5ELEN018W - Tutorial 8 Exercises: Matlab Robotics Toolbox for Forward Kinematics

1 Cell Arrays in Matlab

A cell array is a Matlab data type whose elements are data containers called *cells*.

- Each cell can contain any data type.

Example:

```
c = {"abcde", 1, [3 4], [1 2; 3 4]}

c =

  1×4 cell array

    {["abcde"]}    {[1]}    {[3 4]}    {2×2 double}
```

- To access a cell (element) of a cell array use curly braces

```
>> c{3}

ans =

     3     4
```

2 Creating Cell Arrays

1. Create a cell array with 5 elements. The first cell is a 1×1 array with random integers, the second cell is a 2×2 array with random integers, the third cell a 3×3 array with random integers, the fourth cell a 4×4 array with random integers and the last cell a 5×5 array with random integers.
2. Create the same cell array as in the previous task, but this time using a `for` loop.

3 Build a Manipulator Robot using the Matlab Robotics Toolbox

Use the Denavit-Hartenberg (DH) parameters of the Puma560® manipulator robot to incrementally build a rigid body tree robot model. Specify the relative DH parameters for each joint as you attach them. Visualize the robot frames, and interact with the final model.

As we have seen, the DH parameters define the geometry of how each rigid body attaches to its parent via a joint. The parameters follow a four transformation convention:

- r — Length of the common normal line between the two z-axes, which is perpendicular to both axes
- α — Angle of rotation for the common normal
- d — Offset along the z-axis in the normal direction, from parent to child
- θ — Angle of rotation for the x-axis along the previous z-axis

Specify the parameters for the Puma560 robot using the following matrix:

```
dhparams = [0           pi/2    0        0;
            0.4318      0       0        0
            0.0203      -pi/2   0.15005  0;
            0           pi/2    0.4318   0;
            0           -pi/2   0        0;
            0           0       0        0];
```

Create a rigid body tree object.

```
robot = rigidBodyTree;
```

Create a cell array for the rigid body object, and another for the joint objects. Iterate through the DH parameters performing this process:

1. Create a `rigidBody` object with a unique name.
2. Create and name a revolute `rigidBodyJoint` object.
3. Use `setFixedTransform` to specify the body-to-body transformation of the joint using DH parameters. The function ignores the final element of the DH parameters, `theta`, because the angle of the body is dependent on the joint position.
4. Use `addBody` to attach the body to the rigid body tree.

```
bodies = cell(6,1);
joints = cell(6,1);
for i = 1:6
    bodies{i} = rigidBody(['body' num2str(i)]);
    joints{i} = rigidBodyJoint(['jnt' num2str(i)], "revolute");
    setFixedTransform(joints{i}, dhparams(i,:), "dh");
    bodies{i}.Joint = joints{i};
    if i == 1 % Add first body to base
```

```
        addBody(robot, bodies{i}, "base")
    else % Add current body to previous body by name
        addBody(robot,bodies{i}, bodies{i-1}.Name)
    end
end
```

