Homework 5:
PUMA 260

MEAM 520, University of Pennsylvania
Katherine J. Kuchenbecker, Ph.D.

September 26, 2013

This assignment is due on **Tuesday, October 1, by midnight (11:59:59 p.m.)** Your code should be submitted via email according to the instructions at the end of this document. Late submissions will be accepted until Thursday, October 3, by midnight (11:59:59 p.m.), but they will be penalized by 10% for each partial or full day late, up to 20%. After the late deadline, no further assignments may be submitted.

You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. To help you actually learn the material, what you write down should be your own work, not copied from any other individual or team. Any submissions suspected of violating Penn's Code of Academic Integrity will be reported to the Office of Student Conduct. If you get stuck, post a question on Piazza or go to office hours!

Individual vs. Pair Programming

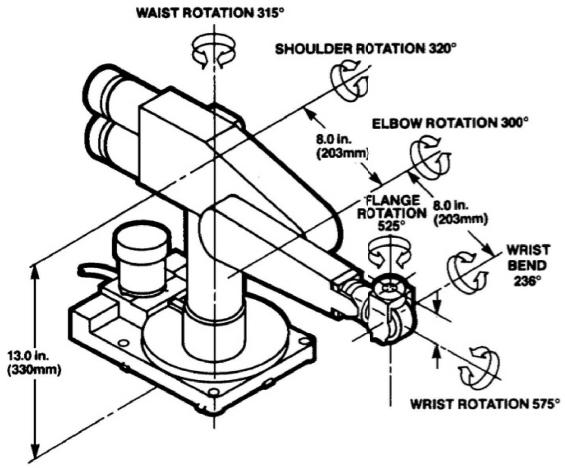
We encourage you to do this assignment with a partner. If you prefer, you may also do this assignment individually. If you do this homework with a partner, you may work with anyone you choose, even someone with substantial MATLAB experience. If you are looking for a partner, consider using the "Search for Teammates!" tool on Piazza.

If you are in a pair, you should work closely with your partner throughout this assignment, following the paradigm of pair programming. You will turn in one MATLAB script for which you are both jointly responsible, and you will both receive the same grade. Please follow these pair programming guidelines, which were adapted from "All I really need to know about pair programming I learned in kindergarten," by Williams and Kessler, *Communications of the ACM*, May 2000:

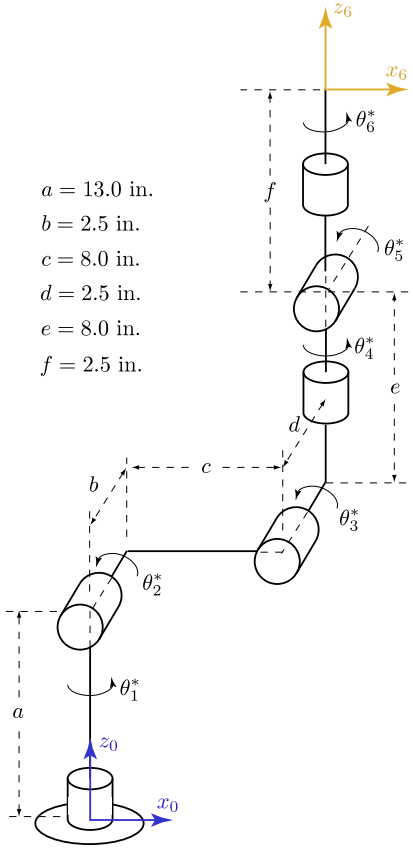
- Start with a good attitude, setting aside any skepticism and expecting to jell with your partner.
- Don't start writing code alone. Arrange a meeting with your partner as soon as you can.
- Use just one computer, and sit side by side; a desktop computer with a large monitor is better for this than a laptop. Make sure both partners can see the screen.
- At each instant, one partner should be driving (using the mouse and keyboard or recording design ideas) while the other is continuously reviewing the work (thinking and making suggestions).
- Change driving/reviewing roles at least every thirty minutes, *even if one partner is much more experienced than the other.* You may want to set a timer to help you remember to switch.
- If you notice a bug in the code your partner is typing, wait until they finish the line to correct them.
- Stay focused and on-task the whole time you are working together.
- Recognize that pair programming usually takes more effort than programming alone, but it produces better code, deeper learning, and a more positive experience for the participants.
- Take a break periodically to refresh your perspective.
- Share responsibility for your project; avoid blaming either partner for challenges you run into.

System Description

The Penn MEAM Department owns a PUMA 260 robot. We will be using this robot for the hands-on manipulator labs in this class. The PUMA is an articulated (RRR) robot with lateral offsets plus a spherical wrist (RRR). The image below on the left shows a photo of the robot, and the image below on the right diagrams the arrangement of its six revolute joints.



The schematic below shows the zero configuration we have designated for the PUMA in this class. All of the joints are shown at $\theta_i = 0$. The joint angle arrows show the positive direction for each revolute joint (θ_1 to θ_6). The diagram also gives the measurements for the constant dimensions (a to f), all in inches, and the minimum and maximum angles for each of the PUMA's six revolute joints.



