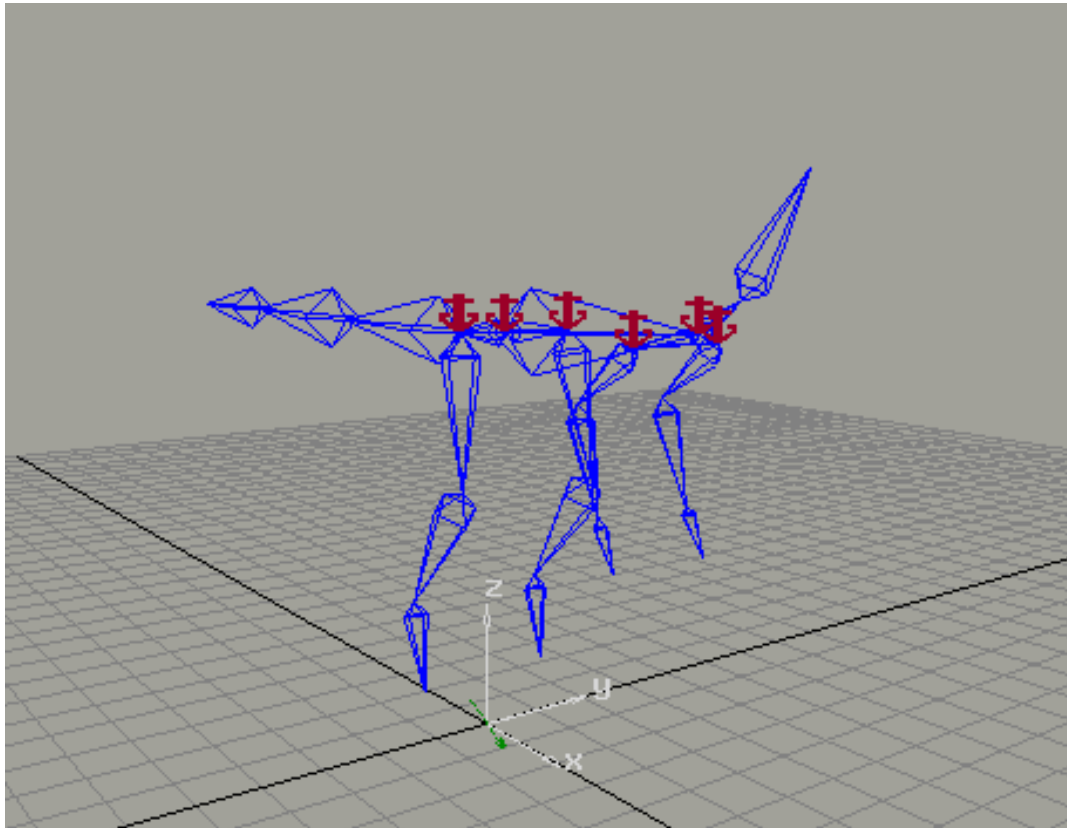
# User Control of Kinematic Characters

## Joint Space

position all joints--fine level of control

## Cartesian Space

specify environmental interactions easily  
most dof computed automatically

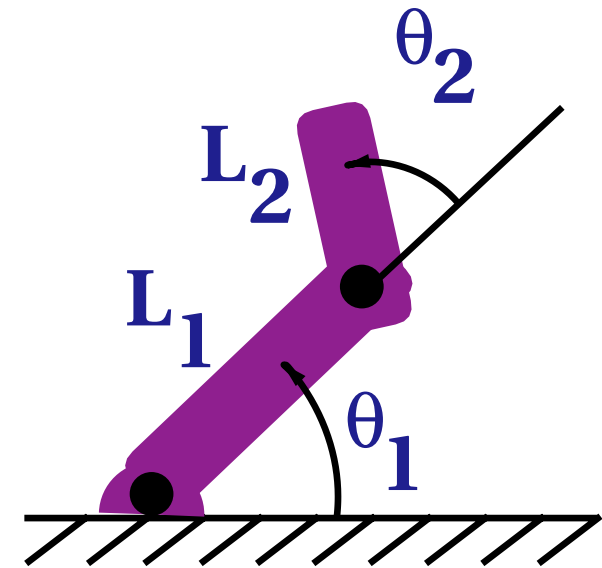


# Forward Kinematics

$$x = L_1 \cos \theta_1 + L_2 \cos (\theta_1 + \theta_2)$$

$$y = L_1 \sin \theta_1 + L_2 \sin (\theta_1 + \theta_2)$$

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



$$\begin{bmatrix} \phantom{x} \\ \phantom{y} \\ \phantom{z} \\ 1 \end{bmatrix} = \begin{bmatrix} \text{rot } \theta_1 \\ \text{trans } L_1 \end{bmatrix} \begin{bmatrix} \text{rot } \theta_2 \\ \text{trans } L_2 \end{bmatrix}$$

