CS 133a **October 26, 2023**

Problem Set 4

Problem 1 (Cubic vs Quintic Spline) - 24 points

part (a) The cubic spline motion was solved numerically with code (refer to the page with the code, function is named
problem_1a). The constants in the cubic function were found in the following manner:

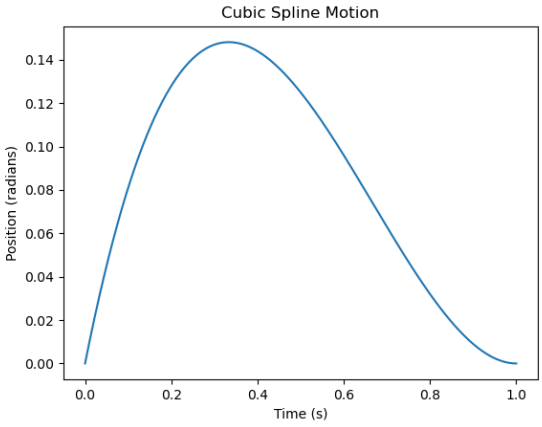
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & T & T^2 & T^3 \\ 0 & 1 & 2T & 3T^2 \end{bmatrix}$$

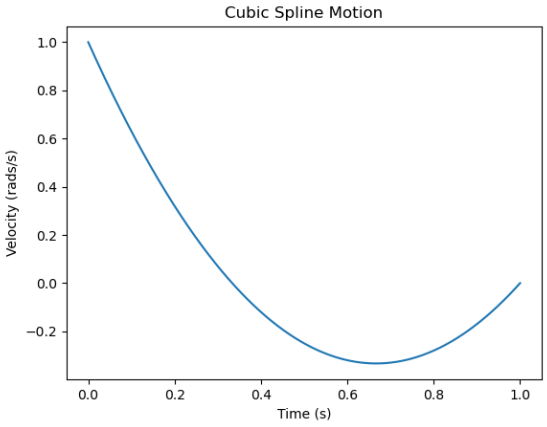
$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = A^{-1} \begin{bmatrix} \theta_0 \\ \dot{\theta_0} \\ \theta_f \\ \dot{\theta_f} \end{bmatrix}$$

where $T = t_f - t_0$, θ_0 is the initial position, and $\dot{\theta_0}$ is the initial velocity. θ_f is the final position, and $\dot{\theta_f}$ is the final velocity. Solving this with code yields

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ -2 \\ 1 \end{bmatrix}$$

Thus we have the following graphs:





From the graph we have that the max position is ≈ 0.148 radians.

part (b) The quintic spline motion was solved numerically with code (refer to the page with code, function is named
problem_1b). The constants in the cubic function were found in the following manner:

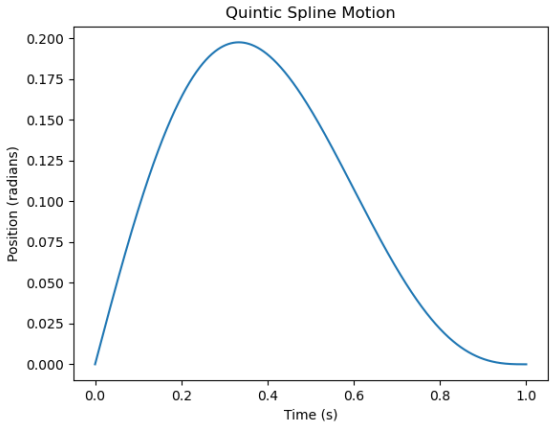
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^5 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 \end{bmatrix}$$

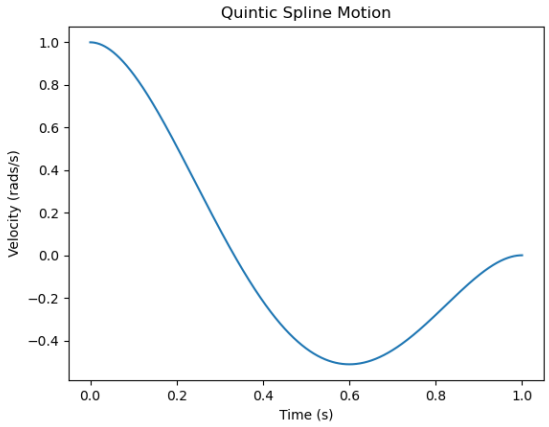
$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = A^{-1} \begin{bmatrix} \theta_0 \\ \dot{\theta}_0 \\ \ddot{\theta}_0 \\ \theta_f \\ \dot{\theta}_f \\ \ddot{\theta}_f \end{bmatrix}$$

where $T = t_f - t_0$, θ_0 is the initial position, and $\dot{\theta}_0$ is the initial velocity, $\ddot{\theta}_0$ is the initial acceleration. θ_f is the final position, and $\dot{\theta}_f$ is the final velocity, $\ddot{\theta}_f$ is the final acceleration. Solving this with code yields

$$\begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -6 \\ 8 \\ -3 \end{bmatrix}$$

Thus we have the following graphs:





From the graph we have that the max position is ≈ 0.1975 radians.