$a = 13.0$ in.
$b = 2.5$ in.
$c = 8.0$ in.
$d = 2.5$ in.
$e = 8.0$ in.
$f = 2.5$ in.

$-180° < \theta_1^* < 110°$
$-75° < \theta_2^* < 240°$
$-235° < \theta_3^* < 60°$
$-580° < \theta_4^* < 40°$
$-120° < \theta_5^* < 110°$
$-215° < \theta_6^* < 295°$

Task Description

Your task is to derive the forward kinematics of the PUMA 260, following the DH convention, and create a simulation that animates the movement of this robot.

1. **DH Parameters for the PUMA 260**
   Annotate the full-page schematic of the PUMA (provided later in this document) with appropriately placed coordinate frames, and then write a table of the corresponding DH parameters. We encourage you to do this in pencil so that you can make corrections if needed. You may find it useful to follow the steps provided in SHV Section 3.4. You do not need to turn this paper in; it is for your reference.

2. **Download PUMA Joint Angle Histories**
   Download `puma_motions.zip` from the Piazza course page under Resources, and unzip it. This folder contains a set of MATLAB data files such as `puma_motion1.mat`. When this assignment was initially released, eight data files were provided (motions 0 through 7); more may become available later.

   If you set MATLAB's current folder to be the unzipped folder, you can load any of these data files into the MATLAB workspace by running the command `load puma_motion1` Insert the number of the motion that you want in place of the `1` in this example.

   Each of the data files includes eight variables:
   - `t_history` is a column vector of time values, measured in seconds. It has n elements, and n differs somewhat across files.
   - `theta1_history` is a column vector of joint angle values for joint 1 of the PUMA, measured in radians (not degrees). It has n elements.
   - `theta2_history` through `theta6_history` are column vectors of joint angle values for the other joints of the PUMA, measured in radians. Each of these has n elements.
   - `T06_history` is a four by four by n matrix that starts out filled with zeros. As described below, part of your task is to fill in this matrix with the transformation matrix T_6^0 that corresponds to each of the n moments in time in the data file.

   All joint angles are set to zero in `puma_motion0`. For each file from `puma_motion1` to `puma_motion6`, only one joint of the robot moves, while the rest are stationary. All of the PUMA's joints move in `puma_motion7`.

   Verify that you can load all of the data files and see the variables before proceeding with this assignment. Post any questions on Piazza.

3. **PUMA 260 Animation** ***(20 points)***
   Write a MATLAB script that accomplishes the following steps:
   - Name your file `puma_robot_pennkey1_pennkey2.m` or just `puma_robot_yourpennkey.m` if you're working alone.
   - Clear the workspace at the start using the `clear` command.
   - Create a variable called `studentNames` and set it equal to a string that contains your full name and your partner's full name, or just your full name if you're working alone.
   - Load one of the provided data files. You should test all of them.
   - Define variables for any robot parameters that you will need. They should be sensibly named.
   - Step through the data file from the start to the end. At each step, calculate the PUMA's current T_6^0 matrix and store it in the appropriate elements of `T06_history`. Use units of inches for the positions. You may use the `dh_kuchenbe.m` function that Professor Kuchenbecker provided with the SCARA robot if you would like. Storing this matrix is required to enable us to grade your assignment.

- At each step, also display a simple three-dimensional plot of the robot's pose in Figure 1, so that you can see how the PUMA would move if controlled to follow this history of joint angles over time. Accomplishing this step will require you to calculate the position of many points along the robot at each step in time and then display them in succession. You may want to look at the `scara_robot.zip` example posted on our Piazza course page under Resources. Like the SCARA, it is fine for your PUMA to be drawn as thick lines with dots at the joints, or you may pick another graphical style that you prefer.
- At each step, also display the current position and orientation of frame 6, so that the viewer can see the orientation of the PUMA's end-effector. You may want to follow the display style of the `visualize_R.m` script provided to the class; the positive x-axis is shown dotted, the positive y-axis dashed, and the positive z-axis solid. You do not need to label the axes with text.
- Make sure your animation is well formatted, including axis labels with units and axis limits that do not change over the course of the script's execution. Call the command `axis vis3d` to make one unit be displayed the same in all directions and to disable scaling during rotation.
- Put your names in the title of the plot.
- If any of the robot's joint angles are outside the acceptable ranges defined earlier in this document, display the robot in a different color for that time step. Return the robot to its regular color when the joint angles are all back within the limits.
- If the tip of the robot moves into the table on which the PUMA sits (below $z = 0$ in.), display the robot in another different color for that time step. Return the robot to its regular color when the tip is no longer touching the table.
- If both the joint angle limits and the table constraint are violated, display the robot in a distinct color for that time step.
- Clean up your code so that it is easy for someone else to follow. Remove any commented-out commands that you are not using. Add comments to explain tricky steps. Use code section headings (e.g., `%% Define Robot Parameters`) to delineate the functional regions of your script.
- If you created any other functions, make sure they contain your (and your partner's) PennKey so that we avoid function name collisions across the class.

