Assignment 4

Name:
UNCC Email:

Submission Format: Submit your solution as a single **.zip** file containing (1) a single PDF file and (2) the source code files for the implementation portions of the assignment. Your source code files should run without having to download any third-party package and should be able to reproduce all figures and results reported in your PDF file.

List of Collaborators and Acknowledgements: List the names of all people you have collaborated with and for which question(s).

Questions about this assignment should be directed to TA Dushyant Pawar, dpawar4@charlotte.edu

1. **(10pts)** Write a program that solves the analytical inverse kinematics for a planar 3R robot with link lengths $L_1 = 3$, $L_2 = 2$, and $L_3 = 1$, given the desired position (x, y) and orientation θ of a frame fixed to the tip of the robot. Each joint has no joint limits. Your program should find all the solutions (how many are there in the general case?), give the joint angles for each, and draw the robot in these configurations. Test the code for the case of $(x, y, \theta) = (4, 2, 0)$.

2. **(5pts)** You are asked to draw a plot of a scalar $x_d - f(\theta)$ versus a scalar θ (similar to Figure 6.7 in the book) with two roots. Draw it so that, for some initial guess θ^0, the iterative process actually jumps over the closest root and eventually converges to the further root. Hand-draw the plot and show the iteration process that results in convergence to the further root. Comment on the basins of attraction of the two roots in your plot.

3. **(10pts)** Implement an A star path planner for a point robot in a plane with obstacles defined by yourself. The robot is assumed to have no motion constraints (i.e. robot is able to move from one node to the other neighboring node on the graph in one step). The planar region is a 200×200 area. Your program should first generate a graph using Probabilistic Roadmap (PRM) consisting of N randomly sampled nodes in the free C-space and E edges, where the value of N is chosen by the user and E is determined after PRM construction. Note that you may need to use a large number N so that the constructed graph is large enough to notice effects. The cost associated with each edge is the Euclidean distance between the nodes. Finally, the program should display the graph, search the graph using A star for the shortest path between nodes 1 and N, and display the shortest path or indicate FAILURE if no path exists. The heuristic cost-to-go is the Euclidean distance to the goal.

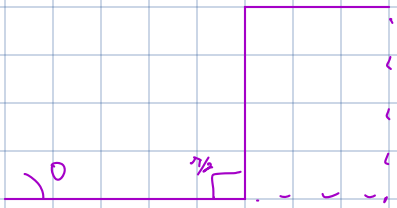
4. **(15pts)** Consider the kinematics of the 7R WAM robot given in Assignment 2, with a detailed description on the Barrett website: https://web.barrett.com/files/B2576_RevAC-00.pdf and Example 4.7 (Page 148, Modern Robotics textbook). We will use the joint conventions from this document. The arm is shown in its all-zeros configuration, and the red arrows show the positive directions for each joint J1-J7. Previously, we have developed the forward kinematics map x = FK(θ), which maps from the joint configuration of the robot θ ∈ CR to the pose of the end-effector x ∈ SE(3). Whereas the FK map is unique and easy to compute analytically, the inverse kinematics map

$$\theta = IK(x)$$

file name

intelroboticshw4p1.m

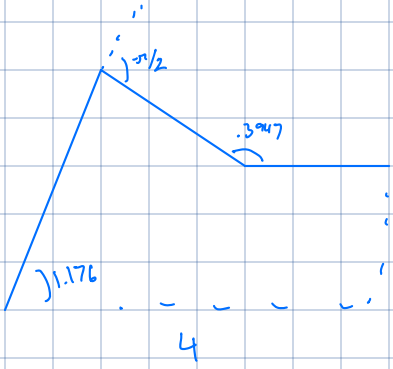
1) desired $(x, y, \theta) = (4, 2, 0)$



$\theta_1 = 0$

$\theta_2 = \pi/2$

$\theta_3 = -\pi/2$



$\theta_1 = 1.176$

$\theta_2 = -\pi/2$

$\theta_3 = .3947$

$L_1\cos(\theta_1) + L_2(\cos(\theta_1+\theta_2) + L_3\cos(\theta_1+\theta_2+\theta_3)$

$L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) + L_3\sin(\theta_1+\theta_2+\theta_3)$