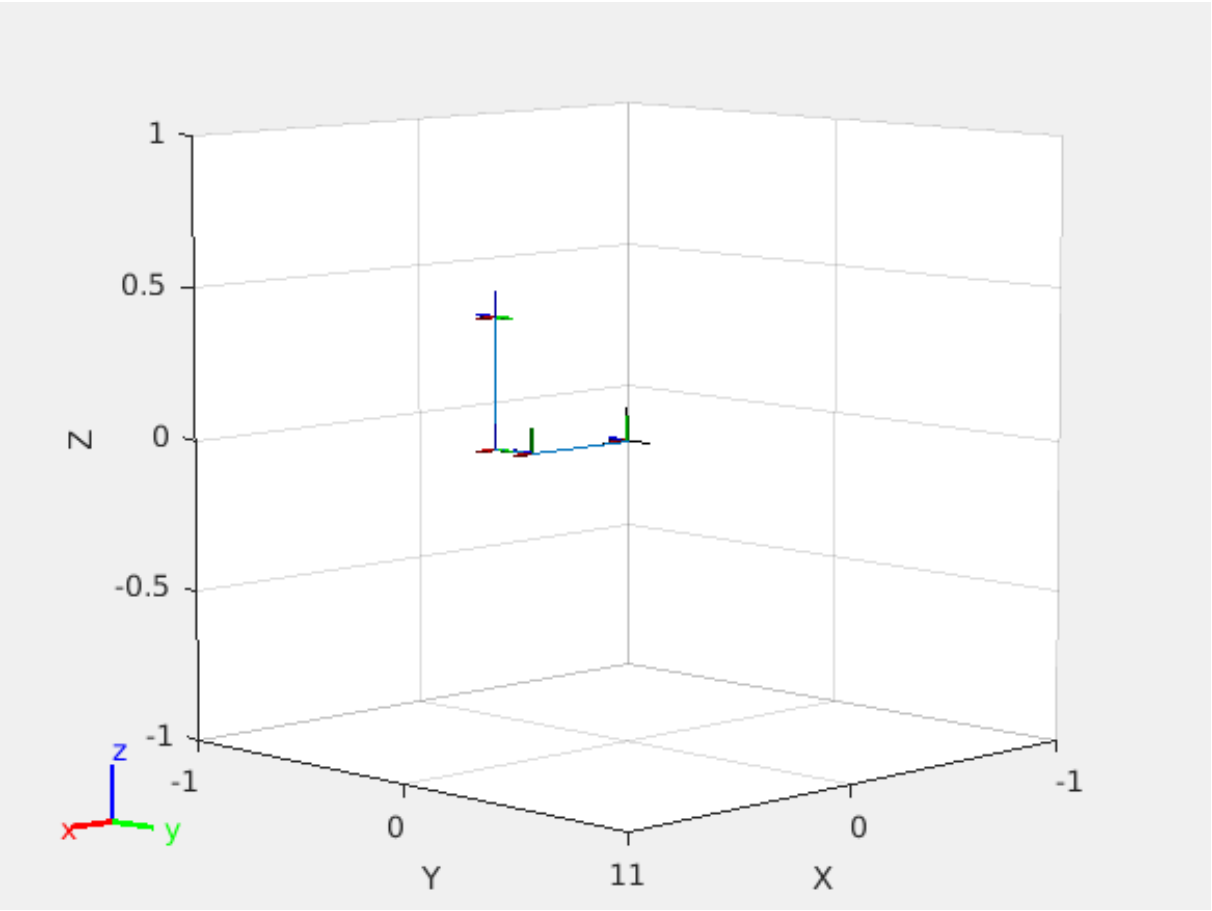
Verify that your robot has been built properly by using the `showdetails` or `show` function. The `showdetails` function lists all the bodies of the robot in the Matlab command window. The `show` function displays the robot with a specified configuration (home by default).

```
showdetails(robot)
```

```
--------------------
Robot: (6 bodies)

 Idx    Body Name   Joint Name   Joint Type    Parent Name(Idx)   Children Name(s)
 ---    ---------   ----------   ----------    ----------------   ----------------
   1        body1         jnt1     revolute              base(0)   body2(2)
   2        body2         jnt2     revolute           body1(1)   body3(3)
   3        body3         jnt3     revolute           body2(2)   body4(4)
   4        body4         jnt4     revolute           body3(3)   body5(5)
   5        body5         jnt5     revolute           body4(4)   body6(6)
   6        body6         jnt6     revolute           body5(5)
--------------------
```