Your calculations may not use any built-in or downloaded functions dealing with rotation matrices, homogeneous transformations, Euler angles, roll/pitch/yaw angles, or related topics. Instead, you must type all your calculations yourself, using only low-level functions such as `sin`, `cos`, *and vector/matrix math.*

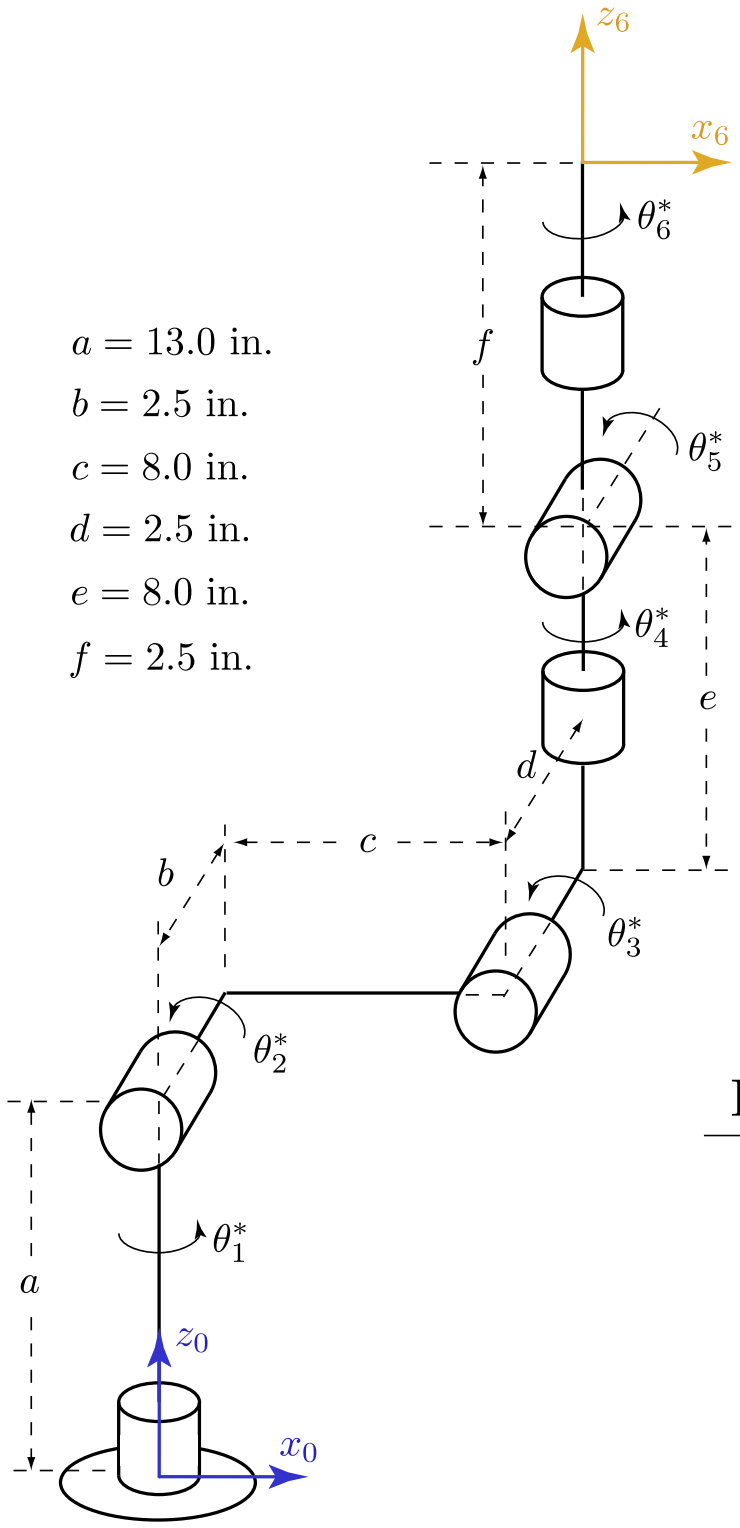
Submitting Your Code

Follow these instructions to submit your code:

1. Start an email to `meam520@seas.upenn.edu`
2. Make the subject *Homework 5: Your Name* or *Homework 5: Your Name and Your Teammate's Name*, replacing *Your Name* and *Your Teammate's Name* with the appropriate full names.
3. Attach your correctly named MATLAB file(s) (`puma_robot_pennkey1_pennkey2.m` plus any other functions you created, similarly identified) to the email. Please do not put them in a zip file or include any other attachments.
4. Optionally include any comments you have about this assignment and the experience of pair programming if you worked with a teammate.
5. Send the email.

You are welcome to resubmit your code if you want to make corrections. To avoid confusion, please state in the new email that it is a resubmission, and include all of your MATLAB files, even if you have not updated all of them.

Zero Configuration for the PUMA 260



$a = 13.0$ in.
$b = 2.5$ in.
$c = 8.0$ in.
$d = 2.5$ in.
$e = 8.0$ in.
$f = 2.5$ in.

Link	a_i	α_i	d_i	θ_i