

## Robot Simulation in Matlab

This tutorial will discuss some of the basic steps in creating a virtual robot control and simulation environment in Matlab. This tutorial consists of several parts:

- 1) 3D graphics (drawing robot primitives) in Matlab using the patch command
- 2) Manipulating these primitives using the transformation operations (rotation and translation)
- 3) Animating a robot move
- 4) Path/Trajectory generation
- 5) Inverse kinematic control of the virtual robot

### Part 1: Using 3D graphics in Matlab

Matlab has several convenient tools to create 3D graphics, in general built on window open GL. The command we will use is the patch command. Everything you would ever want to know about patch can be found in the matlab helpdesk. We will use the following form of the patch command to draw an object:

```
>patch('Vertices',vertices_link1,'Faces',faces_link1,'faceColor',[RGB values])
```

'Vertices': vertices\_link1 is a matrix that contains all the vertices used to describe the object. The size of vertices\_link1 is  $n \times 3$  where  $n$  is the number of vertices, 3 is the number of coordinates for each vertex ( $x,y,z$ ). Each row corresponds to a vertex and each column to its  $x,y,z$  coordinate.

'Faces': face\_link1 is a matrix that defines each face of the object in terms of the vertices that lie on the face. Each row of this matrix corresponds to a face, the columns contain the vertices that define the face.

'FaceColor': gives one means of defining the coloring of the object. It can be in terms of the faces, vertices, etc., or a single overall color (as used here). The examples that follow use a single color defined in terms of RGB values.

Now for an example: First we will create a cylinder to represent the base of our robot. The cylinder will be represented with a certain number of faces defined by a certain number of vertices. It helps to sketch your primitive and label the vertices and faces as in figure 1:

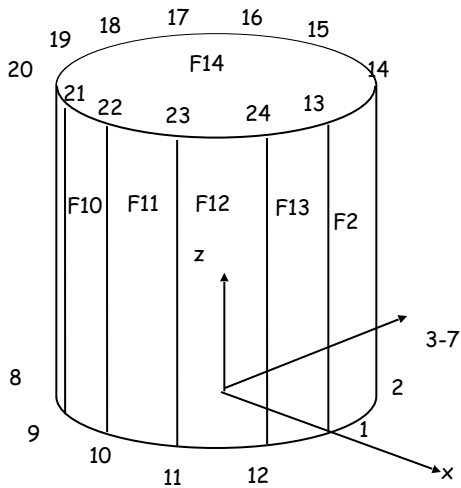


Figure 1: Link 1, a cylinder

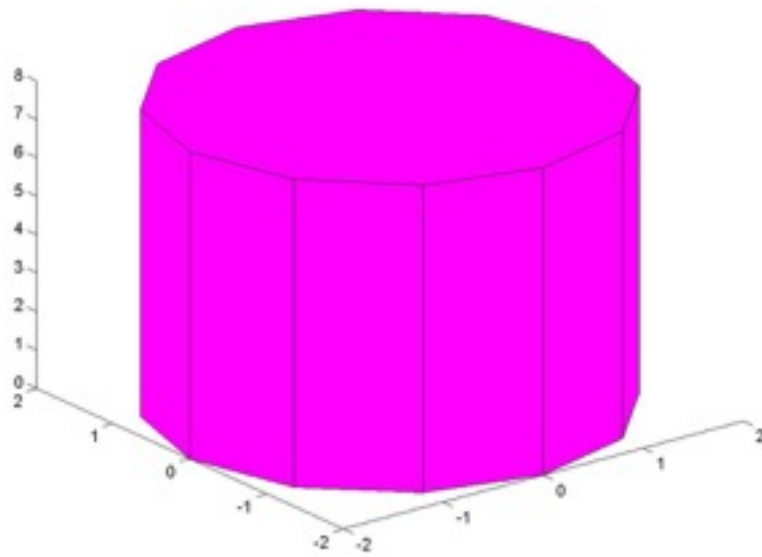
This cylinder will be defined in parametric fashion based on radius ( $r1$ ) and length (11).  
The vertices matrix will look something like:

```
vert_link1=[ r1*cos(0), r1*sin(0), 0
             r1*cos(30), r1*sin(30), 0
etc
```