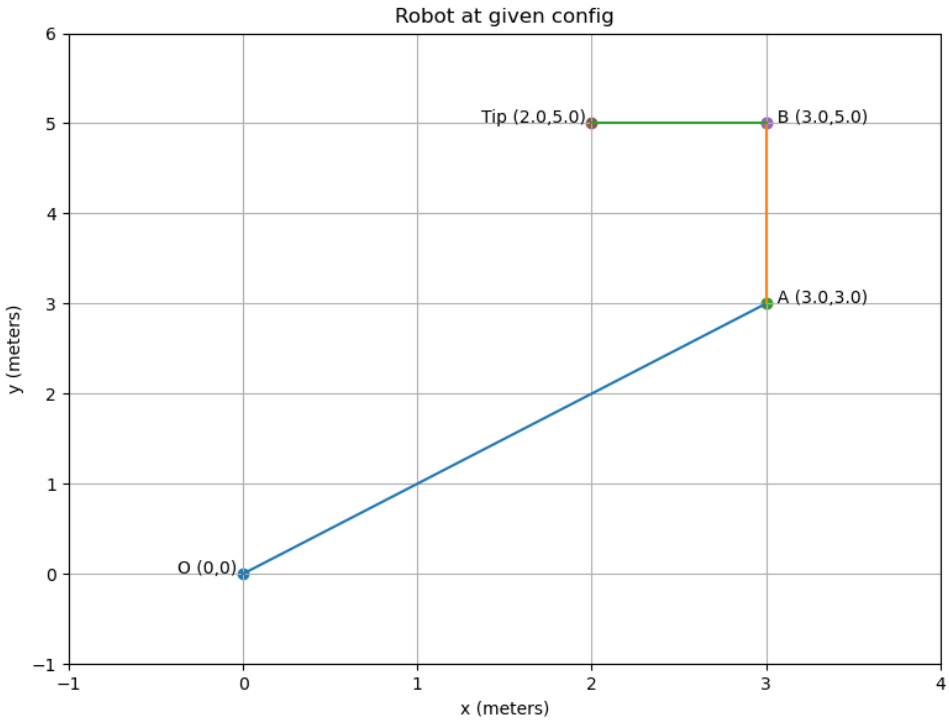
```
import numpy as np
import matplotlib.pyplot as plt


def problem_1a(theta_0, w_0, theta_f, w_f, t0, tf):
    """
    Performs cubic spline motion on
    1 DOF robot. Takes in initial values of
    the motion

    Args:
        theta_0 : initial angular position (radians)
        w_0 : initial angular velocity
        t0 : inital time t
        tf : final time t
    """
    T = tf - t0
    A = np.array([[1,0,0,0],
                  [0,1,0,0],
                  [1,T, T**2, T**3],
                  [0, 1, 2*T, 3*(T**2)]])
    if (np.linalg.det(A) == 0):
        print("Matrix is not invertible")
        return -1
    values = np.array([[theta_0], [w_0], [theta_f], [w_f]])
    consts = np.matmul(np.linalg.inv(A), values)
    print(consts)
    a,b,c,d = consts[0,0], consts[1,0], consts[2,0], consts[3,0]

    N = 1000 #number of points to plot
    thetas = []
    omegas = []
    times = []
    for i in range(N+1):
        t = t0 + (T/N) * i
        theta = a + b*t + c*(t**2) + d*(t**3)
        omega = b + 2*c*t + 3*d*(t**2)

        times.append(t)
        thetas.append(theta)
        omegas.append(omega)

    print("Max position reached = {} radians".format(max(thetas)))
    plt.plot(times, thetas)
    plt.title("Cubic Spline Motion")
    plt.xlabel("Time (s)")
    plt.ylabel("Position (radians)")
    plt.show()

    plt.plot(times, omegas)
    plt.title("Cubic Spline Motion")
```

```
    plt.xlabel("Time (s)")
    plt.ylabel("Velocity (rads/s)")
    plt.show()



def problem_1b(theta_0, w_0, a_0, theta_f, w_f, a_f, t0, tf):
    """
    Performs quintic spline motion on
    1 DOF robot. Takes in initial values of
    the motion

    Args:
        theta_0 : initial angular position (radians)
        w_0 : initial angular velocity
        a_0 : initial angular acceleration
        t0 : inital time t
        tf : final time t
    """
    T = tf - t0
    A = np.array([[1,0,0,0,0,0],
                  [0,1,0,0,0,0],
                  [0,0,2,0,0,0],
                  [1, T, T**2, T**3, T**4, T**5],
                  [0, 1, 2*T, 3*(T**2), 4*(T**3), 5*(T**4)],
                  [0, 0, 2, 6*T, 12*(T**2), 20*(T**3)] ])

    if (np.linalg.det(A) == 0):
        print("Matrix is not invertible")
        return -1
    values = np.array([[theta_0], [w_0], [a_0], [theta_f], [w_f], [a_f]])
    consts = np.matmul(np.linalg.inv(A), values)
    print(consts)
    a,b,c = consts[0,0],consts[1,0],consts[2,0]
    d,e,f = consts[3,0],consts[4,0],consts[5,0]

    N = 1000 #number of points to plot
    thetas = []
    omegas = []
    accels = []
    times = []
    for i in range(N+1):
        t = t0 + (T/N) * i
        theta = a + b*t + c*(t**2) + d*(t**3) + e*(t**4) + f*(t**5)
        omega = b + 2*c*t + 3*d*(t**2) + 4*e*(t**3) + 5*f*(t**4)
        accel = 2*c + 6*d*t + 12*e*(t**2) + 20*f*(t**3)

        times.append(t)
        thetas.append(theta)
        omegas.append(omega)
```

```
        accels.append(accel)

    print("Max position reached = {} radians".format(max(thetas)))
    plt.plot(times, thetas)
    plt.title("Quintic Spline Motion")
    plt.xlabel("Time (s)")
    plt.ylabel("Position (radians)")
    plt.show()

    plt.plot(times, omegas)
    plt.title("Quintic Spline Motion")
    plt.xlabel("Time (s)")
    plt.ylabel("Velocity (rads/s)")
    plt.show()

if __name__ == "__main__":
    problem_1a(0, 1, 0, 0, 0, 1)
    problem_1b(0,1,0, 0,0,0, 0,1)
```

