Capstone: Welding Robot (Wire-Loop Game)

Robot Kinematics and Dynamics

Prof. Howie Choset

Contents

1 Overview **3**

2 Deliverables and Grading **4**
2.1 Competition 4
2.2 Report 5

3 Recommended Approach **9**

4 Resources **11**
4.1 Handout 11
4.2 System Diagram 12

5 Submission Checklist **14**

1 Overview

Welding robots are essential for factory automation. Without them, the cars that help you travel all over the country will not be possible. Welding requires robots that can precisely follow a specified path.

In this capstone project, you will control a 5 degree-of-freedom arm to simulate a welding scenario. To make things interesting, you will compete against your colleagues at the wire-loop challenge.

The final deliverable for this project will be participation in a wire-loop competition as well as a short report where you provide short responses to questions about the project, and what you are learning in class.

This will reinforce the following topics:

- 3D Forward Kinematics
- 3D Inverse Differential Kinematics
- 3D Inverse Kinematics
- Robot Control using Trajectories in 3D
- Denavit-Hartenberg Transformations
- Manipulability

2 Deliverables and Grading

2.1 Competition

Imagine you are working for a company that specializes in welding robots.

A welding robot has been designed with 5 degrees of freedom and the design engineers have created a clever game to simulate the welding situations that the robot might face. A small ring 5 cm in diameter has been connected as the robot's end-effector. The various welding trajectories are presented as wires and have a prescribed speed associated with them. Your job as the lead of the software team is to write a script that will thread the wire through the given trajectory. Your performance will be gauged by:

1. the number of times you touch the wire,
2. the time taken to complete the path.

Tasks:

1. The arm will initially be placed at the start point of the path.
2. Move the arm along the path as fast as possible without connecting the ring to the wire.
3. The robot performance should not degrade when a known mass is attached at the end-effector.

The competition will be worth 50 points (out of 100 total for the capstone). We will break the competition up into two phases:

- Qualifying:
  Your robot will move along the path until it reaches the end or until a non-negligible external force is felt on the end-effector. The TAs will begin the stopwatch at the moment the robot starts along the path. The timer will terminate when the robot reaches the end of the path or when 3 minutes are up. This path will be repeated and during this run, a TA will attach a known mass at the end-effector to simulate a variable mass of the welding apparatus. Grading with be based on the number of times you touch the wire and the time you take.
- Competition:
  The students that qualify, will move on to compete against each other in a series of challenging paths. At the end of each path, the implementation with the lowest number of points will be knocked out. The last student standing will receive bonus points.

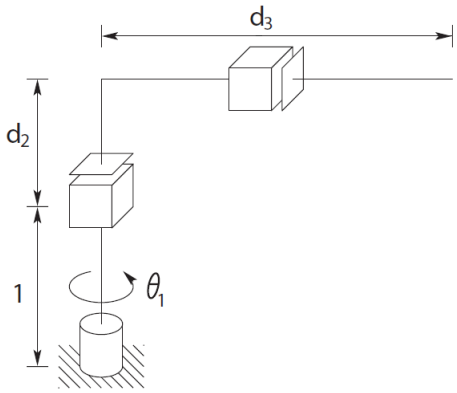
2.2 Report

The report consists of two parts. First is a section with written questions to help as you build up the code for the demo, and the second is a reflection section where you examine the results from the demo and actual data from the running the robots.

