ME 221, Fall 2022
University of California, Riverside
Department of Mechanical Engineering

HW 2
Due: 10/21/2021

1. Find the homogeneous transformation matrices for each of the following:
   a. Translation by +2 units in the Z direction and translation of -4 units along the X direction.
   b. Rotation of 45° about an axis with unit vector $\hat{u} = \frac{\sqrt{3}}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$.
   c. The combined translation and rotation from (a) and (b).

2. Derive the homogeneous transformation matrix that rotates a point P about an axis located **d** from the origin as illustrated in Figure 1.

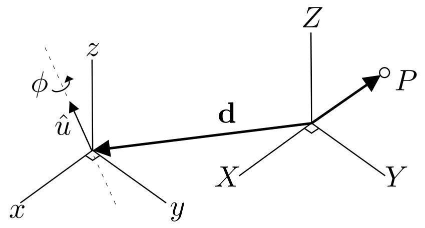


Figure 1: Schematic for Problem 2.

3. Figure 2 illustrates a pick and place scenario. A robot with attached global fixed reference frame-0 is located next to a table (frame-1) with an cube on the table (frame-2). A camera (frame-3) is centered above the table and is used to localize the cube. Suppose that the cube can only move in a planar motion on the tabletop and the camera output consists of x_c and y_c – the cube position in the camera's local coordinate frame-3. Derive the transformation matrix of the cube in the global fixed reference frame-0 as a function of the camera outputs:

$$^0T_2(x_c, y_c) = ?$$

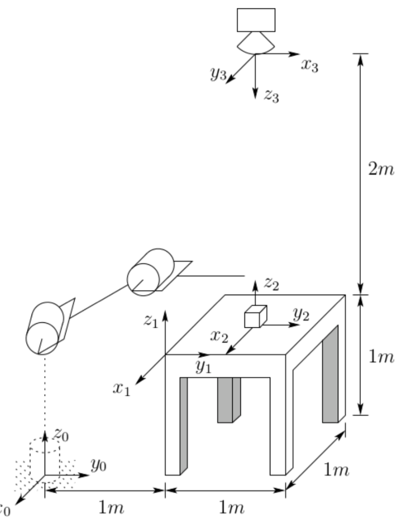
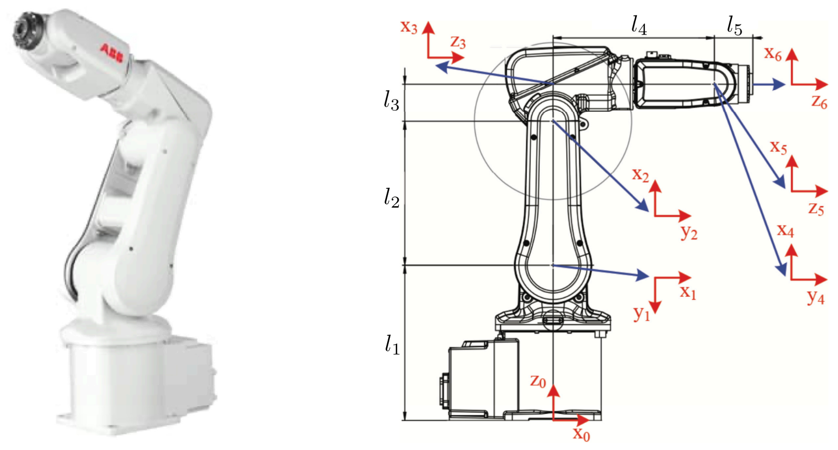


Figure 2: Schematic for Problem 3.



l_1	0.290
l_2	0.270
l_3	0.07
l_4	0.302
l_5	0.072

Figure 3: IRB 120 details. Lengths are reported in meters.

4. Find the Denavit-Hartenberg parameters for the 6-DOF IRB 120 industrial robot shown in Fig. 3. (It is recommend to use the reference frame assignment shown in the image.)