Problem 2 (Jacobian of Planar RPRPRP Robot) - 24 points:

Note that the z coordinate for all positions in depicted in the graph is 0.



Let J_i denote the ith column of the Jacobian J. Using the diagram above (with the position determined) we have (corresponding to the prismatic joints):

$$J_2 = \begin{bmatrix} \frac{\vec{A}-\vec{0}}{||\vec{A}-\vec{0}||} \\ \vec{0} \end{bmatrix} = \frac{1}{\sqrt{3^2+3^2}} \begin{bmatrix} 3 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J_4 = \begin{bmatrix} \frac{\vec{B}-\vec{A}}{||\vec{B}-\vec{A}||} \\ \vec{0} \end{bmatrix} = \frac{1}{\sqrt{2^2}} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$J_6 = \begin{bmatrix} \frac{\vec{Tip}-\vec{B}}{||\vec{Tip}-\vec{B}||} \\ \vec{0} \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Note that all revolute joints rotate about the positive z axis $\vec{e_z}$ (relative to the world frame). Hence for the columns corresponding to the revolute joints we have:

$$J_1 = \begin{bmatrix} \vec{e_z} \times (\vec{Tip} - \vec{0}) \\ \vec{e_z} \end{bmatrix} = \begin{bmatrix} -5 \\ 2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_3 = \begin{bmatrix} \vec{e_z} \times (\vec{Tip} - \vec{A}) \\ \vec{e_z} \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$J_5 = \begin{bmatrix} \vec{e_z} \times (\vec{Tip} - \vec{B}) \\ \vec{e_z} \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Thus for the given q, we have the Jacobian:

$$J = \begin{bmatrix} -5 & \frac{1}{\sqrt{2}} & -2 & 0 & 0 & -1 \\ 2 & \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Dropping all zero rows to get the 3 by 6 Jacobian matrix (for the given q), we have:

$$J = \begin{bmatrix} -5 & \frac{1}{\sqrt{2}} & -2 & 0 & 0 & -1 \\ 2 & \frac{1}{\sqrt{2}} & -1 & 1 & -1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Lastly code that was used to calculate the cross products (code for the graph is on the next page):

```
def cross_products():
    e_i = np.array([0,0,1])
    p0 = np.array([0,0,0])
    pA = np.array([3,3,0])
    pB = np.array([3,5,0])
    pTip = np.array([2,5,0])

    cross1 = np.cross(e_i, pTip - p0) # For J1
    cross3 = np.cross(e_i, pTip - pA) # For J3
    cross5 = np.cross(e_i, pTip - pB) #For J5
```

