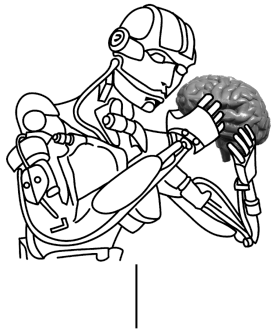
Jacobian methods for inverse kinematics and planning

Slides from Stefan Schaal
USC, Max Planck

The Inverse Kinematics Problem



- Direct Kinematics

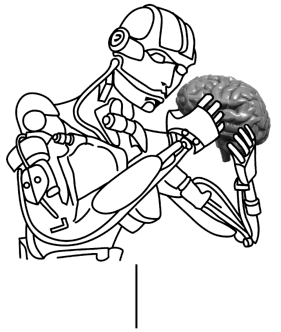
$$\mathbf{x} = f(\theta)$$

- Inverse Kinematics

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

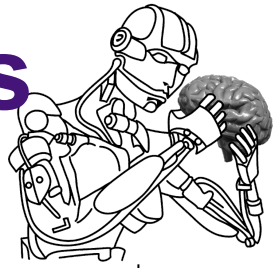
- Possible Problems of Inverse Kinematics
  - Multiple solutions
  - Infinitely many solutions
  - No solutions
  - No closed-form (analytical solution)

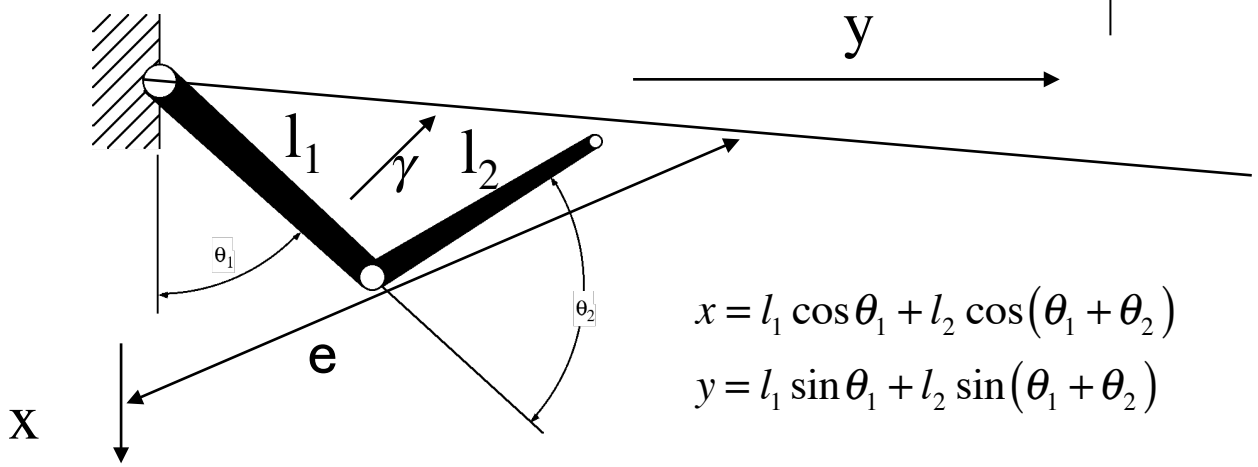
Analytical (Algebraic) Solutions



- Analytically invert the direct kinematics equations and enumerate all solution branches
  - Note: this only works if the number of constraints is the same as the number of degrees-of-freedom of the robot
  - What if not?
    - Iterative solutions
    - Invent artificial constraints
- Examples
  - 2DOF arm
  - See S&S textbook 2.11 ff

Analytical Inverse Kinematics of a 2 DOF Arm



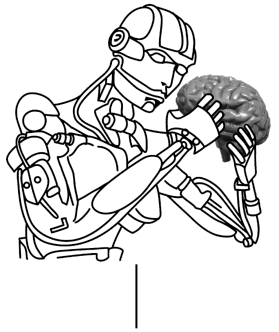


$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

- Inverse Kinematics:

$$l = \sqrt{x^2 + y^2}$$
$$l_2^2 = l_1^2 + l^2 - 2 l_1 l \cos\gamma$$
$$\Rightarrow \gamma = \arccos\left(\frac{l^2 + l_1^2 - l_2^2}{2 l_1 l}\right)$$
$$\frac{y}{x} = \tan\varepsilon \quad \Rightarrow \quad \theta_1 = \arctan\frac{y}{x} - \gamma$$
$$\theta_2 = \arctan\left(\frac{y - l_1 \sin\theta}{x - l_1 \cos\theta_1}\right) - \theta_1$$

