Numerical Inverse Kinematics

Robotics instructional lab #3

Spring 2022

In this lab, we will introduce the Inverse Kinematics (IK) problem. Given a desired position of the end-effector, what are required values of the joints. We will implement Newton-Raphson algorithm to solve IK. This method has the advantage of working for any type of manipulator, although like our previous iterative method, its performance depends on the quality of our initial guess.

1 Background

The idea behind Newton-Raphson algorithm is not to figure out the joint positions required to reach a pose directly, but rather to figure out a joint displacement that will get us closer to the desired pose and then repeat the process until we arrive at our destination.

Let $x = f(\theta)$ be the forward kinematics of the arm where $x \in \mathbb{R}^3$ and $\theta \in \mathbb{R}^n$ are the end-effector **position** and configuration of the arm, respectively. Assuming f is differentiable and with x_d the desired end-effector position, we solve the **Newton-Raphson**[1] to minimize $g(\theta) = x_d - f(\theta)$. That is, we search for joint coordinates θ^* such that

$$g(\theta^*) = x_d - f(\theta^*) = 0.$$

Given the linear Jacobian matrix $J_L \in \mathbb{R}^{3\times n}$ of the arm and using the Taylor expansion, we can solve for $\Delta\theta$

$$\Delta\theta = J_L^{\dagger}(\theta_0)(x_d - \theta_0) \tag{1}$$

where $J_L^{\dagger}$ is the pseudo-inverse of J_L. With this, the new configuration is updated according to $\theta_1 = \theta_0 + \Delta\theta$. The full algorithm to solve this is:

1. **Start:** Initialize guess θ_0 and set $i = 0$.
2. While $\|e\| > \epsilon$:
    - $e = x_d - f(\theta_i)$.
    - $\theta_{i+1} = \theta_i + \lambda J_L^{\dagger}(\theta_i)e$
    - $i = i + 1$

[1] Based on *Kevin M. Lynch and Frank C. Park, Modern Robotics: Mechanics, Planning, and Control, Cambridge University Press.*

- Return θ_i.

Scalar ϵ is a small user-defined accuracy tolerance. λ is a step size value used to control the rate of convergence and is user defined. If there are multiple IK solutions, the iterative process tends to converge to the closest one.

2 Prerequisites

1. Write a function `Jacobian(`θ`)` that receives the joint configuration vector θ and outputs the full Jacobian (linear and angular) J of the arm.
2. Write a function `LinearJacobian(`θ`)` that receives the joint configuration vector θ and outputs the linear Jacobian J_l of the arm.
3. Write a function `IK_NR_position(`θ_0, x_d`)` that receives initial guess θ_0 and **desired end-effector position** x_d, and outputs the joint configuration vector θ^* for the robot to reach x_d.

Prerequisites are a mandatory in order to carry out the lab .

3 Lab instructions

1. Using the GUI from Lab 1, Repeat the following 15 times:
   - Manually move the robot to a random configuration.
   - Write down the joint angles and end-effector pose $x_i, y_i, z_i, \theta_{x_i}, \theta_{y_i}\theta_{z_i}$.
   - Compute the transformation matrix T_i.
2. For each recorded sample i:
   - Solve the IK problem for $x_d = x_i$ with `IK_NR_position` to acquire joint angles $\theta_{IK,i}$. Choose random θ_0.
   - Move the robot to $\theta_{IK,i}$ and write down the resulted end-effector transformation matrix $\tilde{T}_i$.
   - Repeat for a different choice of θ_0.

For the above, fine tune λ and ϵ to achieve best results. If the algorithm fails to converge, try to choose an initial guess closer to the real joint configuration.

4 Report requirements

The lab report should include the following:

1. Describe the process of the lab.

2. Include the following:
   - A table summarizing all results.
   - Mean and standard deviation of the position error for all trials.
   - Explain the reason for the difference in end-effector poses for different choices of initial angles.
   - Explain the reasons for the errors and how could they be improved.
3. Explain the advantages and disadvantages of this IK method compared to the analytical one. Can an analytical IK solution be obtained?
4. Provide a summary for your results.