$3 + 0 + 1 = 4$

$0 + 2 + 0 = 2$

$L_1\cos(\theta_1) + L_2(\cos(\theta_1+\theta_2) + L_3\cos(\theta_1+\theta_2+\theta_3)$

$L_1\sin(\theta_1) + L_2\sin(\theta_1+\theta_2) + L_3\sin(\theta_1+\theta_2+\theta_3)$

$1.1538 + 1.8461 + 1 = 4$

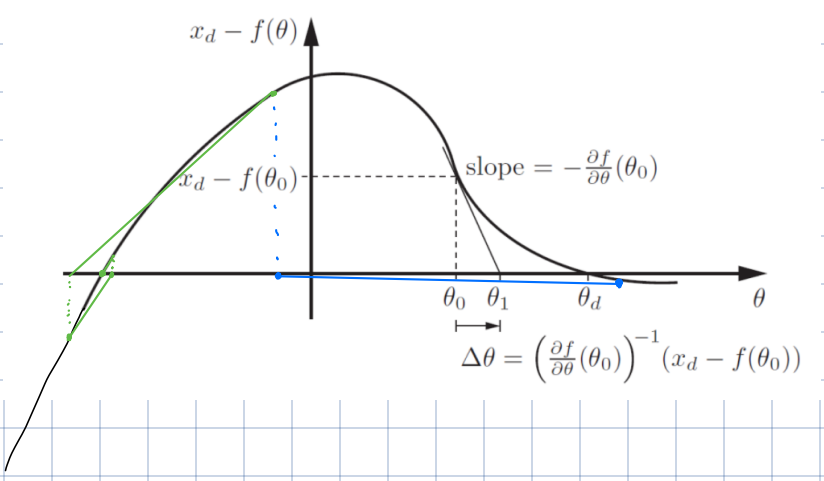
$2.7692 - .7692 + 0 = 2$

```
11
12      L1 = 3;
13      L2 = 2;
14      L3 = 1;
15
16      xb=x_desired;
17      yb=y_desired;
18
19
20      beta= acos((xb^2+yb^2-L1^2-L2^2)/(2*L1*L2));
21      gamma= atan2(yb,xb);
22      alpha= atan2((L2*sin(pi-beta)),L1+L2*cos(pi-beta));
23
24      %beta=.5879;
25
26      theta1=gamma-alpha;
27      theta1_2 = alpha+gamma;
28
29      theta2=pi-beta;
30      theta2_2= beta-pi;
31
32      theta3= theta1+theta2-pi;
33      theta3_2= pi-theta1_2+theta2_2;
34
35      x1_1=L1*cos(theta1) +L2*cos(theta1+theta2) + L3*cos(theta3 +theta1 + theta2);
36      y1_1=L1*sin(theta1) +L2*sin(theta1+theta2) +L3*sin(theta3 +theta1 + theta2);
37
38      x1_2=L1*cos(theta1_2) +L2*cos(theta1_2+theta2_2) + L3*cos(theta3_2 +theta1_2 + theta2_2);
39      y1_2=L1*sin(theta1_2) +L2*sin(theta1_2+theta2_2) +L3*sin(theta3_2 +theta1_2 + theta2_2);
40
```

Command Window

```
first solution
     4

     2

0.000000 1.570796 -1.570796
second solution
     4

     2

1.176005 -1.570796 0.394791
```

2)

