Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

Program structure for main.m:
Main.m is subdivided into sections that can be run one at a time to observe the outputs of each function. The order of the sections reflects the order in which the prompts are given. Both HA bonus functions and the PA bonus function are included at the end of main.m.
Mathematical approach for subfunctions:

1. Angle_axis_func.m:
   Determine validity of input R by making sure its 2D, row length and column length is the same value of 3, and that the determinant is 1.

   Angle = acos(.5*(traceR-1)) *If angle = 0, R is also invalid since there is no rotation.
   Axis = (R-transpose(R))/(2sin(angle)).

2. Quaternion_func.m:
   Determine validity of input R by making sure its 2D, row length and column length is the same value of 3, and that the determinant is 1. Then calculate q terms if valid.

   Q0 = .5*sqrt(R(1,1)+R(2,2)+R(3,3)+1);
   Q1 = .5*sign(R(3,2)-R(2,3))*sqrt(R(1,1)-R(2,2)-R(3,3)+1)
   Q2 = .5*sign(R(1,3)-R(3,1))*sqrt(R(2,2)-R(3,3)-R(1,1)+1)
   Q3 = .5*sign(R(2,1)-R(1,2))*sqrt(R(3,3)-R(1,1)-R(2,2)+1)

3. Euler_angle_func.m:
   Determine validity of input R by making sure its 2D, row length and column length is the same value of 3, and that the determinant is 1. If R = identity matrix 3x3, ZYZ is singularity. Since division by 0 is impossible, R(2,3) and R(1,3) must not equal 0 to be able to calculate ZYZ angles. If able to calculate ZYZ and not in singularity...

   zyz_angles(1)= atan2(R(2,3),R(1,3))
   zyz_angles(2)= atan2(sqrt(R(1,3)^2+R(2,3)^2),R(3,3))
   zyz_angles(3)= atan2(R(3,2),-R(3,1))

   If angle = acos(.5*(trace(R)-1)) = pi/2, -pi/2, or 3*pi/2, ZYX is singularity. If not singularity, ZYX …

   If angle is between -90 and 90 degrees
      zyx_angles(1) = atan2(R(2,1),R(1,1))
      zyx_angles(2) = atan2(-1*R(3,1),sqrt(R(3,2)^2 + R(3,3)^2))
      zyx_angles(3) = atan2(R(3,2),R(3,3));
   Else
      zyx_angles(1) = atan2(-1*R(2,1),-1*R(1,1))
      zyx_angles(2) = atan2(-1*R(3,1),-1*sqrt(R(3,2)^2 + R(3,3)^2))
      zyx_angles(3) = atan2(-1*R(3,2),-1*R(3,3))

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

4. AxisAngle2RotMat.m:
   Normalize the axis -> x,y,z and use terms to solve for R as follows (s = sin(theta),c = cos(theta),t = 1-cos(theta):

   R(1,1)=t*x*x+c
   R(1,2)=t*x*y-z*s
   R(1,3)=t*x*z+y*s

   R(2,1)=t*x*y+z*s
   R(2,2)=t*y*y+c
   R(2,3)=t*y*z-x*s

   R(3,1)=t*x*z-y*s
   R(3,2)=t*y*z+x*s
   R(3,3)=t*z*z+c

5. Quat2RotMat.m:
   Using inputs q0,q1,q2,q3, solve all 9 terms of the rotation matrix as follows:

   R(1,1)=2*(q0^2+q1^2)-1
   R(1,2)=2*(q1*q2-q0*q3)
   R(1,3)=2*(q1*q3+q0*q2)

   R(2,1)=2*(q1*q2+q0*q3)
   R(2,2)=2*(q0^2+q2^2)-1
   R(2,3)=2*(q2*q3-q0*q1)

   R(3,1)=2*(q1*q3-q0*q2)
   R(3,2)=2*(q2*q3+q0*q1)
   R(3,3)=2*(q0^2+q3^2)-1
6. Qsh2screw.m:

   w = s
   v = (-sxq) + h*s
   Screw matrix = [0 -w(3) w(2) v(1);
   w(3) 0 -w(1) v(2);
   -w(2) w(1) 0 v(3);
   0 0 0 0];

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

7. Configuration_calulator.m:
   Define sub angles as quarter divisions of input theta. Calculate/extract all components necessary for calculations such as norm of w, w and v.

   If norm of w is not 0, the following form is needed to calculate the sub transformation matrices.

