INVERSE KINEMATICS

RBT350 – Gateway to Robotics

Roberto Martin-Martin

Assistant Professor of Computer Science.

Recap

- Forward kinematics:
  - Computes the pose of the end-effector frame based on the joint configuration (joint angles for each joint)
  - Configuration vs. Workspace
- Jacobian:
  - ~ Matrix of partial derivatives of “forward kinematics”
  - How does a small change in the joints affect the pose of the end-effector
  - Applications of the Jacobian

Inverting the Jacobian

- Often, the Jacobian is not square
  - 7 DoF robot arm
- We will use the pseudo-inverse J^+ obtained from the Moore-Penrose inversion:

$$J^+ = J^T(J \cdot J^T)^{-1}$$

- $J \in R^{m \times n}$ then $J^+ \in R^{n \times m}$
- The pseudoinverse satisfies:
  - $J \cdot J^+ \cdot J = J$
  - $J^+ \cdot J \cdot J^+ = J^+$
  - $(J \cdot J^+)^T = J \cdot J^+$
  - $(J^+ \cdot J)^T = J^+ \cdot J$

Kinematic Singularities

- Configuration in which the robot loses the ability to move in one of the Cartesian dimensions
- What does that mean for the Jacobian?

How does a singularity look like?

- Trying to follow a smooth trajectory in Cartesian space with the end-effector is not possible:
  - Either robot “stops”

  or
  - Robot moves one/several joint(s) at very fast speed

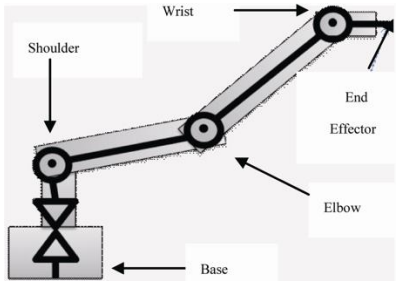
$$\dot{q} = J^{-1\,or\,+}\dot{x}_{ee}$$

What did you learn today?

- Inverse Kinematics
  - What is it?
  - Types of queries/solutions
  - Types of solvers
  - How to compute them

Forward Kinematics

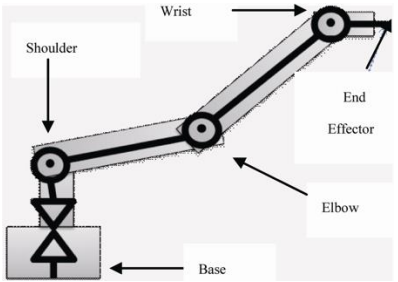
- Given the kinematic model of the robot
  - Links (distance between joints)
  - Joints (type, axis of rotation/translation)
- Given the joint configuration of the robot q = (q1, q2, …)



Forward Kinematics

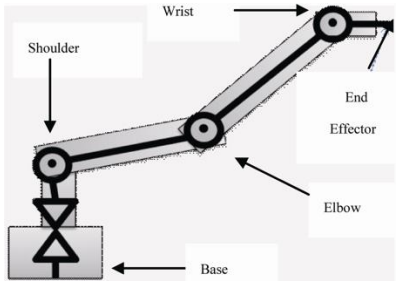
pose of the end-effector frame

$T_{base,ee}(q)$



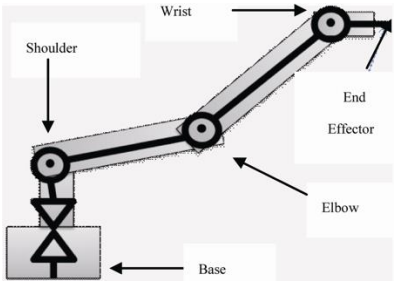
Inverse Kinematics

- Given the kinematic model of the robot
  - Links (distance between joints)
  - Joints (type, axis of rotation/translation)
- Given a desired pose of the end-effector $T_{base,ee}(q)$



Inverse Kinematics

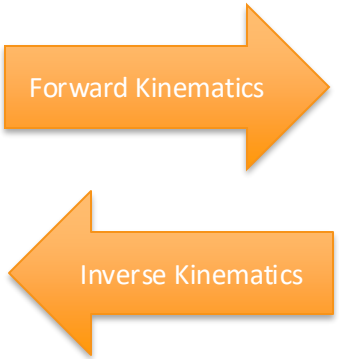
joint configuration of the robot q = (q1, q2, …)



