COL781: Computer Graphics

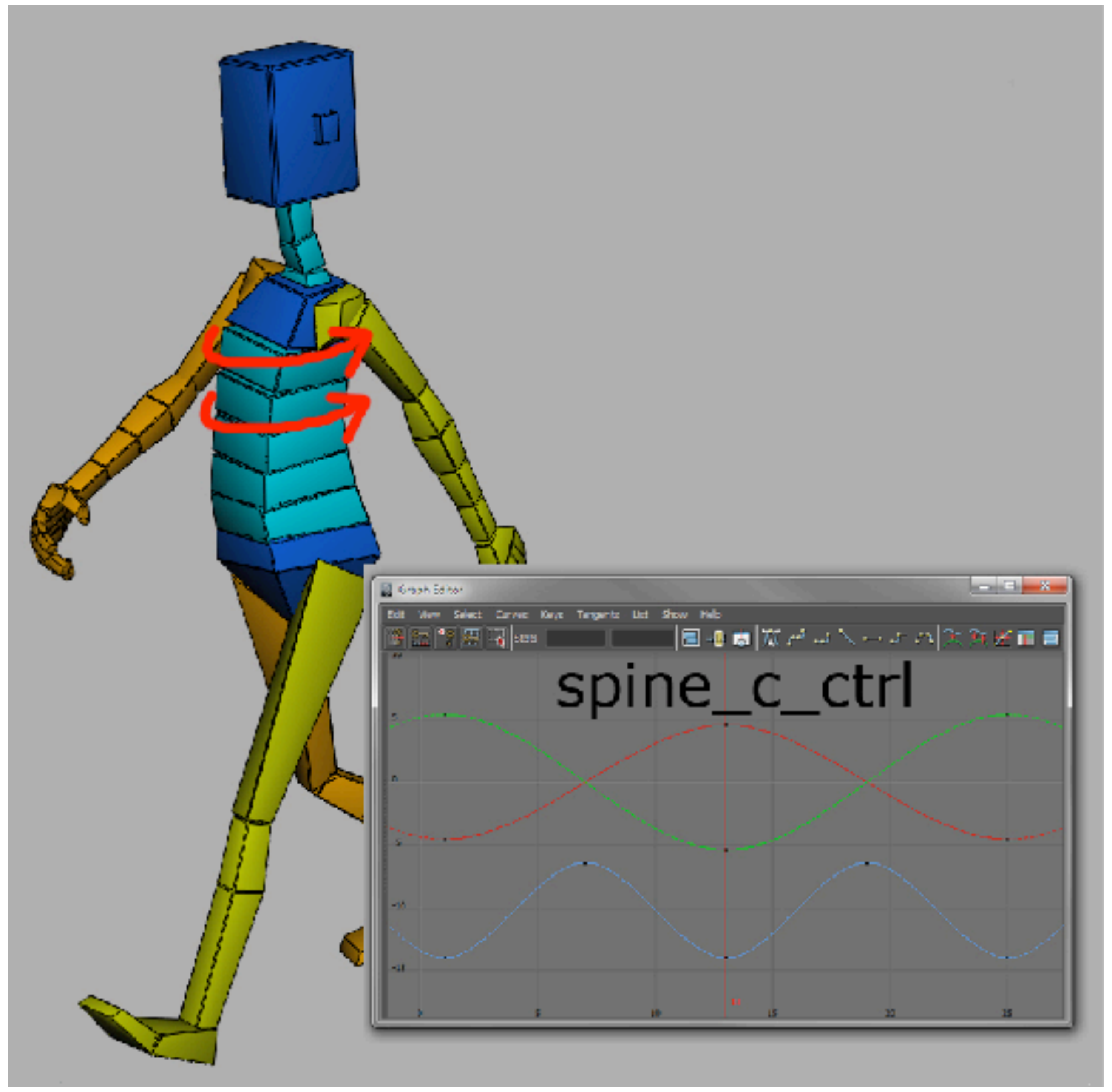
29. Inverse Kinematics

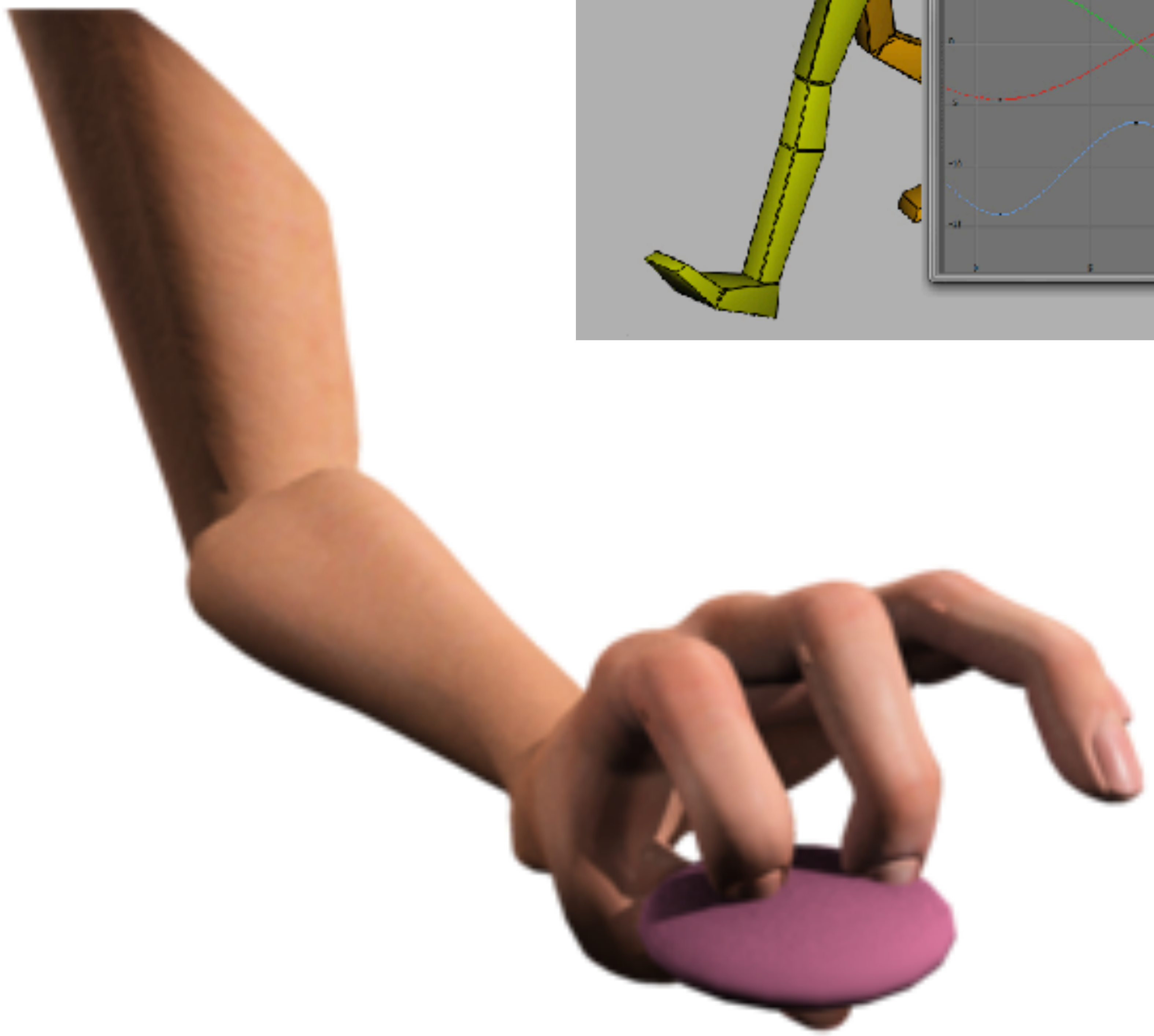
Aristidou et al. 2011

Recap: Skeletal animation

The vector of generalized coordinates **q** (containing joint angles etc.) determines the character's pose.

- We know how to do forward kinematics: find bone transformations from **q**
- Inverse kinematics: find **q** to achieve desired position/rotation of end point(s)