   T =[eye(3)+(w/w_l)*sin(w_l*theta)+(w^2/(w_l)^2)*(1-cos(w_l*theta)), (eye(3)*theta+(1-cos(theta))*w+(theta-sin(theta))*w*w)*v;[0,0,0,1]];

   If norm of w is 0, this is the pure translation with the pitch = infinity. The sub transformation matrices are calculated as follows.

   T = [eye(3),v*theta;[0,0,0,1]]

   Sub configurations are then calculated by premultiplying the initial configuration by the sub transformation matrix.

8. TMatrix2ScrewAngle.m

   Extract R and p from the transformation matrix.
   Solve for omega and theta first by using matrix logarithm of rotations.
   Solve for v with (1/theta*eye(3)-.5*omega+ (1/theta-.5*cot(theta/2))*(omega^2))*p

   Structure output screw matrix as [[w],v;0 0]

9. Screw2qsh.m

   Extract w and v information from the screw matrix and find the q s h configuration variables as follows.

   s = w/w_l;
   h = v_l/w_l;
   q = w x v;

10. Bonus_Math_10.m

    Euler angles are used to navigate a space of rotation matrices. The bisection search method is used to optimize each euler angle, one at a time. The resulting matrix is guaranteed to be a valid rotation matrix. The given equation for matrix similarity is used as the optimized value.

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

11. Matrix_R.m

    Given alpha, beta, gamma, plug into the following matrix formula:
    R=[cos(alp)*cos(bet)*cos(gam)-sin(alp)*sin(gam)-cos(alp)*cos(bet)*sin(gam)-sin(alp)*cos(gam)cos(alp)*sin(bet);sin(alp)*cos(bet)*cos(gam)+cos(alp)*sin(gam)-sin(alp)*cos(bet)*sin(gam)+cos(alp)*cos(gam) sin(alp)*sin(bet);-sin(bet)*cos(gam) sin(bet)*sin(gam) cos(bet)];

12. Matrix_Difference_Norm.m

    Use formula given in problem statement: E=trace((A-R)*transpose(A-R))

13. Bonus_Programming_4.m
    Use the out but of Bonus_Math_10.m and compare the difference value to a user-defined threshold.

Algorithmic steps followed for problem 3:
1. Use sub function sqh2screw.m to get screw matrix from qsh inputs
2. Use sub function configuration_calculator to get sub configurations from applying the screw matrix and theta.
3. Use the last sub configuration to calculate a screw matrix and theta from that configuration to the origin in the world frame.
4. Use subfunction screw2qh to convert this second screw matrix into the q s h terms and then plot the screw axis and sub configurations

Algorithmic steps followed for Bonus_Math_10.m:
1. Euler angles start at [0 0 0]
2. For one euler angle, a range of 0-2pi is established.
3. Four equally spaced points are found within the range
4. Those angles are tested as the euler angle and the matrix similarity to the target is found
5. The results are compared and the lowest performing point is eliminated along with the corresponding quadrant of the unit circle.
6. A new range is established, which includes only the 3 remaining quadrants.
7. Divide the range in half and compare the endpoints and midpoint of the range.
8. Set the new range to be between the two best performing points.
9. Repeat steps 7-8 until the range is small enough to be less than a threshold.
10. Repeat 1-7 again for each of the two remaining euler angles.
11. Output the rotation matrix derived from the euler angles and output the similarity value.

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

Debugging Steps:

The debugging process starts with determining the desired test cases that will be tested. Then inputs are designed whose outputs can be predetermined. We found that our outputs have matched with our expectations and the tabulated results can be seen below. The main.m file can also be run to generate this data in real time.

Debugging with Test Data:

Question 1

- angle_axis_func.m

Test case:	Input:	Output:
4x4 Input Matrix	[0 1 2 3;4 5 6 7;8 9 10 11;12 13 14 15]	Angle = 0; Axis = 0; Valid = Not Valid R
3x3 Input Matrix Rotation About Z axis 90 degrees	[0 -1 0;1 0 0;0 0 1]	Angle = 1.5708; Axis = [0 -1 0;1 0 0;0 0 0]; Valid = Valid R
3x3 Input Matrix Rotation About X 45 and Y 45 degrees	[0 0 1;1 0 0;0 1 0]	Angle = 2.0944; Axis = [0 -.5774 .5774;.5774 0 -.5774;-.5774 .5774 0]; Valid = Valid R
3x3 Input Matrix Invalid Rotation Matrix	[3 3 3;3 3 3;3 3 3]	Angle = 0; Axis = 0; Valid = Not Valid R
Not Square Input Matrix	[3 2 2; 3 4 5]	Angle = 0; Axis = 0; Valid = Not Valid R