All together

robot configuration (joint values)

$$q = (q_0, \cdots, q_{k-1})$$



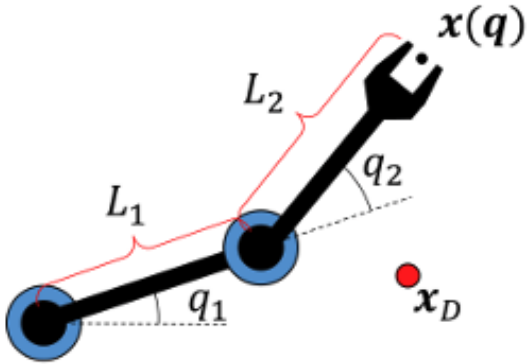
end-effector pose

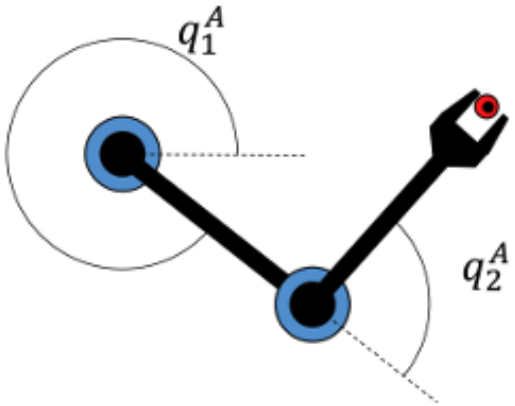
$$T_{base,ee}(q)$$

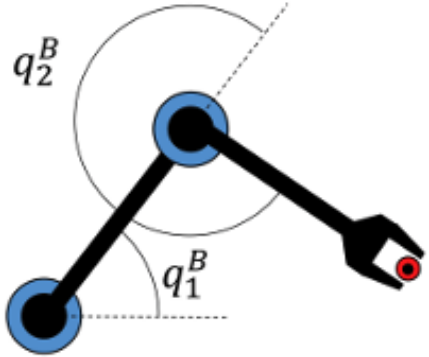
Types of Queries

- One unique solution
- Multiple/infinite solutions
- No solution

Example

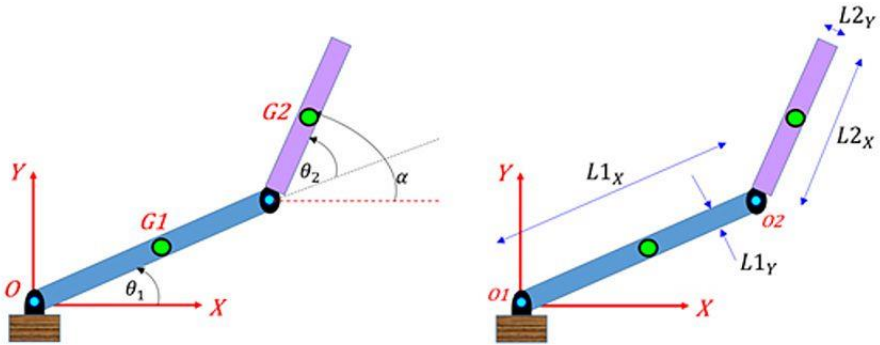




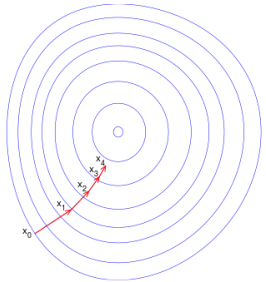


Types of IK Solvers

- Analytical IK Solver
  - Only for simple robots with low number of DoFs
  - Closed form, includes ALL solutions



- Iterative Numerical IK Solver
  - Works with any type of robot
  - Finds only one solution per process (if)