# Why IK?

**pick up an object or place feet on the ground  
hard to do with forward kinematics**

**maybe allow animator to set fewer parameters or  
at least get a good first approximation**

## What do we need?

**skeleton with 1,2,3 dof joints**

**solve from root position to end effector position**

**arbitrary position constraints**

**direction constraints**

**joint limits**

**techniques for handling underconstrained system**

**adding constraints**

**heuristics to push solution into right part of space**

**optimization (energy minimization)**

# Inverse Kinematics

**balance -- keep center-of-mass over support polygon**

**control -- position vaulter's hands on line between shoulder and vault**

**control -- compute knee angles that will give the runner the right leg length**

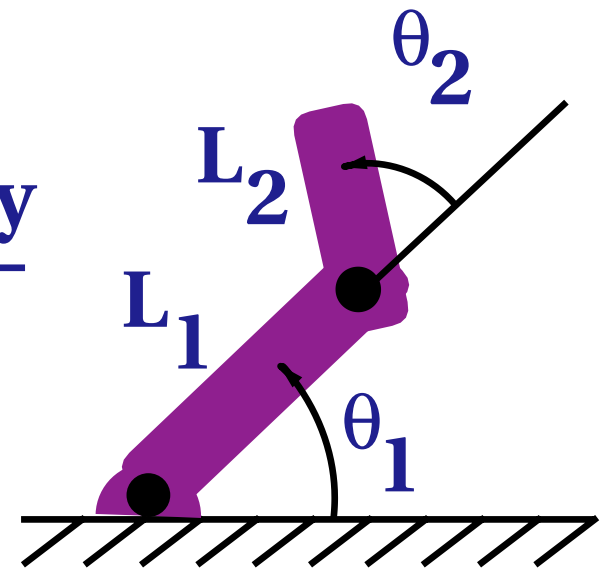


# Inverse Kinematics

$$\theta_2 = \frac{\cos^{-1} (x^2 + y^2 - L_1^2 - L_2^2)}{2 L_1 L_2}$$

$$\theta_1 = \frac{-(L_2 \sin \theta_2)x + (L_1 + L_2 \cos \theta_2)y}{(L_2 \sin \theta_2)y + (L_1 + L_2 \cos \theta_2)x}$$

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



# What makes IK hard? equations hard to solve

$$\begin{bmatrix} x_x & y_x & z_x & p_x \\ x_y & y_y & z_y & p_y \\ x_z & y_z & z_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x & d_x \\ a_y & b_y & c_y & d_y \\ a_z & b_z & c_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