Iterative Solutions of Inverse Kinematics



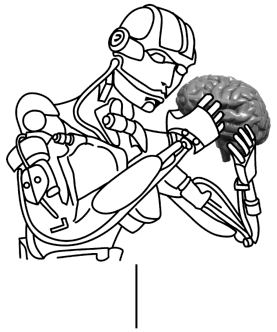
- Resolved Motion Rate Control

$$\dot{\mathbf{x}} = J(\theta)\dot{\theta} \quad \Rightarrow$$

$$\dot{\theta} = J(\theta)^{\#}\,\dot{\mathbf{x}}$$

- Properties
  - Only holds for high sampling rates or low Cartesian velocities
  - “a local solution” that may be “globally” inappropriate
  - Problems with singular postures
  - Can be used in two ways:
    - As an instantaneous solutions of “which way to take “
    - As an “batch” iteration method to find the correct configuration at a target

Essential in Resolved Motion Rate Methods: The Jacobian



- Jacobian of direct kinematics:

Analytical Jacobian

$$\mathbf{x} = f(\theta) \quad \Rightarrow$$

$$\frac{\partial \mathbf{x}}{\partial \theta} = \frac{\partial f(\theta)}{\partial \theta} = J(\theta)$$

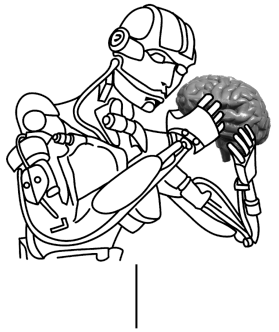
- In general, the Jacobian (for Cartesian positions and orientations) has the following form (geometrical Jacobian):

$$J(\theta) = \begin{pmatrix} j_{P1} & & j_{Pn} \\ & \cdots & \\ j_{O1} & & j_{On} \end{pmatrix}$$

where $$\begin{bmatrix} j_{Pi} \\ j_{Oi} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ \mathbf{0} \end{bmatrix} & \text{for a prismatic joint} \\ \begin{bmatrix} z_{i-1} \times (\mathbf{p} - \mathbf{p}_{i-1}) \\ z_{i-1} \end{bmatrix} & \text{for a revolute joint} \end{cases}$$

$\mathbf{p_i}$ is the vector from the origin of the world coordinate system to the origin of the i-th link coordinate system, p is the vector from the origin to the endeffector end, and **z** is the i-th joint axis (p.72 S&S)

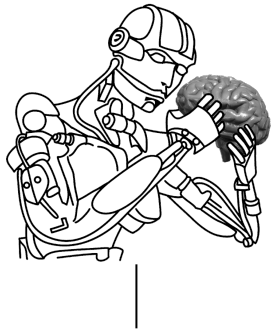
The Jacobian Transpose Method



$$\Delta\theta = \alpha J^T(\theta)\Delta\mathbf{x}$$

- Operating Principle:
  - Project difference vector D**x** on those dimensions **q** which can reduce it the most
- Advantages:
  - Simple computation (numerically robust)
  - No matrix inversions
- Disadvantages:
  - Needs many iterations until convergence in certain configurations (e.g., Jacobian has very small coefficients)
  - Unpredictable joint configurations
  - Non conservative

Jacobian Transpose Derivation



Minimize cost function $F = \frac{1}{2}\left(\mathbf{x}_{target} - \mathbf{x}\right)^T \left(\mathbf{x}_{target} - \mathbf{x}\right)$

$$= \frac{1}{2}\left(\mathbf{x}_{target} - f(\boldsymbol{\theta})\right)^T \left(\mathbf{x}_{target} - f(\boldsymbol{\theta})\right)$$

with respect to $\boldsymbol{\theta}$ by gradient descent:

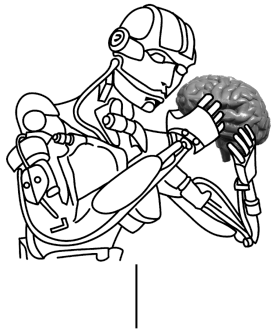
$$\Delta\boldsymbol{\theta} = -\alpha\left(\frac{\partial F}{\partial \boldsymbol{\theta}}\right)^T$$

