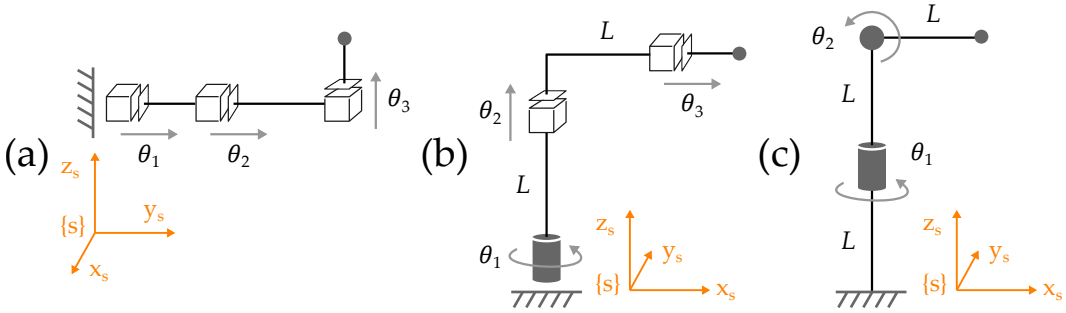
Problem Set 5

Robotics & Automation
Dylan Losey, Virginia Tech

Instructions. Please write legibly and do not attempt to fit your work into the smallest space possible. It is important to show all work, but basic arithmetic can be omitted. You are encouraged to use Matlab when possible to avoid hand calculations, but print and submit your commented code for non-trivial calculations. You can attach a pdf of your code to the homework, use live scripts or the publish feature in Matlab, or include a snapshot of your code. Do not submit .m files — we will not open or grade these files.

For this assignment we are asking you to also submit **videos** of your simulations. Follow the instructions to **label** these videos based on the problem number, and then submit them all within a **single zipped folder**.

1 Analytical Inverse Kinematics



You are considering purchasing an industrial robot arm. Before you buy, you want to find the inverse kinematics of the available options. For each of the robots shown above find equations for θ in terms of $p = [x, y, z]^T$, the position of the end-effector in the world frame.

1.1 (5 points)

Find the inverse kinematics of robot (a).

If you are unsure about where to start, try writing the position of the end-effector $p = [x, y, z]^T$ in terms of joint position θ:

$$y = \theta_1 + \theta_2 \tag{1}$$

$$z = \theta_3 \tag{2}$$

Then solve these equations for θ. Remember to write your answer in terms of x, y, and z as far as possible. Here we cannot disentangle the first two prismatic joints:

$$\theta_1 + \theta_2 = y \tag{3}$$

$$\theta_3 = z \tag{4}$$

1.2 (10 points)

Find the inverse kinematics of robot (b).

Write θ in terms of x, y, and z. Use atan2$(\cdot)$ for the revolute joint.

$$\theta_1 = \text{atan2}(y, x) \tag{5}$$

$$\theta_2 = z - L \tag{6}$$

For the second prismatic joint, use the Pythagorean theorem to get the total radius of the horizontal arm, and then subtract L:

$$\theta_3 = \sqrt{x^2 + y^2} - L \tag{7}$$

1.3 (15 points)

Find the inverse kinematics of robot (c).

The first revolute joint is the same as in part (b):

$$\theta_1 = \text{atan2}(y, x) \tag{8}$$

The second revolute joint is another instance of atan2$(\cdot)$, but now in a different plane:

$$\theta_2 = \text{atan2}\left(z - 2L, \sqrt{x^2 + y^2}\right) \tag{9}$$

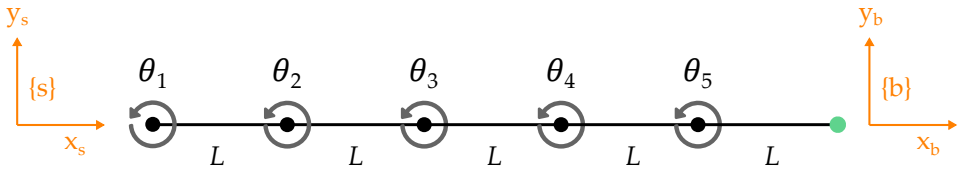
Here the $2L$ comes from the vertical offset between joint two and the ground, and $\sqrt{x^2 + y^2}$ is the radius of the final link projected into the (x, y) plane.

1.4 (5 points)

Imagine that the end-effector of robot (a) is at position $p = [0, 2L, 0]^T$. How many inverse kinematics solutions are there at this position?

There are an **infinite** number of solutions. Since $z = 0$, we know that θ_3 must equal 0. But every combination of $\theta_1 + \theta_2$ that add up to $2L$ is a valid solution!

2 Numerical Inverse Kinematics



You are building a snake robot. This snake robot moves in a plane and has 5 joints, making it a redundant robot. You are using this redundancy to mimic the motion of real snakes.