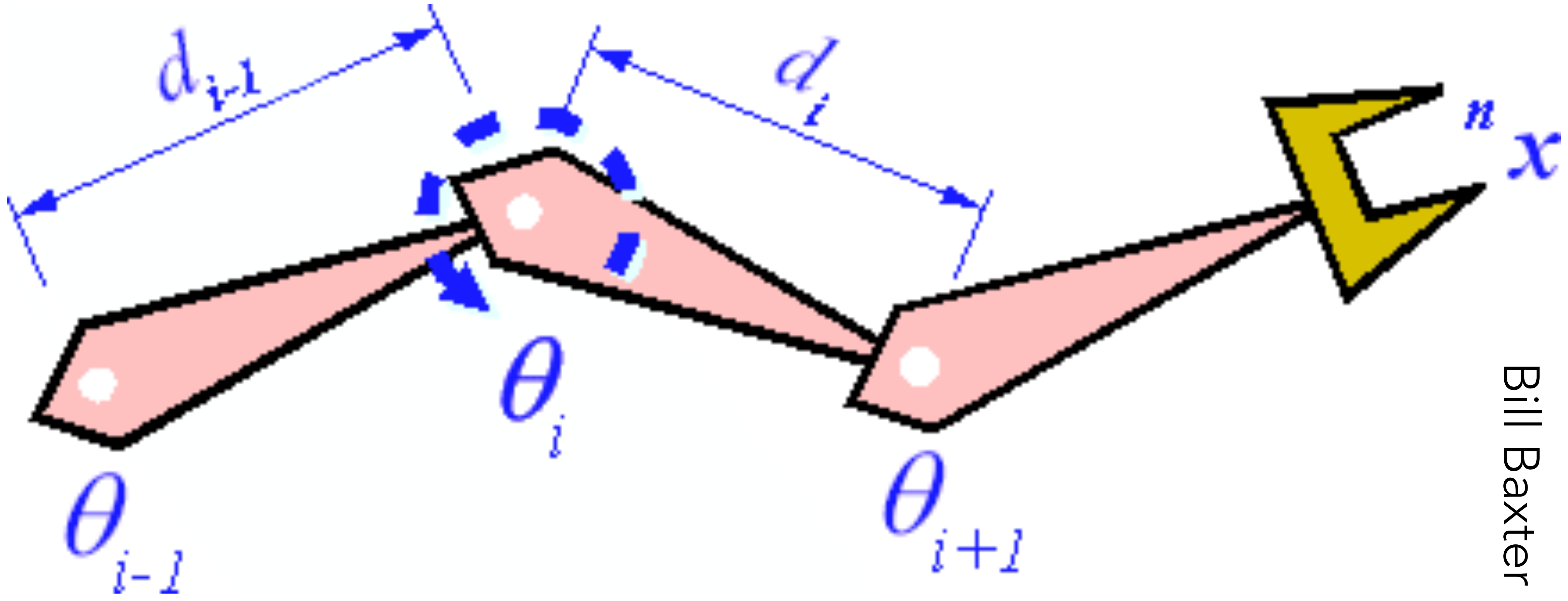


Inverse kinematics

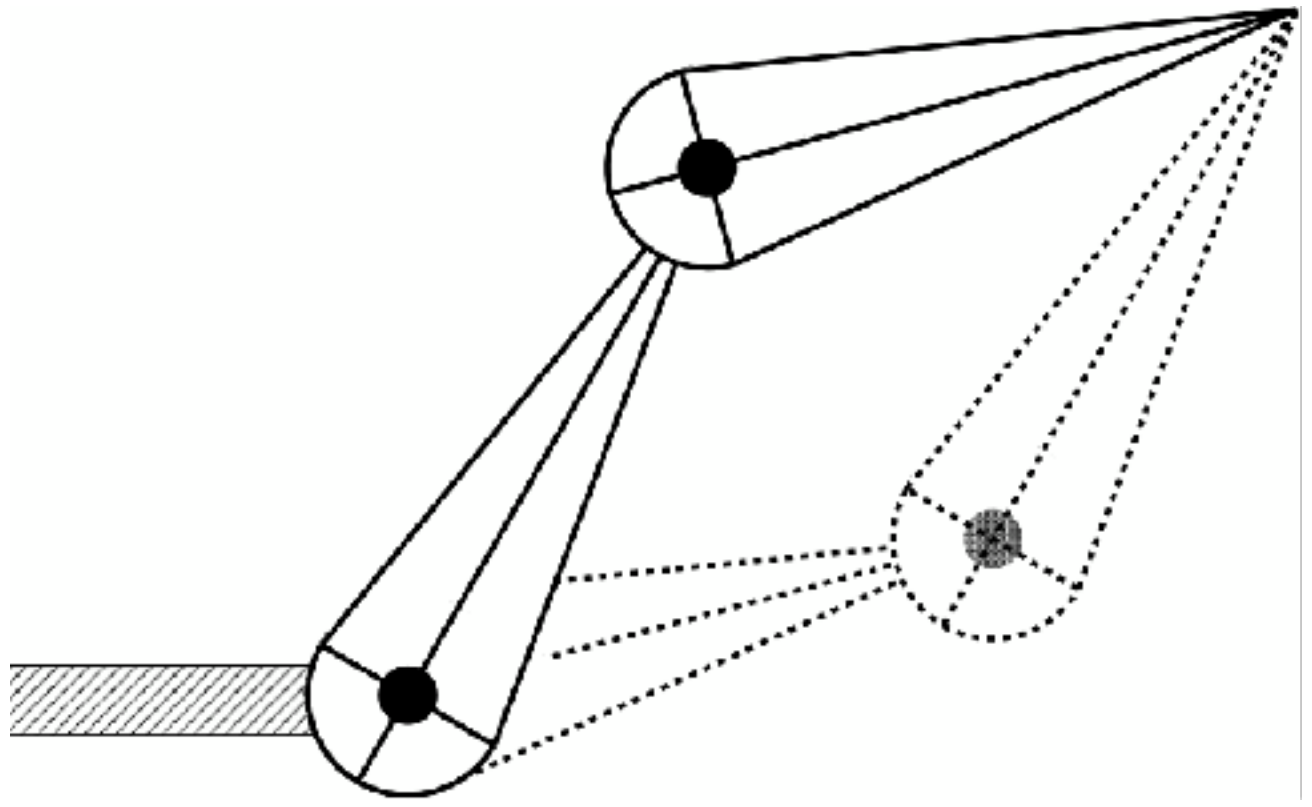
Position of target point (**end effector**) depends nonlinearly on joint angles

$$\mathbf{x} = f(\mathbf{q})$$

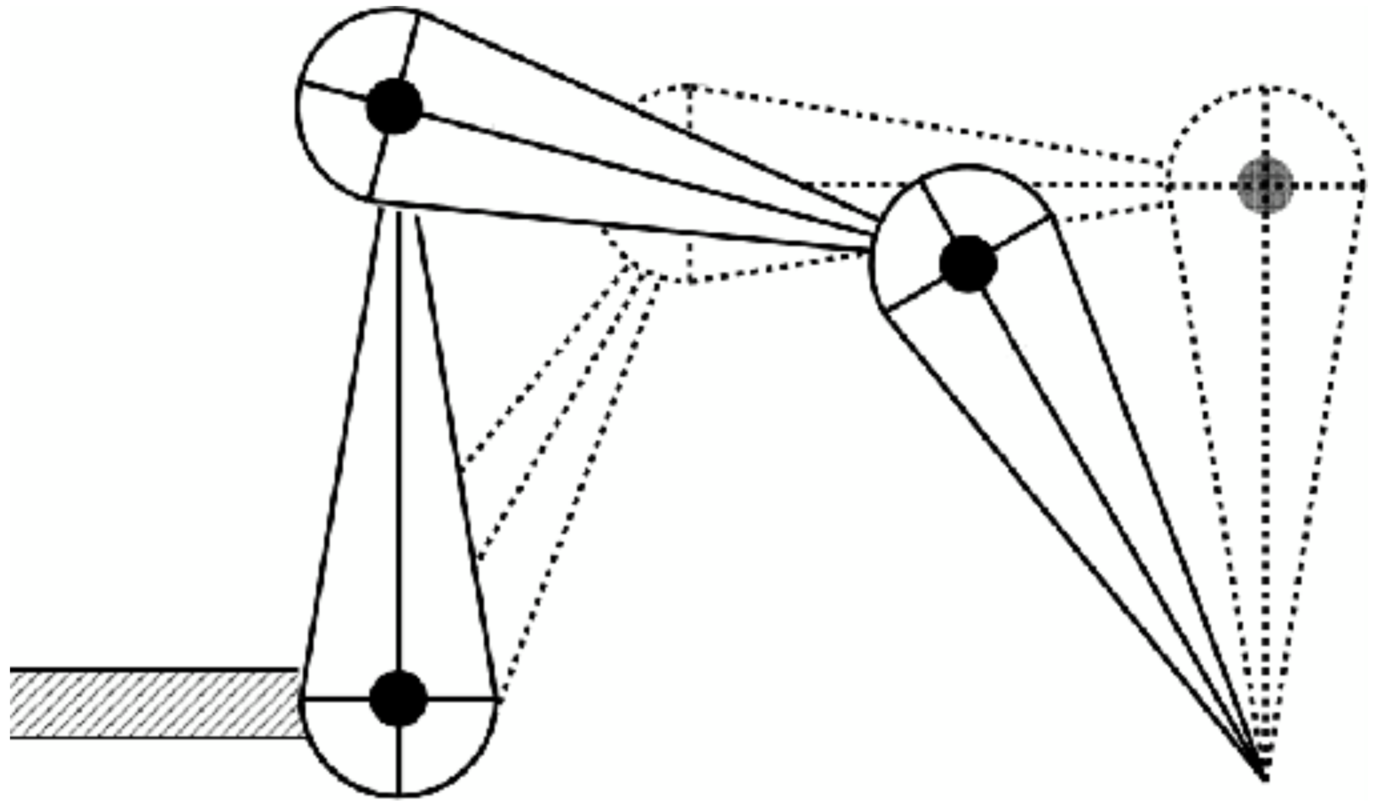
Given desired **x**, how to compute **q**?