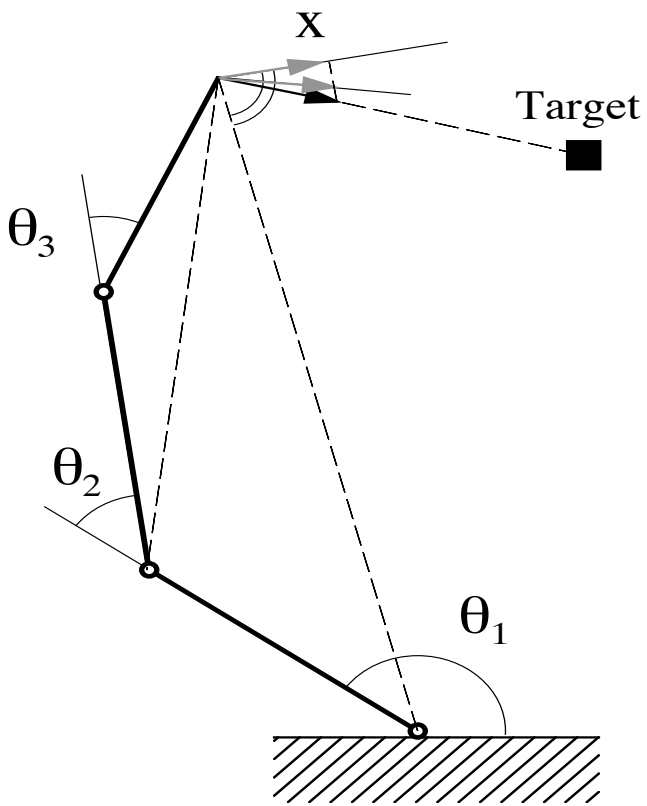
$$= \alpha\left(\left(\mathbf{x}_{target} - \mathbf{x}\right)^T \frac{\partial f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right)^T$$

$$= \alpha J^T(\boldsymbol{\theta})\left(\mathbf{x}_{target} - \mathbf{x}\right)$$

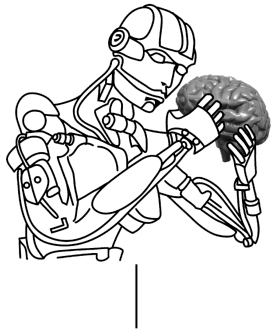
$$= \alpha J^T(\boldsymbol{\theta})\Delta\mathbf{x}$$

Jacobian Transpose Geometric Intuition





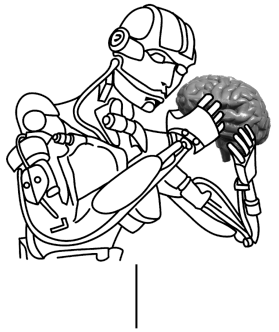
The Pseudo Inverse Method



$$\Delta\boldsymbol{\theta} = \alpha J^T(\boldsymbol{\theta})\left(J(\boldsymbol{\theta})J^T(\boldsymbol{\theta})\right)^{-1}\Delta\mathbf{x} = J^{\#}\Delta\mathbf{x}$$

- Operating Principle:
  - Shortest path in **q**-space
- Advantages:
  - Computationally fast (second order method)
- Disadvantages:
  - Matrix inversion necessary (numerical problems)
  - Unpredictable joint configurations
  - Non conservative

Pseudo Inverse Method Derivation



For a small step $\Delta\mathbf{x}$, minimize with repect to $\Delta\boldsymbol{\theta}$ the cost function:

$$F = \frac{1}{2}\Delta\boldsymbol{\theta}^T\Delta\boldsymbol{\theta} + \boldsymbol{\lambda}^T\left(\Delta\mathbf{x} - J(\boldsymbol{\theta})\Delta\boldsymbol{\theta}\right)$$

where $\boldsymbol{\lambda}^T$ is a vector of Lagrange multipliers.

Solution:

(1) $$\frac{\partial F}{\partial \boldsymbol{\lambda}} = 0 \quad \Rightarrow \quad \Delta\mathbf{x} = J\Delta\boldsymbol{\theta}$$

(2) $$\frac{\partial F}{\partial \Delta\boldsymbol{\theta}} = 0 \quad \Rightarrow \quad \Delta\boldsymbol{\theta} = J^T\boldsymbol{\lambda} \quad \Rightarrow \quad J\Delta\boldsymbol{\theta} = JJ^T\boldsymbol{\lambda}$$

$$\Rightarrow \quad \boldsymbol{\lambda} = \left(JJ^T\right)^{-1} J\Delta\boldsymbol{\theta}$$

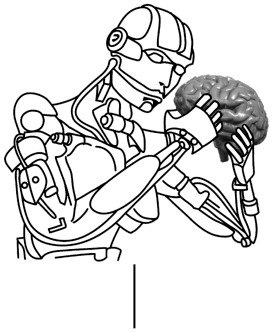
insert (1) into (2):