5. Download the MATLAB code `hw2_5.m`. The code renders the IRB 120 robot in a virtual environment. The function call:

```
r = loadrobot('abbIrb120','DataFormat','column');
```

loads the robot model object into the variable `r` and specifies that a column vector format will be used to describe the *pose* of the robot (e.g., the joint parameters are specified as a column vector). Since the IRB 120 is 6 DOF, the pose vector is dimension 6×1. The function call:

```
ax = show(r,pose, ...
        'Visuals','on', ...
        'PreservePlot',0, ...
        'Fastupdate',1); hold all;
```

renders the robot with joint parameters specified by `pose`.

a. Use the DH-parameters from Problem 4 to create a function that takes as input the joint parameters (pose) and returns the homogeneous transformation matrix from the end-effect frame to the base frame: ${}^0T_6$. That is, solve the *forward kinematics* problem for the IRB 120. The function should take the form:

```
function T06 = fkine(pose)
  ...
end
```

One suggestion is to define a helper function that converts the DH-parameters to the corresponding transformation ${}^{i-1}T_i$:

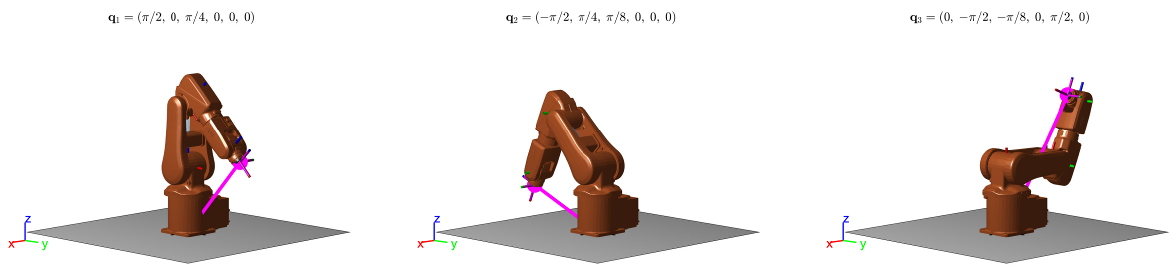

$\mathbf{q}_1 = (\pi/2,\ 0,\ \pi/4,\ 0,\ 0,\ 0)$
$\mathbf{q}_2 = (-\pi/2,\ \pi/4,\ \pi/8,\ 0,\ 0,\ 0)$
$\mathbf{q}_3 = (0,\ -\pi/2,\ -\pi/8,\ 0,\ \pi/2,\ 0)$
z
x
y

Figure 4: Images for Problem 5a.. Test pose plots.

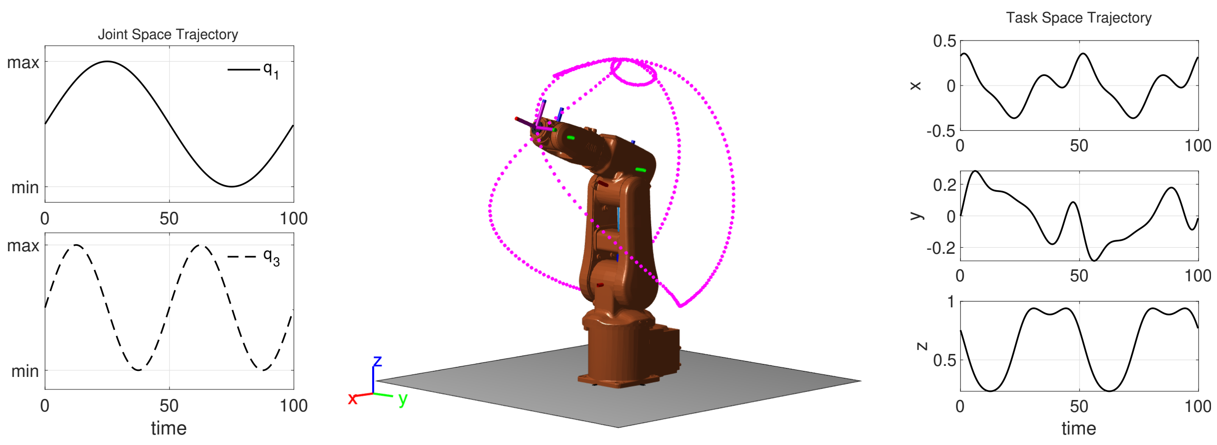


Figure 4: Images for Problem 5b.. Joint space (pose) trajectory, Cartesian trajectory in the virtual environment, and Cartesian trajectory in time.