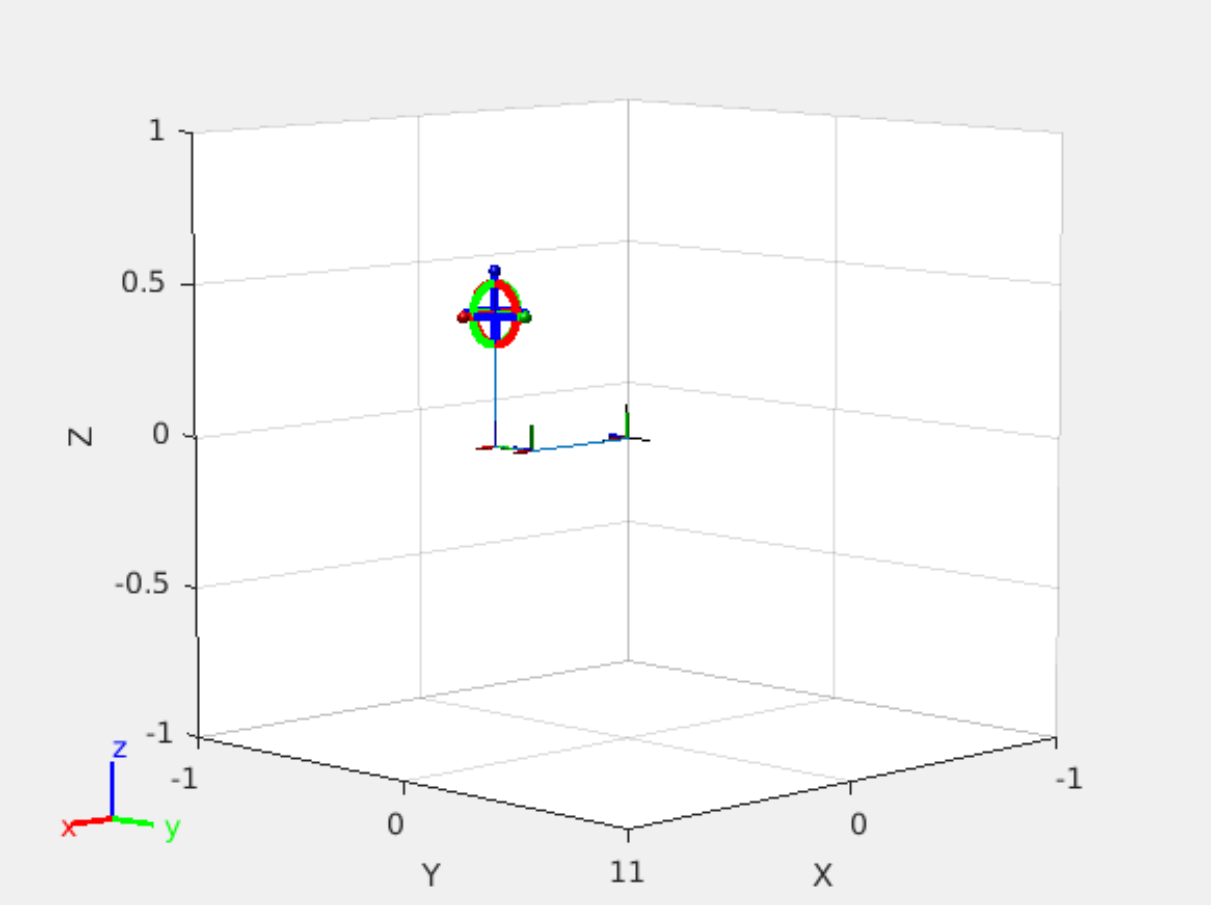
```
figure(Name="PUMA Robot Model")
show(robot);
```



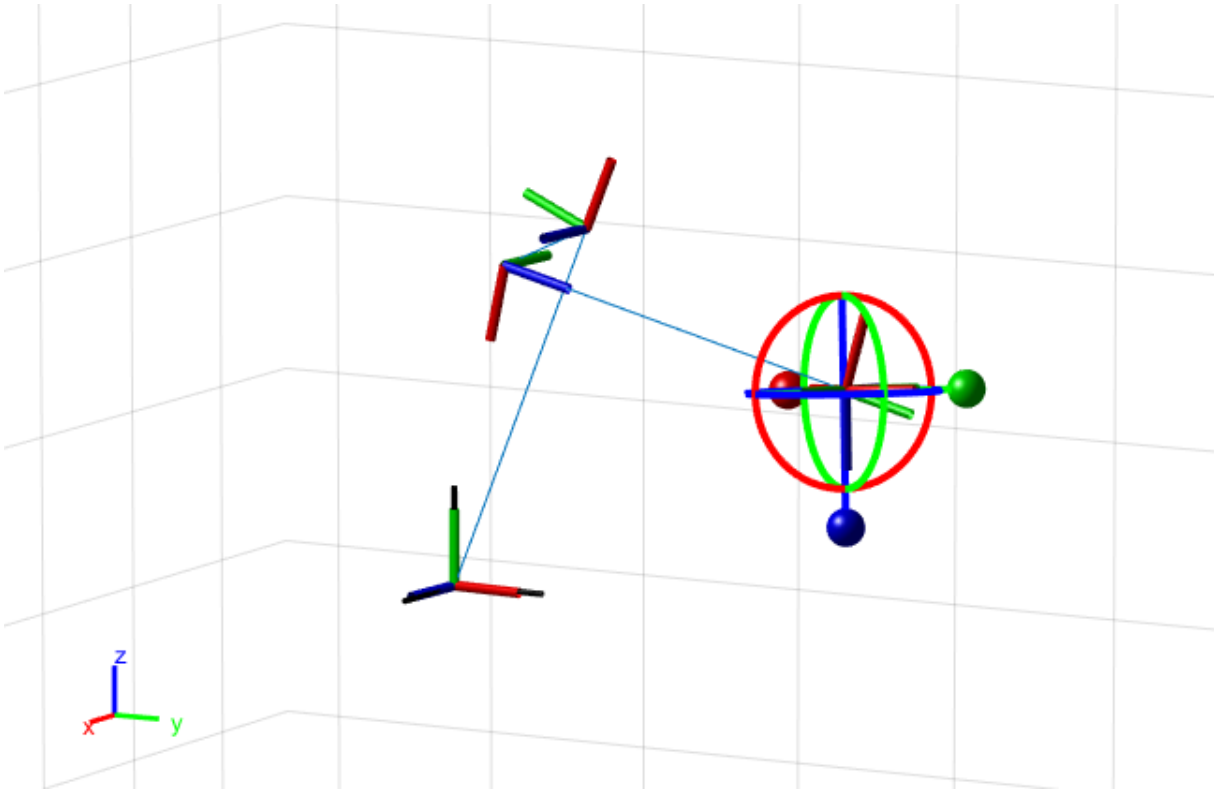
Interact with the Robot Model

Visualize the robot model to confirm its dimensions by using the `interactiveRigidBodyTree` object.

```
figure(Name="Interactive GUI")
gui = interactiveRigidBodyTree(robot,MarkerScaleFactor=0.5);
```



