Week 4: DH Table

Author: Keith Chester

E-mail: kchester@wpi.edu

Table of Contents

Introduction

In the fourth week of *RBE 500 - Foundations of Robotics*, we learn the application of the Denavit–Hartenberg (DH) Convention. From this method, we can generate a DH Table, which allows us to quickly dervie the transformations between coordinate frames of the joints and actuator of a robot arm, and thus its forward kinematics.

IV-1

In our first problem, we are tasked with taking the robot demonstrated in the figure below, and analyze its motion by moving its links a prescriebd amount - θ_1 from 0 to 180 degres, and d_2 and $d_3$from 0 to 0.1 meters in sequence.

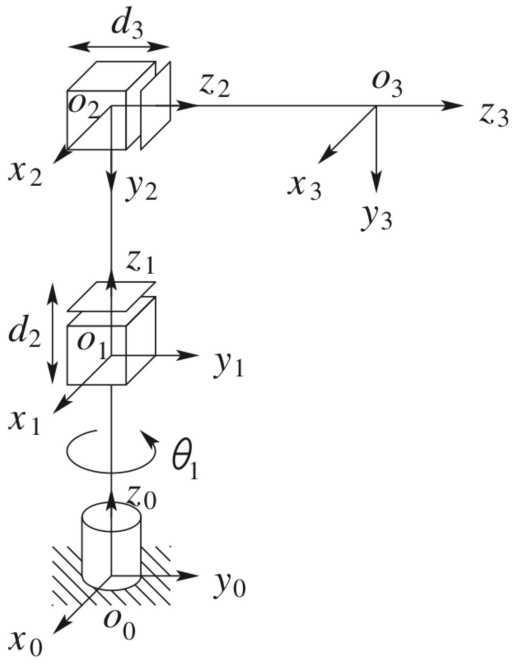


Figure 3.7: Three-link cylindrical manipulator.

Figure 1: A three link cylindrical manipulator from Robot Modeling and Control, 2nd edition, page 90

After analyzing its motion, we animate the arm using the transforms dervied in a prescribed manner per the problem to demonstrate understanding of its motion through the DH method.

IV-2

For this problem, we need to consider the **RRR** robot arm presented in the following figure.

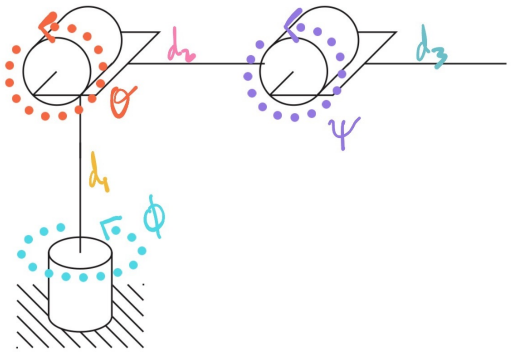


Figure 3.16: Three-link articulated robot.

Figure 2: Figure 3.16 from Robot Modeling and Control, 2nd edition, page 90 with additional markings to denote variables and axis

The above figure was additionally marked to add in ϕ, θ and ψ as joint angles, and d_1, d_2, and d_3 as specified lengths as well. From this figure, we need to apply the DH procedure to generate a DH table, and demonstrate an analysis from o_i as we change the motion parameters.

IV-3

For our third problem, we will be again looking at the Denso HSR Robot and specifications listed in their robot catalog.

Figure 3: Denso HSR Robot from product catalog

We will be utilizing the specifications and technical drawings in the catalog to build a DH table, and then analyze the movement of the arm in an animation by our derived transformations.

Methods

In this section, we discuss how we approached and performed each problem.

IV-1

Referencing *Figure 1*, we can see that coordinate axes have already been assigned. We follow these axes to determine the values for the DH convention. Thankfully the robot arm as presented follows the DH convention and thus is easy to determine the values of the DH table.

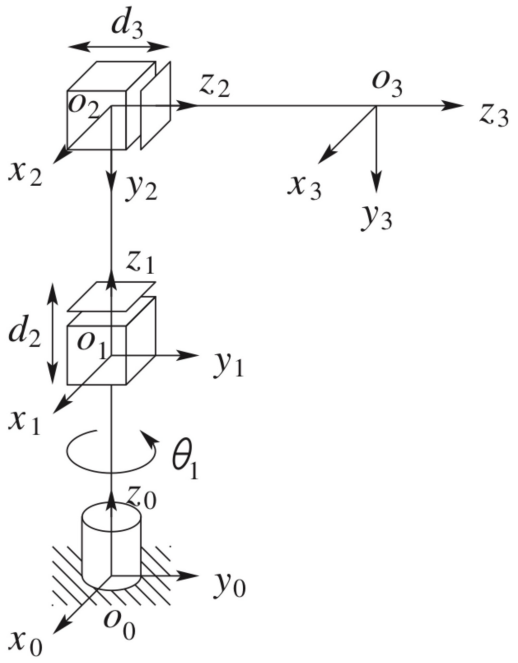


Figure 3.7: Three-link cylindrical manipulator.

Figure 1: Reposted for reference

1. In our first axes, o_0 to o_1, we see that we have a potential rotation around the z-axis (θ) of prescribed term **theta1.** Thus our θ is **theta1**. The z axes for both frames are directly above one-another, denoting a **0** *a* term. The two axes are apart z-axis by assumed length of **d1**. Finally, there is no rotation around the x-axis between the two frames, so α is 0.
2. From frame o_1 to o_2, we see no rotation in the z axis, denoting a **0** for θ. The coordinate axis does see a rotation around the x-axis, however, resulting in a **-90** degree α. There is no difference in position along the x-axis, so *a* is **0**. The prismatic joint in o_2 moves along the z-axis at a variable distance of **d2**, so that is the *d* value.
3. From frame o_2 to o_3, we see no movement or rotation along the x-axis, so *a* and α are both **0**. We move along the z-axis based on the variable length of the prismatic joint **d3**, so that is our *d* value. We see no rotation along the z-axis, so θ is **0**.

Link	Θ	d	a	α
1	theta1	d1	0	0
2	0	d2	0	-90
3	0	d3	0	0

Figure 4: DH Table for IV-1

From this table we can determine a transformation for any given link (and its coordinate frames from o_{i-1} to o_i via the following equation:

$$A = \text{Rot}_{z,\theta}\text{Trans}_{z,d}\text{Trans}_{x,a}\text{Rot}_{x,\alpha}$$

Equation 1: Transformation Matricies from a DH Table row

A is thus the transformation matrix for each endpoint of a given link, where link i represents the connection between frames o_{i-1} and o_i. For example, the following transformation matricies are created from the table above:

$$\begin{aligned}
A_1 = H_1^0 &= \text{Rot}_{z,\theta}\text{Trans}_{z,d}\text{Trans}_{x,a}\text{Rot}_{x,\alpha} \\
&= \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

Equation 2: Calculating a transformation matrix from Figure 4's link 1

After performing this for each link, we have the transformation between each frame moving from base to actuator. In order to animate it, we must calculate the H_i^0 transform for each axis, which can be caclulated by just multiplying it by prior transforms within the path to the actuator. From that point on we can determine the plottable location of each point within frame o_0. Thus we get the following transformations:

$$A_1 = H_1^0 = \begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_2 = H_2^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90) & \sin(90) & 0 \\ 0 & -\sin(90) & \cos(90) & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_3 = H_3^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H_2^0 = H_1^0 H_2^1$$
$$H_3^0 = H_1^0 H_2^1 H_3^2$$

Equation 3: Calculating transforms from Figure 4, using Equation 2's approach

For each transformation calculation we used the methods described in *Equation 2*.

