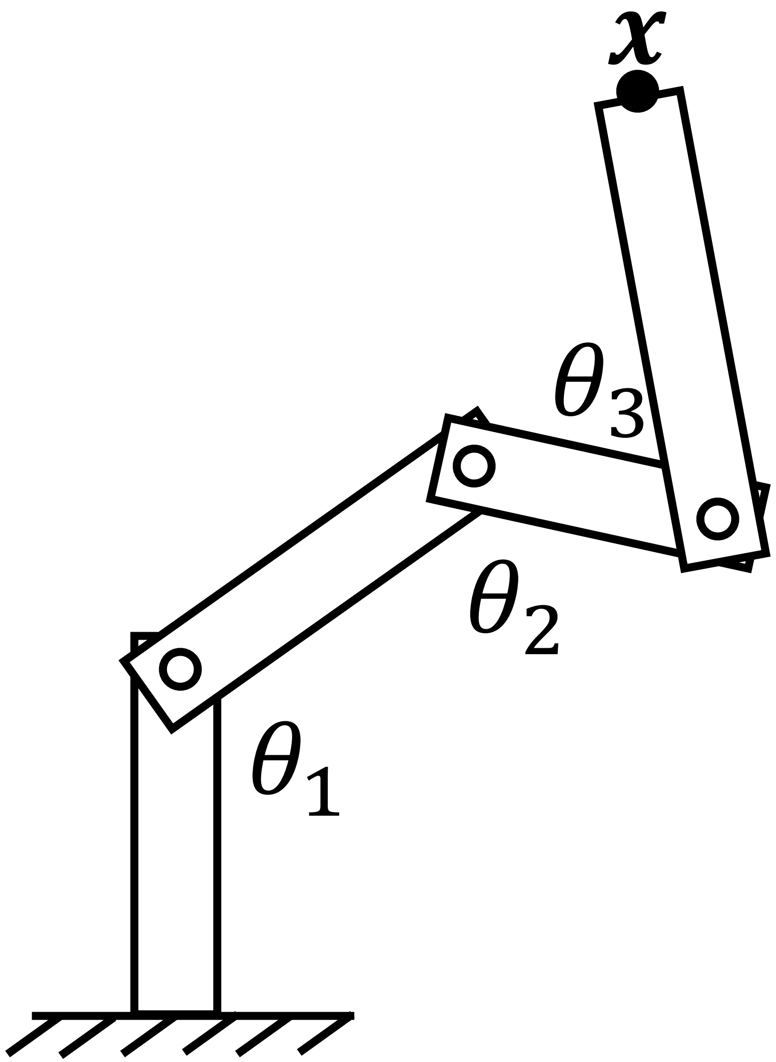
derivatives for design and control

with Jim and Simon

review: serial manipulator



end effector position $x \in \mathbb{R}^2$

motor angles $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_K \end{bmatrix}$

forward kinematics (FK)

what is $\boldsymbol{x}(\boldsymbol{\theta})$?

i.e., given joint angles $\boldsymbol{\theta}$, what is the corresponding tip position $\boldsymbol{x}$?

→ something like $\boldsymbol{x}(\boldsymbol{\theta}) = \boldsymbol{T}_K \boldsymbol{R}_K \cdots \boldsymbol{T}_1 \boldsymbol{R}_1 \boldsymbol{O}$ // some big analytic

// expression with a bunch

// of $\sin(\theta_i)$'s and $\cos(\theta_j)$'s

inverse kinematics (IK)

what is $\boldsymbol{\theta}^*(\widetilde{\boldsymbol{x}})$?

i.e., given joint target tip position $\widetilde{\boldsymbol{x}}$, what is <u>an optimal choice</u> of joint angles $\boldsymbol{\theta}^*$?