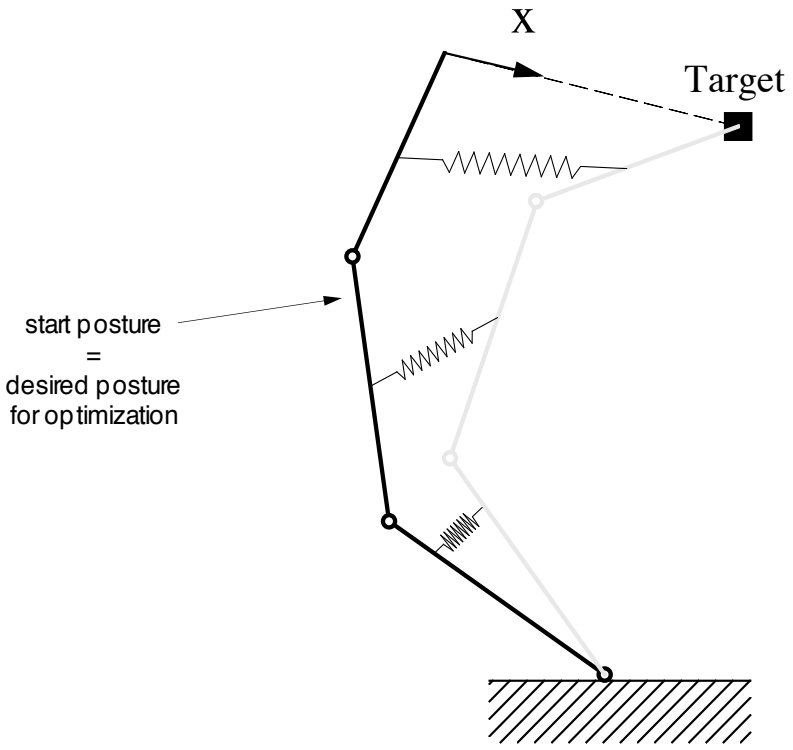
(3) $$\boldsymbol{\lambda} = \left(JJ^T\right)^{-1}\Delta\mathbf{x}$$

insertion of (3) into (2) gives the final result:

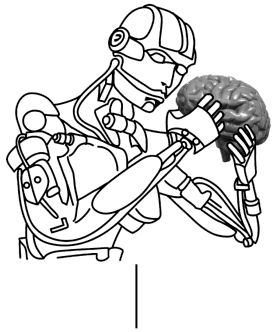
$$\Delta\boldsymbol{\theta} = J^T\boldsymbol{\lambda} = J^T\left(JJ^T\right)^{-1}\Delta\mathbf{x}$$

Pseudo Inverse Geometric Intuition





Pseudo Inverse with explicit Optimization Criterion



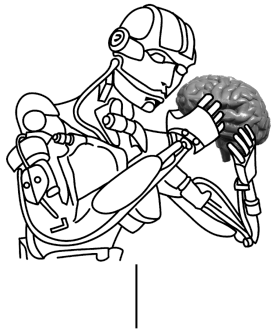
$$\Delta\boldsymbol{\theta} = \alpha J^{\#}\Delta\mathbf{x} + \left(I - J^{\#}J\right)\left(\boldsymbol{\theta}_{\mathrm{O}} - \boldsymbol{\theta}\right)$$

- Operating Principle:
  - Optimization in null-space of Jacobian using a kinematic cost function

$$F = g(\boldsymbol{\theta}), \quad e.g., F = \sum_{i=1}^{d}\left(\theta_i - \theta_{i,0}\right)^2$$

- Advantages:
  - Computationally fast
  - Explicit optimization criterion provides control over arm configurations
- Disadvantages:
  - Numerical problems at singularities
  - Non conservative

Pseudo Inverse Method & Optimization Derivation



For a small step $\Delta\mathbf{x}$, minimize with repect to $\Delta\boldsymbol{\theta}$ the cost function:

$$F = \frac{1}{2}\left(\Delta\boldsymbol{\theta} + \boldsymbol{\theta} - \boldsymbol{\theta}_o\right)^T \left(\Delta\boldsymbol{\theta} + \boldsymbol{\theta} - \boldsymbol{\theta}_o\right) + \boldsymbol{\lambda}^T \left(\Delta\mathbf{x} - J(\boldsymbol{\theta})\Delta\boldsymbol{\theta}\right)$$

where $\boldsymbol{\lambda}^T$ is a vector of Lagrange multipliers.