Code for the plot in problem 2:

```
def problem_2():
    x_pos = [0]
    y_pos = [0]

    theta1 = 45 * np.pi/180
    d2 = 3 * np.sqrt(2)
    theta3 = 45 * np.pi/180
    d4 = 2
    theta5 = 90 * np.pi/180
    d6 = 1

    x1 = d2 * np.cos(theta1)
    y1 = d2 * np.sin(theta1)
    x_pos.append(x1)
    y_pos.append(y1)

    x2 = x1 + d4 * np.cos(theta1 + theta3)
    y2 = y1 + d4 * np.sin(theta1 + theta3)
    x_pos.append(x2)
    y_pos.append(y2)

    x3 = x2 + d6 * np.cos(theta1 + theta3 + theta5)
    y3 = y2 + d6 * np.sin(theta1 + theta3 + theta5)
    x_pos.append(x3)
    y_pos.append(y3)

    label = 'O ({},{}) '.format(round(0,3),round(0,3))
    plt.text(0, 0, label, horizontalalignment='right')
    for i in range(1, len(x_pos)):
        x0,y0 = x_pos[i-1], y_pos[i-1]
        xf,yf = x_pos[i], y_pos[i]
        plt.scatter(x0, y0, marker = 'o')
        plt.scatter(xf, yf, marker='o')
        if i != len(x_pos) - 1:
            if i == 1:
                plt.text(xf, yf, '  A ({},{})'.format(round(xf,3),round(yf,3)))
            if i == 2:
                plt.text(xf, yf, '  B ({},{})'.format(round(xf,3),round(yf,3)))

        plt.plot([x0,xf], [y0,yf])

    label = 'Tip ({},{}) '.format(round(x_pos[-1],3),round(y_pos[-1],3))
    plt.text(x_pos[-1], y_pos[-1], label , horizontalalignment='right')
    plt.xlabel("x (meters)")
    plt.ylabel('y (meters)')
    plt.xlim(-1, 4)
    plt.ylim(-1, 6)
    plt.title("Robot at given config")
    plt.grid()
    plt.show()
```

Problem 3 (URDF for 4DOF Robot) - 24 points

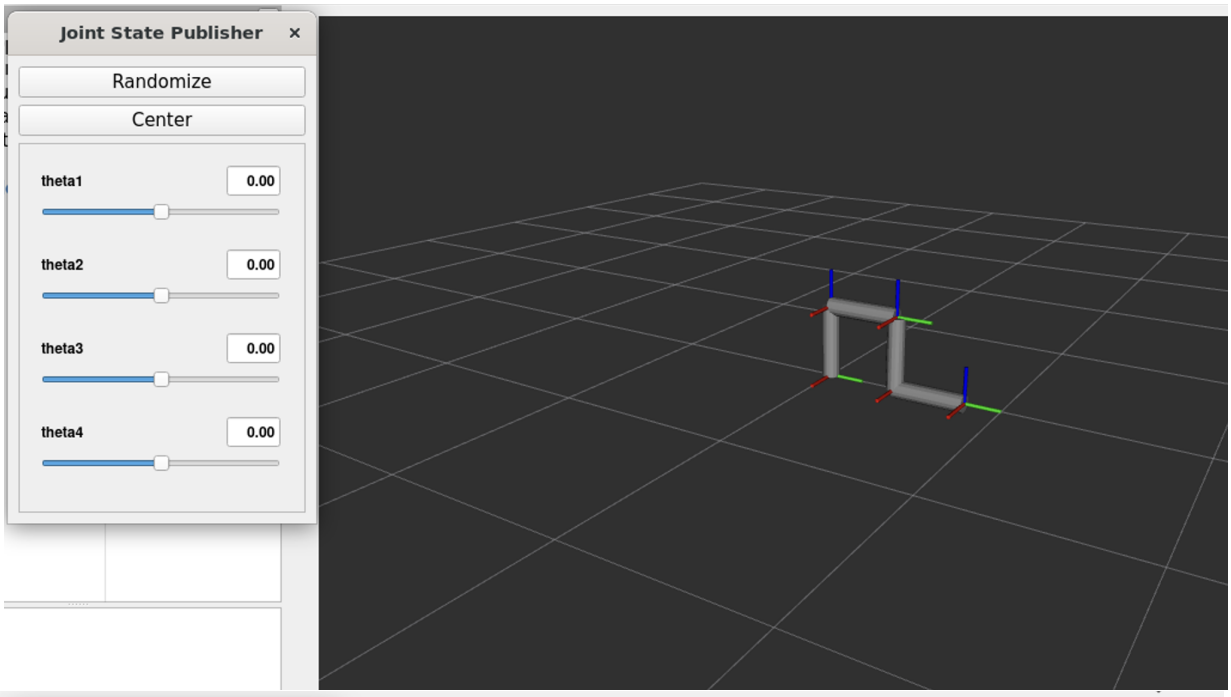
part (a) Complete URDF file shown below:

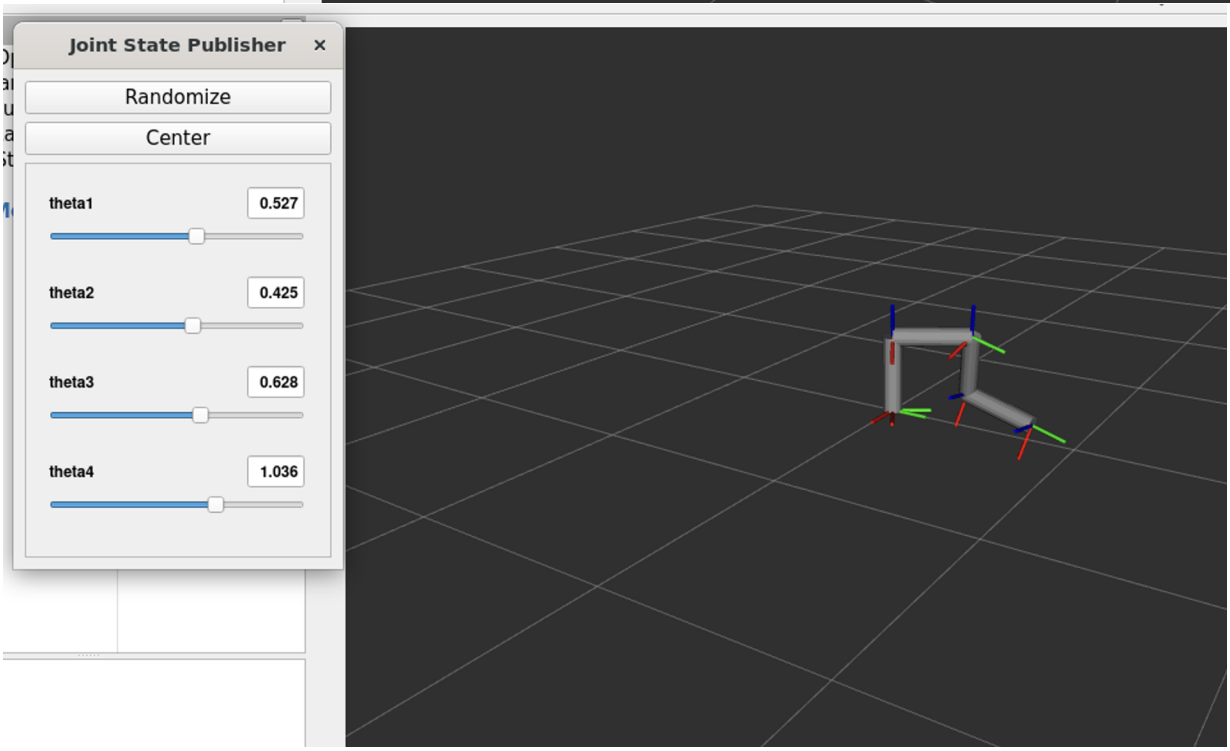
```
<?xml version="1.0"?>

<robot name="FourDOF">

  <!-- ************************************************ -->
  <!-- ***************** Kinematic Chain ***************** -->
  <!-- ************************************************ -->
  <!-- <link name="..."> is really a frame!              -->
  <!-- <joint name="..." type="fixed">      is a fixed shift.    -->
  <!-- <joint name="..." type="continuous"> is a revolute joint. -->
  <!-- <joint name="..." type="prismatic">  is a linear   joint. -->
  <!-- Note the 'origin' tag in the joints defines the shift and
       reorientation, i.e. the transform.  For moving joints, this
       happens before the continuous joint rotates around 'axis'. -->

  <link name="world">
  </link>

  <!-- Shift 0 -->
  <joint name="shift0" type="fixed">
    <parent link="world"/>
    <child  link="frame0prime"/>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </joint>

  <link name="frame0prime">
  </link>

  <!-- Joint 1 -->
  <joint name="theta1" type="continuous">
    <parent link="frame0prime"/>
    <child  link="frame1"/>
    <axis   xyz="0 0 1"/>
  </joint>

  <link name="frame1">
  <visual>
      <origin xyz="0 0 0.2" rpy="0 0 0"/>       <!-- Center of cylinder -->
      <geometry>
            <cylinder length="0.4" radius="0.05"/>      <!-- Size of cylinder -->
      </geometry>
      <material name="gray"/>                   <!-- Color -->
    </visual>
  </link>

  <!-- Shift 1 and Joint 2 in one step... -->
  <joint name="theta2" type="continuous">
    <parent link="frame1"/>
```

```
  <child  link="frame2"/>
  <origin xyz="0 0 0.4" rpy="0 0 0"/>
  <axis   xyz="0 1 0"/>
</joint>

<link name="frame2">
<visual>
    <origin xyz="0 0.2 0" rpy="1.57 0 0"/>    <!-- Center of cylinder -->
    <geometry>
      <cylinder length="0.4" radius="0.05"/> <!-- Size of cylinder -->
    </geometry>
    <material name="gray"/>                   <!-- Color -->
  </visual>
</link>

<!-- Shift 2 and Joint 3-->
<joint name="theta3" type="continuous">
  <parent link="frame2"/>
  <child  link="frame3"/>
  <origin xyz="0 0.4 0" rpy="0 0 0"/>
  <axis   xyz="0 0 -1"/>
</joint>

<link name="frame3">
<visual>
    <origin xyz="0 0 -0.2" rpy="0 0 0"/>      <!-- Center of cylinder -->
    <geometry>
      <cylinder length="0.4" radius="0.05"/> <!-- Size of cylinder -->
    </geometry>
    <material name="gray"/>                   <!-- Color -->
  </visual>
</link>

<!-- Shift 3 and Joint 4-->
<joint name="theta4" type="continuous">
  <parent link="frame3"/>
  <child  link="frame4"/>
  <origin xyz="0 0 -0.4" rpy="0 0 0"/>
  <axis   xyz="0 1 0"/>
</joint>

<link name="frame4">
<visual>
    <origin xyz="0 0.2 0" rpy="1.57 0 0"/>    <!-- Center of cylinder -->
    <geometry>
      <cylinder length="0.4" radius="0.05"/> <!-- Size of cylinder -->
    </geometry>
    <material name="gray"/>                   <!-- Color -->
  </visual>
</link>
```

