# CSE4421: Lab 2

Burton Ma

Thu 27 Jan, 2011

## 1 Preliminaries

This lab introduces you to the A255 robot and the Matlab controller for the A255 robot. The A255 API is somewhat different from that of the A150; in particular, there is no native function such as `ma` that allows the user to set all of the joint angles simultaneously.

## 2 Instructions to Use the A255 Arm

The safety precautions discussed for the A155 robot also apply to the A255. The A255 has brakes on most of the joints that will hold the links stationary when the arm power is turned off.

1. Log in to the workstation connected to the arm.
2. Start a console; in the console start the minicom program by entering the command `minicom`.
3. Find the teach pendant that plugs into the socket located in the upper-left corner of the controller. Plug the teach pendant into the controller (the arm will not power on without the pendant plugged in).
4. Power on the robot by pressing the power switch in the middle-left side of the control panel. Wait for it to finish its boot process.
5. Check the big red kill switch (kills power to the arm motors). If it is pressed in, unlock it by turning it.
6. Check the kill switch on the pendant. If it is pressed in, unlock it by turning it.
7. Press the arm power button located in the upper-right corner of the controller. You should hear a click and see a light on the button.
8. Check if the robot is in a suitable pose so that it can be homed (all of the joint markers should be approximately aligned). If it is not, manually rotate the joints by entering the `joint` command in Minicom (type `help joint` for documentation or see Chapter 3 of the *CROS and System Shell* manual).
9. Tell the robot to home itself by entering the command `home` in minicom. This will take some time.
10. Once the robot is correctly homed it is ready for use. Enter the command `ready` in Minicom.
11. If you are using the Matlab controller you can exit Minicom.

## 2.1 Instructions to Turn Off the A255 Arm

1. If the robot joints are limped, unlimp them.
2. Press the kill switch; the robot has joint brakes that will lock the joints in place.
3. Turn off the main power on the controller.

## 2.2 Using the Application Shell

The user can interact with the A255 using the application shell (see the *Application Shell ASH* manual). To start the application shell type the following command into Minicom:

```
ash test
```

Once in the application shell, you can list all of the available commands by typing *help*, and you can get help on a particular command (say the *joint* command) by typing *help joint*. Of course, you can always refer to the *Application Shell ASH* manual. Like the A150, there are numerous commands that you may find useful:

- JOINT
- LIMP and NOLIMP
- GRIP_CLOSE and GRIP_OPEN (not that WGRIP does not work properly on this arm)
- W0 and W2 (what distance units are being used?)
- WX, WY, and WZ
- TX, TY, and TZ
- ROLL and PITCH
- HERE
- MOVE
- APPRO and DEPART

## 2.3 Using the Matlab Controller

It is impossible (?) to write a program that runs in the ASH shell as we do not have the software tools needed to compile and send RAPL3 programs to the hardware controller. However, a controller written in Matlab is available for your use:

```
/cs/dept/www/course/4421/src/matlab/A255.m
```

(also available from the Source Code section of the course web pages). It is very similar to the controller for the A150 robot with the important exception that only one joint at a time can be directly manipulated (using the `joint` command). The lab instructor will show you the effects of using the `joint` command.

## 3 Matlab Programming Problems

### 3.1 Address the Difference Between the Java Simulator and the Matlab A150 Controller

There is an important difference between the Java controller and the Matlab controllers. The Java controller and simulator use the DH convention to specify the joint angles. The Matlab controllers use the robot's native convention for the joint angles; specifically, the joint angles for the joints 2, 3, and 4 (the shoulder, elbow, and first wrist joint) are specified relative to the horizon. In the ready position, the robot's convention is that the first four joint angles are waist $0°$, shoulder $90°$, elbow $0°$, and wrist $0°$; these are same values you should use when using the Matlab controller function `madeg`. The same joint angles will cause the arm to point stright up when using the Java simulator.

It turns out that it is very easy to write a new function in the Matlab controller that behaves the same as the `madeg` function in the Java simulator. All the function needs to do is to take as input a set of joint angles specified using the Java simulator convention and convert them to the robot's native convention, and then use the robot's version of `madeg` with the converted joint angles.

Modify the A150 Matlab controller so that it implements a function named `simmadeg` that behaves identical to the Java simulator version.

### 3.2 Implement the Denavit-Hartenberg Transformations

Implement a function that computes the forward kinematics of an $n$-joint serial manipulator using the Denavit-Hartenberg convention.

As a first step, you should implement the three functions `rx`, `rz`, and `translate` that return the $4 \times 4$ homogeneous matrices for rotation about the $x$ axis, rotation about the $z$ axis and translation by a vector, respectively.

When you think that you are finished, test your Denavit-Hartenberg implementation by using values for the A150 or A255 robots; note that both robots have a function named `w0` that returns the position of the end effector connector in the base coordinates of the robot.