IV-2

To analyze the setup of this **RRR** arm, we first look at *Figure 2* and then expand upon it in order to define the coordinate axes for our DH table derivation. Here, we place the axes on the robot with the following rules:

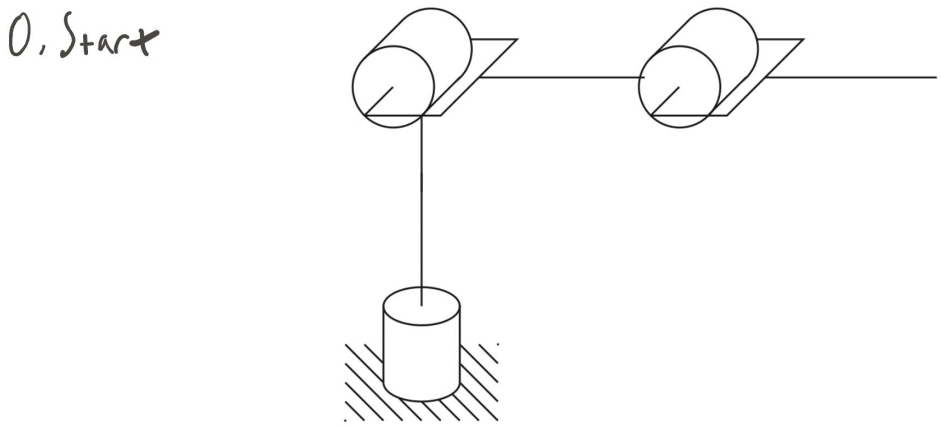


Figure 3.16: Three-link articulated robot.

Figure 4: DH Convention applied to RRR arm

1. While not required for the DH Method, we attempt to place the axis in line with the actual point of rotation. Thankfully for this arm, this method works out.
2. The z-axis is defined parallel to the motion of rotation for each revolute joint.
3. We set the x-axis to point towards the next frame if possible, perpendicular to the z-axis.
4. The y-axis is set appropriately acording to the right-hand rule.
5. We place a frame on the endpoint of the actuator to coplete the final link.

From this, we can use the DH method to determine values for the DH table. We define first the axis rotations of each frame, from base to actuator, as ϕ, θ, and ψ respectively. The static lengths of the linkages are defined as static variables d_1, d_2, and d_3 respectively. Using this, we can determine for the links that:

1. The z-axis rotates according to **phi**, so this is θ. The next frame o_0 is directly above the z-axis, resulting in a **0** *a*. The distance between o_0 and $o_1$in the z-direction is defined statically as variable **d1**, so that will be d. Finally, the frame is rotateable in the x-axis by **90** degrees, so that is α.

2. The z axis rotates by **theta** for the value of θ. The x-axis points towards the next frame, and along this frame the distance is defined as the static variable **d2** - so this is the value for *a*. There is no change in alignment between the coordinate frames in the z-axis, so *d* is **0**. Finally, no rotation occurs in the x-axis to orient the frames together, so α is **0**.
3. Between these two frames acts exactly as the prior frame, the only difference being the static variable **d3** is used for *a* and **psi** for θ.

Thus, our resulting DH table is:

Link	Θ	d	a	α
1	phi	d1	0	90
2	theta	0	d2	0
3	psi	0	d3	0

Figure 6: DH Table for IV-2

Much like our previous problem, we can solve for the transformation matricies needed to plot and show this figure. Using *Equation 1*, we can find the transformation matrix for each link, and then multiply them together to determine the H_i^0 transformation for each frame.

$$A_1 = H_1^0 = \begin{pmatrix} \cos(\phi) & 0 & sin(\phi) & 0 \\ \sin(\phi) & 0 & -\cos(\phi) & 0 \\ 0 & \sin(90) & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_2 = H_2^1 = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & d_2\cos(\theta) \\ \sin(\theta) & \cos(\theta) & 0 & d_2\sin(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

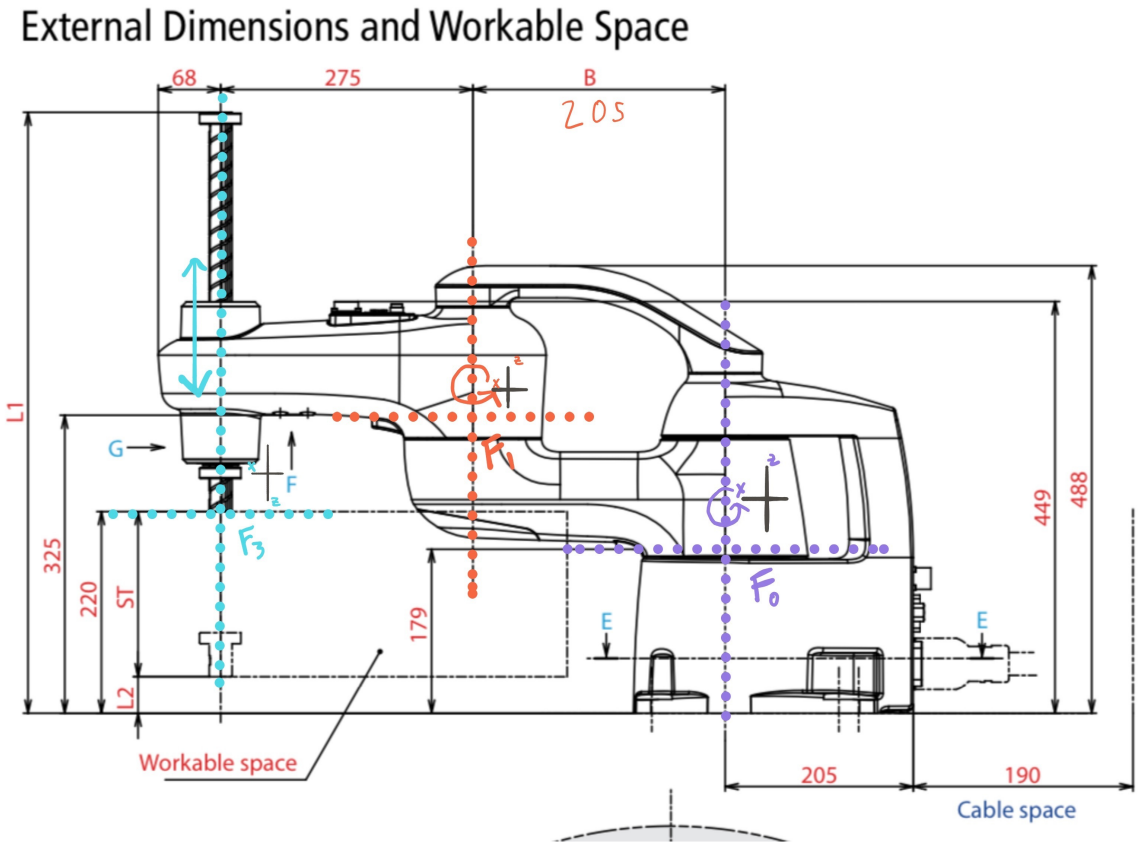
$$A_3 = H_3^2 = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 & d_3\cos(\psi) \\ \sin(\psi) & \cos(\psi) & 0 & d_3\sin(\psi) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H_2^0 = H_1^0 H_2^1$$
$$H_3^0 = H_1^0 H_2^1 H_3^2$$