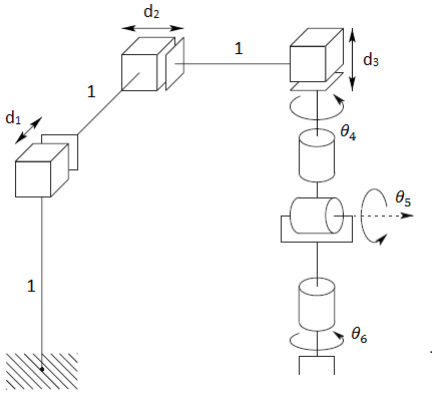
1) Part I: Written Questions

(1) [5 points] **Denavit-Hartenberg Transformations**

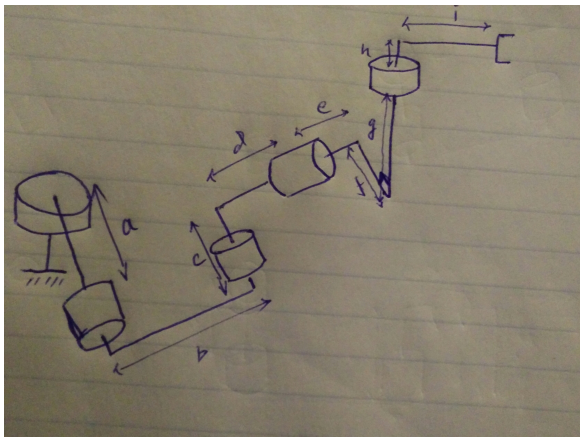
1. Robot A



2. Robot B



3. Robot C (each joint i is labelled θ_i following the right-hand rule)



Above are three diagrams, A-C, of robot systems. Robot 'C' is a top-down of the physical system you will use for the capstone. The system parameters for robot 'C' are given in Table 1.

For each diagram above, provide

1. DH parameters of this robot
2. The 4x4 transformation at the end-effector (with respect to the base joint), given the following angles:
   (a) Robot A: $\theta_1 = 45^o, d_2 = 2, d_3 = 3$
   (b) Robot B: $d_1 = 2, d_2 = 3, d_3 = 1, \theta_4 = 20^o, \theta_5 = 45^o, \theta_6 = 90^o$
   (c) Robot C: $\theta_1 = 20^o, \theta_2 = 45^o, \theta_3 = 45^o, \theta_4 = -45^o, \theta_5 = 90^o$

Note that by following the recommended approach below, your code can help you generate these transformations given the DH parameters. To help check your work, here are (for each system) two example configurations and the resulting end effector transform.

If you do not use your code to generate the answers here, you will need to show your work clearly.

(2) [5 points] **Workspace**

We will compute a slice of the workspace of the robot you will use in the demo. To do this, let's lock the base joint(1st) and the 4th joint at 90 degrees. The remainder of the arm becomes a planar RRR robot.

For a J1 and J4 at an angle of 90 degrees, plot the (x,y) workspace of the robot arm. This can be done using similar code to that from earlier labs, by iterating through the free joints and computing the forward kinematics in each case. Add this plot to the report. Choose the step size wisely as iterating through three variables can be time consuming on matlab.

(3) [5 points] **Manipulability**

Now, additionally compute the manipulability for each of the points in the constrained slice of the 2D workspace. Clearly visualize this information (e.g., you can use color, size

of the markers, a 3D plot, or other methods, as long as you clearly show the differences in manipulability magnitude and label your plot).

(4) [5 points] **Determine Initial Home Position**

Using the information in these workspace and manipulability plots, choose a suitable location for the initial home position for the end-effector to go. In a sentence or two, describe why you chose this location.

(5) [5 points] **3D Jacobians**

Using the techniques you learned in class, write the analytic Jacobian for the physical robot you will use in the demo.

If you follow the recommended approach below, you can cross-check your analytic Jacobian with the numerical one written in the 3D Robot class. This is a good way to check your answers!

(6) [10 points] **Inverse Kinematics**

Note that the 5 DOF demo robot is well suited for analytic IK, due to the ability to geometrically decompose the IK problem into subproblems.

Given a position in the workspace (x,y,z) (relative to the base frame), and an orientation of the end-effector pointing straight down (e.g., the positive z axis of the end effector is in the negative z axis direction of the base frame[1]), compute the joint angles θ_1, θ_2, θ_3, and θ_4 which achieve this position and orientation. Your answer should be in terms of the link lengths l_1 and l_2; assume the other dimensions of the robot are fixed.

(Note that θ_5 is unconstrained given this problem description).

If you follow the recommended approach below, you can cross-check your IK with results from the numerical one written in the 3D Robot class.

2) Part II: Reflection

(1) [10 points] **Description of Approach**

Concisely outline the approach you took. How did you pick the waypoints? What type of trajectories did you use to follow the waypoints? Did you only send position commands? Did you also send velocity commands? Did you add any torques to help compensate for gravity? Did you tweak any of the control gains?

Keep this section brief – no more than a few paragraphs – but be sure to justify *why* you made the choices you did during the project.

(2) [5 points] **Plotting Data**

After running the demo (this does not have to be the same time you show it to the TAs), examine the logged data from the run. Plot the position error (commanded - feedback) for each joint, and identify any areas where this seems higher than others. Add this plot, and comment on the cause for any such areas. Add these plots to the report.