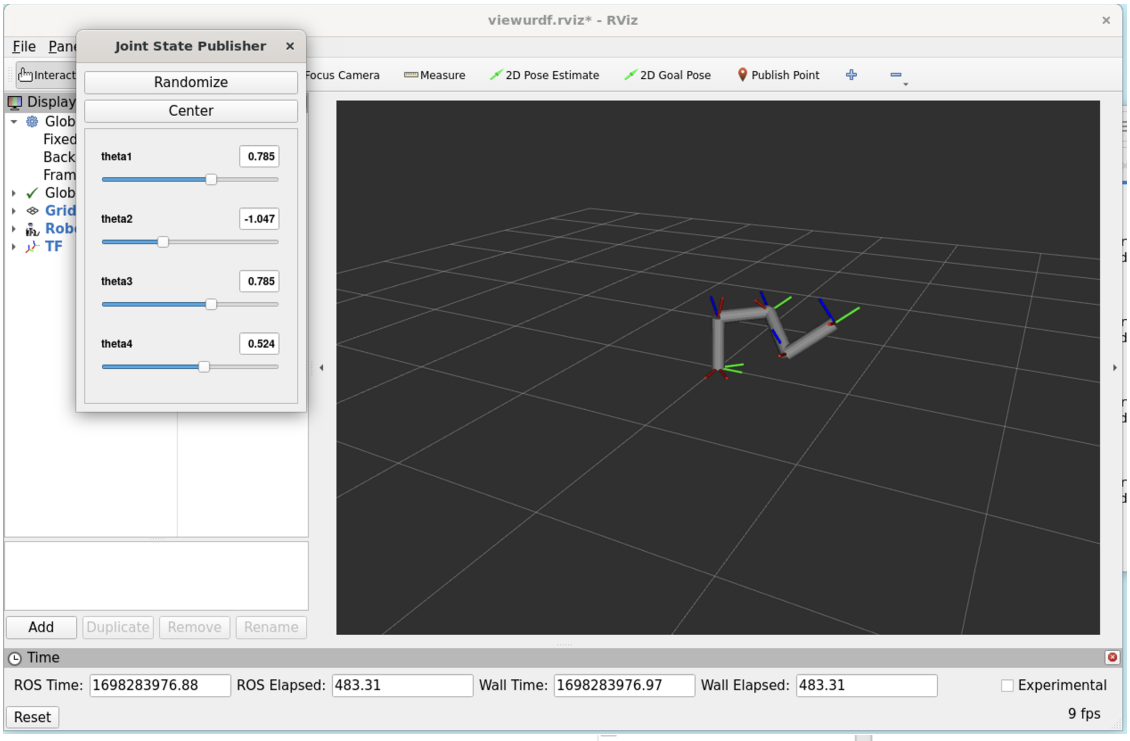
```
  <!-- Shift 4 -->
  <joint name="shift4" type="fixed">
   <parent link="frame4"/>
   <child  link="tip"/>
   <origin xyz="0 0.4 0" rpy="0 0 0"/>
  </joint>


  <!-- you probably want to end with tip -->
  <link name="tip">
  </link>


  <!-- ================================================ -->
  <!-- ================= Named Colors ================= -->
  <!-- ================================================ -->

  <material name="white">    <color rgba="1.00 1.00 1.00 1"/>  </material>
  <material name="gray">     <color rgba="0.60 0.60 0.60 1"/>  </material>
  <material name="table">    <color rgba="0.85 0.77 0.77 1"/>  </material>
  <material name="black">    <color rgba="0.00 0.00 0.00 1"/>  </material>

  <material name="red">      <color rgba="1.00 0.00 0.00 1"/>  </material>
  <material name="green">    <color rgba="0.00 1.00 0.00 1"/>  </material>
  <material name="blue">     <color rgba="0.00 0.00 1.00 1"/>  </material>

  <material name="cyan">     <color rgba="0.00 1.00 1.00 1"/>  </material>
  <material name="magenta">  <color rgba="1.00 0.00 1.00 1"/>  </material>
  <material name="yellow">   <color rgba="1.00 1.00 0.00 1"/>  </material>

  <material name="orange">   <color rgba="1.00 0.65 0.00 1"/>  </material>


  <!-- ================================================ -->
  <!-- ========== Additional Visual Elements ========== -->
  <!-- ================================================ -->

</robot>
```