Solution:

(1) $$\frac{\partial F}{\partial \boldsymbol{\lambda}} = 0 \quad \Rightarrow \quad \Delta\mathbf{x} = J\Delta\boldsymbol{\theta}$$

(2) $$\frac{\partial F}{\partial \Delta\boldsymbol{\theta}} = 0 \quad \Rightarrow \quad \Delta\boldsymbol{\theta} = J^T\boldsymbol{\lambda} - \left(\boldsymbol{\theta} - \boldsymbol{\theta}_o\right) \quad \Rightarrow \quad J\Delta\boldsymbol{\theta} = JJ^T\boldsymbol{\lambda} - J\left(\boldsymbol{\theta} - \boldsymbol{\theta}_o\right)$$

$$\Rightarrow \quad \boldsymbol{\lambda} = \left(JJ^T\right)^{-1} J\Delta\boldsymbol{\theta} + \left(JJ^T\right)^{-1} J\left(\boldsymbol{\theta} - \boldsymbol{\theta}_o\right)$$

insert (1) into (2):

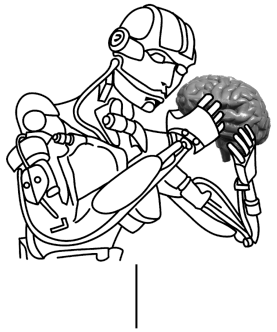
(3) $$\boldsymbol{\lambda} = \left(JJ^T\right)^{-1} \Delta\mathbf{x} + \left(JJ^T\right)^{-1} J\left(\boldsymbol{\theta} - \boldsymbol{\theta}_o\right)$$

insertion of (3) into (2) gives the final result:

$$\Delta\boldsymbol{\theta} = J^T\boldsymbol{\lambda} - \left(\boldsymbol{\theta} - \boldsymbol{\theta}_o\right) = J^T\left(JJ^T\right)^{-1}\Delta\mathbf{x} + J^T\left(JJ^T\right)^{-1} J\left(\boldsymbol{\theta} - \boldsymbol{\theta}_o\right) - \left(\boldsymbol{\theta} - \boldsymbol{\theta}_o\right)$$

$$= J^{\#}\Delta\mathbf{x} + \left(I - J^{\#}J\right)\left(\boldsymbol{\theta}_o - \boldsymbol{\theta}\right)$$

The Extended Jacobian Method



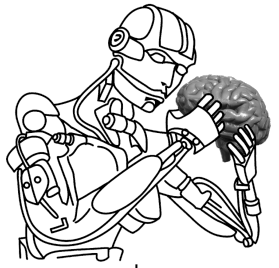
$$\Delta\boldsymbol{\theta} = \alpha\left(J^{ext.}(\boldsymbol{\theta})\right)^{-1}\Delta\mathbf{x}^{ext.}$$

- Operating Principle:
  - Optimization in null-space of Jacobian using a kinematic cost function

$$F = g(\boldsymbol{\theta}), \quad e.g., F = \sum_{i=1}^{d}\left(\theta_i - \theta_{i,0}\right)^2$$

- Advantages:
  - Computationally fast (second order method)
  - Explicit optimization criterion provides control over arm configurations
  - Numerically robust
  - Conservative
- Disadvantages:
  - Computationally expensive matrix inversion necessary (singular value decomposition)
  - Note: new and better ext. Jac. algorithms exist

Extended Jacobian Method Derivation



The forward kinematics $\mathbf{x} = f(\boldsymbol{\theta})$ is a mapping $\Re^n \rightarrow \Re^m$, e.g., from a n-dimensional joint space to a m-dimensional Cartesian space. The singular value decomposition of the Jacobian of this mapping is:

$$J(\boldsymbol{\theta}) = \mathbf{USV}^T$$

The rows $[\mathbf{V}]_i$ whose corresponding entry in the diagonal matrix $\mathbf{S}$ is zero are the vectors which span the Null space of $J(\boldsymbol{\theta})$. There must be (at least) n-m such vectors ($n \geq m$). Denote these vectors $\mathbf{n}_i, i \in [1, n-m]$. The goal of the extended Jacobian method is to augment the rank deficient Jacobian such that it becomes properly invertible. In order to do this, a cost function $F=g(\boldsymbol{\theta})$ has to be defined which is to be minimized with respect to $\boldsymbol{\theta}$ in the Null space. Minimization of F must always yield:

$$\frac{\partial F}{\partial \boldsymbol{\theta}} = \frac{\partial g}{\partial \boldsymbol{\theta}} = 0$$