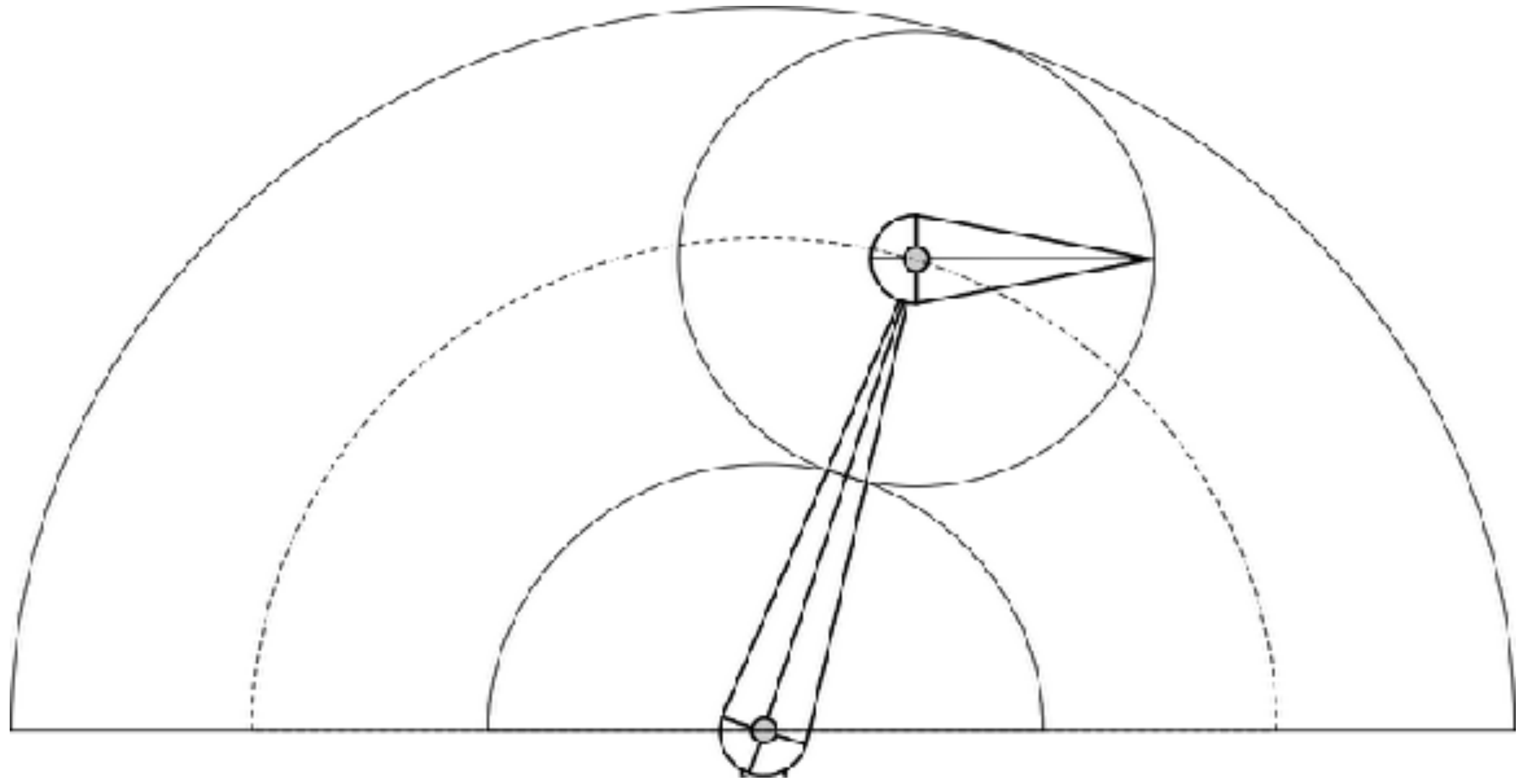
Bill Baxter



Multiple solutions



Infinitely many solutions



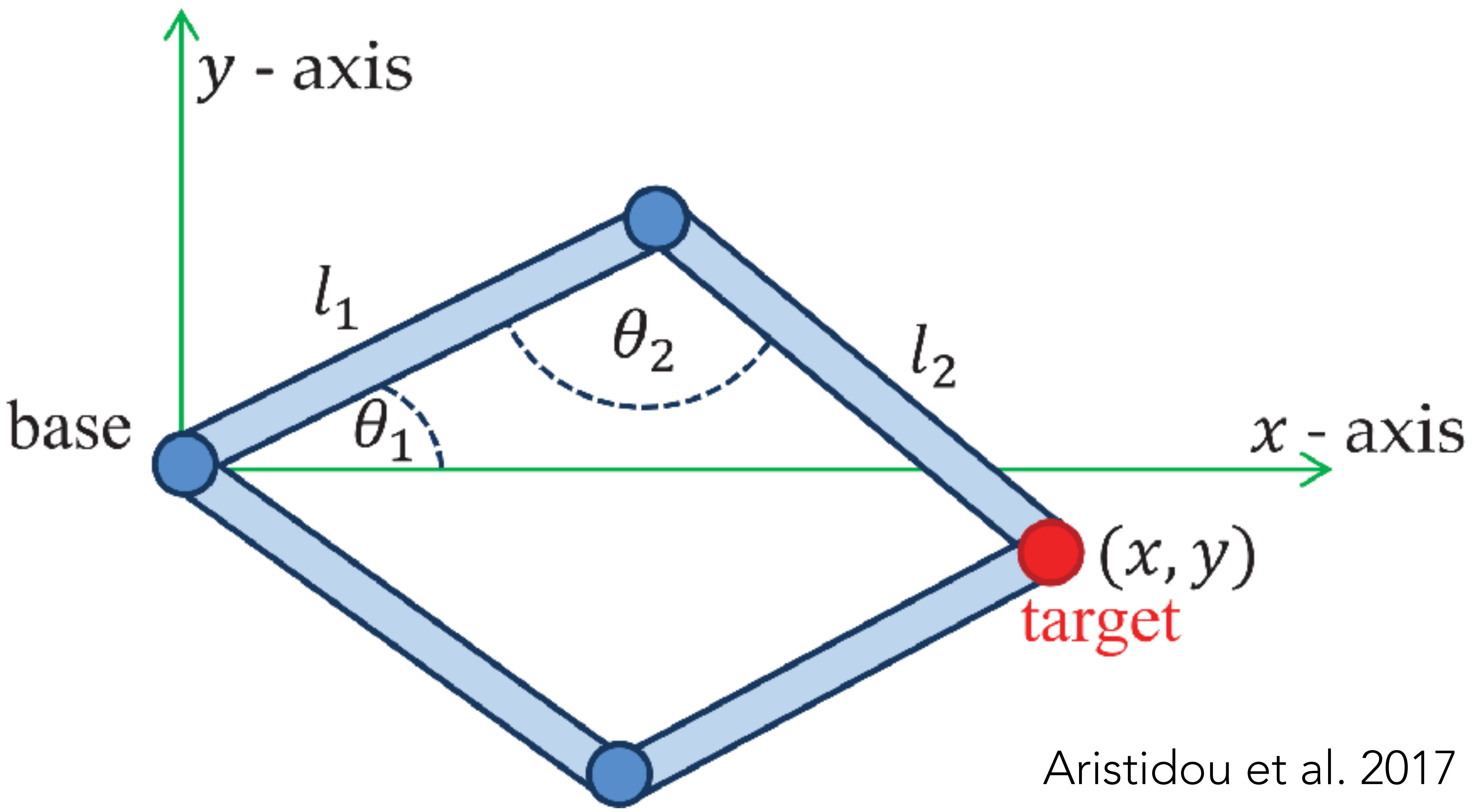
No solution

James O'Brien

Closed-form solution for 2 segments in 2D:

$$\theta_1 = \cos^{-1}\left(\frac{l_1^2 + x^2 + y^2 - l_2^2}{2l_1\sqrt{x^2 + y^2}}\right)$$

$$\theta_2 = \cos^{-1}\left(\frac{l_1^2 + l_2^2 - x^2 - y^2}{2l_1 l_2}\right)$$



Aristidou et al. 2017

Does not generalize!

Solving nonlinear equations

Warm-up: 1 equation in 1 variable

$f : \mathbb{R} \to \mathbb{R}$ is some nonlinear function. We want to find x such that $f(x) = 0$

- Usually no analytical solution (no formula for f^{-1})
- Assume f is smooth: we can evaluate $f(x)$, $f'(x)$, $f''(x)$, … at any x

One way to solve: **Newton's method**

Here's a general problem-solving strategy (not specific to Newton's method):

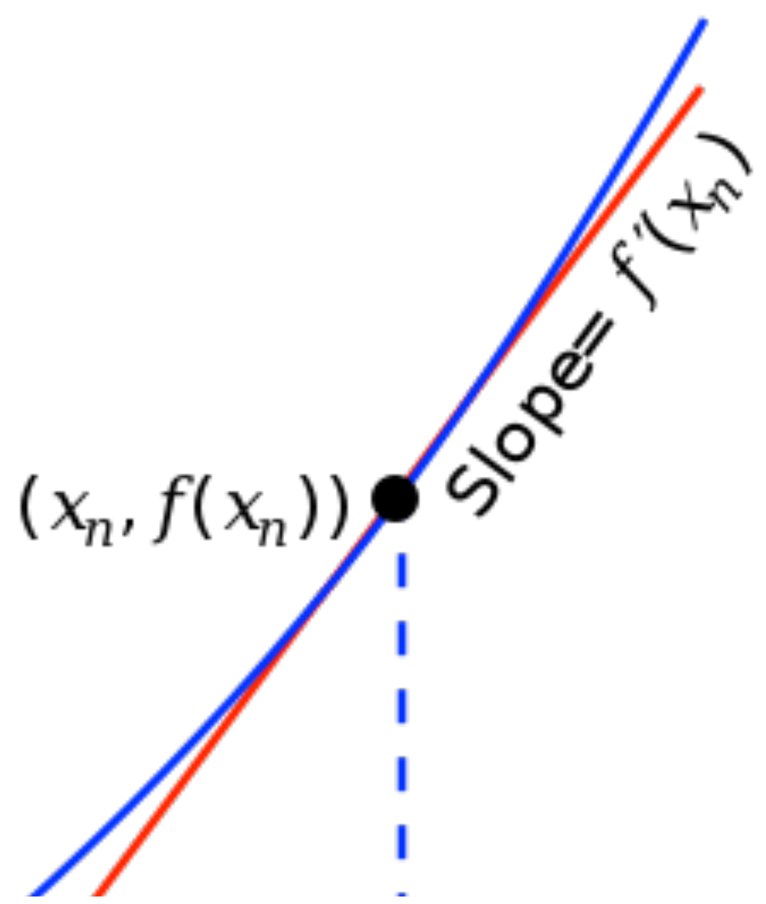
Say you have a problem you don't know how to solve exactly.

1. Approximate the problem.
2. Solve the approximation exactly.
3. If possible, use the solution to improve the approximation, and repeat…

In Newton's method, approximation = 1st-order Taylor series

$$f(x + \Delta x) \approx f(x) + f'(x)\,\Delta x$$

when Δx is small



Say you have a nonlinear equation you don't know how to solve exactly:
Find x such that $f(x) = 0$.

Start with a guess: $\tilde{x}$.

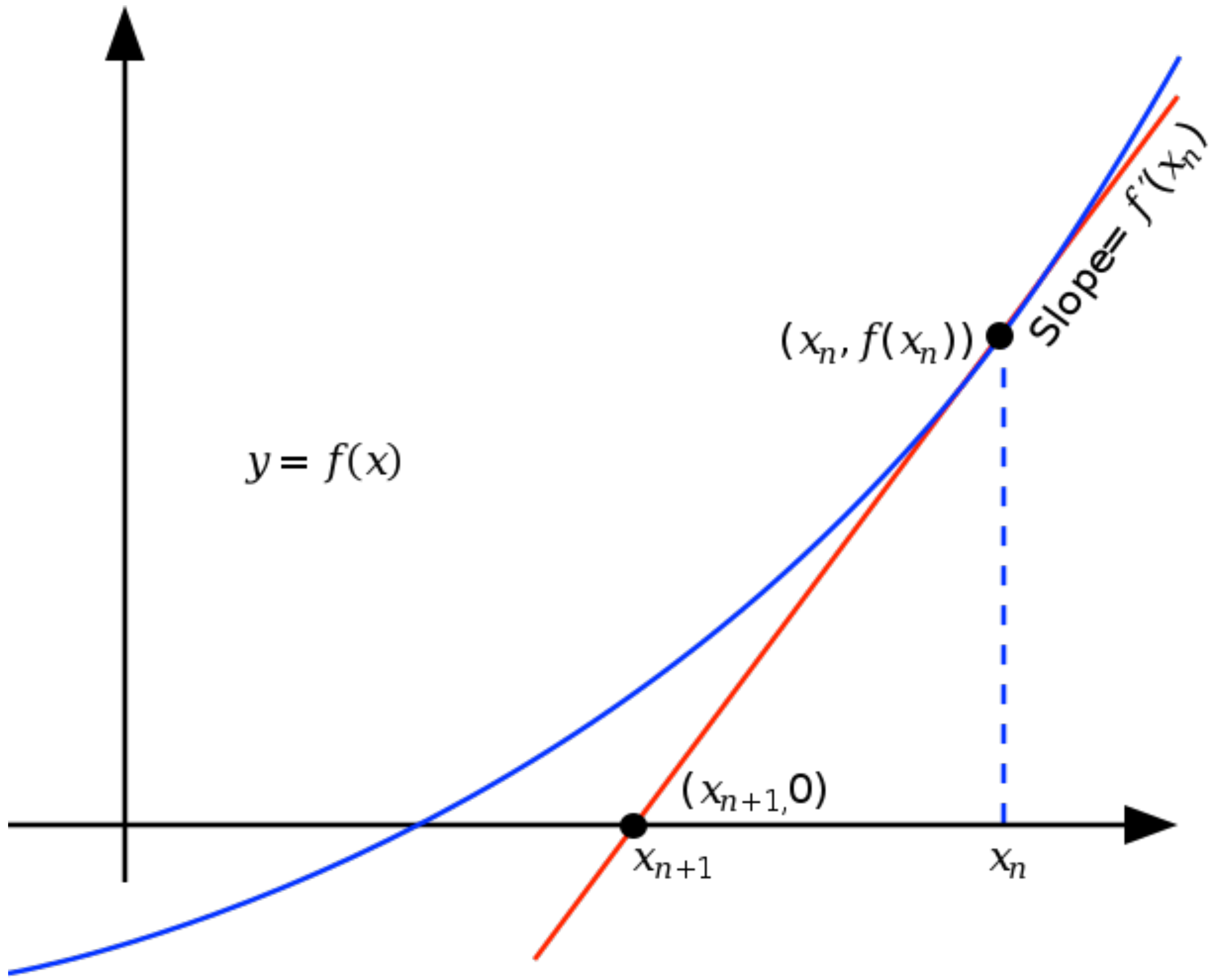
1. Approximate the problem near the guess:

$$0 = f(\tilde{x} + \Delta x) \approx f(\tilde{x}) + f'(\tilde{x})\,\Delta x$$

2. Solve the approximation exactly:

$$\Delta x = -f(\tilde{x})/f'(\tilde{x})$$

3. Improve the guess and repeat: $\tilde{x} \leftarrow \tilde{x} + \Delta x$



Newton's method is not guaranteed to work:

- Can overshoot the solution
- Can move in the wrong direction
- Can diverge when $f'(\tilde{x})$ is close to 0

Converges rapidly when initial guess $\tilde{x}$ is close to the solution

Going to *n* dimensions

Say we have a function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$

$$\mathbf{y} = \mathbf{f}(\mathbf{x})$$

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{bmatrix}$$

Can we do the same thing?