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

- quaternion_func.m

Test case:	Input:	Output:
4x4 Input Matrix	[0 1 2 3;4 5 6 7;8 9 10 11;12 13 14 15]	Valid = Not valid R; q = [0;0;0;0]
3x3 Input Matrix Rotation About Z axis 90 degrees	[0 -1 0;1 0 0;0 0 1]	Valid = Valid R; q = [.7071;0;0;.7071]
3x3 Input Matrix Rotation About X 45 and Y 45 degrees	[0 0 1;1 0 0;0 1 0]]	Valid = Valid R; q = [.5;.5;.5;.5]
3x3 Input Matrix Invalid Rotation Matrix	[3 3 3;3 3 3;3 3 3]	Valid = Not valid R; q = [0;0;0;0]
Not Square Input Matrix	[3 2 2; 3 4 5]	Valid = Not valid R; q = [0;0;0;0]

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

- euler_angle_func.m

Test case:	Input:	Output:
4x4 Input Matrix	[0 1 2 3;4 5 6 7;8 9 10 11;12 13 14 15]	ZYZ_angles = [0;0;0]; ZYX_angles = [0;0;0]; Valid = Not Valid R
3x3 Input Matrix Rotation About Z axis 90 degrees	[0 -1 0;1 0 0;0 0 1]	ZYZ_angles = [0;0;0]; ZYX_angles = [0;0;0]; Valid = Not possible to calculate ZYZ Singularity for ZYX
3x3 Input Matrix Rotation About X 45 and Y 45 degrees	[0 0 1;1 0 0;0 1 0]]	ZYZ_angles = [0;1.5708;1.5708]; ZYX_angles = [-1.5708;-3.416;-1.5708]; Valid = Valid
3x3 Input Matrix Invalid Rotation Matrix	[3 3 3;3 3 3;3 3 3]	ZYZ_angles = [0;0;0]; ZYX_angles = [0;0;0]; Valid = Not Valid R
Not Square Input Matrix	[3 2 2; 3 4 5]	ZYZ_angles = [0;0;0]; ZYX_angles = [0;0;0]; Valid = Not Valid R
Singularity for ZYZ	[1 0 0;0 1 0;0 0 1]	ZYZ_angles = [0;0;0]; ZYX_angles = [0;0;0]; Valid = Singularity for ZYZ

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

Question 2

- AxisAngle2RotMat.m

Test case:	Input:	Output:
X axis, pi	[1 0 0], pi	a3 = 1.0000 0 0 0 -1.0000 -0.0000 0 0.0000 -1.0000
Y axis, pi/2	[0 1 0], p/2	b3 = 0.0000 0 1.0000 0 1.0000 0 -1.0000 0 0.0000
Z axis, -pi	[0 0 1], -pi	c3 = -1.0000 0.0000 0 -0.0000 -1.0000 0 0 0 1.0000

- Quat2RotMat.m

Test case:	Input:	Output:
Expected output: 3x3 Rotation About Z axis 90 degrees	q = [.7071;0;0;.7071]	a4 = -0.0000 -1.0000 0 1.0000 -0.0000 0 0 0 1.0000
Expected output: 3x3 Matrix Rotation About X 45 and Y 45 degrees	q = [.5;.5;.5;.5]	b4 = 0 0 1 1 0 0 0 1 0

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

Question 3

- Question_3.m (images are given below table; the perspective is adjusted to highlight the observations made in the Output column)

Test case:	Input:	Output (visual plot):
1. The data given in HW	Q1=[0 2 0]; S1=[0 0 1]; H1=2; theta1=pi; Tinit1=[1 0 0 2;0 1 0 0;0 0 1 0;0 0 0 1];	Points extend down in a helix, pink screw axis is in between last point and origin
2. T init at origin, same orientation	Q1=[0 2 0]; S1=[0 0 1]; H1=2; theta1=pi; Tinit2=[1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];	Pink screw axis is equidistant from all points
3. Negative h	Q1=[0 2 0]; S1=[0 0 1]; H1=-2; theta1=pi; Tinit1=[1 0 0 2;0 1 0 0;0 0 1 0;0 0 0 1];	Points appear above origin
4. q and Tinit are coincident	Q1=[2 0 0]; S1=[0 0 1]; H1=-2; theta1=pi; Tinit1=[1 0 0 2;0 1 0 0;0 0 1 0;0 0 0 1];	Points form a line down
5. theta=pi/2	Q1=[0 2 0]; S1=[0 0 1]; H1=2; theta1=pi/2; Tinit1=[1 0 0 2;0 1 0 0;0 0 1 0;0 0 0 1];	Points do not extend down as far