Since we are only interested in zeroing the gradient in Null space, we project this gradient onto the Null space basis vectors:

$$G_i = \frac{\partial g}{\partial \boldsymbol{\theta}} \mathbf{n}_i$$

If all G_i equal zero, the cost function F is minimized in Null space. Thus we obtain the following set of equations which are to be fulfilled by the inverse kinematics solution:

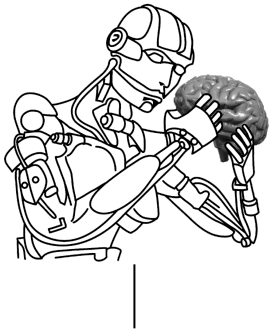
$$\begin{pmatrix} f(\boldsymbol{\theta}) \\ G_1 \\ \dots \\ G_{n-m} \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

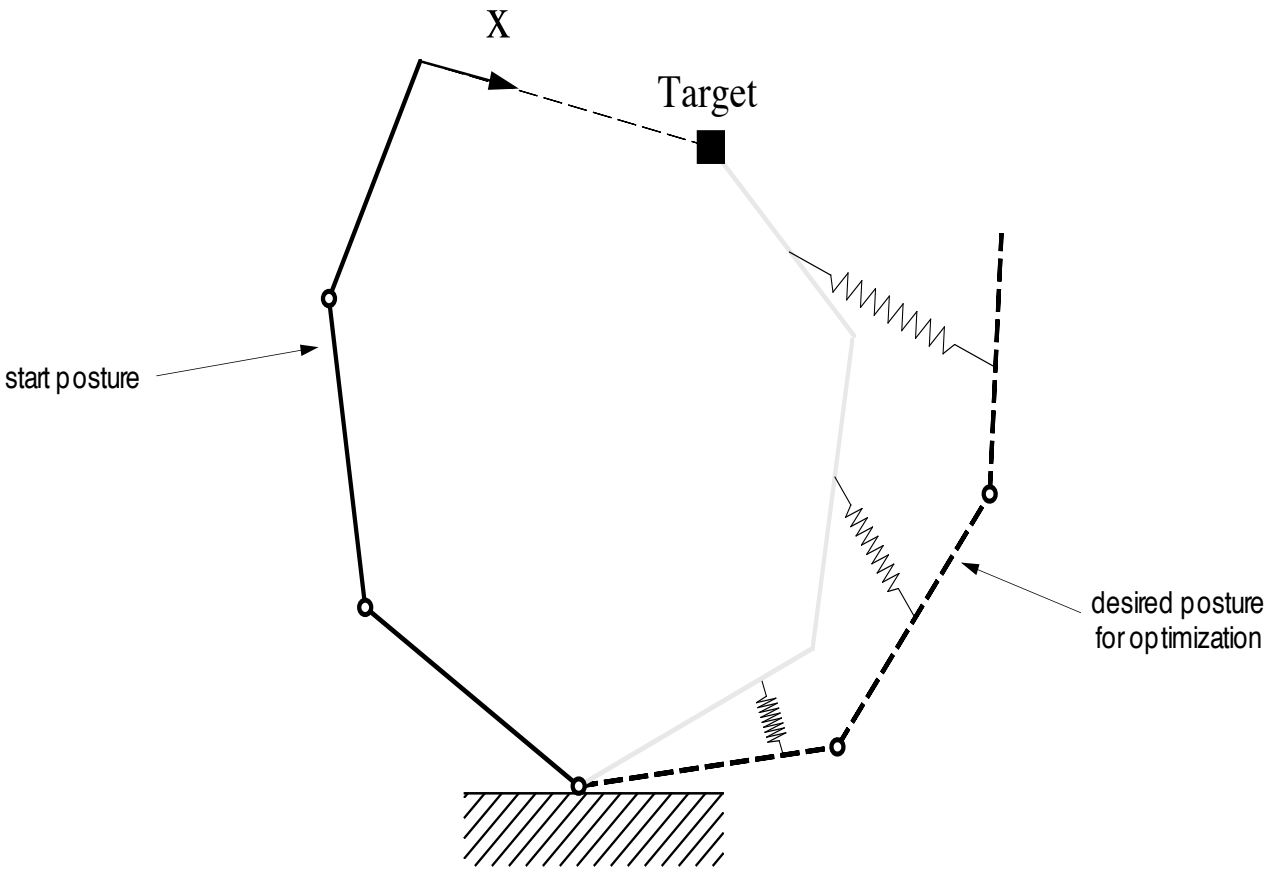
For an incremental step $\Delta\mathbf{x}$, this system can be linearized:

$$\begin{pmatrix} J(\boldsymbol{\theta}) \\ \frac{\partial G_1}{\partial \boldsymbol{\theta}} \\ \dots \\ \frac{\partial G_{n-m}}{\partial \boldsymbol{\theta}} \end{pmatrix} \Delta\boldsymbol{\theta} = \begin{pmatrix} \Delta\mathbf{x} \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad or \quad J^{ext.} \cdot \Delta\boldsymbol{\theta} = \Delta\mathbf{x}^{ext.}.$$

The unique solution of these equations is: $\Delta\boldsymbol{\theta} = \left(J^{ext.}\right)^{-1} \Delta\mathbf{x}^{ext.}$.