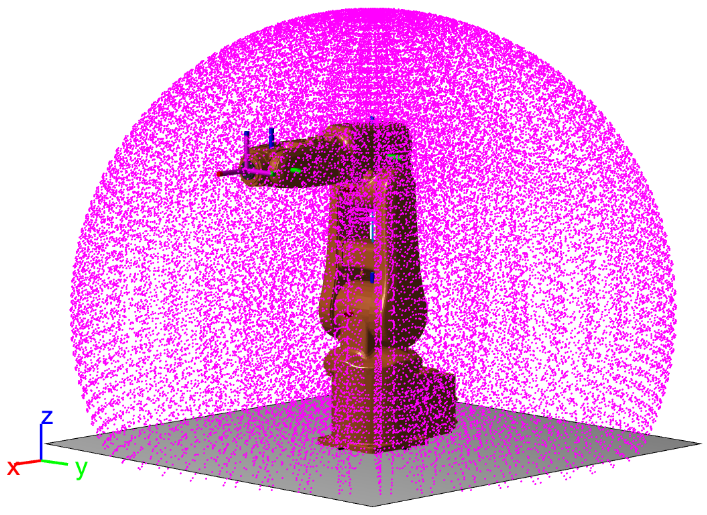


Figure 4: Images for Problem 5c.. Robot workspace.

```
function im1_T_i = dh2T(d_i,theta_i,a_i,alpha_i)
  ...
end
```

You can call this function inside `fkine` to create each transformation. Evaluate your forward kinematics function and create the plots for the following *test* poses:

i. $\mathbf{q}_1 = (\pi/2,\ 0,\ \pi/4,\ 0,\ 0,\ 0)$
ii. $\mathbf{q}_2 = (-\pi/2,\ \pi/4,\ \pi/8,\ 0,\ 0,\ 0)$
iii. $\mathbf{q}_3 = (0,\ -\pi/2,\ -\pi/8,\ 0,\ \pi/2,\ 0)$

To create the plots move the robot to the test pose and use your forward kinematics function to get the Cartesian position ${}^{G}\mathbf{r}_e$ of the end-effect:

```
GTe = fkine(pose1);
Gre = GTe(1:3,4);
```

Plot a line from the origin to ${}^{G}\mathbf{r}_e$ and place a large marker at the end effector position as shown in Fig. 4.

b. Create a time varying pose trajectory with the following characteristics:

i. $q_1(t) = A_1 \sin\left(2\pi \frac{1}{T} t\right) + K_1$

ii. $q_3(t) = A_3 \sin(2\pi \frac{2}{T} t) + K_2$

where T is the simulation period. Choose A_1, A_2, K_1, K_2 such that the sinusoids span the full range of joint motion for each joint. You can find the joint ranges uses:

```
jointConstraints = constraintJointBounds(r);
limits = jointConstraints.Bounds;
```

The variable `limits` contains the min/max of each joint, stored as a 2×6 matrix. In a `for` loop, plot the end-effector position using your function `fkine` (use a smaller marker this time) and rendered the robot pose for each time step. Save the final figure to show the trajectory in the virtual environment and plot the resulting Cartesian end-effector trajectory as a function of time. Fig. 4 shows the joint space trajectory in time, and the Cartesian trajectory in the virtual environment and in time. An example video is posted on the website.

c. Generate a visual representation of the reachable workspace as shown in Fig. 4. Fix the last three joints to reduce the number of test points. Also, don't render the robot for each test point, rather use your `fkine` function to generate all the points in a `for` loop, then plot all of them simultaneously using `scatter`. (*Hint*: the function `meshgrid` can help.)