Image 1:

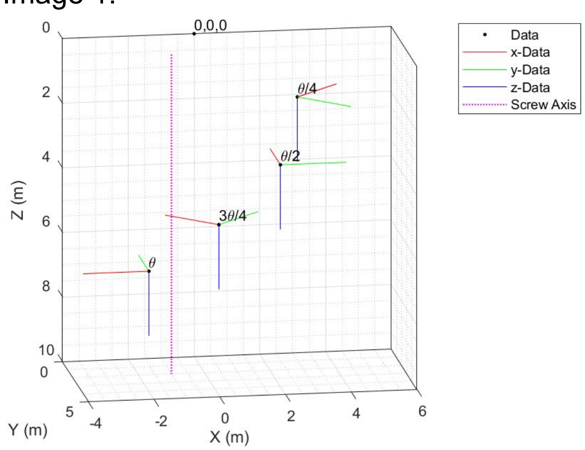


Image 2:

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

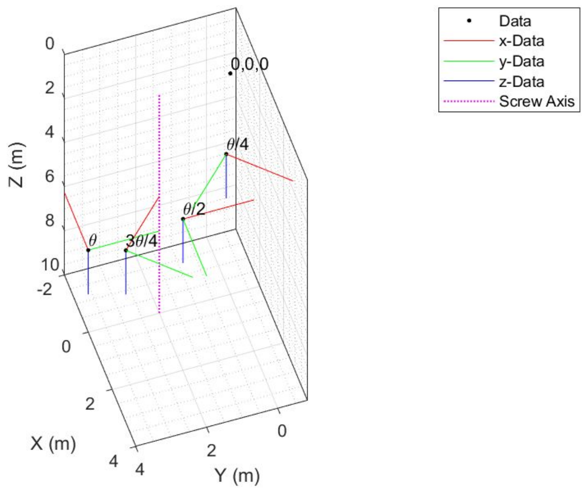


Image 3:

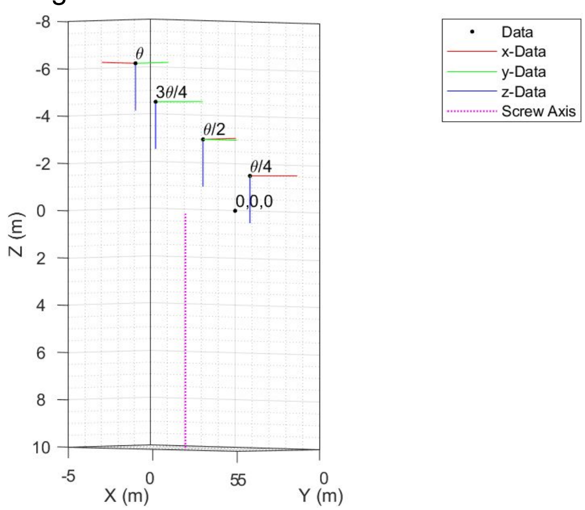


Image 4:

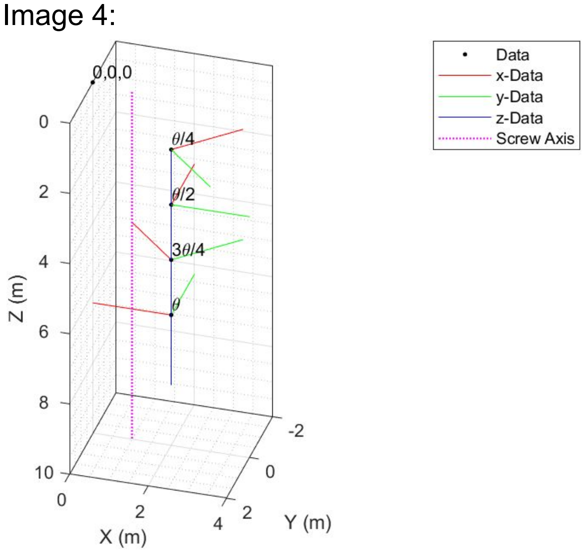
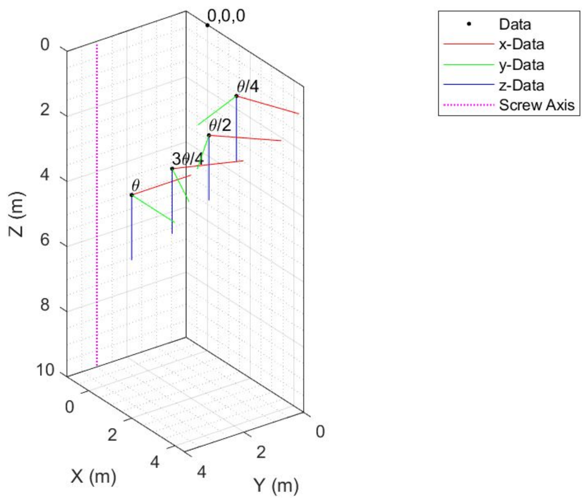


Image 5:

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21



- qsh2screw.m

Test case:	Input:	Output:
Qsh given in problem 3	Q1a=[0 2 0]; S1a=[0 0 1]; H1a=2;	a5 = 0 -1 0 2 1 0 0 0 0 0 0 2 0 0 0 0
Same q and s, h=0	Q1a=[0 2 0]; S1a=[0 0 1]; H1a=0;	b5 = 0 -1 0 2 1 0 0 0 0 0 0 0 0 0 0 0
q=0, same s and h,	Q1a=[0 0 0]; S1a=[0 0 1]; H1a=2;	c5 = 0 -1 0 0 1 0 0 0 0 0 0 2 0 0 0 0
Same q and h, s along x axis	Q1a=[0 2 0]; S1a=[1 0 o]; H1a=2;	d5 = 0 0 0 2 0 0 -1 0 0 1 0 -2 0 0 0 0

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

- configuration_calulator.m

Test Case:	Input:	Output
Theta = 0 Screw has w term	Screw = [0 0 3 0;0 0 0 1;-3 0 0 0;0 0 0 0], Theta = 0 Tinit = [0,-2,0,1;2,0,0,0;0,0,0,0;0,0,0,1]	Config1 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1] :: Config2 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1] :: Config3 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1] :: Config4 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1]
Theta = 1 Screw has w term	Screw = [0 0 3 0;0 0 0 1;-3 0 0 0;0 0 0 0], Theta = 1 Tinit = [0,-2,0,1;2,0,0,0;0,0,0,0;0,0,0,1]	Config1 = [0 -1.4634 0 .7317;2 0 0 .25;0 1.36 0 -.6816;0 0 0 1] :: Config2 = [0 -.1415 0 .0707;2 0 0 .5;0 1.995 0 -.9975;0 0 0 1] :: Config3 = [0 1.2563 0 -.6282;2 0 0 .75;0 1.5561 0 -.7781; 0 0 0 1] :: Config4 = [0 2 0 -1;2 0 0 1;0 .2822 0 -.1411; 0 0 0 1]
Theta = 1 Screw does not have w term	Screw = [0 0 0 0;0 0 0 1;0 0 0 0;0 0 0 0], Theta = 1 Tinit = [0,-2,0,1;2,0,0,0;0,0,0,0;0,0,0,1]	Config1 = [0 -2 0 1;2 0 0 .25;0 0 0 0;0 0 0 1] :: Config2 = [0 -2 0 1;2 0 0 .5;0 0 0 0;0 0 0 1] :: Config3 = [0 -2 0 1;2 0 0 .75;0 0 0 0;0 0 0 1] :: Config4 = [0 -2 0 1;2 0 0 1;0 0 0 0;0 0 0 1]
Theta = 0 Screw does not have w term	Screw = [0 0 0 0;0 0 0 1;0 0 0 0;0 0 0 0], Theta = 0 Tinit = [0,-2,0,1;2,0,0,0;0,0,0,0;0,0,0,1]	Config1 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1] :: Config2 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1] :: Config3 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1] :: Config4 = [0 -2 0 1;2 0 0 0;0 0 0 0;0 0 0 1]