Equation 4: Generating the transforms from Figure 6 via Equation 1

IV-3

For this problem, we look at the technical drawings of within the robot arm catalog for Denso, and derive key measurements from the specified model (the 480mm stroke version of the arm):



Model	A	B	C	D	E
HSR048A1-N*	480	205	164.4	287°	406.53
HSR055A1-N*	550	275	142.4	300°	364.32
HSR065A1-N*	650	375	194.0	300°	287.62

Figure 5: The Denso HSR arm measurements and specifications

From this we can build a simplistic model representation, and place our axes according to DH convention.

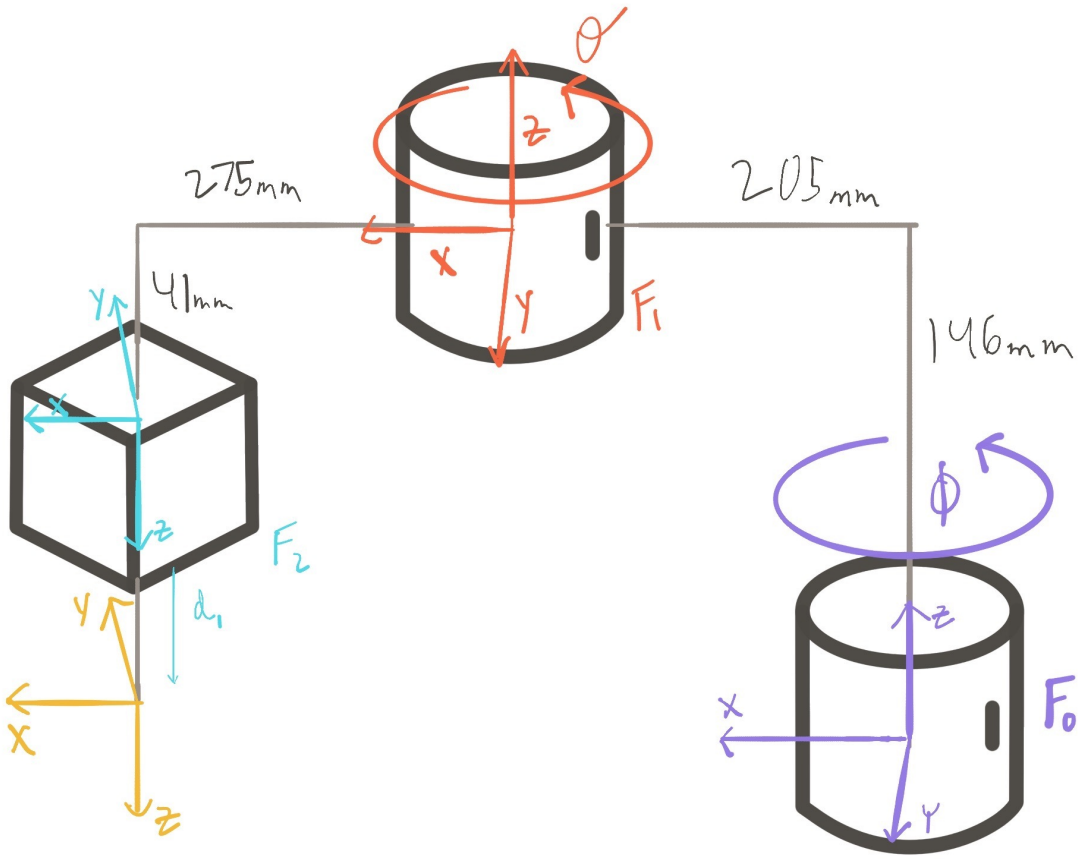


Figure 6: The Denso HSR Robot arm as a simplified model representation

Here, we place our axes for each of the frames according to DH convention:

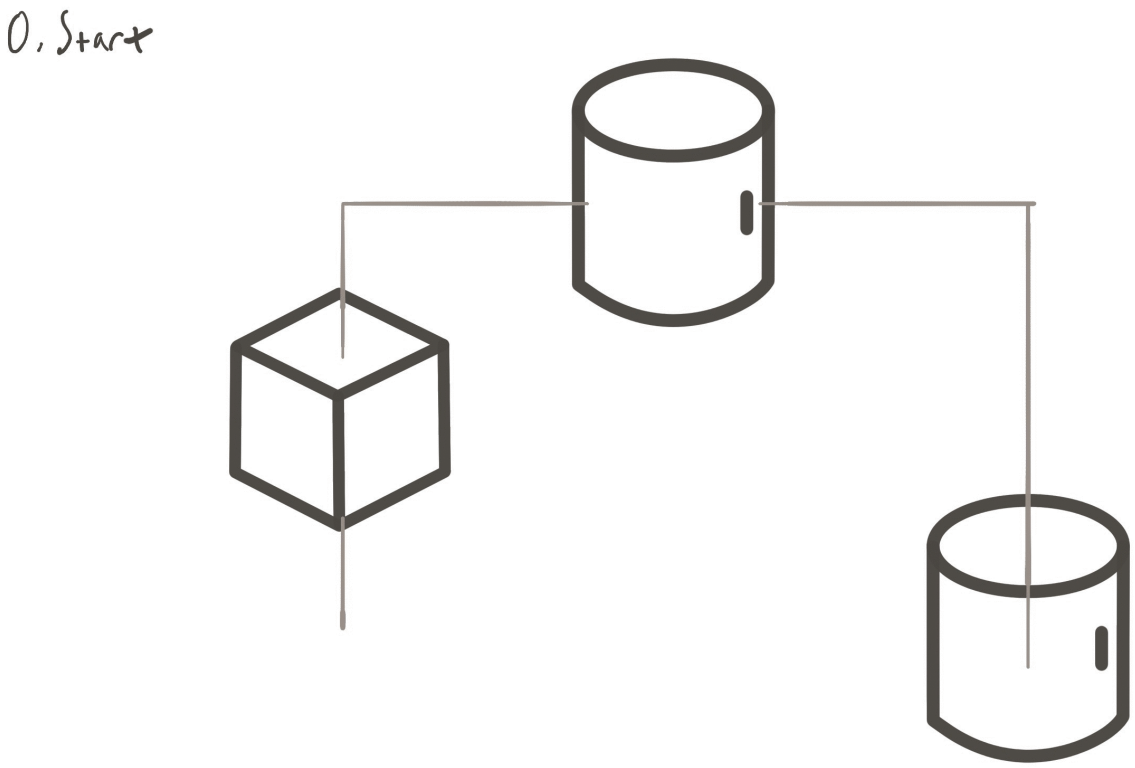


Figure 7: Applying DH Convention to drawing axis for the Denso HSR

1. For frame o_0, we place the z-axis in line with the axis of rotation itself. We place the x-axis perpendicular to the z-axis frame, pointing towards the next frame. The y-axis is placed according to the right-hand rule.
2. For frame o_1, we place the z-axis in line with the axis of rotation itself. We orient it in the same manner as the prior frame to simplify the math around the transformations. We place the x-axis perpendicular to the z-axis frame, pointing towards the next frame along the 275mm link. The y-axis is placed according to the right-hand rule.
3. For frame o_2, we place the z-axis in line with the prismatic joint's movement, oriented in the direction of its movement. Most notably this means a 180 degree rotation on the x to point down vs previous frame's z-axis pointing up. We place the x-axis perpendicular to the z-axis frame, pointing in the same direction past x-axis have. The y-axis is placed according to the right-hand rule.
4. For frame o_3, we perform no rotations or displacements from the prior frame, save for the displacement along the z-axis for the prismatic joint between its base and the tip of the actuator. This distance (specified by **d1**) is the sole change in the frame.

Finally, once we have defined these axes, we can use our axes and specified measurements in *Figure 8* to create our DH table:

Link	Θ	d	a	α
1	phi	146	205	0
2	theta	-41	275	180
3	0	d1	0	0

Figure 9: The derived DH table from Figure 8.

From here, we can gather a transform between each frame - H_1^0, H_2^1, and H_3^2. We can determine H_2^0 and H_3^0 as well, which is needed to animate the robotic arm within the context of our world frame (which is o_0).

This, our resulting transformations. These are then used to perform the plotting necessary to animate the arm.

$$A_1 = H_1^0 = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 & 205\cos(\phi) \\ \sin(\phi) & \cos(\phi) & 0 & 205\sin(\phi) \\ 0 & 0 & 1 & 146 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_2 = H_2^1 = \begin{pmatrix} \cos(\theta) & \sin(\theta) & -\sin(\theta) & 275\cos(\theta) \\ \sin(\theta) & -\cos(\theta) & \cos(\theta) & 275\sin(\theta) \\ 0 & 0 & -1 & -41 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_3 = H_3^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$H_2^0 = H_1^0 H_2^1$$
$$H_3^0 = H_1^0 H_2^1 H_3^2$$

Equation 5 Transformations derived from Figure 9 via Equation 1

Results

IV-1

From the methods described above, we have a DH table defined as

```
syms theta1 d1 d2 d3
dh_table = {  %theta        %d          %a      %alpha
            [  theta1        d1          0       0       ],
            [   0             d2          0       -90     ],
            [   0             d3          0       0       ]
           };
```

Which we can utilize the **transform_from_dh_row** function, defined and described below in the **Functions** section,

```
A_1 = transform_from_dh_row(dh_table, 1, true)
```

A_1 =

$$\begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
A_2 = transform_from_dh_row(dh_table, 2, true)
```

A_2 =

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90) & \sin(90) & 0 \\ 0 & -\sin(90) & \cos(90) & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
A_3 = transform_from_dh_row(dh_table, 3, true)
```

A_3 =

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
H_0_3 = A_1 * A_2 * A_3
```

H_0_3 =

$$\begin{pmatrix} \cos(\theta_1) & -\cos(90)\sin(\theta_1) & -\sin(90)\sin(\theta_1) & -d_3\sin(90)\sin(\theta_1) \\ \sin(\theta_1) & \cos(90)\cos(\theta_1) & \sin(90)\cos(\theta_1) & d_3\sin(90)\cos(\theta_1) \\ 0 & -\sin(90) & \cos(90) & d_1 + d_2 + d_3\cos(90) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Once we have these transforms, we can create a range of valus for the values of **theta1**, **d2**, and **d3**, and for those values generate the necessary values for each transform at each stage. From that, we can then plot the resulting points for an animation of the robot arm in action, as demonstrated below.

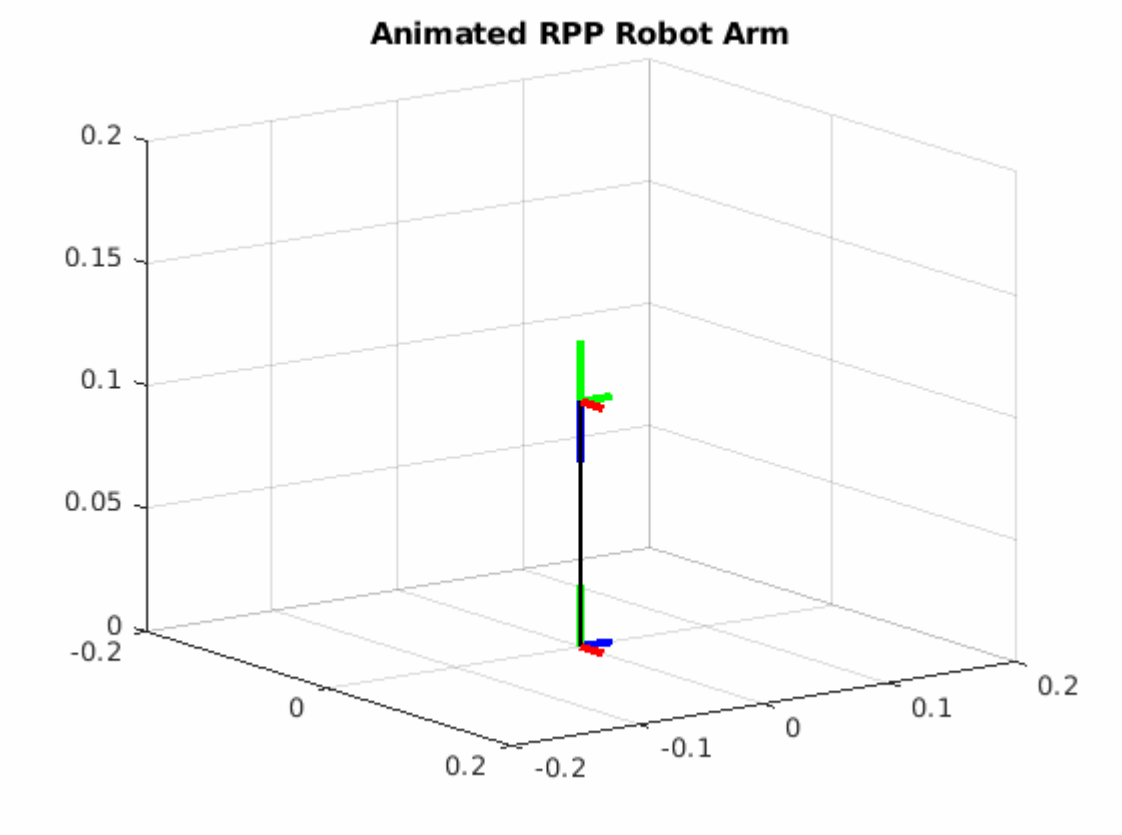


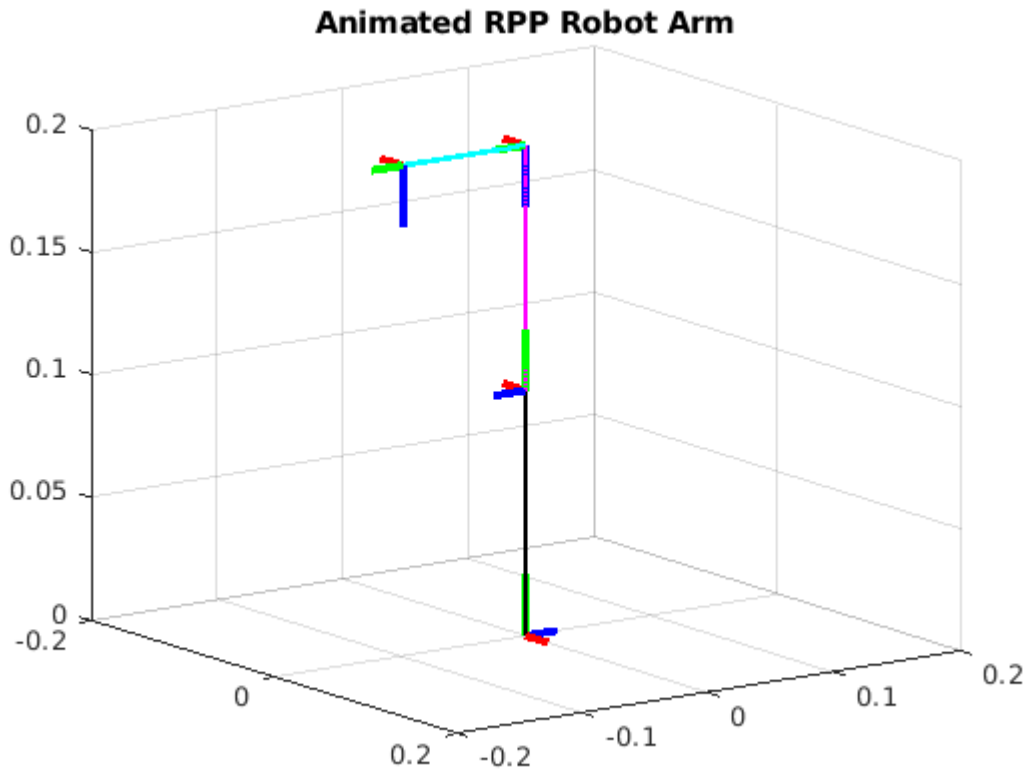
Figure 8: The animated RPP Robot Arm

The below code will run the animation pictured here. It will first generate all values for each step, and then work to draw the animation.