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) \approx \, ?$$

$$f_1(\mathbf{x} + \Delta\mathbf{x}) \approx f_1(\mathbf{x}) + \frac{\partial f_1}{\partial x_1}\Delta x_1 + \cdots + \frac{\partial f_1}{\partial x_n}\Delta x_n$$

$$\vdots$$

$$f_n(\mathbf{x} + \Delta\mathbf{x}) \approx f_n(\mathbf{x}) + \frac{\partial f_n}{\partial x_1}\Delta x_1 + \cdots + \frac{\partial f_n}{\partial x_n}\Delta x_n$$

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\,\Delta\mathbf{x}$$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Jacobian matrix

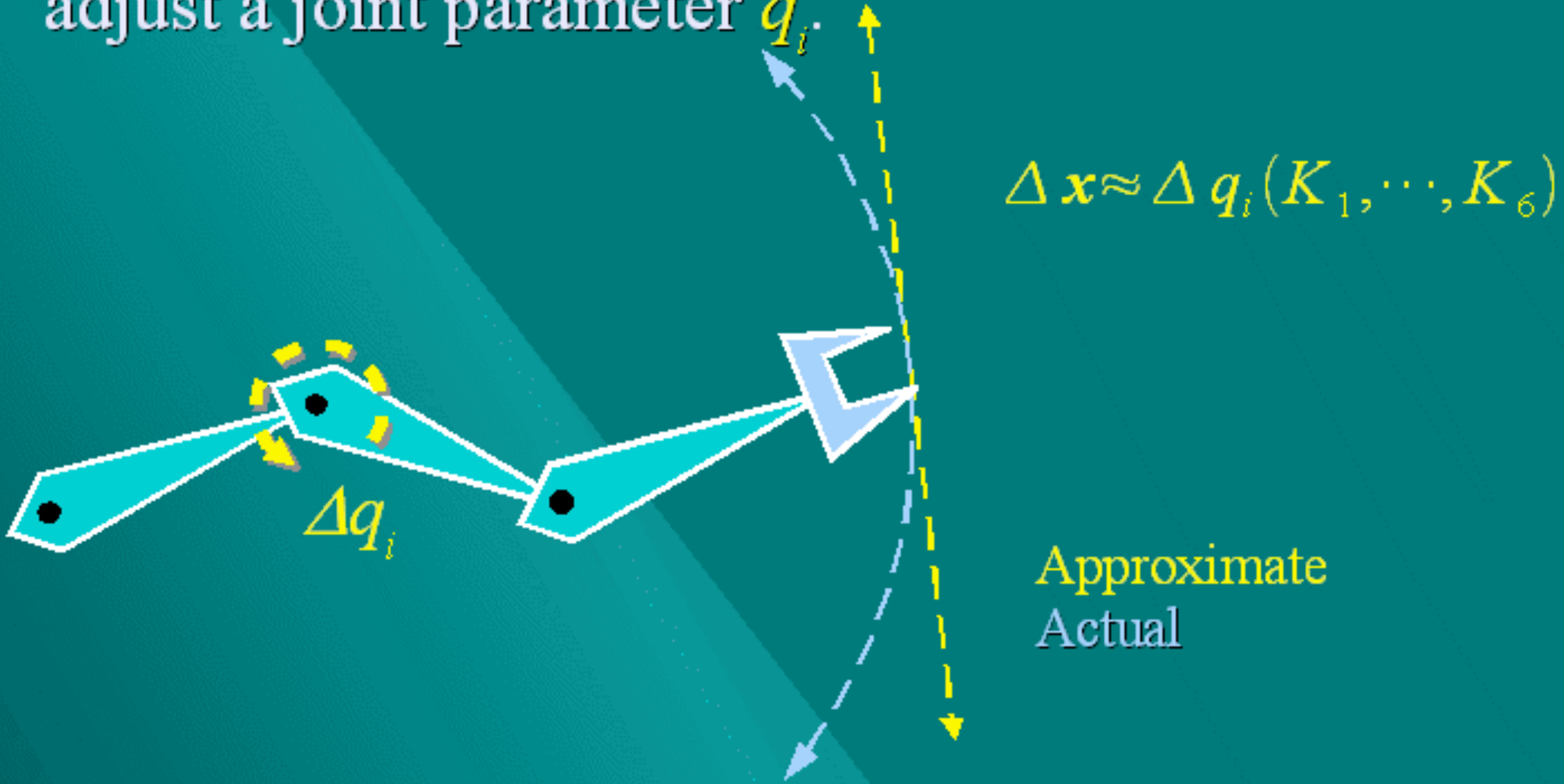
The Jacobian matrix

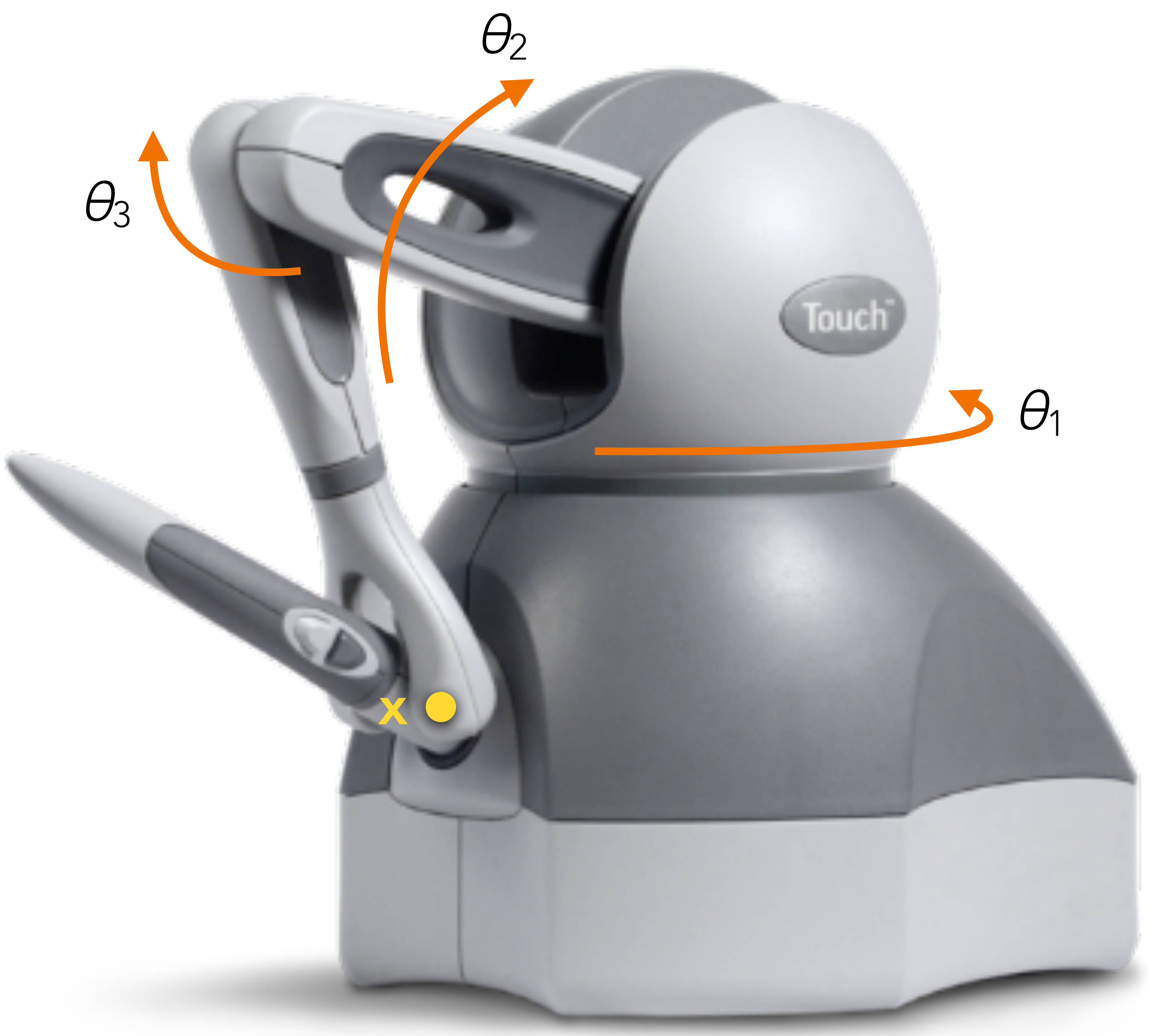
$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

What does this mean, geometrically?

- jth column = $\frac{\partial \mathbf{f}}{\partial x_j}$: how the output $\mathbf{f} = [f_1, \ldots, f_n]$ changes if one coordinate x_j is changed
- ith row = ∇f_i : gradient of f_i with respect to changes in all coordinates $\mathbf{x} = [x_1, \ldots, x_n]$

- Put another way, J tells us approximately how much x will change in world space when we adjust a joint parameter q_i.





Example:

$$\mathbf{x} = \mathbf{f}(\theta_1, \theta_2, \theta_3) = \mathbf{M}_1(\theta_1)\, \mathbf{M}_2(\theta_2)\, \mathbf{M}_3(\theta_3)\, \mathbf{x}^0$$

$$\frac{\partial \mathbf{f}}{\partial \theta_1} = \frac{\mathrm{d}\mathbf{M}_1}{\mathrm{d}\theta_1}\, \mathbf{M}_2(\theta_2)\, \mathbf{M}_3(\theta_3)\, \mathbf{x}^0$$

$$\frac{\partial \mathbf{f}}{\partial \theta_2} = \mathbf{M}_1(\theta_1)\, \frac{\mathrm{d}\mathbf{M}_2}{\mathrm{d}\theta_2}\, \mathbf{M}_3(\theta_3)\, \mathbf{x}^0$$

$$\frac{\partial \mathbf{f}}{\partial \theta_3} = \mathbf{M}_1(\theta_1)\, \mathbf{M}_2(\theta_2)\, \frac{\mathrm{d}\mathbf{M}_3}{\mathrm{d}\theta_3}\, \mathbf{x}^0$$

Newton's method for systems of nonlinear equations: Find $\mathbf{x}$ such that $\mathbf{f}(\mathbf{x}) = 0$.

Start with a guess $\tilde{\mathbf{x}}$.

