Homework 2

ECE 172A
Introduction to Intelligent Systems

February 4, 2017

Make sure you follow these instructions carefully during submission :

- Homework 2 is due by by 11:59 PM, February 15, 2016.
- Submit your homework electronically by email to arangesh@ucsd.edu with the subject line **ECE 172A HW2**. The email should have one file attached.

  Name this file: **ECE_172A_hw2_lastname_studentid.zip**.

  The contents of the file should be:

  1. A pdf file with your write-up. This should include your code outputs along with your answer to each question. Include your MATLAB code (only the ones you made changes to) in an Appendix at the end of your write-up.

     Name this file: **ECE_172A_hw2_lastname_studentid.pdf**.
  2. All of your MATLAB code in a folder called **code** (Note that you have to include your code in the write-up in addition to this). This should include all files necessary to run your code out of the box.
- All problems are to be solved using MATLAB unless mentioned otherwise.
- You should avoid using loops in your MATLAB code unless you are explicitly permitted to do so.
- Finally, carefully read and include the following sentences at the top of your report:

  Academic Integrity Policy: Integrity of scholarship is essential for an academic community. The University expects that both faculty and students will honor this principle and in so doing protect the validity of University intellectual work. For students, this means that all academic work will be done by the individual to whom it is assigned, without unauthorized aid of any kind.

  By including this in my report, I agree to abide by the Academic Integrity Policy mentioned above.

Problem 1. Better Robot Traversal (15 points)

In the last homework assignment, we looked at robots traversing a room using the sense-act paradigm discussed in class. One of the downsides to this approach is the restrictions it places on the space the bot is trying to traverse. In this problem, we will explore bot traversal using a more general approach. Make use of the script *Problem_1.m* as a template while working on each of the following parts.

(i) **Plotting the potential field**

(a) Plot the contour plot and mesh of the vector field as separate figures. Useful functions are *contour()*, *mesh()*, *surf()*, *meshc()*, etc.

(b) Plot the gradient (dx, dy) as *quivers* on the **same** plot as the contour plot. Quiver plots are extremely useful for analyzing the gradients of a differentiable function. It is constructed by plotting a small arrow at every point on the domain of the function, with the arrow pointing in the direction of the gradient at that point. You can create such a plot using MATLABs' *quiver()* function.

(c) Using the quiver plot as reference, indicate where the bot will begin (with a blue asterisk), and where the bot is intended to finish (with a red asterisk) on both the contour/quiver plot, and the mesh plot. Useful functions are *plot()*, *plot3()*.

Provide answers to the following question in no more than two sentences each:

- Where are the obstacles in the room?
- Why do the obstacles look like they do on the potential field?
- What happens to the gradient as you approach the end location?

(ii) **Gradient descent algorithm**

(a) Implement the gradient descent algorithm as discussed in discussion section to navigate the potential field. Be sure to plot the position of the bot at every iteration.

(b) Change the initial position of the bot to a different location within the room and run the gradient descent algorithm for the new case.

(c) Change the position of the obstacles to different locations within the room and run the gradient descent algorithm for the new case. Make sure you revert the initial position of the robot back to the original location.

Provide answers to the following question in no more than three sentences each:

- Why is this method better than the sense-act paradigm?
- Explain what the algorithm is doing and why it works to get the bot across the room while avoiding obstacles.

In you report, include:

- A figure for the contour/quiver plot and a figure for the mesh plot with the initial and final location of the bot.
- A figure for each of the following scenarios:

  - The completed bot trajectory for the given problem.
  - The completed bot trajectory for a different initial location.
  - The completed bot trajectory with different obstacle locations.
- Answers to the 3 questions in part (i), and the 2 questions in part (ii).

Problem 2. Swarms (15 points)

As robots get smaller, the ability to maintain a large number of them becomes easier and easier. Increasing developments in machine intelligence have had a significant effect on the development of *robotic swarms*. In this problem, we'll look at the advantages of intelligent swarms versus non-intelligent ones.