option 0: solve analytically

option 1: use optimization

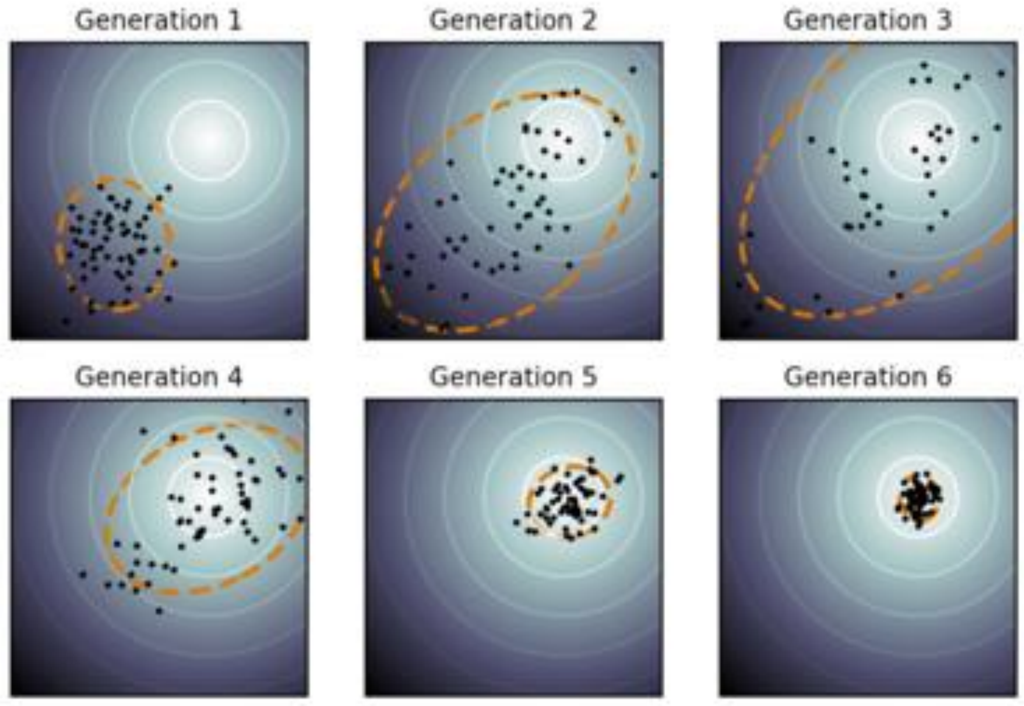
minimize a suitable objective

$$\boxed{\boldsymbol{f}_0(\boldsymbol{\theta}) = \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{x}(\boldsymbol{\theta}) - \widetilde{\boldsymbol{x}})^{\boldsymbol{T}}(\boldsymbol{x}(\boldsymbol{\theta}) - \widetilde{\boldsymbol{x}})}$$