```
clf
animated_figure = figure;
title("Animated RPP Robot Arm")
axis([-.2 .2 -.2 .2 0 0.2])
grid on
hold on
lines = [];
origin = [0; 0; 0; 1;];
view([54 16])

animation_length_seconds = 5;
frames_per_second = 24;
steps = animation_length_seconds * frames_per_second;

theta1_delta = (180/steps)*3;
d2_delta = (0.1/steps)*3;
d3_delta = (0.1/steps)*3;

theta1 = 0;
d1 = 0.1;
```

```
d2 = 0;
d3 = 0;

H_0_1s = {};
H_0_2s = {};
H_0_3s = {};

% In this section, we will pre-generate all of the transforms for each step
% of the animation
for step = 1:steps
    % Create our d_h_table for the instance
    dh_table = { %theta        %d          %a      %alpha
                 [theta1        d1          0       0   ],
                 [0             d2          0       -90 ],
                 [0             d3          0       0   ]
               };

    % Get H_0_1, H_1_2, and H_2_3 from the dh table
    H_0_1 = transform_from_dh_row(dh_table, 1);
    H_1_2 = transform_from_dh_row(dh_table, 2);
    H_2_3 = transform_from_dh_row(dh_table, 3);

    % Find H_N_0 to get everyting back to the 0 frame
    H_0_2 = H_0_1 * H_1_2;
    H_0_3 = H_0_2 * H_2_3;

    % Save our coordinate frames
    H_0_1s{end + 1} = H_0_1;
    H_0_2s{end + 1} = H_0_2;
    H_0_3s{end + 1} = H_0_3;

    % Increment per our calculated deltas, but only if it's the third of
    % the animation for that link's sequential movement
    if step <= steps / 3
        theta1 = theta1 + theta1_delta;
    elseif step <= 2 * steps / 3
        d2 = d2 + d2_delta;
    else
        d3 = d3 + d3_delta;
    end

end

% Finally, with all of the transforms calculated, we shall move on and
% animate each step!
for step = 1:steps

    % Clear previous lines
    for index = 1:length(lines)
        delete(lines(index));
```

```
    end
    lines = [];

    [x_line, y_line, z_line] = draw_coordinate_axis(eye(4), 0.025, 'red', 'blue', 'gree
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;

    [x_line, y_line, z_line] = draw_coordinate_axis(H_0_1s{step}, 0.025, 'red', 'blue',
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;

    [x_line, y_line, z_line] = draw_coordinate_axis(H_0_2s{step}, 0.025, 'red', 'blue',
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;

    [x_line, y_line, z_line] = draw_coordinate_axis(H_0_3s{step}, 0.025, 'red', 'blue',
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;

    % Draw the limbs
    origin_1 = H_0_1s{step} * origin;
    origin_2 = H_0_2s{step} * origin;
    origin_3 = H_0_3s{step} * origin;
    lines(end+1) = line('XData', [origin(1) origin_1(1)], 'YData', [origin(2) origin_1(
    lines(end+1) = line('XData', [origin_1(1) origin_2(1)], 'YData', [origin_1(2) origi
    lines(end+1) = line('XData', [origin_2(1) origin_3(1)], 'YData', [origin_2(2) origi

    pause(1/frames_per_second)

end

hold off;
```



IV-2

In this problem, we will be looking at our generated DH Table for *Figure 5*. First we look at the table with symbolic variables to see how to calculate the transforms and resulting base-to-actuator transformation.

```
syms phi theta psi d1 d2 d3
dh_table = { %theta        %d          %a      %alpha
             [phi          d1          0       90   ],
             [theta        0          d2       0    ],
             [psi          0          d3       0    ]
               };
```

Now that we have the dh_table defined, we'll use **transform_from_dh_row**, defined below in **Functions**, to generate the transform for each linkage per *Equation 1*.