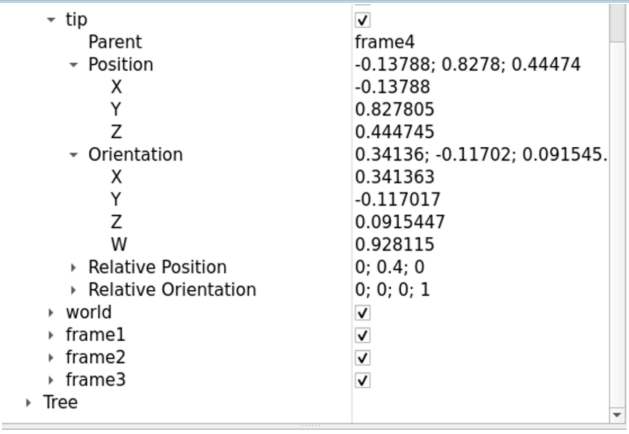
part (b) Yes all joints look and function correctly. Here are two examples:





part (c) Image of the robot given the values for angles. The position and orientation of the tip frame also shown in the bottom image





part (c) The position and orientation of the tip frame also shown in the image on the previous page. For the position we have:

$$p = \begin{bmatrix} -0.13788 \\ 0.827805 \\ 0.444745 \end{bmatrix}$$

And for the orientation (in quaternion form) we have

$$R_{quaternion} = \begin{bmatrix} 0.341363 \\ -0.117017 \\ 0.0915447 \\ 0.928115 \end{bmatrix}$$

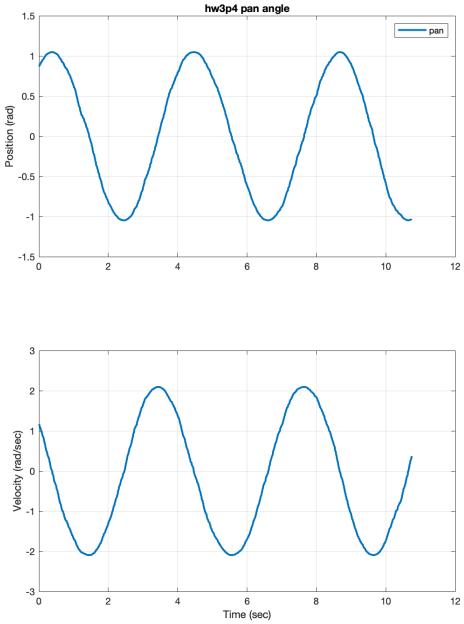
Converting the orientation to a rotation matrix we have:

$$R = \begin{bmatrix} 0.95585316 & -0.24981868 & -0.15471059 \\ 0.09003751 & 0.75018163 & -0.6550731 \\ 0.27971054 & 0.61222394 & 0.73955653 \end{bmatrix}$$