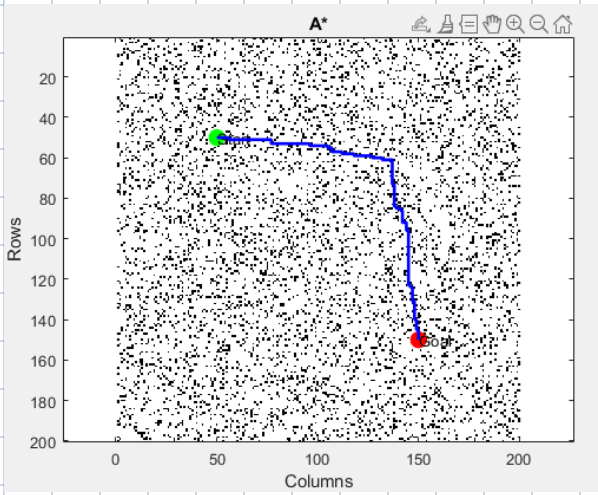


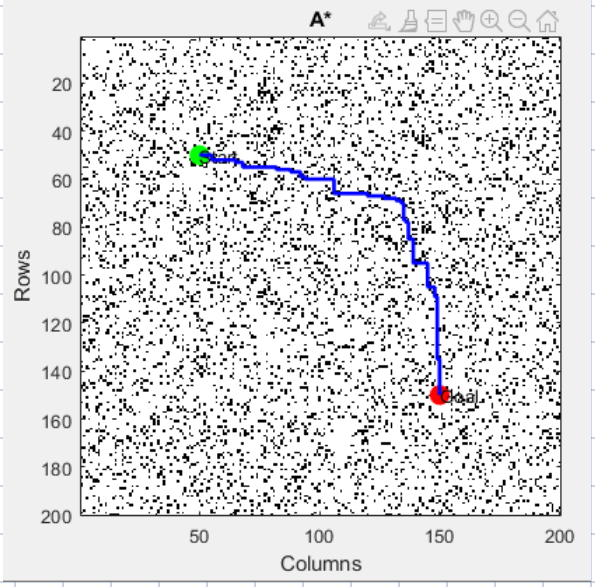
Starting just to the right of the solution will cause the next guess to come on the negative side of the graph, because this side is even it will converge to the negative theta value.

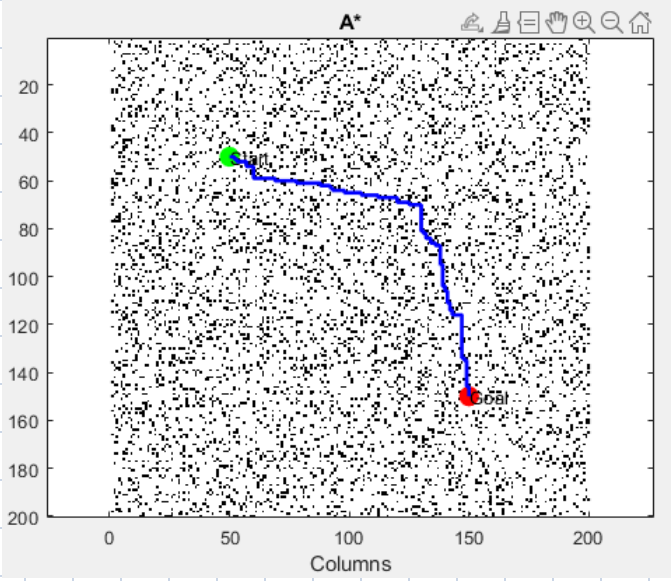
$x_{n+1} = x_n - \left(\frac{f(x_n)}{f'(x_n)}\right)$

A* implementation

file name: intelrobotisastar.m







can be quite difficult to compute analytically, and is generally not unique. In this problem, we will explore numerical methods which use the manipulator's Jacobian matrix to iteratively solve the IK problem of the 7R WAM robot arm. We will ignore joint limits.

In our imagined scenario, the WAM is mounted upright to a tabletop whose surface is 1 meter above the floor. With respect to the origin (on the floor), the back-right corner of the robot (the location of the "Base" frame in the Barrett PDF) is located at x = 0.75m, y = 0.5m, z = 1.0m. The front of the robot is facing the positive y direction.

A whiteboard marker has been attached rigidly to the robot's end plate, such that the marker is vertical and centered on the end plate when all joints are at zero (as shown in the PDF). The marker is 12cm long, so the drawing marker tip is 12cm from the end plate.

(a) (2pts) In assignment 3 and assignment 4, you have developed functions to compute the forward kinematics mapping $x = FK(\theta)$ and the body Jacobian matrix for the marker tip velocity (both the linear and angular components). Evaluate your functions at the following joint configuration θ_s to compute (i) the configuration of the end-effector in $SE(3)$ (i.e. homogeneous transformation matrix $x_s = T_{sb}(\theta_s)$) and (ii) the body twist $\mathcal{V}_b$ of the end-effector when $\dot{\theta}_1 = \dot{\theta}_2 = \dot{\theta}_3 = \dot{\theta}_4 = \dot{\theta}_5 = \dot{\theta}_6 = \dot{\theta}_7 = 1$. Include the result in your submission.

$$\theta_s = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]^T$$

(b) (3pts) Consider the marker tip configuration $x_s \in SE(3)$ for the joint configuration θ_s above as the starting configuration. Suppose the desired marker tip configuration in $SE(3)$ is given as

$$x_d = \begin{bmatrix} -0.7954 & -0.4634 & -0.3906 & 0.46320 \\ -0.0277 & -0.6160 & 0.7873 & 1.16402 \\ -0.6054 & 0.6370 & 0.4771 & 2.22058 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and compute a body twist which would move the marker tip from x_s in the direction of x_d.