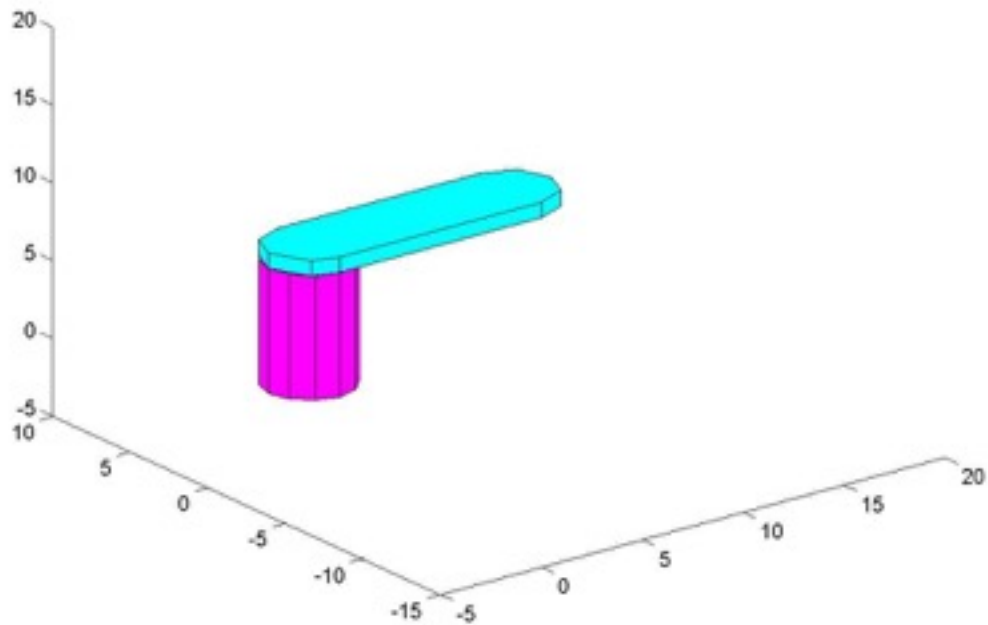
The faces matrix will look like:

```
Face_link1=[1,2,3,4,5,6,7,8,9,10,11,12
            1,2,14,13,13,13,13,13,13,13,13,13
etc
```

These are defined in matlab, and plotted using the patch command to yield the following result:



A second link is created, this one looks like a bar with rounded ends, with parameters  $r_2$  (end radius),  $l_2$  (bar length)  $t_2$  (bar thickness) to yield the following result (plotted with the cylinder):



One final note is made in defining these objects. Each object is defined relative to its own frame and has an origin location. Subsequent rotation operations will act on this frame and about this origin. Therefore, locate the origin and z axis of the body to align with a joint that could be envisioned on the body.