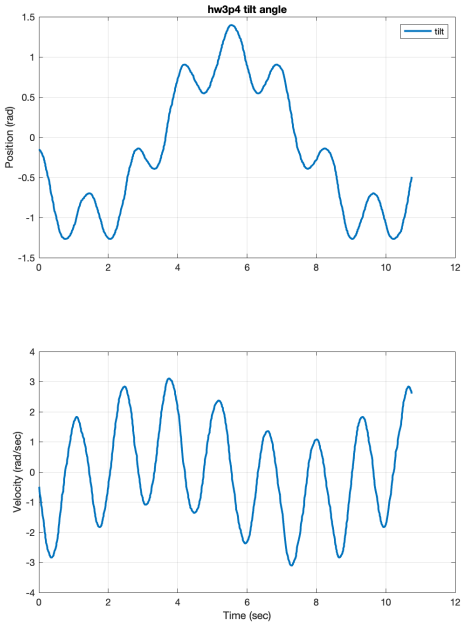
This rotation matrix and the position of the tip match the results from last week's problem set. The content of the URDF file is in part a.

Problem 4 (Plotting Data for Last Week's Animations) - 24 points:

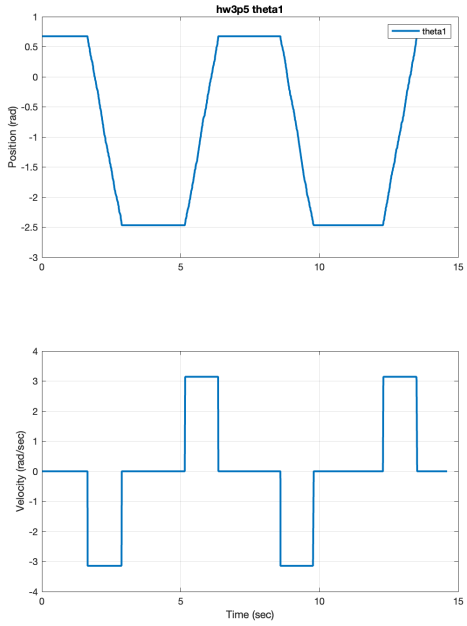
Plots for problem 4 of homework 3. The plots for the pan angle is shown below:



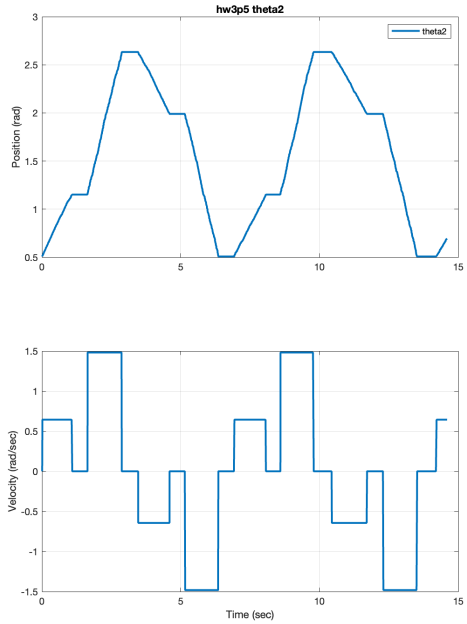
Plots for problem 4 of homework 3. The plots for the tilt angle is shown below:



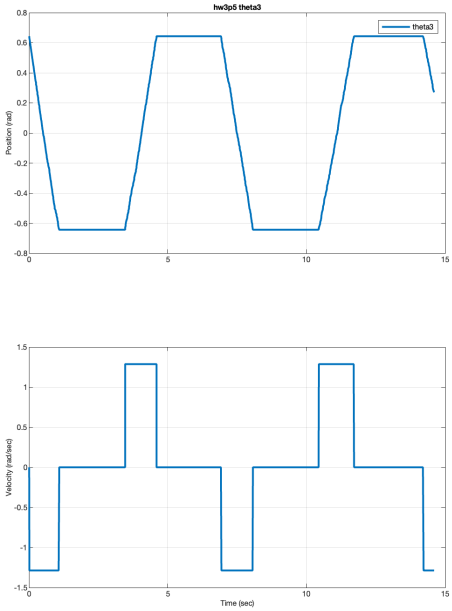
Plots for problem 5 of homework 3. The plots for theta1 is shown below:



Plots for problem 5 of homework 3. The plots for theta2 is shown below:



Plots for problem 5 of homework 3. The plots for theta3 is shown below:



Problem 5 (Time Spent) - 4 points:

I spent about 4 hours on this problem set. I did not have any particular bottlenecks, just spent some time grasping the concept of the Jacobian.