(c) (10pts) Implement the numerical inverse kinematics method using your results from questions a and b. Indicate any choices for parameters that you chose (step sizes, stopping conditions, etc). Use your implementation to move from the starting configuration θ_s above to each of these goal marker tip poses:

$$x_{d1} = \begin{bmatrix} -0.7954 & -0.4634 & -0.3906 & 0.46320 \\ -0.0277 & -0.6160 & 0.7873 & 1.16402 \\ -0.6054 & 0.6370 & 0.4771 & 2.22058 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$x_{d2} = \begin{bmatrix} -0.3741 & -0.9211 & -0.1082 & 0.49796 \\ 0.8848 & -0.3894 & 0.2560 & 0.98500 \\ -0.2779 & 0.0000 & 0.9606 & 2.34041 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

For each goal pose, submit a joint trajectory file, giving the joint values at each iteration (each file should start with θ_s). Indicate the final joint configurations θ_{di} for each problem. Briefly discuss the performance of your algorithm.

code: intelroboticshw4.m

a)

$\mathcal{V}_b = J(\theta)\,\dot{\theta}$

$J(\theta) = \begin{bmatrix} J_v(\theta) \\ J_\omega(\theta) \end{bmatrix}$

$T(\theta) \Rightarrow$

```
T0_7 =

    0.2464    0.6429   -0.7253   -0.6699
    0.2390    0.6849    0.6883    0.6338
    0.9392   -0.3429    0.0151    0.3113
         0         0         0    1.0000
```

$\mathcal{V}(\theta) \Rightarrow$

```
Jv =

   1.0e+03 *

   -0.0046   -0.0053    0.1266         0         0         0
    0.0053   -0.0046    0.5171         0         0         0
         0         0    1.0966         0         0         0
         0         0         0   -0.0046   -0.0053    0.1266
         0         0         0    0.0053   -0.0046    0.5171
         0         0         0         0         0    1.0966
```

b)

```
193  %% hw 4 part b %%
194  T0_7;
195  Txd = [ -0.7954 -0.4634 -0.3906 0.46320; -0.0277 -0.6160 0.7873 1.16402; -0.6054 0.6370 0.4771 2.22058;0 0 0 1 ];
196
197
198  R1 = T0_7(1:3, 1:3);
199  p1 = T0_7(1:3, 4);
200
201
202  R2 = Txd(1:3, 1:3);
203  p2 = Txd(1:3, 4);
204
205
206  R = R1' * R2;
207  p = p2 - p1;
208  w_hat = logm(R);
209
210  theta = acos((trace(R) - 1) / 2);
211  w = theta / (2 * sin(theta)) * [w_hat(3,2); w_hat(1,3); w_hat(2,1)];
212  V = [w; p]
213
214  %% hw 4 part c %%
215
```

Command Window

```
         0         0         0    0.0053   -0.0046    0.5171
         0         0         0         0         0    1.0966


V =

   -3.3107
   -0.1409
  -10.7710
    1.1331
    0.5302
    1.9093
```

c) :)

5. **(extra 20pts)** [RRT algorithm] Write an RRT planner for a point robot moving in a plane with obstacles (you can reuse your planar region with 200x200 area in Question 1 without PRM). Free space and obstacles are represented by a two-dimensional array, where each element corresponds to a grid cell in the two-dimensional space. The occurrence of a 1 in an element of the array means that there is an obstacle there, and a 0 indicates that the cell is in free space. Your program should plot the obstacles, the tree that is formed, and show the path that is found from the start to the goal. Note that for the RRT to converge quickly, you can define a goal region instead of using a particular node as the goal.

RRT implementation

file name: inelroboticsrrt.m