Now plot the torque feedback from each joint during the demo. Save these figures, and then run the demo again at 1/4 the speed. Plot the torque feedback again, and

[1] Note that this constraint ensures the point is in the robot's workspace for some link lengths, and helps with the geometric decomposition

compare these plots. What explains the difference between the torque when the robot is moving slowly versus when it is moving fast? Add these plots to the report.

What is the equation that could compute the approximate torques experienced by the robot when it was moving very slowly?

When the robot moves faster, what are the additional torques caused by? You do not have to provide an equation for these torques, but briefly (1-2 sentences) describe how you could compute them given knowledge of the robot and the joint angle trajectory.

3 Recommended Approach

As long as you meet the deliverables and complete the demo, you can take any approach you would like. However, here is our recommendation.

1. Using the diagram shown in section 4.2 (System Diagram), extract the DH parameters for the 5-DOF robot that will be used for this demonstration.

2. Start by adapting the 2D Robot.m class you wrote before to a new 3D version.
   - Use a matrix of DH parameters as the argument for the class constructor, rather than just link lengths.
   - Replace your numerical IK optimizer with a built-in MATLAB optimizer: look into function **fmincon**. You can use analytical method to do inverse kinematics but we would highly recommend using numerical methods.
   - Test the forward kinematics, Jacobian, and inverse kinematics functions by solving some simple test cases, and by using this code to help solve the problems in the first part of the report.
   - Test your extraction of DH parameters for the demo robot by using your code to help solve the problems in the first part of the report.

3. Use the robot class to help generate answers for the first part of the report, writing small scripts as needed.

4. Using the sample code and code you have written previously, write a script that reads a comma separated file containing the definition of the trajectory and queries the initial position of the robot. With these two inputs, follow this trajectory as accurately as possible to prevent contact with the wire. You might have to tune motor controller gains and/or implement feed-forward gravity compensation to balance the extra load on the end-effector. For a more detailed description of the task, refer to section 2.1 above.
   - Use IK to build the sequence of commanded angles you will send to the robot.
   - You can call the getNextFeedback() function, as in the demo code, to get the position of the robot at any time to help identify these joint angles.
   - Think about the path you want the robot to take as it moves throughout the task. What requirements must you have on your trajectories?

5. Improve your code to ensure your final demo is robust and gets you the most points possible.
   - Can you make the process faster? Can you speed up the motion of the robot without adding too much error?
   - Are you consistently within the level of acceptable error? Do you need to slow down or change the trajectory waypoints for any portions of the task?
   - Does adding gravity compensating joint torque commands improve the accuracy or let you speed up the motion of the system?
   - Does using velocity commands instead of or in addition to position commands improve the accuracy or let you speed up the motion of the system?

- No two robots are exactly the same! Does your code work on both robots, or are there tweaks you need to make for one or the other?

6. Complete the second half of the report (the 'reflection' section) using plots and data taken from running your code.

4 Resources

4.1 Handout

Download the assignment handout from the course website. Copy the code handout folder to some location of your choice. Open Matlab and navigate to that location.

This assignment is fairly open ended, but we have provided some sample code to help you get started. Note that code from previous assignments – especially for creating trajectories – will be very useful when completing this assignment.

The main sample file is `welding_trajectories.m`. This is a short example which runs through a couple of waypoints. **NOTE: this requires the spline trajectory function that you wrote for the last homework in order to run.**