1. Approximate the problem near the guess:

$$0 = \mathbf{f}(\tilde{\mathbf{x}} + \Delta\mathbf{x}) \approx \mathbf{f}(\tilde{\mathbf{x}}) + \mathbf{J}(\tilde{\mathbf{x}})\, \Delta\mathbf{x}$$

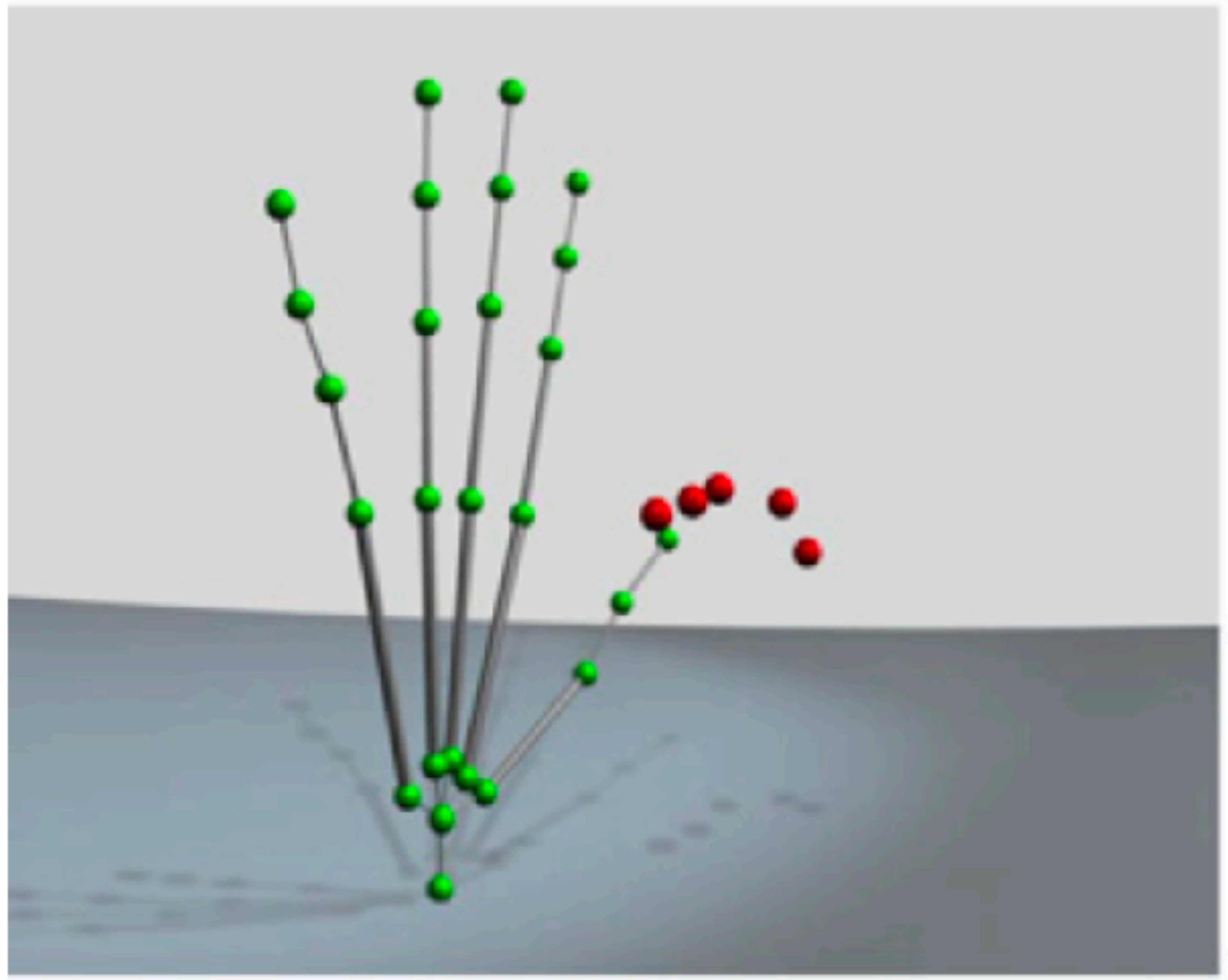
2. Solve the approximation exactly:

$$\Delta\mathbf{x} = -\mathbf{J}(\tilde{\mathbf{x}})^{-1}\, \mathbf{f}(\tilde{\mathbf{x}})$$

3. Improve the guess and repeat: $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} + \Delta\mathbf{x}$

Back to inverse kinematics

In IK, we usually have more joints than end effector DOFs…



$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} f_1(x_1, \ldots, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, \ldots, x_n) \end{bmatrix}$$

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Jacobian becomes rectangular, can't do Newton update $\Delta\mathbf{x} = -\mathbf{J}(\tilde{\mathbf{x}})^{-1}\,\mathbf{f}(\tilde{\mathbf{x}})$!

Jacobian-based strategies:

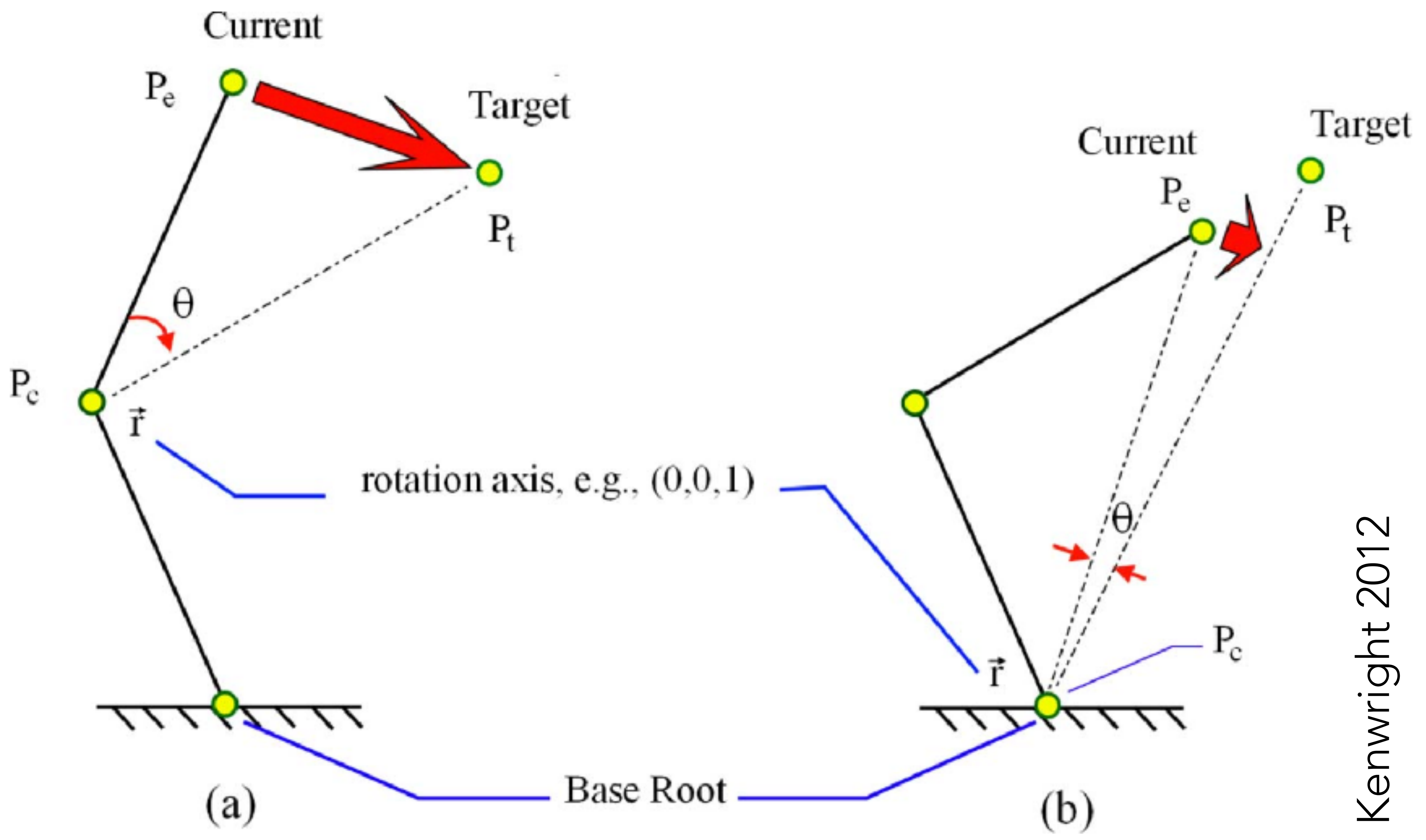
We know $\Delta\mathbf{f} = \mathbf{J}(\tilde{\mathbf{x}})\,\Delta\mathbf{x}$, choose $\Delta\mathbf{x}$ to get desired $\Delta\mathbf{f} = -\mathbf{f}(\tilde{\mathbf{x}})$

- Pseudoinverse of Jacobian: $\Delta\mathbf{x} = \mathbf{J}^{T}\,(\mathbf{J}\,\mathbf{J}^{T})^{-1}\,\Delta\mathbf{f}$
- Jacobian transpose: $\Delta\mathbf{x} = \alpha\,\mathbf{J}^{T}\,\Delta\mathbf{f}$
- Damped least squares: $\Delta\mathbf{x} = \mathbf{J}^{T}\,(\mathbf{J}\,\mathbf{J}^{T} + \lambda\,\mathbf{I})^{-1}\,\Delta\mathbf{f}$

Cyclic coordinate descent

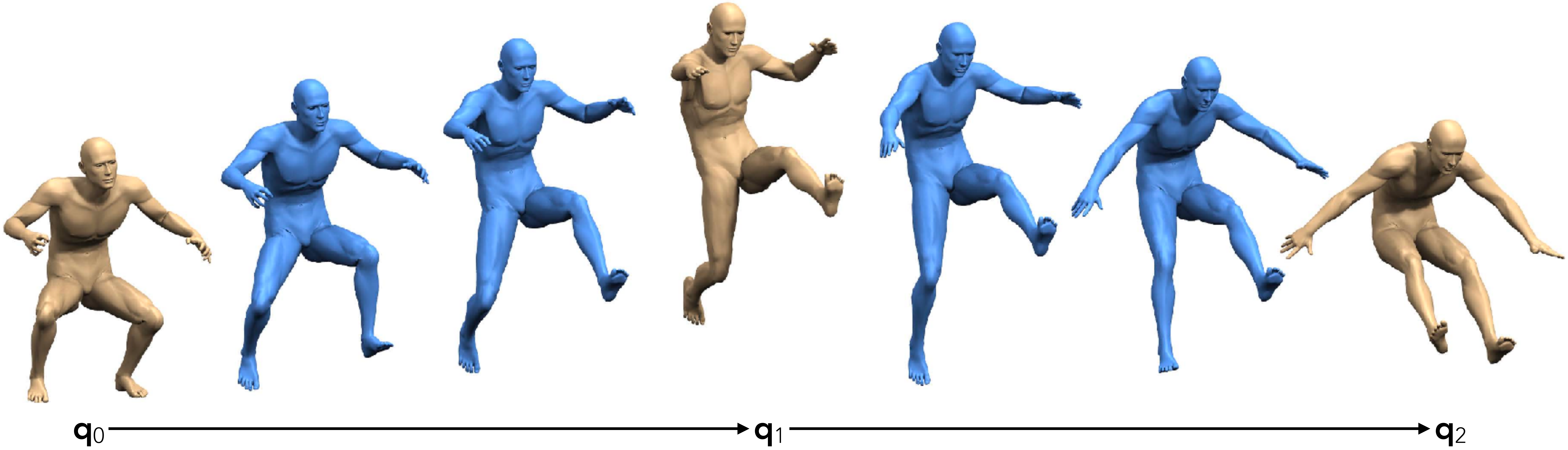
- Pick one coordinate *j*, hold all others fixed.
- Update q_j to best value
- Repeat with new choice of *j*