2.1 (25 points)

Implement the numerical inverse kinematics algorithm. Leave $b = 0$ within the Jacobian pseudoinverse. Using your code, find the inverse kinematics solutions when:

Case 1 $L = 1$ and the desired end-effector pose is:

$$T_{sb} = \begin{bmatrix} rotz(\pi/4) & \begin{bmatrix} 3 \\ 2 \\ 0 \end{bmatrix} \\ 0 & 1 \end{bmatrix} \tag{10}$$

Case 2 $L = 1$ and the desired end-effector pose is:

$$T_{sb} = \begin{bmatrix} rotz(\pi/2) & \begin{bmatrix} -2 \\ 4 \\ 0 \end{bmatrix} \\ 0 & 1 \end{bmatrix} \tag{11}$$

Case 3 $L = 1$ and the desired end-effector pose is:

$$T_{sb} = \begin{bmatrix} rotz(0) & \begin{bmatrix} 3 \\ -1 \\ 0 \end{bmatrix} \\ 0 & 1 \end{bmatrix} \tag{12}$$

For each of these cases let your initial joint position be $\theta_0 = [\pi/8, \pi/8, \pi/8, \pi/8, \pi/8]^T$.

My code for solving this problem is shown in Figures 1, 2, 3, and 4. Your code should provide answers close to the ones listed below:

- **Case 1**: $\theta = \begin{bmatrix} -0.61 & 0.60 & 1.10 & 0.48 & -0.77 \end{bmatrix}^T$
- **Case 2**: $\theta = \begin{bmatrix} 1.38 & 0.84 & 0.33 & -0.12 & -0.85 \end{bmatrix}^T$
- **Case 3**: $\theta = \begin{bmatrix} -1.51 & 0.29 & 1.36 & 0.78 & -0.92 \end{bmatrix}^T$

```
close all
clear
clc

% create figure
figure
axis([-6, 6, -6, 6])
grid on
hold on

% save as a video file
v = VideoWriter('problem3_1.mp4', 'MPEG-4');
v.FrameRate = 25;
open(v);

```

Figure 1: Clear variables and initialize the video file.

3 Simulating the Robot

This problem uses the same redundant snake robot as in Problem 2. Here you will make videos of the robot to visualize your inverse kinematics solutions. Start by **downloading** the Matlab file `make_video.m` that was provided with this assignment.

```
16      % initial conditions
17 -    L = 1;
18 -    theta = [pi/8;pi/8;pi/8;pi/8;pi/8];
19
20      % screws and home transformation matrix
21 -    S1 = [0 0 1 0 0 0]';
22 -    S2 = [0 0 1 0 -1*L 0]';
23 -    S3 = [0 0 1 0 -2*L 0]';
24 -    S4 = [0 0 1 0 -3*L 0]';
25 -    S5 = [0 0 1 0 -4*L 0]';
26 -    S = [S1, S2, S3, S4, S5];
27 -    M = [eye(3), [5*L;0;0]; 0 0 0 1];
28      % for plotting purposes:
29      % home transformation matix for each link
30 -    M1 = [eye(3), [L;0;0]; 0 0 0 1];
31 -    M2 = [eye(3), [2*L;0;0]; 0 0 0 1];
32 -    M3 = [eye(3), [3*L;0;0]; 0 0 0 1];
33 -    M4 = [eye(3), [4*L;0;0]; 0 0 0 1];
34
```

Figure 2: Set the initial joint position and write the screws for each joint. Here I include the home transformation matrix to the end of each link, which will be useful for plotting.

```
35      % desired end-effector pose
36 -    Td = [rotz(pi/4), [3;2;0]; 0 0 0 1];
37 -    Xd = [R2axisangle(Td(1:3, 1:3)); Td(1:3, 4)];
38
39      % initial end-effector pose
40 -    T = fk(M, S, theta);
41 -    X = [R2axisangle(T(1:3, 1:3)); T(1:3, 4)];
42
43
```

Figure 3: Set the desired transformation matrix and get the initial end-effector pose. The function `R2axisangle` is included in our practice problems for numerical inverse kinematics.

3.1 (25 points)

Combine `make_video.m` with your numerical inverse kinematics code. You will need to get the (x, y) position of each link (in addition to the end-effector position) to plot the snake robot (see example at top of next page). Turn in the following MP4 videos:

- Video for **Case 1** in Problem 2. Title this video **Problem3_1.mp4**
- Video for **Case 2** in Problem 2. Title this video **Problem3_2.mp4**
- Video for **Case 3** in Problem 2. Title this video **Problem3_3.mp4**

Please adjust the video frame rate (`v.FrameRate = your_rate_here;`) so that each video is a few seconds long (more than 3 secs, less than 10 secs).