```
H_0_1 = transform_from_dh_row(dh_table, 1, true)
```

```
H_0_1 =
```

$$\begin{pmatrix} \cos(\phi) & -\cos(90)\sin(\phi) & \sin(90)\sin(\phi) & 0 \\ \sin(\phi) & \cos(90)\cos(\phi) & -\sin(90)\cos(\phi) & 0 \\ 0 & \sin(90) & \cos(90) & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
H_1_2 = transform_from_dh_row(dh_table, 2, true)
```

H_1_2 =

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & d_2\cos(\theta) \\ \sin(\theta) & \cos(\theta) & 0 & d_2\sin(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
H_2_3 = transform_from_dh_row(dh_table, 3, true)
```

H_2_3 =

$$\begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 & d_3\cos(\psi) \\ \sin(\psi) & \cos(\psi) & 0 & d_3\sin(\psi) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Thus our end result, once multiplied through, gives us our base to actuator transform:

```
H_0_3 = H_0_1 * H_1_2 * H_2_3
```

H_0_3 =

$$\begin{pmatrix} \cos(\psi)\,\sigma_2 - \sin(\psi)\,\sigma_4 & -\cos(\psi)\,\sigma_4 - \sin(\psi)\,\sigma_2 & \sin(90)\,\sin(\\ \cos(\psi)\,\sigma_3 - \sin(\psi)\,\sigma_1 & -\cos(\psi)\,\sigma_1 - \sin(\psi)\,\sigma_3 & -\sin(90)\,\cos(\\ \sin(90)\cos(\psi)\sin(\theta) + \sin(90)\cos(\theta)\sin(\psi) & \sin(90)\cos(\psi)\cos(\theta) - \sin(90)\sin(\psi)\sin(\theta) & \cos(90) \\ 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = \sin(\phi)\sin(\theta) - \cos(90)\cos(\phi)\cos(\theta)$$

$$\sigma_2 = \cos(\phi)\cos(\theta) - \cos(90)\sin(\phi)\sin(\theta)$$

$$\sigma_3 = \cos(\theta)\sin(\phi) + \cos(90)\cos(\phi)\sin(\theta)$$

$$\sigma_4 = \cos(\phi)\sin(\theta) + \cos(90)\cos(\theta)\sin(\phi)$$

To analyze this arm, we provide controls for each specified variable length and rotation below. Once the values are changed, we perform analysis via the DH table and the values specified to create our transformations, which are in turn used to generate an image of our robot arm in a static position.

```
phi = 0;
theta = -15;
psi = 15;
d1 = 5;
d2 = 3;
d3 = 2;
dh_table = { %theta        %d          %a      %alpha
```

```
            [phi          d1          0       90],
            [theta        0           d2      0 ],
            [psi          0           d3      0 ]
              };
```

Below we run the drawing code to create our image:

```
figure
title("Forward Kinematics of RRR Arm")
grid on
hold on
axis([-7 7 -7 7 0 7])
view([45 25])

origin = [ 0; 0; 0; 1; ];

H_0_1 = transform_from_dh_row(dh_table, 1)
```

```
H_0_1 = 4×4
     1     0     0     0
     0     0    -1     0
     0     1     0     5
     0     0     0     1
```

```
H_1_2 = transform_from_dh_row(dh_table, 2)
```

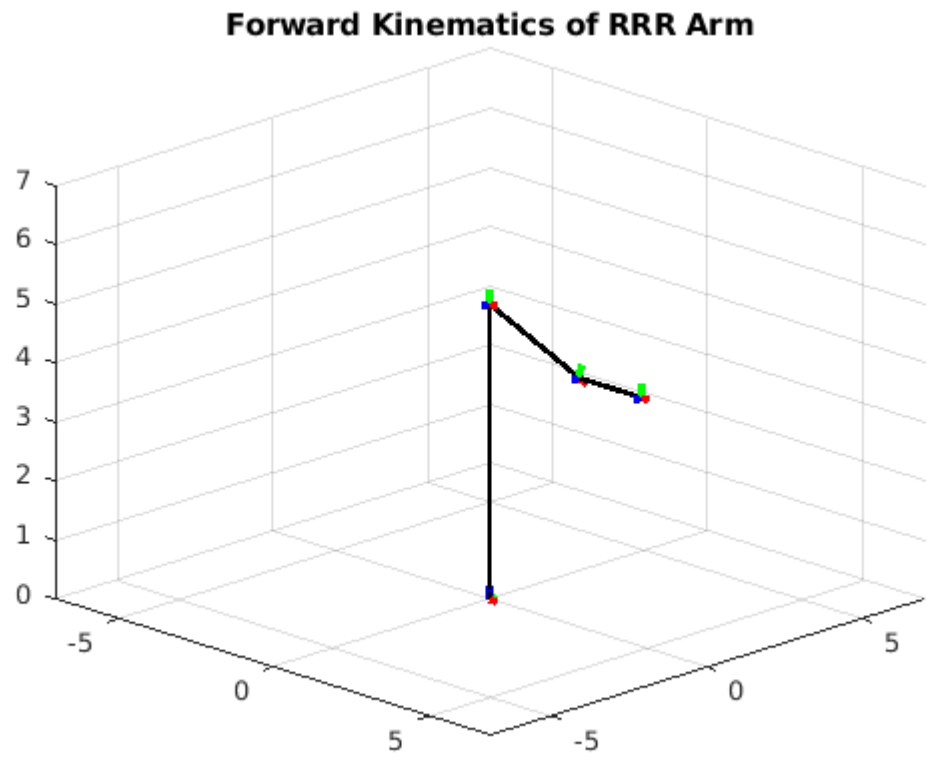
```
H_1_2 = 4×4
    0.9659    0.2588         0    2.8978
   -0.2588    0.9659         0   -0.7765
         0         0    1.0000         0
         0         0         0    1.0000
```

```
H_2_3 = transform_from_dh_row(dh_table, 3)
```

```
H_2_3 = 4×4
    0.9659   -0.2588         0    1.9319
    0.2588    0.9659         0    0.5176
         0         0    1.0000         0
         0         0         0    1.0000
```

```


H_0_2 = H_0_1 * H_1_2;
H_0_3 = H_0_2 * H_2_3;

draw_coordinate_axis(eye(4), 0.25, 'red', 'green', 'blue');
draw_coordinate_axis(H_0_1, 0.25, 'red', 'green', 'blue');
draw_coordinate_axis(H_0_2, 0.25, 'red', 'green', 'blue');
draw_coordinate_axis(H_0_3, 0.25, 'red', 'green', 'blue');

origin_1 = H_0_1 * origin;
origin_2 = H_0_2 * origin;
origin_3 = H_0_3 * origin;

line('XData', [origin(1) origin_1(1)], 'YData', [origin(2) origin_1(2)], 'ZData', [orig
line('XData', [origin_1(1) origin_2(1)], 'YData', [origin_1(2) origin_2(2)], 'ZData', [
```

```
line('XData', [origin_2(1) origin_3(1)], 'YData', [origin_2(2) origin_3(2)], 'ZData', [

hold off;
```



IV-3

Here we look at the symbolic values of our dh_table, using measurements taken from the catalog specifications discussed above.