a,b,c,d are functions of  $(\theta_1, \dots, \theta_6)$

x,y,z,p are desired orientation, position of end effector

12 equations, 6 unknowns  $(\theta_1, \dots, \theta_6)$

only 3 of the 9 rotation terms are independent

non-linear, transcendental equations

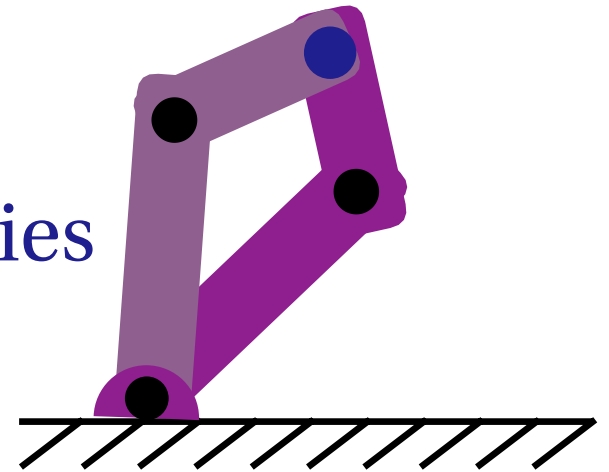
# What makes IK hard? -- Redundancy

a subspace  $\{\theta_x\}$  defined by

$$\theta(\theta_1, \dots, \theta_n) \in \theta_x \text{ if } f(\theta) = X$$

Add constraints to reduce redundancies

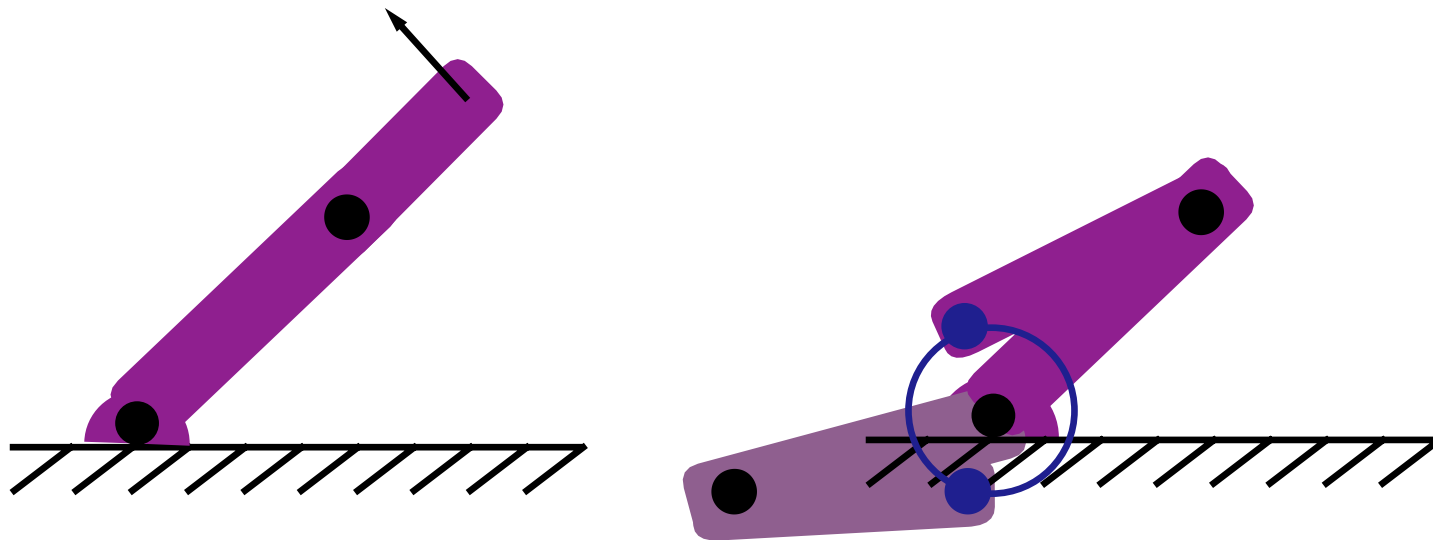
Choose solution that is  
"closest" to current configuration  
move outermost links the most  
energy minimization  
minimum time



# What makes IK hard? -- singularities

ill-conditioned near singularities

high state space velocities for  
low cartesian velocities



# What makes IK hard?

goal of "natural looking" motion

minimum jerk

equilibrium point trajectories

# The Jacobian

$f(\theta) = x$        $x$  is of dimension  $n$  (generally 6)  
                          $\theta$  is of dimension  $m$  (# of dof)

Jacobian is the  $n \times m$  matrix relating differential changes of  $\theta$  ( $d\theta$ ) to differential changes of  $x$  ( $dx$ )

$J(\theta) d\theta = dx$  where the  $ij$ th element of  $J$  is

$$J_{ij} = \frac{\delta f_i}{\delta x_j}$$

Jacobian maps velocities in state space to velocities in cartesian space

# Solutions

no solution (outside workspace, too few dof)

multiple solutions (redundancy)

single solution

# Methods

closed form

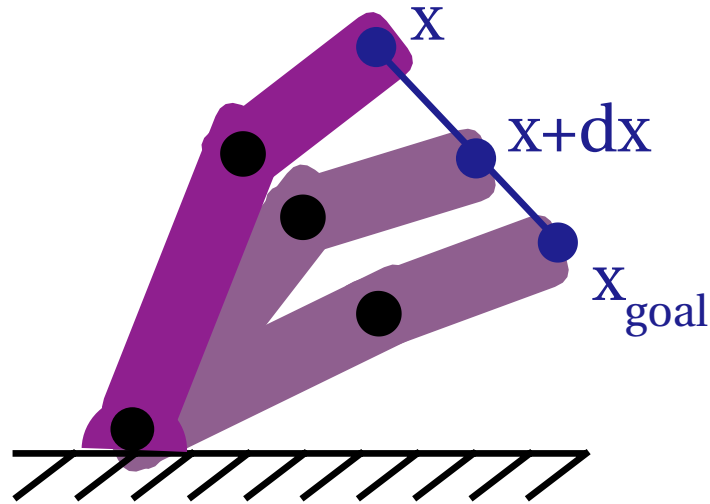
iterative

# IK and the Jacobian

$$\theta = f^{-1}(x)$$

$$dx = J d\theta$$

$$d\theta = J^{-1} dx$$



$$\theta_{k+1} = \theta_k + \Delta t J^{-1} dx$$

linearize about  $\theta_k$

# Inverting the Jacobian

**J is  $n \times m$  -- not square in general  
compute pseudo-inverse**

**Singularities cause the rank of the  
Jacobian to change**

**Damped Least Squares:  
find solution that minimizes**

$$\|J - dx\|^2 + \lambda^2 \|d\theta\|^2$$

tracking error + joint velocities

# Non-linear Optimization

Zhao and Badler, TOG 1994

solution is a (local) minima of some non-linear function

objective function

constraints

non-linear optimization routine

# Objective Function

position and orientation of end effector

$$P(\mathbf{x}) = (\mathbf{p} - \mathbf{x})^2$$

$$\nabla_{\mathbf{x}} P(\mathbf{x}) = 2(\mathbf{x} - \mathbf{p})$$

or just position, or just orientation,  
or aiming at

# Formulation

minimize  $G(\theta)$   
subject to  $a_i \theta = b_i$   
 $a_i \theta < b_i$

# Solution

$G(\theta)$  and  $\nabla G(\theta)$

use a standard numerical technique to solve