> See Figures 1, 2, 3, and 4 for the simulation code. Videos of each **Case** are uploaded in a separate folder in solutions. Your videos should match the uploaded versions, give or take some movement speed and final precision.

```
while norm(Xd - X) > 1e-2
    % plot the robot
    % 1. get the position of each link
    p0 = [0; 0];
    T1 = fk(M1, S1, theta(1));
    T2 = fk(M2, [S1, S2], [theta(1), theta(2)]);
    T3 = fk(M3, [S1, S2, S3], [theta(1), theta(2), theta(3)]);
    T4 = fk(M4, [S1, S2, S3, S4], [theta(1), theta(2), theta(3), theta(4)]);
    P = [p0, T1(1:2,4), T2(1:2,4), T3(1:2,4), T4(1:2,4), T(1:2,4)];
    % 2. draw the robot and save the frame
    cla;
    plot(P(1,:), P(2,:), 'o-', 'color',[1, 0.5, 0],'linewidth',4)
    drawnow
    frame = getframe(gcf);
    writeVideo(v,frame);
    % calculate the jacobian
    Js = JacobianSpace(S, theta);
    Jb = Adjoint(inv(T)) * Js;
    J = [T(1:3,1:3) zeros(3); zeros(3) T(1:3,1:3)] * Jb;
    % numerical inverse kinematics
    V = Xd - X;
    delta_theta = pinv(J) * V;
    theta = theta + 0.1 * delta_theta;
    T = fk(M, S, theta);
    X = [R2axisangle(T(1:3, 1:3)); T(1:3, 4)];
end
close(v);
close all
```

Figure 4: Main loop. At each iteration we first plot the robot and save a video frame. Then we calculate the Jacobian and perform numerical inverse kinematics. The loop terminates when the actual pose is close to the desired pose.

4 Jacobian Pseudoinverse and Redundancy

This problem continues exploring the redundant snake robot introduced in Problem 2. So far we have left $b = 0$ in our Jacobian pseudoinverse. More generally, choosing b allows us to set a **secondary objective** for the inverse kinematics of redundant robots. Recall that numerical inverse kinematics finds a solution for θ such that $T_{sb}(\theta)$ equals the desired end-effector pose. But when working with redundant robots, multiple solutions are often possible. Choosing b affects which of these solutions the algorithm selects.

4.1 (15 points)

Set b as the following vector (and update b as θ_1 changes):

$$b = \begin{bmatrix} -\theta_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{13}$$

This choice of b **minimizes the first joint angle**. In other words, the robot will look for an inverse kinematics solution where $|\theta_1| \to 0$. Show that this actually works:

- Make a video for **Case 3** in Problem 2 with b chosen as Equation 13. Title this video **Problem4.mp4**
- Compare the final joint position θ for **Case 3** when $b = 0$ and when b is Equation 13. Show that $|\theta_1|$ is smaller with the secondary objective.

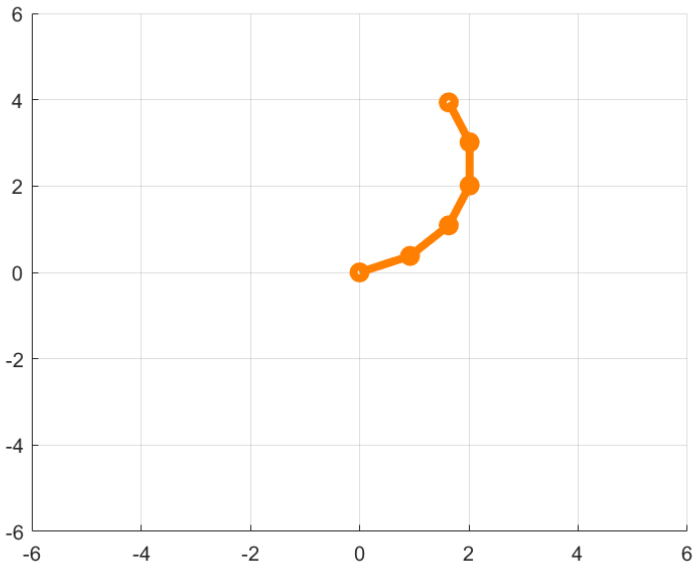


Figure 5: Plot of the snake robot in its initial position $\theta = [\pi/8, \pi/8, \pi/8, \pi/8, \pi/8]^T$

- State one reason why you might want to minimize the first joint angle (or any joint angle). What is a practical benefit?

First we need to implement the secondary objective by selecting the vector b. Modify line 65 of the code in Figure 4 to the following:

```
delta_theta = pinv(J)*V + (eye(5)-pinv(J)*J)*[-theta(1);0;0;0;0];
```

Then run the code and save the video. You can find an example of **Problem4.mp4** in the solution videos folder. Your final joint positions θ should be close to:

- **Case 3 with** $b = 0$: $\theta = \begin{bmatrix} -1.51 & 0.29 & 1.36 & 0.78 & -0.92 \end{bmatrix}^T$
- **Case 3 with** $b = -\theta_1$: $\theta = \begin{bmatrix} -0.38 & -1.40 & 1.20 & 1.70 & -1.12 \end{bmatrix}^T$

Comparing these two results, we have that $|\theta_1|$ is smaller (and θ_1 is closer to zero) with the secondary objective:

$$0.38 < 1.51 \tag{14}$$

There are several reasons why we may want to minimize a joint angle:

- The actuator at that joint moves more slowly than the other actuators.
- Moving the actuator at that joint consumes more power as compared to the other actuators along the robot arm.
- We want to avoid colliding with an obstacle, and we need to keep one or more joints at a specific angle to avoid that obstacle.