```
syms phi theta d1
dh_table = {    %theta      %d      %a      %alpha
             [  phi         146     205     0   ],
             [  theta       -41     275     180 ],
             [  0           d1       0      0   ]
           };
A_1 = transform_from_dh_row(dh_table, 1, true)
```

A_1 =

$$\begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 & 205\cos(\phi) \\ \sin(\phi) & \cos(\phi) & 0 & 205\sin(\phi) \\ 0 & 0 & 1 & 146 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
A_2 = transform_from_dh_row(dh_table, 2, true)
```

A_2 =

$$\begin{pmatrix} \cos(\theta) & -\cos(180)\sin(\theta) & \sin(180)\sin(\theta) & 275\cos(\theta) \\ \sin(\theta) & \cos(180)\cos(\theta) & -\sin(180)\cos(\theta) & 275\sin(\theta) \\ 0 & \sin(180) & \cos(180) & -41 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
A_3 = transform_from_dh_row(dh_table, 3, true)
```

A_3 =

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
H_0_3 = A_1 * A_2 * A_3
```

H_0_3 =

$$\begin{pmatrix} \cos(\phi)\cos(\theta) - \sin(\phi)\sin(\theta) & -\cos(180)\cos(\phi)\sin(\theta) - \cos(180)\cos(\theta)\sin(\phi) & \sigma_1 & 205\cos(\\ \cos(\phi)\sin(\theta) + \cos(\theta)\sin(\phi) & \cos(180)\cos(\phi)\cos(\theta) - \cos(180)\sin(\phi)\sin(\theta) & \sigma_2 - \sigma_3 & 205\sin(\phi) + \\ 0 & \sin(180) & \cos(180) & \\ 0 & 0 & 0 & \end{pmatrix}$$

where

$$\sigma_1 = \sin(180)\cos(\phi)\sin(\theta) + \sin(180)\cos(\theta)\sin(\phi)$$

$$\sigma_2 = \sin(180)\sin(\phi)\sin(\theta)$$

$$\sigma_3 = \sin(180)\cos(\phi)\cos(\theta)$$

With the values dictated in the problem, we can create an animation of the arm moving.

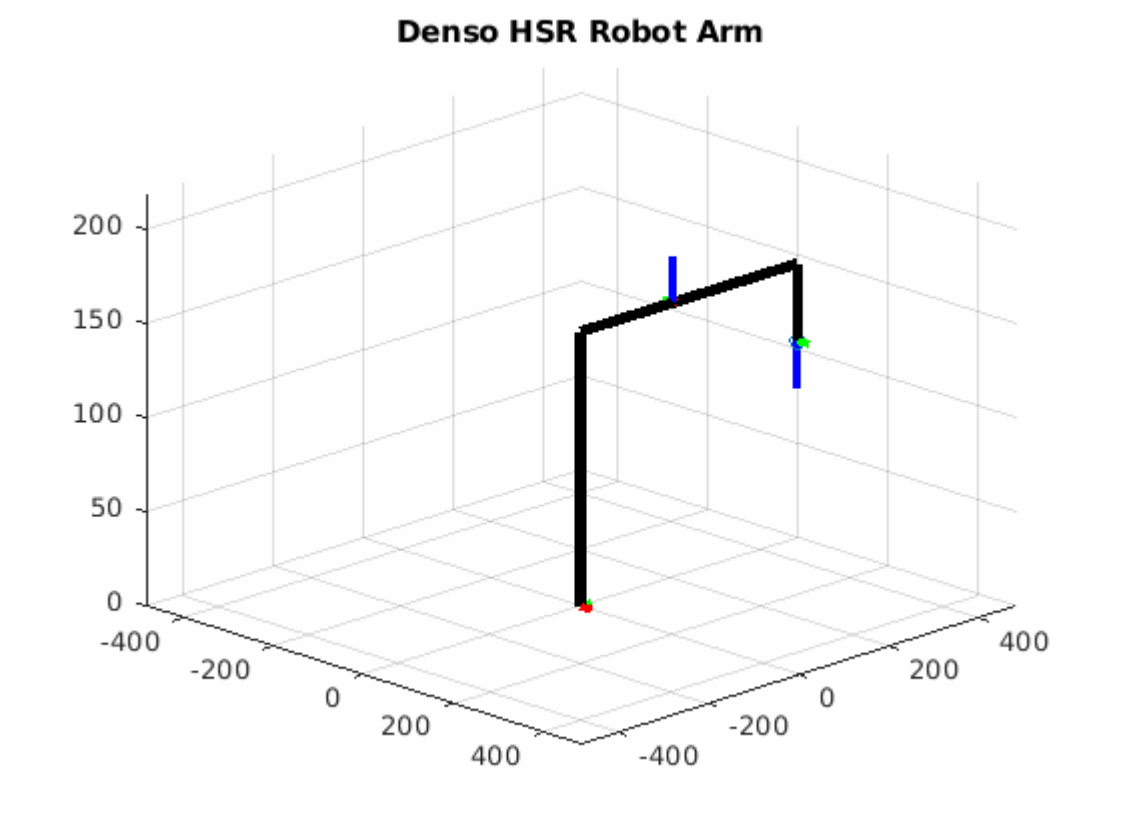
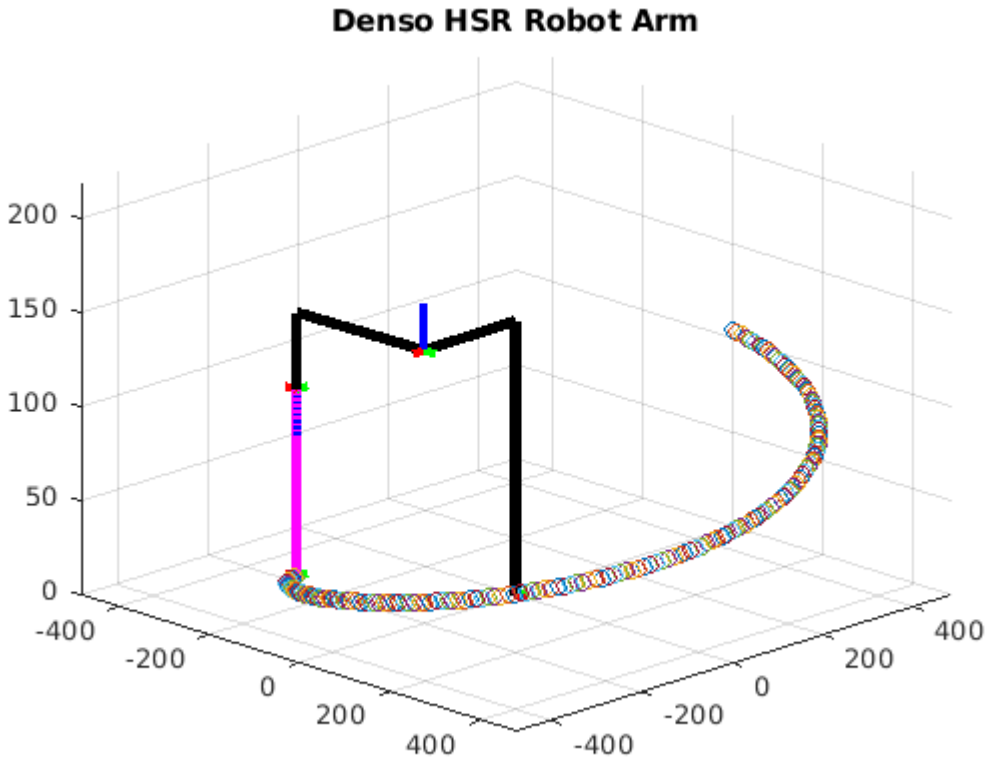