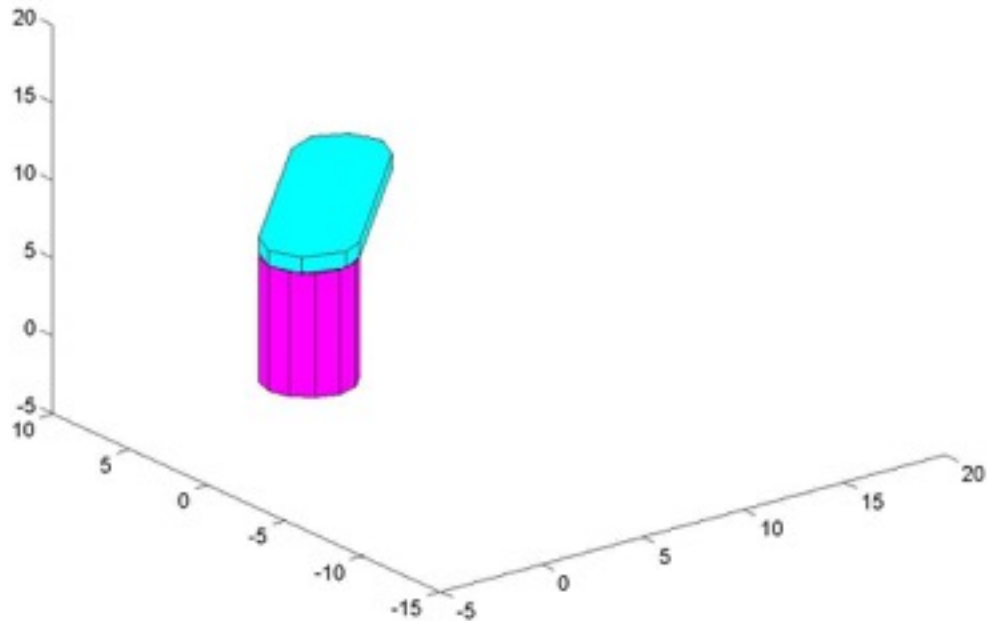
An example of object drawing for a cylinder and rounded bar is shown in the attached code, draw\_robot.m

### Part 2: Manipulating 3D objects in Matlab:

Once the objects are created, they can easily be manipulated by operating on the vertices with rigid body transformations (rotation + translation). Each vertex (corresponding to each row in the vertex matrix) is operated on. A few lines of pseudo code demonstrating this process follow:

```
For i=1 to # of rows of vertex_link1
    vertex_link1(i,:) = (R * vertex_link_o(i,:))' + d
end
```

where **R** is 3x3 rotation matrix, **d** is the 3x1 translation vector, and the ' indicates transpose. Note that the rotation operation provides a rigid body rotation to all vertices about the origin of the object, and the translation provides a rigid body translation or offset to all points. Also note that we have acted on the original vertices to create a new or current vertex matrix. A rotation of 45 degrees is applied to the bar (link 2) and plotted as shown in the following figure.



An example of object manipulation is shown in the attached code, draw\_robot.m

### Part 3: Animating a robot move

Animation is simply a sequence of rigid body motions performed in a continuous manner. This can be performed by embedding the transformation and drawing commands in a loop that slowly increments the amount of rotation/translation. A few side notes on animation;

- 1) 15 frames per second will provide a smooth animation
- 2) The pause(.075) command should force the current plot to show and create a pause of .075 seconds
- 3) Use the hold on command to plot multiple parts of a robot. Use the hold off to force new plotting commands to redraw the screen

### Part 4: Path/Trajectory generation

A nice feature you can add to your program would plot the path of end effector over a given move. This can be performed by saving the end effector positions in a matrix (each row a new position) and then using the plot3 command to plot these points. If you plot the current tool position matrix each time in your animation, you can generate the trajectory curve in real time.

### Part 5: Inverse Kinematic control

With the inverse kinematics for a manipulator solved, you can use these equations to find the inputs (joint angles) necessary to plot your robot. You can define a trajectory in terms of a tool space move, say a straight-line move of the end effector, and then animate your robot over this move. As an example, say you define a 200 cm straightline move for the tool to occur over 5 seconds. You would like to show 10 frames per second. Divide the total move by the total number of frames to show ( $200/(5*10) = 4$ ) to get the step size in tool space. Start the tool at 0, calculate the IK's and draw the robot. Pause .1 seconds, move the tool to 4 cms, calculate the IK's and redraw the robot. Continue until you have reached the final position of 200 cm.

Assignment:

Create a virtual robot in Matlab based on the SCARA configuration (you may choose another robot, with a min. of 4 dof). Your first task is to write a program to allow a user to type values for theta 1,2,4 and d3, and then plot the robot.