option 1a: derivative-free optimization

requires no derivatives

- when in doubt just use CMA-ES



option 1b: derivative-based optimization

may require 1 derivative (gradient)…

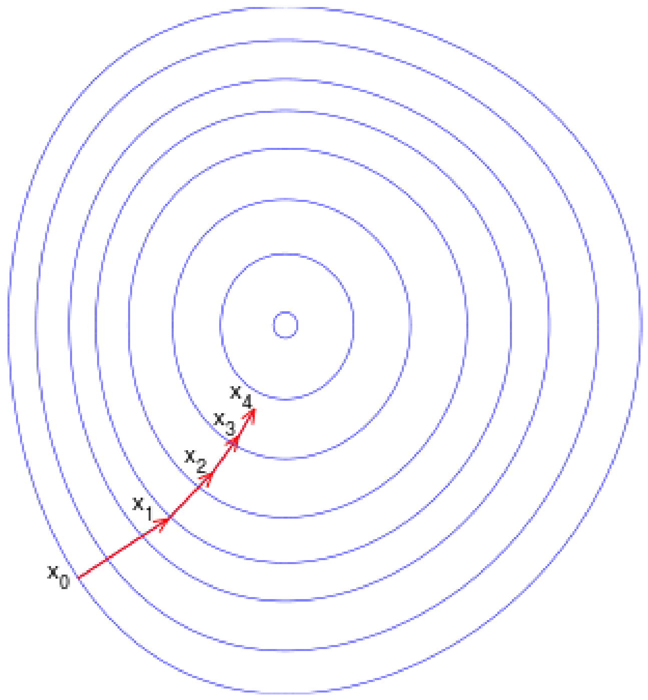
gradient descent

may require 2 derivatives (gradient and Hessian)…

Newton’s method

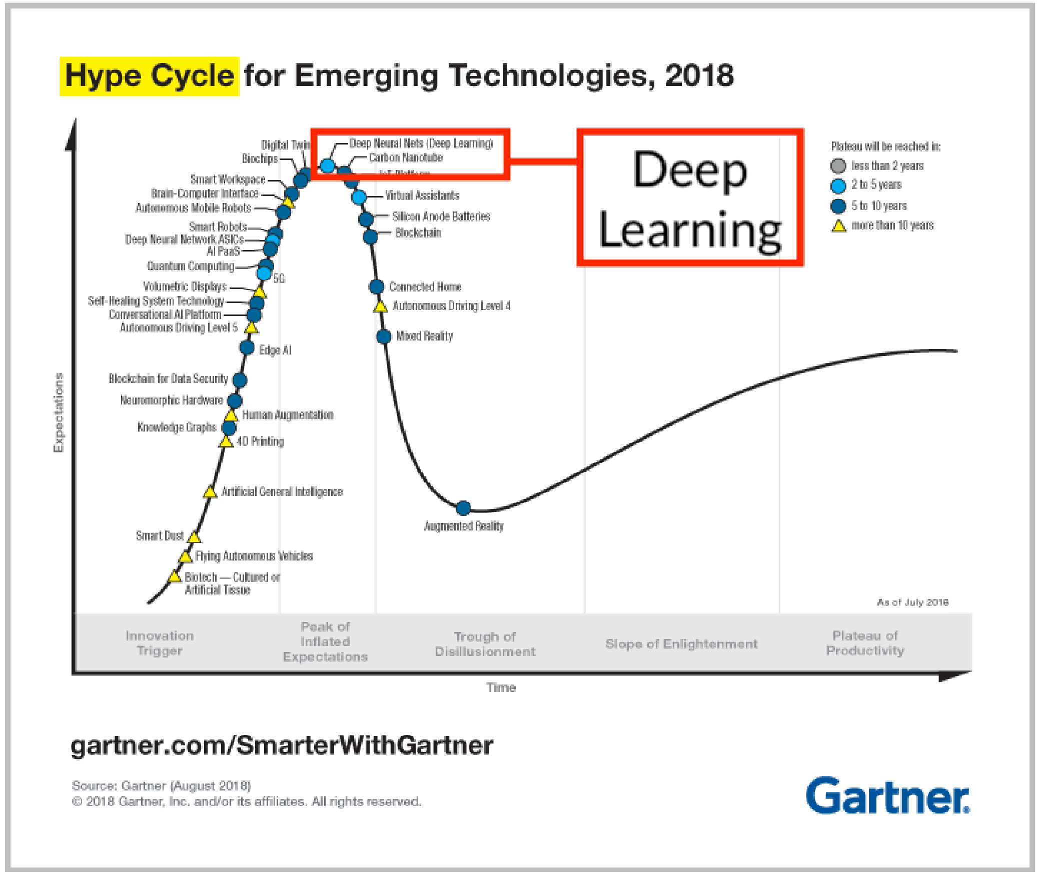
or be somewhere in the middle…

Gauss-Newton, L-BFGS

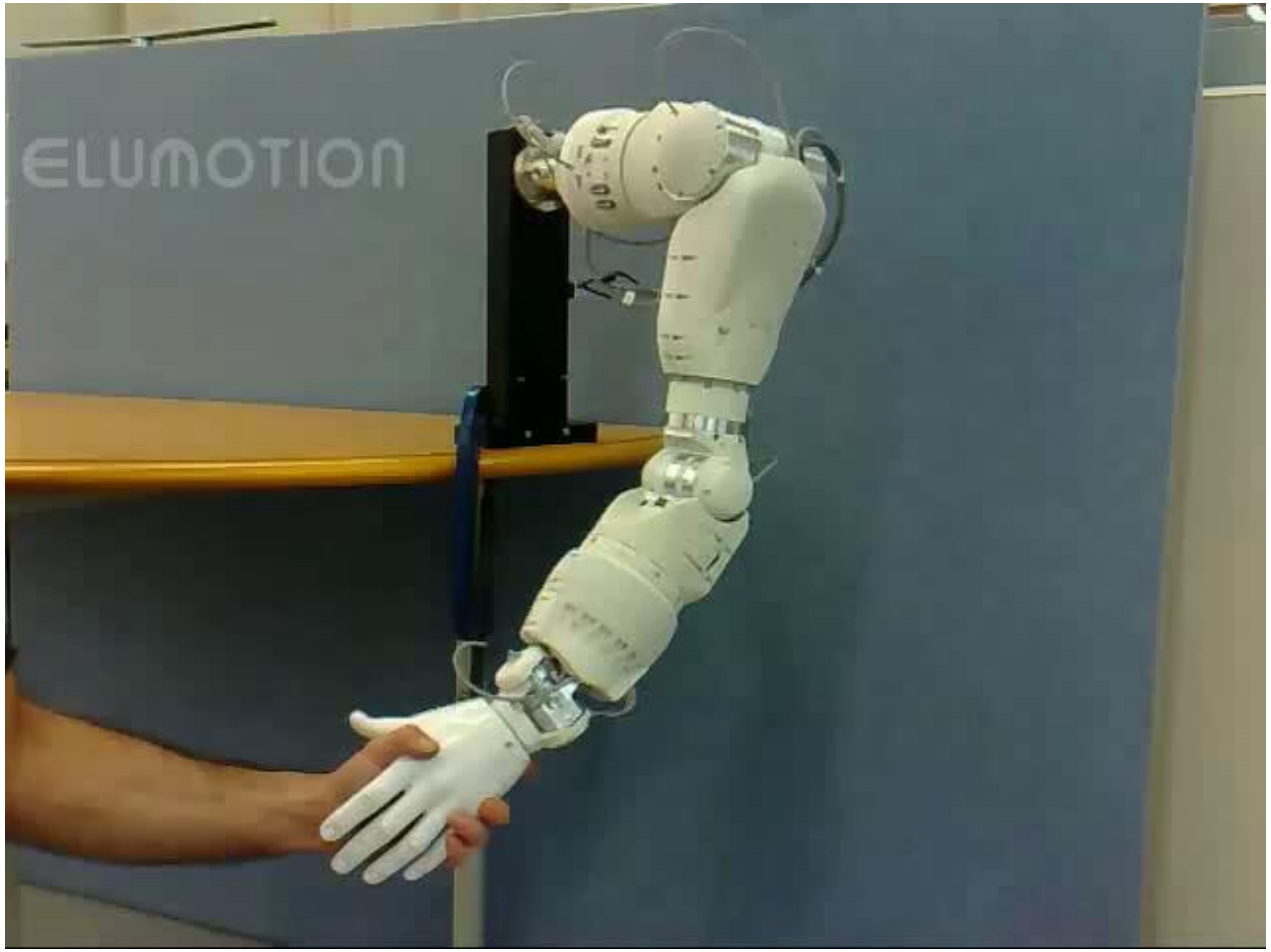


option 2: learn it

build a large set of training data $\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i)\}_{i=1}^{N}$ using forward kinematics, then train a deep net using tensor flow, and evaluate the deep net at $\boldsymbol{\theta}$



option 3: invert kinematics using the *real world*



option 1b: derivative-based optimization

Say the objective is $f(\boldsymbol{x}(\boldsymbol{p})) = \frac{1}{2}(\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})^T(\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})$

gradient is $\frac{df}{d\boldsymbol{p}} = \frac{\partial f}{\partial \boldsymbol{x}} \frac{d\boldsymbol{x}}{d\boldsymbol{p}}$ // chain rule

$\frac{\partial f}{\partial \boldsymbol{x}} = (\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})$ is trivial to compute

and for a serial manipulator, $\frac{d\boldsymbol{x}}{d\boldsymbol{p}}$ can be computed analytically

But what if $x(p)$ does not have an analytic expression?