The goal of this problem is to implement behaviors that increase the efficiency of the swarm. Upon first running the code, you will realize how long it takes the swarm to explore the entire space. Some bots don't even leave the center of the room. Follow the instructions given in README.txt to implement a more efficient swarm of robots. You may use loops in your code if necessary. Also, take a look at *exampleOutput.avi* to get an idea about what you're working towards.

In your report, simply include the code (in the Appendix) for all the functions you modified.

Problem 3. Robot Kinematics (15 points)

In this problem, we aim to design a simple 2-D robotic arm and learn how to control its movement. Figure 1 shows the robot to be designed:

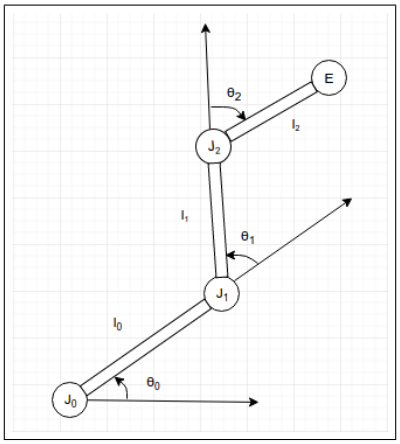

Figure 1: 2-D robotic arm

It has 3 links of lengths l_0, l_1 and l_2 that are free to rotate in the 2-D plane about the three joints J_0, J_1 and J_2. The joint J_0 remains fixed at the origin. The end point of the last link of the robot is called the 'end effector' E. We can control our robotic arm by varying the joint angles θ_0, θ_1 and θ_2

(i) **Forward Kinematics:**

Your first task will be to find out where the end effector and the joints of your robot arm move as you control the joint angles. This is called 'Forward Kinematics'.

Write a MATLAB function *ForwardKinematics()* that takes as inputs the joint angles θ_0, θ_1, θ_2 and the link lengths l_0, l_1, l_2 and outputs the positions of the J_1, J_2 and E. As a guideline, you can use the template for *ForwardKinematics()* that has been provided. Once you have the joint positions, you can use *drawRobot* for visualizing the robot arm. Try varying the values of theta and l_1, l_2, l_3 and observe what happens to the robot.

In your report, include:

- Images of your robotic arm for:
  (a) $\theta_0 = \pi/6$, $\theta_1 = \pi/4$, $\theta_2 = -\pi/3$, $l_1 = 5$, $l_2 = 5$, $l_3 = 5$
  (b) $\theta_0 = \pi/6$, $\theta_1 = \pi/6$, $\theta_2 = \pi/6$, $l_1 = 3$, $l_2 = 5$, $l_3 = 7$
- Your code for *ForwardKinematics* (in the Appendix).

(ii) **Inverse Kinematics:**

Now imagine that your robotic arm is to be deployed in an imaginary 2-D factory to carry objects at given locations. This requires the end effector of the robot to be matched to the location of the object of interest. This falls under 'Inverse Kinematics'. Essentially, you need to find the angles θ_0, θ_1, θ_2 for a given position (x_{target}, y_{target}) of the end effector.

Write a matlab function *InverseKinematics()* that takes as inputs the target end effector position (x_{target}, y_{target}) and link lengths l_0, l_1, l_2, and outputs the angles θ_0, θ_1, θ_2 that will lead to that end effector position. As a guideline, use the template provided for *InverseKinematics()*. The comments in the template will walk you through the 'Jacobian method' for inverse kinematics covered in the discussion section.

Now, using your function, find the values of θ_0, θ_1 and θ_2 for $l_1 = l_2 = l_3 = 5$ and (x_{target}, y_{target}) = (8,8). Try changing the initialization of the θ values in the function (lines 27 to 29) and see if it leads to different solutions. In particular try these two initializations: (1) $\theta_0 = \pi/6$, $\theta_1 = 0$, $\theta_2 = 0$, (2) $\theta_0 = \pi/3$, $\theta_1 = 0$, $\theta_2 = 0$.

In your report, include:

- Images of your robot for the solutions obtained with each of the two initializations.
- A plot showing the end effector positions through all the iterations. (There will be two plots, one for each initialization).
- Your code for *InverseKinematics()* and any other function that you may write (in the Appendix).