Inverse Kinematics with Trajectory Planning

This project will expand upon your program to simulate and visualize robotic manipulators. So far, you program should compute the direct/forward (position and differential) kinematics, and display a manipulator in three dimensions with DH frames and end-effector velocity. The ability to generate trajectories and compute inverse kinematics will be added to your robot simulator and visualizer. A different approach will be taken for each of two robotic manipulators.

1. Incorporate the following items into a complete simulation and visualization of the Adept Cobra s800 SCARA Robot:

   (a) compute the inverse kinematics (let's choose the left-elbow configurations) of the robot for the desired initial pose T_{ei} and final pose T_{ef} given below

$$T_{ei} = \begin{bmatrix} 1 & 0 & 0 & 500 \\ 0 & -1 & 0 & -500 \\ 0 & 0 & -1 & 350 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_{ef} = \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & -250 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 500 \\ 0 & 0 & -1 & 50 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   (b) connect the initial and final joint variables with Linear-Segment Parabolic-Blends using a start time of $t_o = 0$ sec, an end time of $t_f = 4$ sec, and velocities $V = 1.2\frac{q_f - q_o}{t_f}$

   (c) assume you have perfect joint controllers such that your joint variables track your desired values in the trajectory, and simulate (compute forward position and differential kinematics) and visualize your robot for these joint trajectories

   (d) turn in the following along with showing me your visualization

      i. initial and final values of joint variables that correspond to initial and final poses

      ii. plot of desired trajectories for joint variables and velocities

      iii. plot of end-effector's position, orientation (as angle-axis), linear velocity and angular velocity

      iv. some snap shots of your animation (maybe a start-middle-end or bread-crumb trail)

2. Incorporate the following items into a complete simulation of the Stanford Arm:

   (a) connect the desired initial and final poses of the robot given below with quintic polynomials (start and end at rest), a start time of $t_o = 0$ sec and an end time of $t_f = 4$ sec making use of angle-axis for orientation; note you'll need to generate desired trajectories for the end-effector's position $\vec{p}_{de}$, orientation $\vec{\phi}_{de}$ as angle-axis, linear velocity $\dot{\vec{p}}_{de}$ and angular velocity $\vec{\omega}_{de}$

$$T_{ei} = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & -2 \\ -1 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad T_{ef} = \begin{bmatrix} \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & -2 \\ \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 2 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   (b) compute the inverse kinematics (let's choose the left-arm configuration) of the robot for the desired initial pose T_{ei}; this will serve as the initial values for the joint variables

   (c) iteratively compute the inverse kinematics via the method of damped-least-squares with numerical integration (you can get fancy or use Euler as shown below)

$$\begin{aligned} \dot{\vec{q}}_d &= (W + J^T J)^{-1} J^T (\vec{v}_{de} + K(\vec{x}_d - \vec{x})) \\ \vec{q}_d &= \vec{q}_d + \dot{\vec{q}}_d \Delta t \end{aligned}$$

   where we'll start with $\Delta t = 0.01$ sec, $W = \text{diag}(0.01)$, $K = \text{diag}(1)$

   (d) assume you have perfect joint controllers such that your joint variables track your desired values in the trajectory, and simulate (compute forward position and differential kinematics) and visualize your robot for these joint trajectories

   (e) turn in the following along with showing me your visualization

      i. plot of end-effector's desired position, orientation, linear velocity and angular velocity

      ii. plot of desired trajectories for joint variables and velocities

      iii. plot of end-effector's position, orientation (as angle-axis), linear velocity and angular velocity

      iv. some snap shots of your animation (maybe a start-middle-end or bread-crumb trail)