Iterative Numerical IK Solver

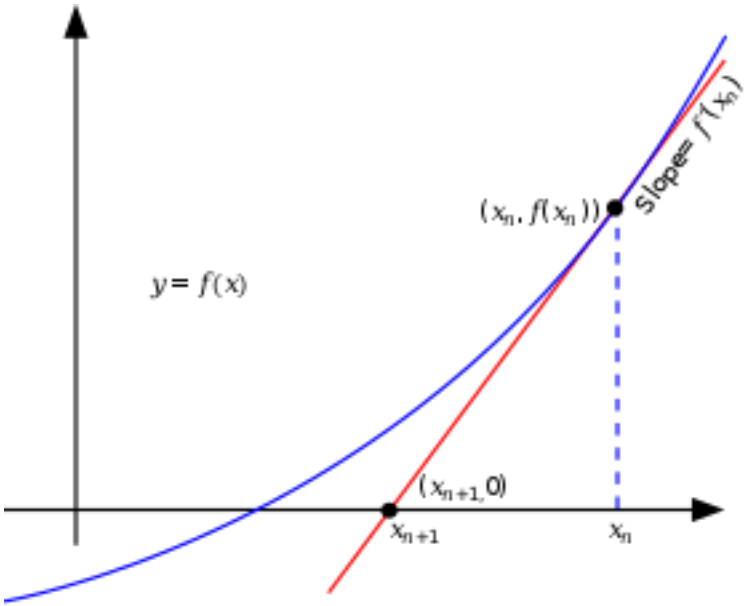
- Also called Newton-Raphson root-finding method
- Main idea similar to other numerical optimization approaches
- Use the derivatives!

Newton-Raphson Method

- Method to find roots of a function f(x)
  - x such that f(x)=0
- Starting at a point x_n, takes the linear approximation of the function f at that point: $f'(x_n)$
- Trigonometry:

$$f'(x_0) = \frac{f(x_n) - 0}{x_n - x_{n+1}} \rightarrow$$

$$\rightarrow x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$



Iterative Numerical IK Solver

- $f(x) = x_d - x = x_d - FK(q) = f(q)$
  - x: end-effector pose
  - $FK(q)$: forward kinematics
  - The "-" operation is a bit more complex → subtraction of poses (but we will ignore for now)
  - We search for roots of that function: values of q that make the function 0
- $f'(\mathrm{q}) = \frac{\delta f}{\delta q} = -\frac{\delta FK(q)}{\delta q} = -\frac{\delta x}{\delta q} = -J$
- $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ →

→ $q_{n+1} = q_n + J^{-1\backslash+}(q_n) f(q_n) = q_n + J^{-1\backslash+}(q_n)(x_d - FK(q_n))$