Click and drag the marker in the interactive GUI to reposition the end effector. The GUI uses inverse kinematics to solve for the joint positions that achieve the best possible match to the specified end-effector position. Right-click a specific body frame to set it as the target marker body, or to change the control method for setting specific joint positions.



4 The Stanford Arm

The Stanford arm robot manipulator shown in Figure 1 consists of 6 joints. The first two joints are revolute, the third is prismatic, and the last three wrist joints are all revolute joints. The home position of the arm is when it is vertical.

The home (reset) position of the robot is when the robot is pointed up and vertical.

The DH table of the Stanford robot manipulator is given by:

Joint	$\boldsymbol{\theta}$	$\boldsymbol{r}$	$\boldsymbol{d}$	$\boldsymbol{\alpha}$
1	θ_1	0	0	-90^o
2	θ_2	0	d_2	90^o
3	0	0	d_3	0
4	θ_4	0	0	-90^o
5	θ_5	0	0	90^o
6	θ_6	0	0	0

1. Using similar steps with the previous Exercise, build a rigid body tree robot Matlab model.
2. Visualise the model built by creating an `interactiveRigidBodyTree` object.
3. Using the Matlab function you created in Exercise 5 of the Tutorial 7, calculate the pose of the end-effector with reference to the base frame. Use the following values: $\theta_1 = -45^o$, $\theta_2 = 90^o$, $\theta_4 = -45^o$, $\theta_5 = 135^o$, $\theta_6 = 90^o$, $d_2 = 5$, $d_3 = 10$.

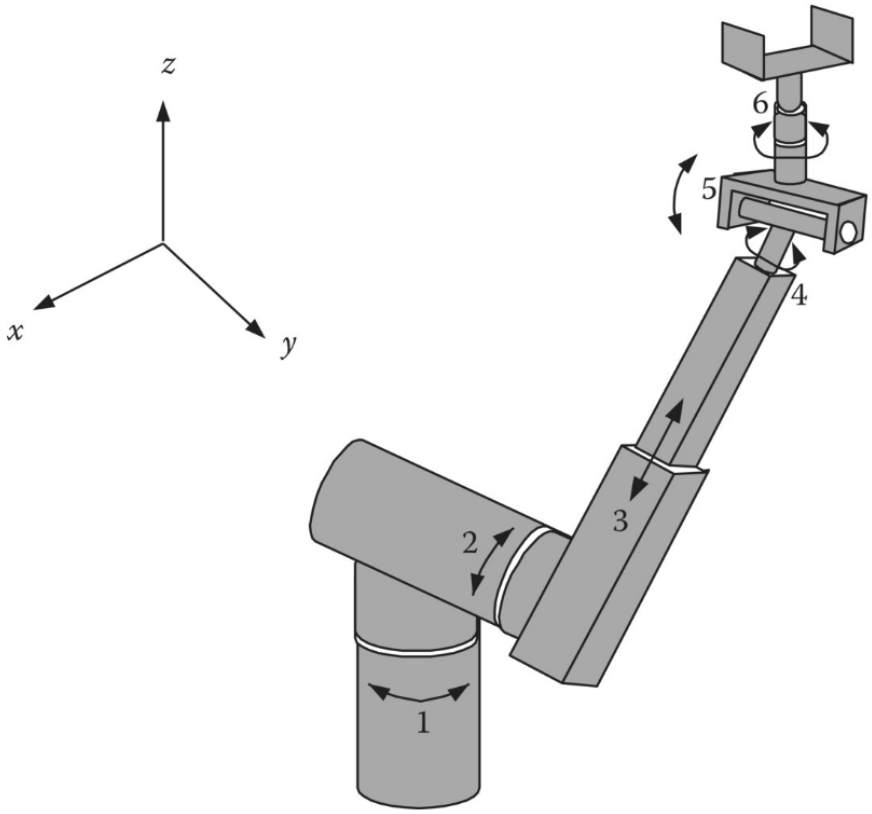


Figure 1: The Stanford arm used in Exercise 4.