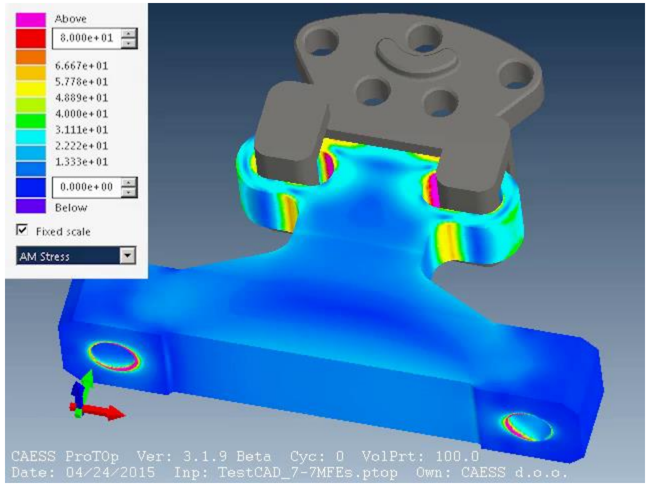
For example, static equilibrium of a finite element mesh:

$$\boldsymbol{x}(\boldsymbol{p}) = arg \min_{\boldsymbol{x}} E(\boldsymbol{x}, \boldsymbol{p})$$

Still want to solve optimization problems of this form:

$$\min_{\boldsymbol{p}} f\big(\boldsymbol{x}(\boldsymbol{p})\big)$$

An example: topology optimization



Modeling continuous Relation between Parameters and State

- **Observation**: when we set parameters $\boldsymbol{p}$, we observe the state $\boldsymbol{x}$ as the result of simulation.
- Although $\boldsymbol{x}$ are problem variables, they are not real DOFs – they are functions of the parameters, i.e.,

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{p})$$

- Map from parameters to state is

$$\boldsymbol{x} = \text{simulate}(\boldsymbol{p})$$

- For design, we need derivatives of $\boldsymbol{x}(\boldsymbol{p})$,

$$\frac{\partial f}{\partial \boldsymbol{p}} = \frac{\partial f}{\partial \boldsymbol{x}} \frac{\mathrm{d}\boldsymbol{x}}{d\boldsymbol{p}}$$

- But how to compute these derivatives,

$$\frac{d\boldsymbol{x}}{d\boldsymbol{p}} = \frac{d\text{simulation}}{d\boldsymbol{p}}?$$

- The derivative of an argmin...?

Differentiating the Map

- Although we can evaluate the map $\boldsymbol{x} \rightarrow \boldsymbol{x}(\boldsymbol{p})$, this map is not available in closed-form (i.e., *analytically*)
- $\boldsymbol{x} \rightarrow \boldsymbol{x}(\boldsymbol{p})$ requires minimizing a function, i.e., solving a system of nonlinear equations.
- In general, it is impractical to compute derivatives of the minimization process.
- But even though $\boldsymbol{x} \rightarrow \boldsymbol{x}(\boldsymbol{p})$ is not given *explicitly*, the gradient of the objective

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{p}) = \nabla_{\mathbf{x}}\mathrm{E} = \frac{\boldsymbol{dE}}{\boldsymbol{dx}} = \mathbf{0}$$

provides this map *implicitly*.

Differentiating the Map

- Suppose that $(\boldsymbol{x},\boldsymbol{p})$ is a feasible pair, i.e., $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{p}) \;=\; \mathbf{0}$. In other words, $\boldsymbol{x}$ is an equilibrium configuration for $\boldsymbol{p}$.
- If we apply a parameter perturbation $\Delta\boldsymbol{p}$, the system will undergo displacements $\Delta\boldsymbol{x}$ such that it is again in equilibrium,

$$\boldsymbol{g}(\boldsymbol{x}+\Delta\boldsymbol{x},\boldsymbol{p}+\Delta\boldsymbol{p}) \;=\; \mathbf{0}$$

- Since this has to hold for arbitrary parameter variations, we have

$$\frac{d\boldsymbol{g}}{d\boldsymbol{p}} = \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{x}}\frac{d\boldsymbol{x}}{d\boldsymbol{p}} + \frac{\partial\boldsymbol{g}}{\partial\boldsymbol{p}} = \mathbf{0} \quad \text{// total derivative}$$

