AME 50551: Introduction to Robotics

Project 1: Numerical Inverse Kinematics and the MATLAB Robotics Systems Toolbox

Upload your project report PDF and MATLAB code to Gradescope by Friday, March 27th, 11:59 PM

Objective: After completing the project you should be able to:

(a) Code a backtracking line-search to select step sizes in iterative numerical IK.

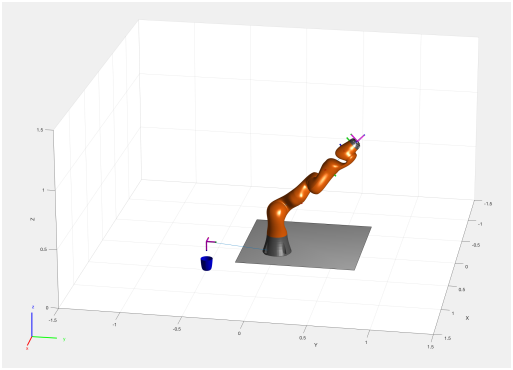
(b) Solve IK problems numerically using iterative approaches.

(c) Approximately solve IK problems when the desired end-effector pose is not in the workspace of the manipulator.

This project requires the Robotics Systems Toolbox in MATLAB R2017b. If you do not have this version (or later) on your computer, you should install it (and the Toolbox) before getting started. You may work in pairs for this project. Each project team should hand in a report that includes deliverables specified in this document. In addition to the listed deliverables, **include a copy of your final code**.

Part 1: (IK for a Single Target Pose)

- Download the Project 1 starter code from Sakai.
- Run the starter code. You should see a visualization like the one shown below.



- Uncomment the function `computeError`. Read the documentation to understand the inputs and outputs. After reading, fill in the blanks as identified by `???` in the code. The goal of this function is to compute a 6×1 vector for the error between the desired end-effector pose (position and orientation) and the actual end-effector pose. The error vector should be formed as follows: the 3×1 angle-axis orientation error goes on top, and the 3×1 position error goes on the bottom. All quantities should be expressed with respect to the base frame.
- Next, uncomment the function `IK_step`. Again, read the documentation for the function and the fill in the blanks. The goal of this function is to update the joint angles so that the gripper pose approaches the desired pose. The magnitude of the error should decrease from this update operation.
- Finally uncomment the function `IK`. Again, read the documentation and fill in the blanks. Once you are done, run `[theta, iteration_errors] = IK(T_desired, theta_0, lbr, gripper, 1);` If all goes correctly, the manipulator should iterate toward the goal and converge in 10-20 iterations.

Deliverables:

1. A plot of the error norm versus iteration number for your iterative numerical IK solution. (10 points)
2. A screenshot showing the manipulator in the solution pose from iterative IK. (10 points)
3. Code (35 points)

Part 2: (IK for a Target Trajectory)

- Consider a modified IK problem with the same desired orientation as before, but with a desired position along the edge of the cup.

```
p_desired = cupPosition + [-cupRadius*cos(polar_angle_for_cup), ...
                           -cupRadius*sin(polar_angle_for_cup), ...
                           .2];
```

- Write a `for` loop that solves the IK problem for 100 evenly spaced values of `polar_angle_for_cup` in the interval $[0, 2\pi]$. As you move around the circle, use each solution as an initial guess for the next. Use the command `show(lbr, theta , 'PreservePlot', false); drawnow; pause(0.05);` to show the robot in the solution configuration after each solution.

Deliverables:

1. A plot of IK solution joint angles versus `polar_angle_for_cup`. All joint angles should be reported on the interval $[-\pi, \pi]$. (*Hint: use* `wrapToPi`.) (10 points)
2. How does the performance change if you use the same initial condition for all IK solves? What is the effect on the joint angle trajectories and the overall time it takes to solve the IK problems? Provide an explanation for your findings. (5 points)
3. Code (10 points)

Part 3: (Approximate IK for Poses Beyond the Workspace)

- Modify the radius of the cup to `cupRadius = 0.35;`. With this parameter setting, the far edge of your previous trajectory should no longer be within the workspace of the manipulator.
- Re-run your code from Part 2. Describe what problem arises.
- Propose a modification to your IK routine that allows the manipulator to solve the IK problem as closely as possible for poses beyond the workspace. For the purposes of this project, you can consider the IK problem solved as closely as possible if the `IK_step` function fails to reduce the error by more than 10^{-6}.

Deliverables:

1. Answer: What problem originally occurs when the desired pose is beyond the workspace? (5 points)
2. A plot of solution joint angles versus `polar_angle_for_cup` with parameters `cupRadius = 0.35;` and `cupPosition = [0.5, -0.5, .1];`. Joint angles should be reported on the interval $[-\pi, \pi]$. (5 points)
3. A parametric plot of the x and y positions of the end effector along the solution trajectory. (5 points)
4. Code (5 points)