Smarter version: FABRIK (Aristidou et al. 2011)



Kenwright 2012

Keyframe animation



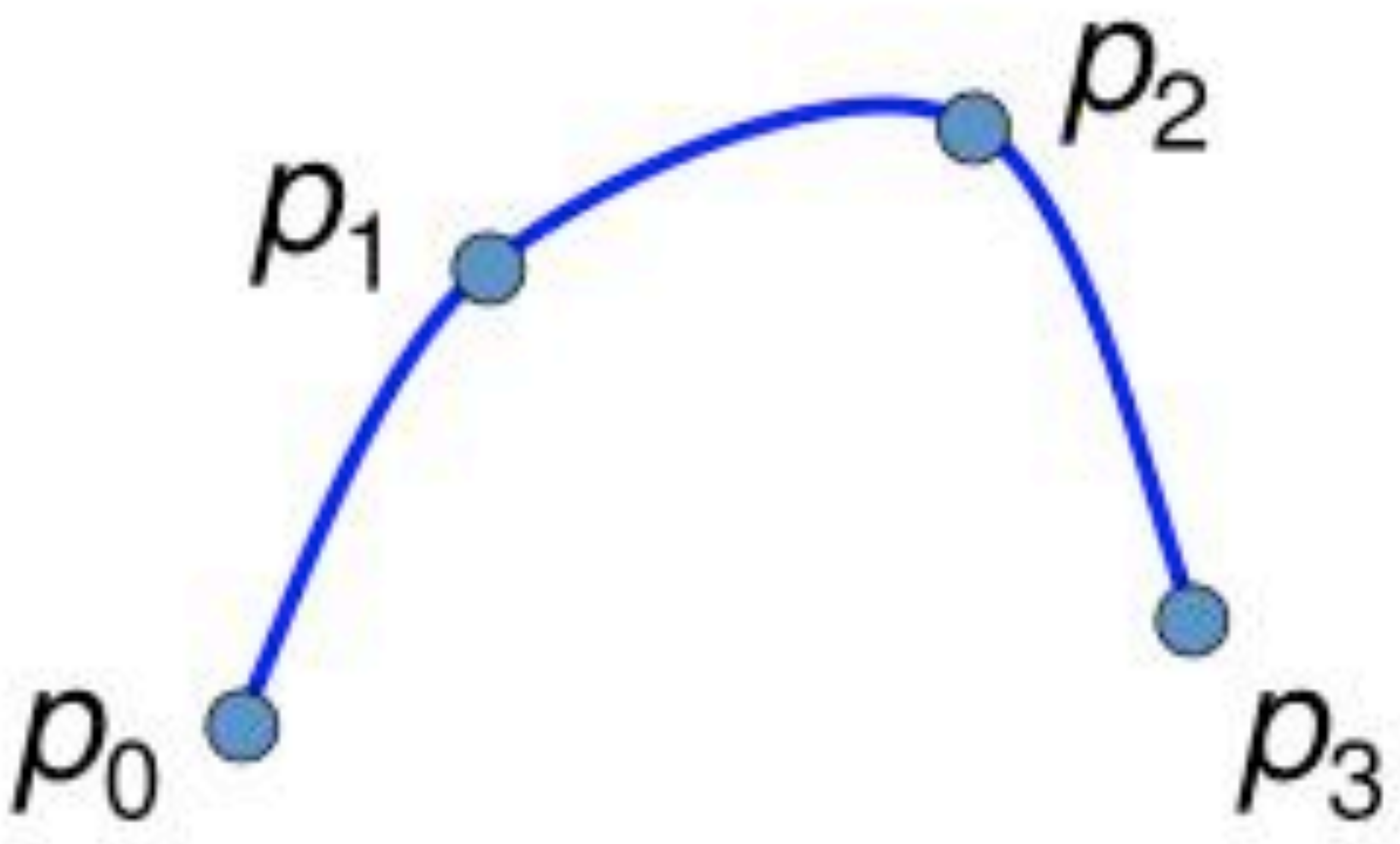
Chu & Lee 2009

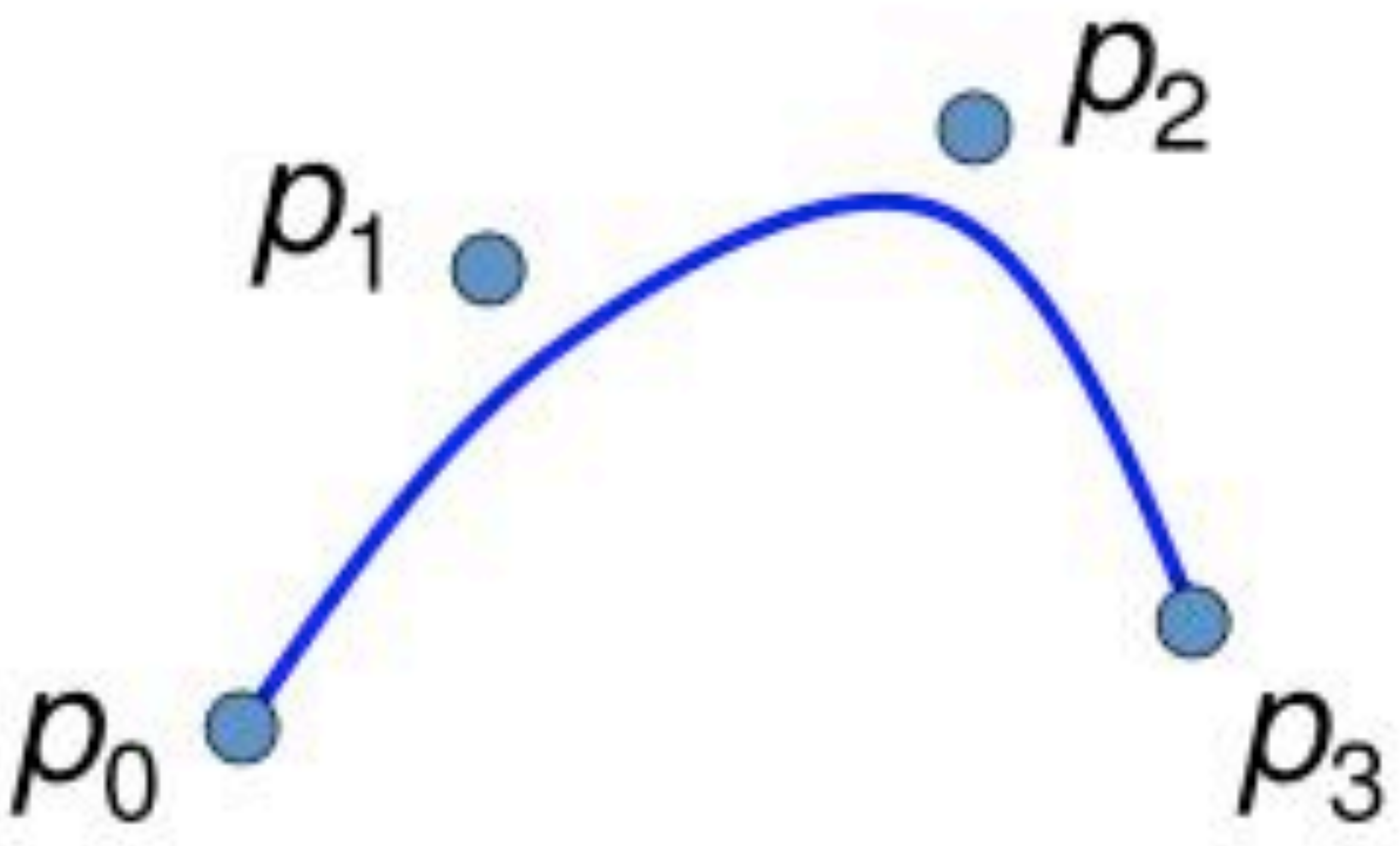
Animator specifies character pose (i.e. values of animation controls) at specific keyframes.

How to interpolate to arbitrary times?

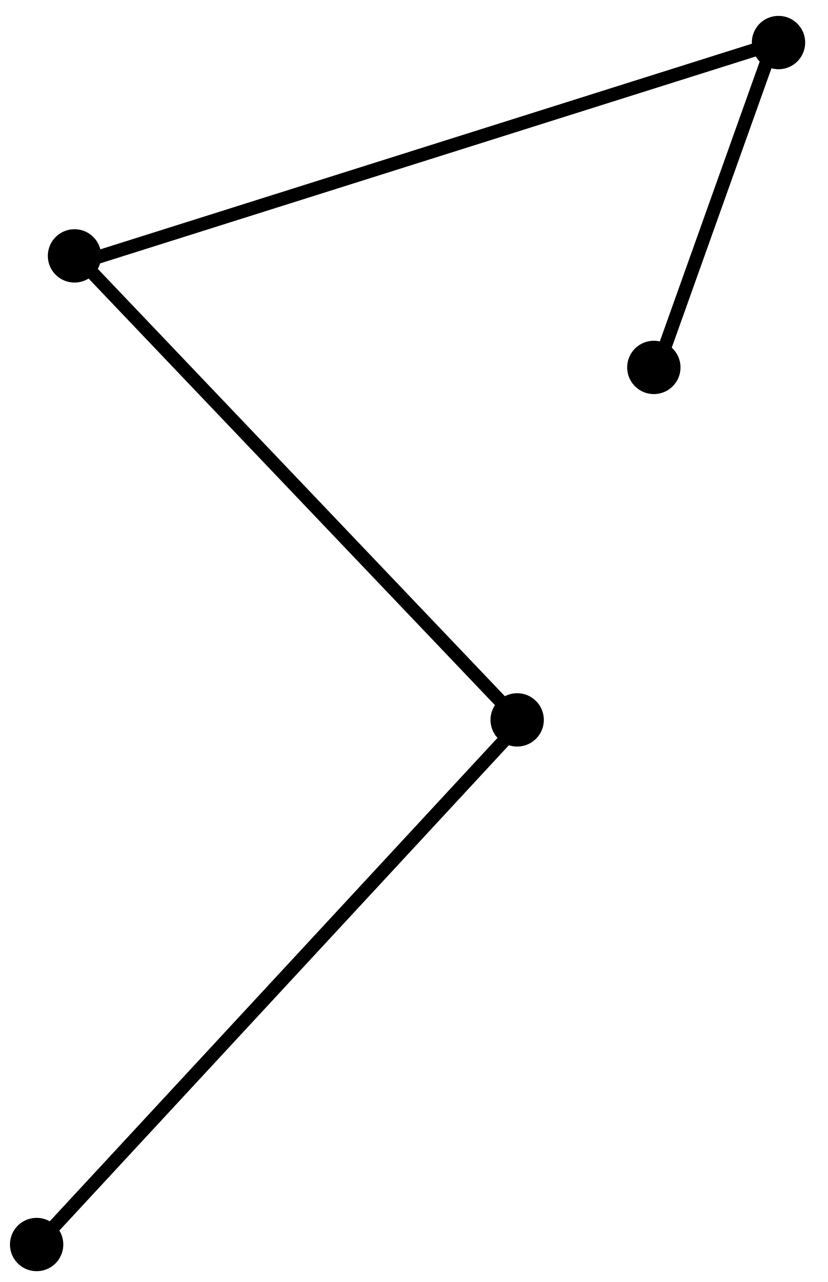
Recall **splines**: piecewise polynomial functions with some continuity/differentiability