Extended Jacobian Geometric Intuition





What makes control hard?

Emo Todorov

Applied Mathematics
Computer Science and Engineering

University of Washington

What makes control hard?

Some of the usual suspects are:

- **non-linearity**
- **high dimensionality**
- **redundancy**
- **noise and uncertainty**

These properties can make an already hard problem harder,
however none of them is a root cause of difficulty.

A control problem can have all these properties and still be easy,
in the sense that there exists a simple strategy that always works:

Push towards the goal!

The problem is hard when this strategy is infeasible, due to **constraints**.

Easy example: Reaching with a redundant arm

joint space configuration q

end effector position $y(q)$

end effector Jacobian $J(q) = \frac{\partial y(q)}{\partial q}$

Jacobian null space $N(q)$

Push hand towards target:

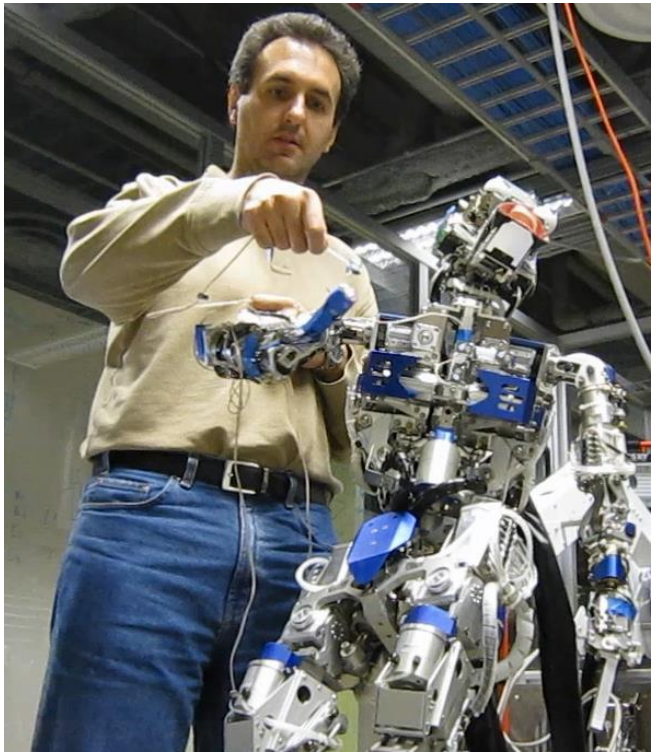
$$u = k\, J(q)^T \left(y^* - y(q)\right)$$

Push hand towards target,
while staying close to default configuration:

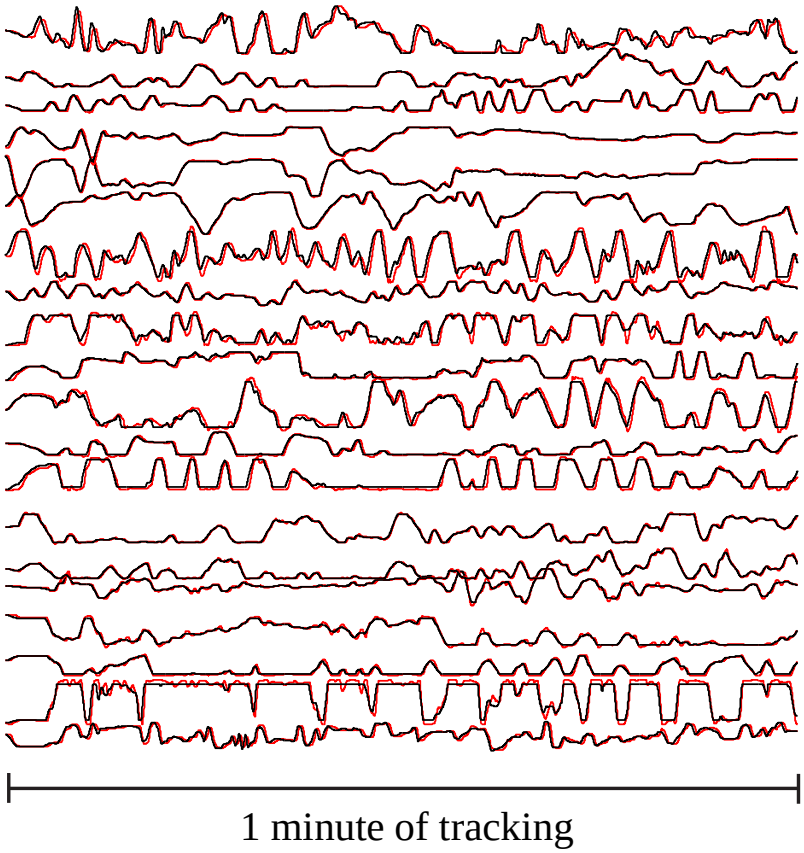
$$u = k_1\, J(q)^T \left(y^* - y(q)\right) + k_2\, N(q)\left(q^* - q\right)$$