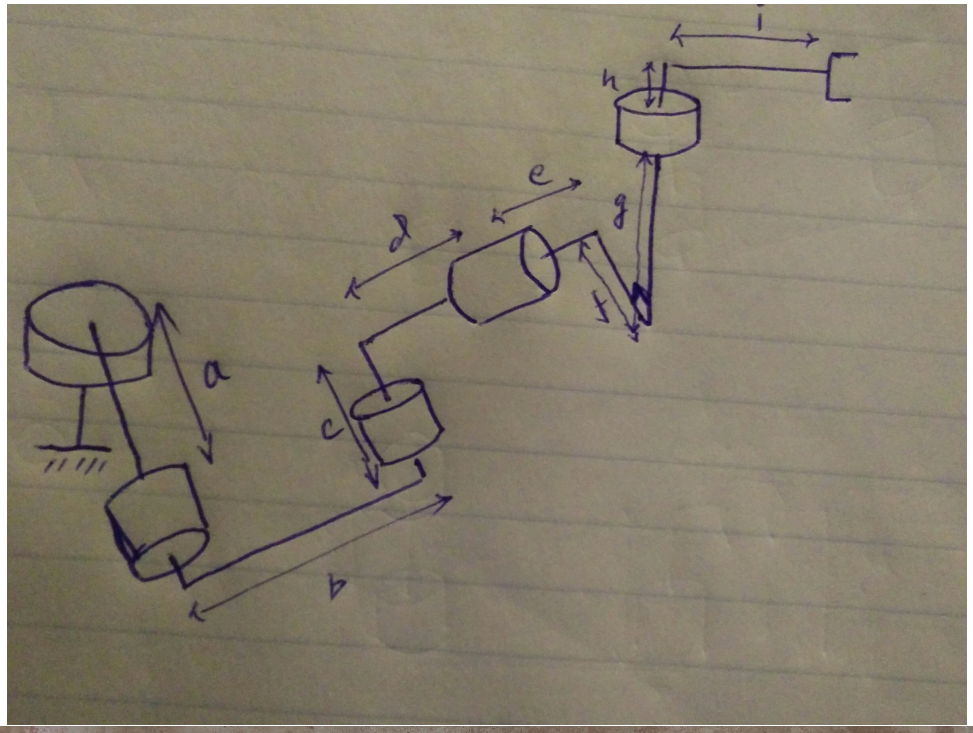
This file also demonstrates:

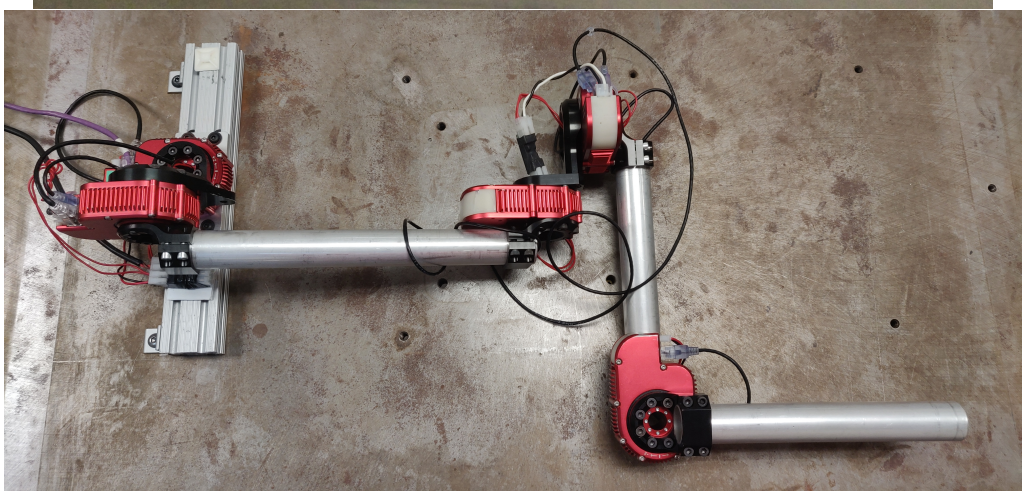
- Basic robot access and control.
- Logging commands and feedback, and plotting this data.
- Use of intermediate waypoints and trajectory segments to improve approach and drop off accuracy.

Additionally, we have provided `numerical_IK_sample.m`, which demonstrates how to use built-in MATLAB optimization functions to quickly solve IK problems without the speed, stability, and parameter tuning issues of a custom gradient descent implementation. Note that this just demonstrates solving for a 3D position (not orientation), but you can extend this error function to include orientation as well.

When using this numerical optimization code, be aware that the joint angles resulting from this optimization may be out of the commandable range of the robot – you should limit the results for all joints (except the end-effector) to within $\pm\pi$. For example, if the result is 9 radians for joint 1, you should instead use $9 - 2\pi$ when commanding the robot.

4.2 System Diagram





The following diagram may not be precisely to scale and all measurements are in meters. You are encouraged to re-measure the exact geometry of the robotic system for further accuracy. The width of the motor is 31.1mm.

Parameter	Value (mm)
a	53.34
b	317.5
c	53.34
d	73.66
e	15.24
f	254
g	50.8
h	15.24
i	241.3

Table 1: System parameters for Welding Robot

5 Submission Checklist

- ☐ After completing the entire capstone, fill out the feedback form[2], and:
  - ☐ Run the demo for grading by the TAs.
  - ☐ Upload a .tar or .zip file of your code.
  - ☐ Upload a PDF of your report to Autolab.

[2] https://goo.gl/hKFctm