Except now, we really want interpolation instead of approximation:
We want the animation to exactly match the specified pose at the keyframes





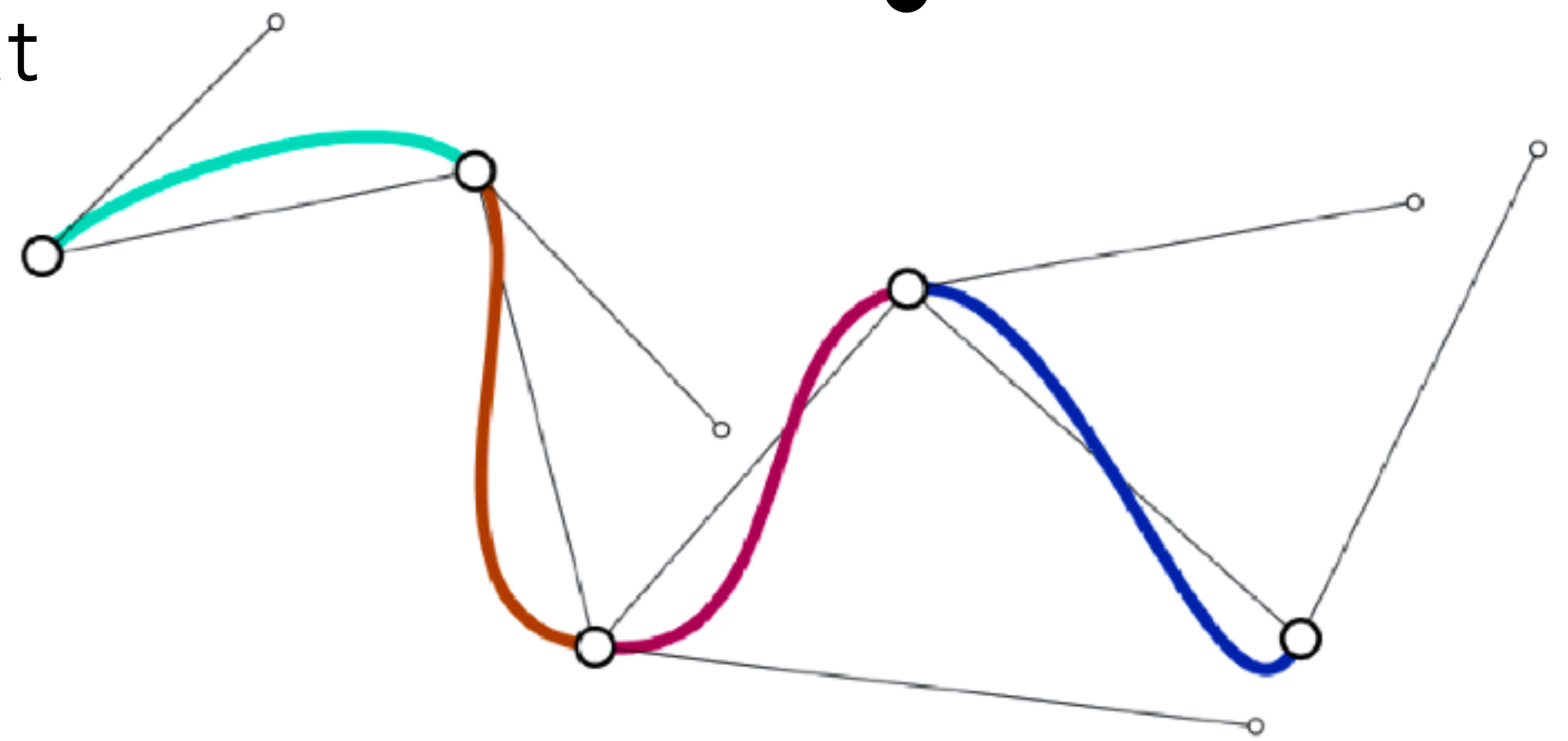
- Piecewise linear interpolation



$$q(t) = \frac{t_{i+1} - t}{t_{i+1} - t_i}\, q_i + \frac{t - t_i}{t_{i+1} - t_i}\, q_{i+1}$$

- Cubic Hermite spline: assume positions q_i and velocities q_i' are given.

Let $q(t) = at^3 + bt^2 + ct + d$, solve for coefficients so that

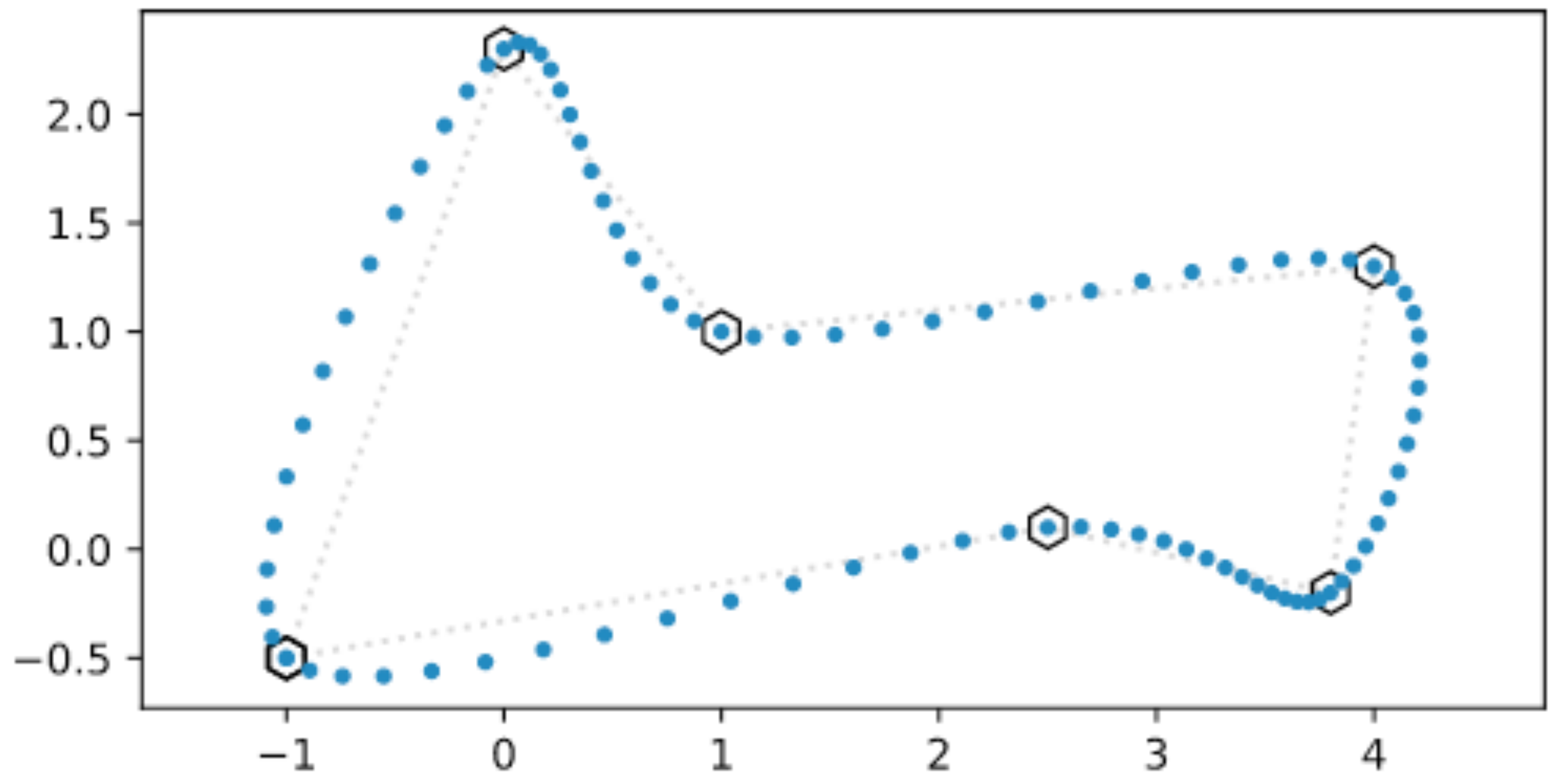


$$q(t_i) = q_i,\ q(t_{i+1}) = q_{i+1},$$
$$q'(t_i) = q_i',\ q'(t_{i+1}) = q'_{i+1}$$

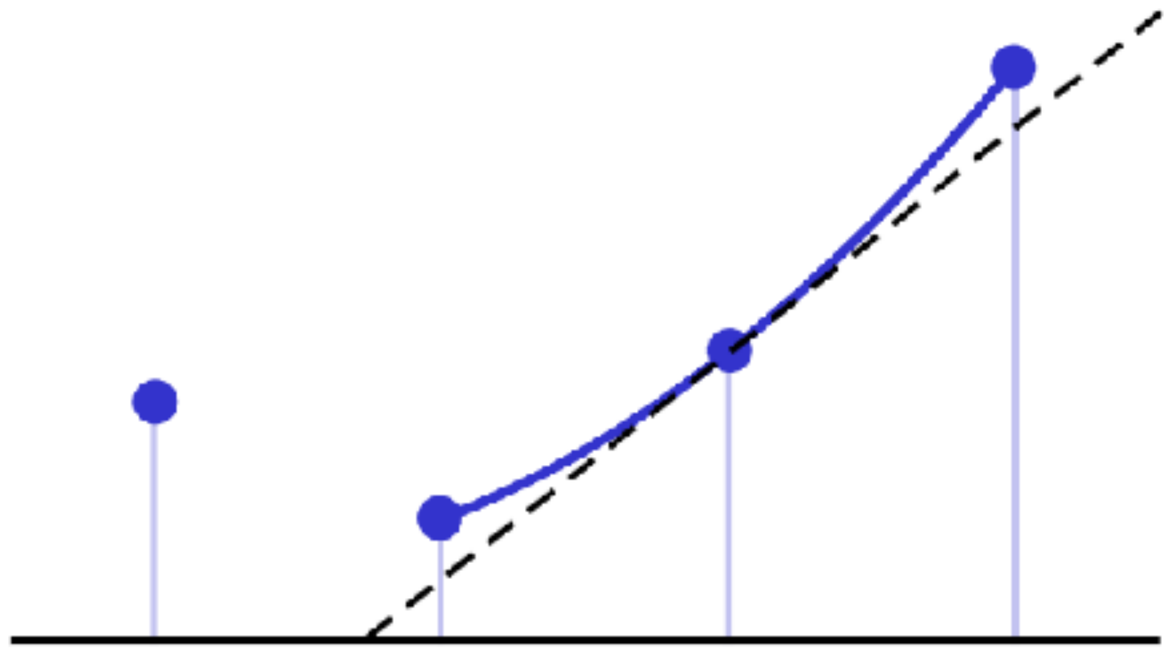
Closed-form solution:

$$q(t) = (2t^3-3t^2+1)\, q_i + (t^3-2t^2+t)\, q_i' + (-2t^3+3t^2)\, q_{i+1} + (t^3-t^2)\, q'_{i+1}$$