- If the Jacobian $\nabla_{\boldsymbol{x}}\boldsymbol{g}$ is non-singular, we have

$$\frac{d\boldsymbol{x}}{d\boldsymbol{p}} = -\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{x}}^{-1}\frac{\partial\boldsymbol{g}}{\partial\boldsymbol{p}}$$

Sensitivity Analysis

…d to parameters

We could in principle *approximate* the sensitivity…

$$D_{\Delta p} \boldsymbol{f}(\boldsymbol{p}) \sim \frac{\boldsymbol{f}(\boldsymbol{p}+h\Delta\boldsymbol{p}) - \boldsymbol{f}(\boldsymbol{p})}{\boldsymbol{h}}$$ **// "finite difference"**

…but this approach is typically real slow

option 1b: derivative-based optimization

Say the objective is $f(\boldsymbol{x}(\boldsymbol{p})) = \frac{1}{2}(\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})^T(\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})$

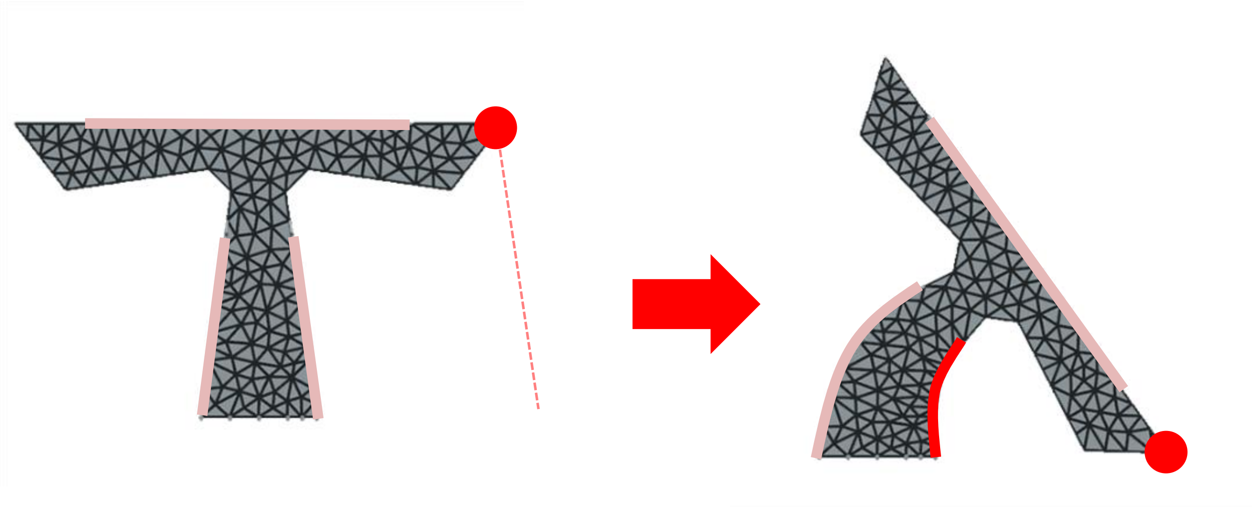
gradient is $\frac{df}{d\boldsymbol{p}} = \frac{\partial f}{\partial \boldsymbol{x}}\frac{d\boldsymbol{x}}{d\boldsymbol{p}}$ // chain rule

$\frac{\partial f}{\partial \boldsymbol{x}} = (\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})$ is trivial to compute