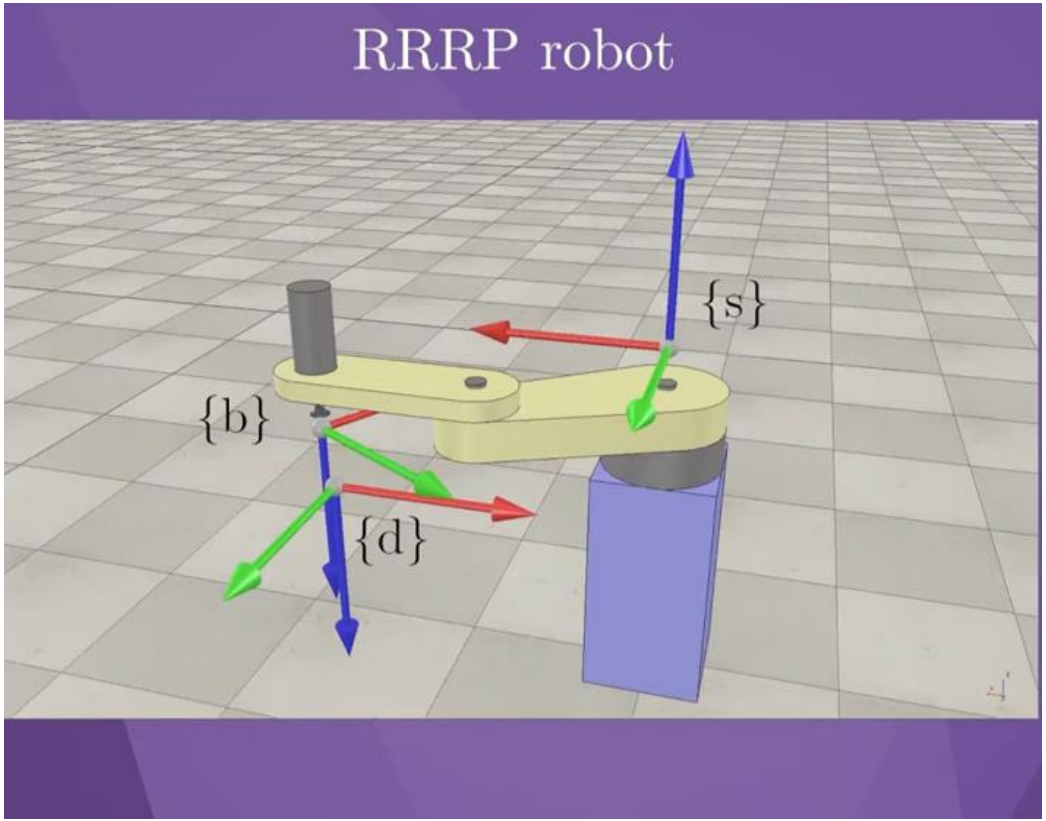
A note about the pseudoinverse

- If the Jacobian is invertible, the pseudoinverse = inverse
- If not, the PM pseudoinverse provides the change in joint configuration with the smallest L2 distance to the initial step
- If there is no solution (for example, the commanded goal is out of reach), using the pseudoinverse we obtain the closest solution possible, that is, the solution that minimizes the "L2" distance between the desired and achieved end-effector pose

The iterative loop

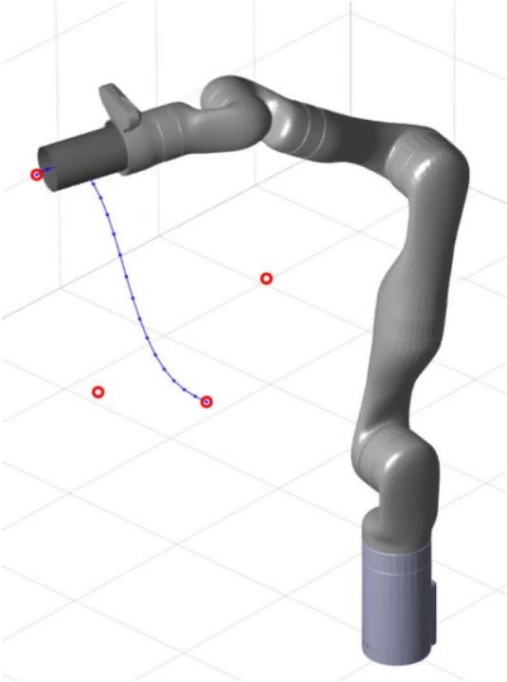
- Start at n=0, with a first approximation of q
- Set $e = x_d - FK(q_n)$, while $\|e\| > \epsilon$:
  - Set $\mathrm{q_{n+1}} = \mathrm{q_n} + J^{-1\backslash+}(q_n)\mathrm{e}$
  - Increment n

Visualization of the steps



Some practical uses of IK

1. We want to query a possible joint configuration for a desired end-effector pose
   - The pose can come from perception (vision)
2. We want to control the robot to move the end-effector a small “delta” in Cartesian space from where it is now
   - Many policy learning algorithms (RL, IL) use this action space
   - We can just use a P(I)D controller in joint space then to move our robot from the current q to the desired new q that corresponds to the new end-effector pose
3. We want to follow a Cartesian path



Problems with IK

- Sensitivity to the initialization
- Not guaranteed to converge
- The method that we studied can lead to huge steps when $J^{-1\backslash+}$ is close to 0
  - Singularity!

https://pollev.com/robertomartinmartin739

Exercise

- Assuming that the end-effector configuration of a 2 DoF robot has only two variables, x and y
- $FK(q) = \begin{pmatrix} q_0^2 \\ q_0 q_1 \end{pmatrix}$
- Reminder:
  - Inverse of a 2x2 matrix of the form $\mathrm{M} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is $\mathrm{M}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$