What if derivatives are not given, but still want a C^1 curve? **Catmull-Rom splines**



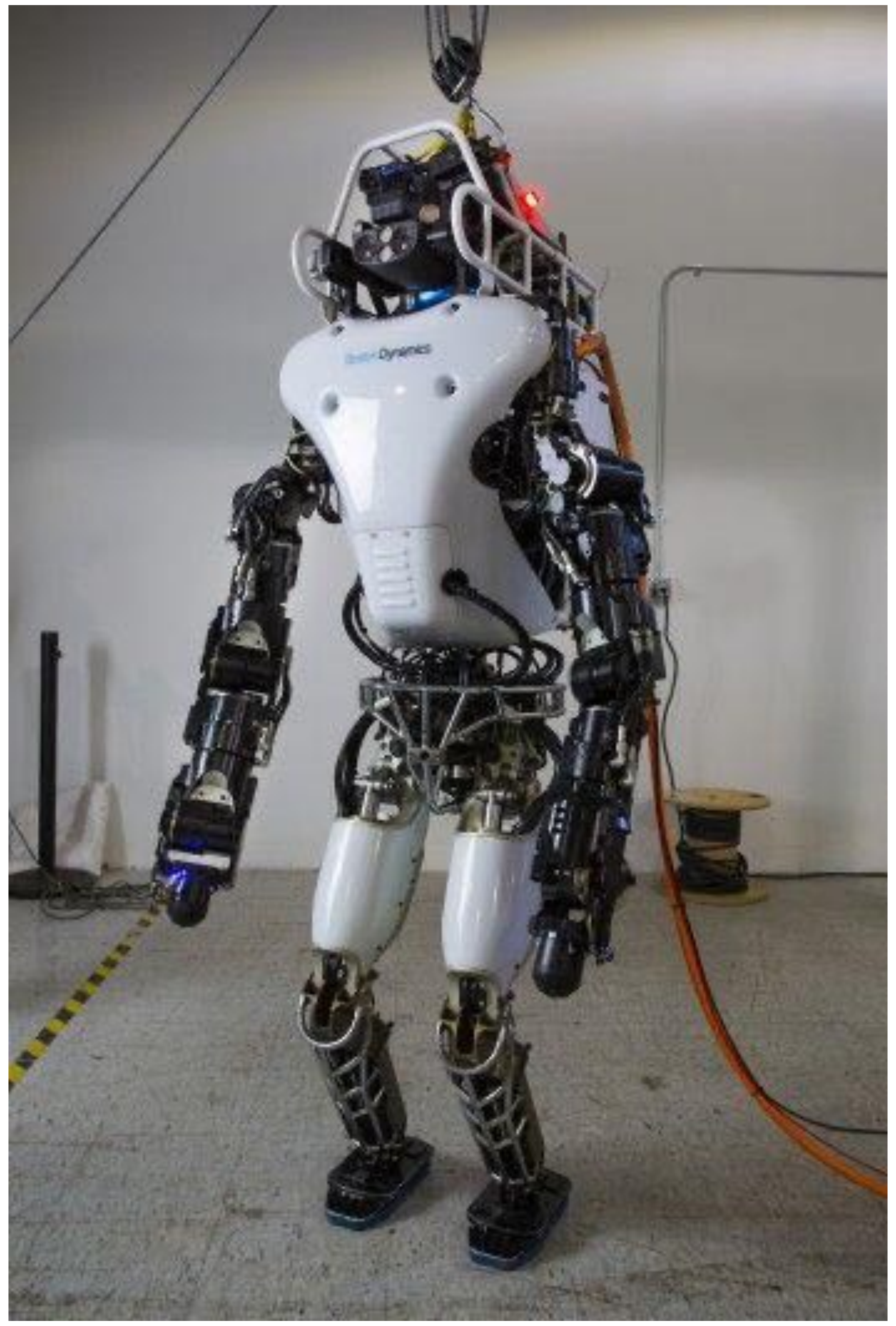
Estimate derivatives from neighbouring points, e.g. slope at t_i of quadratic passing through q_{i-1}, q_i, q_{i+1}

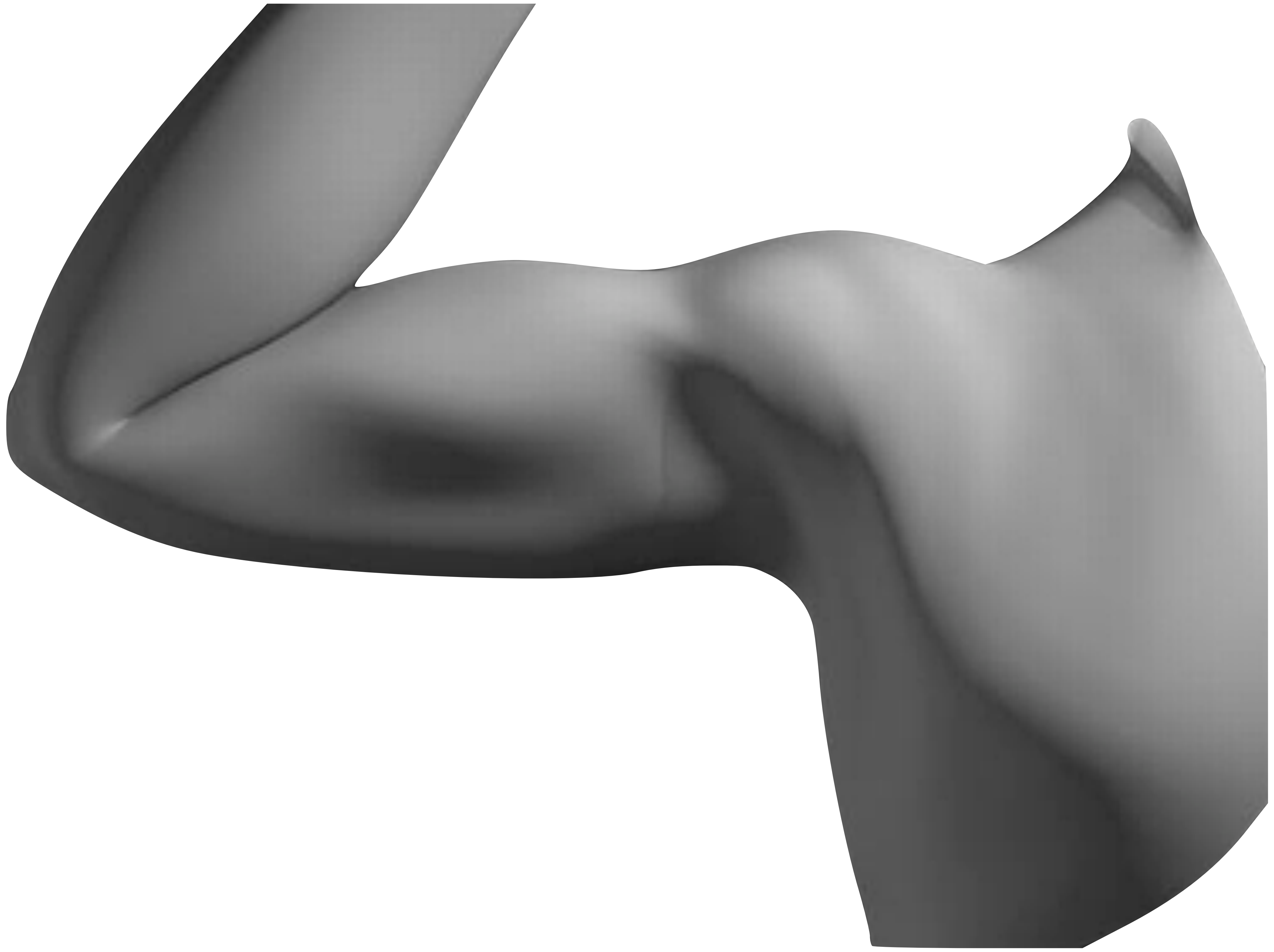


- Equally spaced points: $q_i' = \dfrac{q_{i+1} - q_{i-1}}{2\Delta t}$
- Unequally spaced points: not as simple (work it out yourself)

Rigid bone transformations may be sufficient for robots and toys with rigid parts.

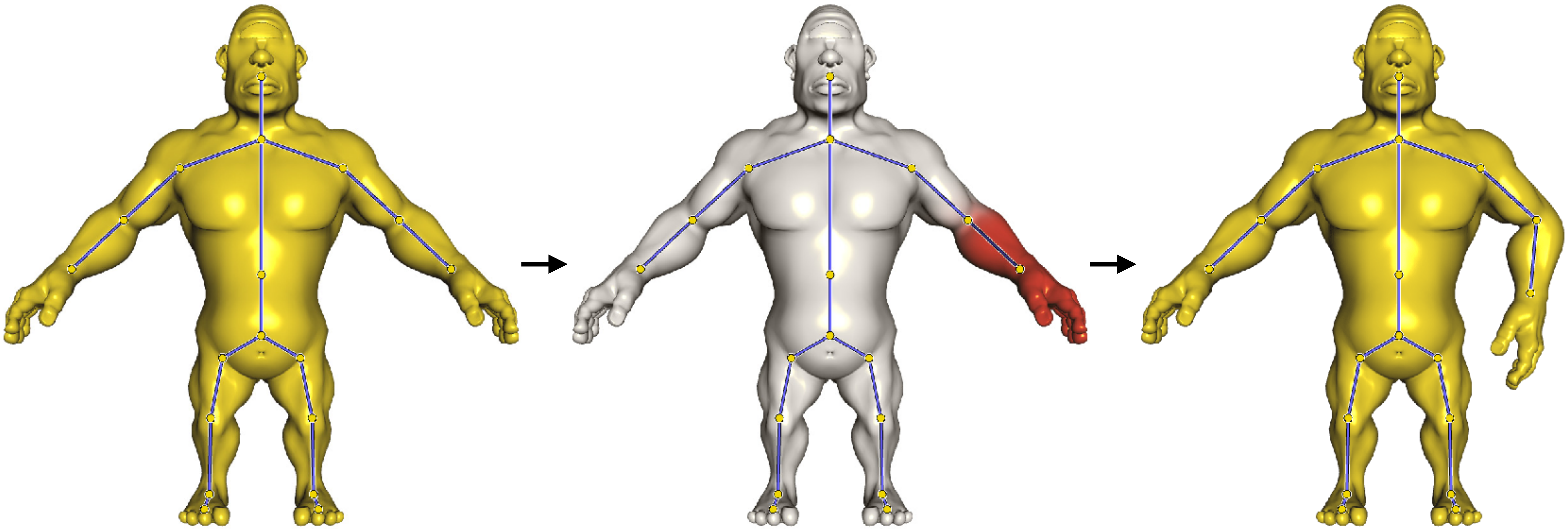
What about organic characters?





J.P. Lewis

Skinning



Skeleton → Skinning weights → Deformed shape

Alec Jacobson