Figure 9: The animated HSR Robot Arm

```
animation_length_seconds = 5;
steps = 180;

phi = 90;
theta = 0;
d1 = 0;
phi_delta = -180 / 180;
theta_delta = -90 / 180;
d1_delta = 100 / 180;

figure3 = figure;
title("Denso HSR Robot Arm")
grid on
hold on
% axis([-300 300 -300 300 0 150])
axis([-480 480 -480 480 0 220])
view([45 25])
lines = [];

for step = 1:steps
    % Clear previous lines
    for index = 1:length(lines)
        delete(lines(index));
```

```
    end
    lines = [];

    % Set up our dh table for the step
    dh_table = {    %theta      %d      %a      %alpha
                 [  phi         146     205     0   ],
                 [  theta       -41     275     180 ],
                 [  0           d1       0      0   ]
               };

    H_0_1 = transform_from_dh_row(dh_table, 1);
    H_1_2 = transform_from_dh_row(dh_table, 2);
    H_2_3 = transform_from_dh_row(dh_table, 3);

    % Calculate our H_02 and H_0_3
    H_0_2 = H_0_1 * H_1_2;
    H_0_3 = H_0_2 * H_2_3;

    % Draw coordinate axes
    [x_line, y_line, z_line] = draw_coordinate_axis(eye(4), 25, 'red', 'green', 'blue')
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;
    [x_line, y_line, z_line] = draw_coordinate_axis(H_0_1, 25, 'red', 'green', 'blue');
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;
    [x_line, y_line, z_line] = draw_coordinate_axis(H_0_2, 25, 'red', 'green', 'blue');
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;
    [x_line, y_line, z_line] = draw_coordinate_axis(H_0_3, 25, 'red', 'green', 'blue');
    lines(end+1) = x_line;
    lines(end+1) = y_line;
    lines(end+1) = z_line;

    % Draw links
    origin = [ 0; 0; 0; 1; ];
    corner_1 = origin + [0; 0; 146; 0;];
    shoulder = H_0_1 * origin;
    corner_2 = H_0_1 * [cosd(theta)*275; sind(theta)*275; 0; 1;];
    elbow = H_0_2 * origin;
    wrist = H_0_3 * origin;
    lines(end+1) = line('XData', [origin(1) corner_1(1)], 'YData', [origin(2) corner_1(
    lines(end+1) = line('XData', [corner_1(1) shoulder(1)], 'YData', [corner_1(2) shoul
    lines(end+1) = line('XData', [shoulder(1) corner_2(1)], 'YData', [shoulder(2) corne
    lines(end+1) = line('XData', [corner_2(1) elbow(1)], 'YData', [corner_2(2) elbow(2)
    lines(end+1) = line('XData', [elbow(1) wrist(1)], 'YData', [elbow(2) wrist(2)], 'ZD

    % Mark the actuator endpoint
```

```
    plot3(wrist(1), wrist(2), wrist(3), 'o')

    phi = phi + phi_delta;
    theta = theta + theta_delta;
    d1 = d1 + d1_delta;

    pause(5/180)

end

hold off
```



Functions and Classes

transform_from_dh_values

This function takes the expected DH values - θ, *d*, *a*, and α. It then performs *Equation 1*, listed above.

The use of *vargin* is used to determine if an optional parameter is passed. This optional parameter is used to determine if the caller of the function wishes to make use of symbolic values. If so, we use *cos* and *sin* instead of *cosd* and *sind* respectively. This results in slightly better outputs for symbolic functions.

```
function H = transform_from_dh_values(theta, d, a, alpha, varargin)
    if nargin == 4
        R_z_theta = [ cosd(theta)      -sind(theta)     0     0;
                      sind(theta)       cosd(theta)     0     0;
                      0                 0               1     0;
```

```
                    0                0              0     1; ];
        R_x_alpha = [ 1     0              0           0;
                      0     cosd(alpha)    -sind(alpha) 0;
                      0     sind(alpha)    cosd(alpha)  0;
                      0     0              0           1; ];
    else
        R_z_theta = [ cos(theta)      -sin(theta)     0     0;
                      sin(theta)       cos(theta)     0     0;
                      0                0              1     0;
                      0                0              0     1; ];
        R_x_alpha = [ 1     0              0           0;
                      0     cos(alpha)     -sin(alpha)  0;
                      0     sin(alpha)     cos(alpha)   0;
                      0     0              0           1; ];
    end


    T_z_d = [ 1     0      0    0;
              0     1      0    0;
              0     0      1    d;
              0     0      0    1; ];

    T_x_a = [ 1     0      0    a;
              0     1      0    0;
              0     0      1    0;
              0     0      0    1; ];



    H = R_z_theta * T_z_d * T_x_a * R_x_alpha;
end
```

transform_from_dh_row

This is a helper function that takes a DH table as defined above (in the form of a cell of matricies of expected DH values in a set standard order of θ, *d*, *a*, and α. It then separates out the θ, *d*, *a*, and α of the specified row, and passes it to **transform_from_dh_values**, as described above. The use of *vargin* is passed through, for reasons specified above.

```
function H = transform_from_dh_row(table, row, vargin)
    theta = table{row}(1);
    d = table{row}(2);
    a = table{row}(3);
    alpha = table{row}(4);
    if nargin == 2
        H = transform_from_dh_values(theta, d, a, alpha);
    else
        H = transform_from_dh_values(theta, d, a, alpha, true);
    end
end
```

draw_coordinate_axis

This function accepts a specified transform, length of the desired axis lines, and the color for each axis. It then draws on the current active figure the origin for that axis based on the transform passed.

```
function [x_line, y_line, z_line] = draw_coordinate_axis(transform, length, x_color, y_
    origin = transform * [0; 0; 0; 1;];
    x = transform * [length; 0;         0;      1; ];
    y = transform * [0;      length;    0;      1; ];
    z = transform * [0;      0;         length; 1; ];
    x_line = line('XData', [origin(1) x(1)], 'YData', [origin(2) x(2)], 'ZData', [origi
    y_line = line('XData', [origin(1) y(1)], 'YData', [origin(2) y(2)], 'ZData', [origi
    z_line = line('XData', [origin(1) z(1)], 'YData', [origin(2) z(2)], 'ZData', [origi
end
```

Discussion

The assignment's purpose was to establish understanding and application of the DH Convention and Table in order to easily determine the transformations required to do forward kinematics (and later inverse kinemtaics) for each joint in a robot arm.

In our first problem, **IV-1**, we analyze an **RPP** robot arm by using provided coordinate axes and generating a DH table for its design. To demonstrate our analysis we created an animation of sequential movement within its revolute and two prismatic joints.

In our second problem, **IV-2**, we analyze an **RRR** three axis robot arm. This time we specified the coordinate axes for each frame in a manner that follows DH convention and creates an easy-to-work with set of axes that generate our DH table, and thus, our required transforms. To demonstrate the analysis, we created a tool that allows different values to be set for each component of the robot arm, and then generate a static image of the arm in that configuration.

In our third and final problem, **IV-3**, we analyze the Denso HSR 480mm stroke robot arm once again. We take the specifications and technical drawings provided through the manufacturer's robot arm catalog and specify axes according to the DH convention. We then create a DH table for each linkage, and demonstrate analysis of the arm through a generated animation of its movement by adjusting its joints by a set amount for 180 steps.

Citations

- SPONG, M. W., HUTCHINSON, S., & VIDYASAGAR, M. (2020). *Robot modeling and control*. *2nd edition* Hoboken, NJ, John Wiley & Sons.