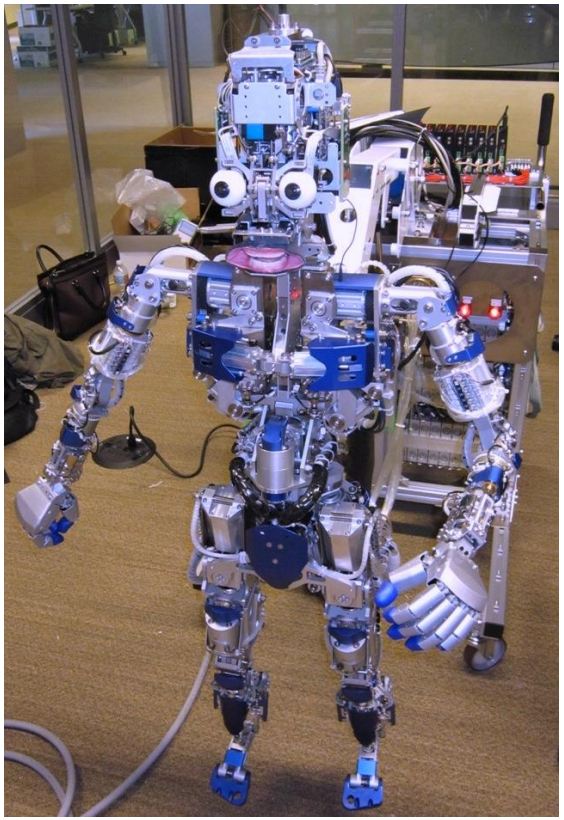
The controller does not need to worry about the path, or the speed profile, or stability, or anything else - it all emerges from the nicely damped dynamics.

Pneumatic robot (Diego-san)
air pressure similar to muscle activation,
but with longer time constant (~ 80 ms)



Easy example: Trajectory tracking with PD control





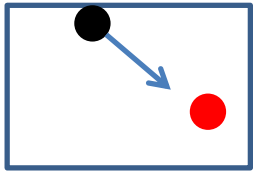
reference trajectory

actual trajectory

Constraints that are (mostly) benign

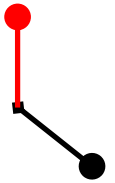
Joint limits

The goal is inside the convex feasible region, so pushing towards the goal will not violate the joint limits.



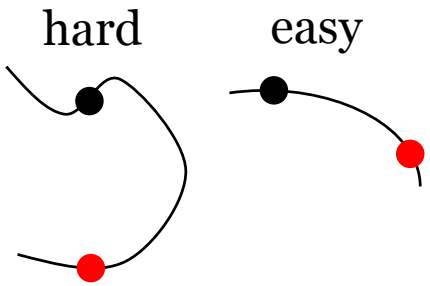
Actuation limits

This is a big problem for under-powered systems, but most robots are sufficiently strong.



Equality constraints

Such constraints restrict the state to a manifold. If the simple push-towards-the-goal action *projected on the manifold* always gets us closer to the goal, then the problem is still easy. Gently curved manifolds are likely to have this property.



Constraints that make the problem hard

Under-actuation

In tasks such as locomotion and object manipulation, some DOFs cannot be controlled directly. These un-actuated DOFs are precisely the ones we would like to control.

Obstacles

Obstacles can turn a control problem into a complicated maze. Solving such problems requires path planning, along with dynamic consistency.

Contact dynamics

Physical contacts change the plant dynamics qualitatively. Making and breaking contacts is usually required for the task, thus the controller has to operate in many dynamic regimes, and handle abrupt transitions.