and for statically stable FEM (and for many, many other systems),

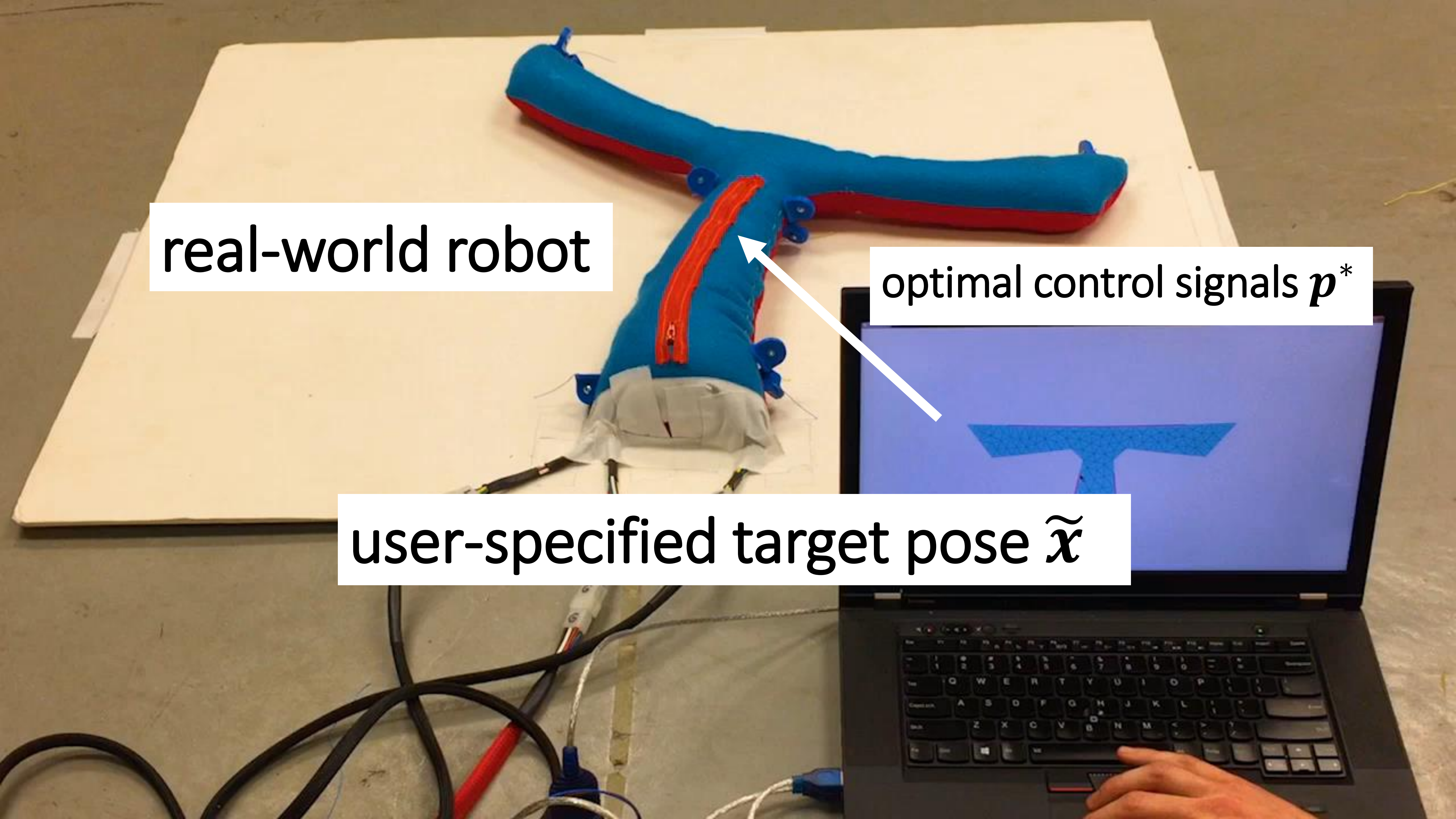
$\frac{d\boldsymbol{x}}{d\boldsymbol{p}}$ can be computed using sensitivity analysis

application: soft IK



say the control input $\boldsymbol{p}$ are the contacted lengths of cables in a soft robot...
given a target pose $\widetilde{\boldsymbol{x}}$, what is the optimal control $\boldsymbol{p}^*$?

$$f(\boldsymbol{x}(\boldsymbol{p})) = \frac{1}{2}(\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})^T \boldsymbol{Q}(\boldsymbol{x}(\boldsymbol{p}) - \widetilde{\boldsymbol{x}})$$


real-world robot
optimal control signals $\boldsymbol{p}^*$
user-specified target pose $\widetilde{\boldsymbol{x}}$