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

- TMatrix2ScrewAngle.m

Test case:	Input:	Output:
Translation only transformation matrix. In this case, omega should be undefined.	T1=[1 0 0 1;0 1 0 1;0 0 1 1; 0 0 0 0];	a7 = NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN 0 0 0 0
Rotation only transformation matrix	T2=[1 0 0 0;0 cos(2) -sin(2) 0;0 sin(2) cos(2) 0;0 0 0 0];	b7 = 0 0 0 0 0 0 -1 0 0 1 0 0 0 0 0 0
Both translation and rotation	T3=[1 0 0 1;0 cos(2) -sin(2) 1;0 sin(2) cos(2) 1;0 0 0 0];	c7 = 0 0 0 0.5000 0 0 -1.0 0.8210 0 1.0 0 -0.1790 0 0 0 0

- Screw2qsh.m

Test Case:	Input:	Output:
W has length >1	S = [0 -1 .333 0;1 0 -.333 0;-.333 .333 0 , 1;0 0 0 0]	q= [.333;-.333;0] s=[.3013;.3013;.9047] h= .9047
W has length = 1	S = [0 -.333 .333 0;.333 0 -.333 0;-.333 .333 0 , 1;0 0 0 0]	q= [.333;-.333;0] s=[.5774;.5774;.5774] h= 1.7338

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

Bonuses(Math#10 and Programming#4)

- Bonus_Math_10.m

Test Case:	Input:	Output:
A is a rotation matrix	A1=[cos(2) 0 sin(2);0 1 0;-sin(2) 0 cos(2)];	a9 = -0.4162 -0.0000 0.9093 0.0000 1.0000 0.0001 -0.9093 0.0001 -0.4162 b9 = 1.5653e-08
Arbitrary A	A2=[1 2 3;4 5 6;7 8 9];	c9 = 0.9736 -0.1305 -0.1872 0.1208 0.9907 -0.0624 0.1936 0.0381 0.9803 d9 = 256.6066

- Matrix_R.m

Test Case:	Input:	Output:
Euler angles (0,0,0)	E1=[0 0 0];	a10 = 1 0 0 0 1 0 0 0 1
Euler angles (0,0,1)	E2=[0 0 1];	b10 = 0.5403 -0.8415 0 0.8415 0.5403 0 0 0 1.0000
Euler angles (1,1,1)	E3=[1 1 1];	c10 = -0.5503 -0.7003 0.4546 0.7003 -0.0906 0.7081 -0.4546 0.7081 0.5403

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

- Matrix_Difference_Norm.m

Test Case:	Input:	Output:
A and R are identical	A1=[1 1 1;1 1 1;1 1 1]; R1=[1 1 1;1 1 1;1 1 1];	a11 = 0
A and R are slightly different	A1=[1 1 1;1 1 1;1 1 1]; R2=[1 1 1;1 1.01 1;1 1 1];	b11 = 1.0000e-04
A and R are arbitrary	A2=[4 6 2;7 6 2;3 4 5]; R3=[3 5 0;1 6 7;6 1 3];	c11 = 89

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

- Bonus_Programming_4.m

Test Case:	Input:	Output:
A is rotation matrix, threshold=.001	A1=[cos(2) 0 sin(2);0 1 0;-sin(2) 0 cos(2)];	a12 = logical 1 b12 = -0.4162 -0.0000 0.9093 0.0000 1.0000 0.0001 -0.9093 0.0001 -0.4162
A is almost rotation matrix, threshold=.001	A2=[cos(2) 0 sin(2);0 1.01 0;-sin(2) 0 cos(2)];	c12 = logical 1 d12 = -0.4162 -0.0000 0.9093 0.0000 1.0000 0.0001 -0.9093 0.0001 -0.4162
A is arbitrary, threshold=.001	A3=[1 2 3;4 5 6;7 8 9];	e12 = logical 0 f12 = 0.9736 -0.1305 -0.1872 0.1208 0.9907 -0.0624 0.1936 0.0381 0.9803

Zahin Nambiar, Maxim Gurevich
THA 1 Programming Documentation
3-2-21

Discussion of Results:

The results tabulated above match our expectations and the Question_3.m file successfully combined multiple functions while producing results that make sense.

Contributions :

1a. R to axis-angle Representation: Zahin Nambiar
1b. R to Quaternion Representation: Zahin Nambiar
1c. R to Euler Representations: Zahin Nambiar

2a. Axis-angle Representation to R: Maxim Gurevich
2b. Quaternion to R: Maxim Gurevich

3a. Multi Step Configuration Calculator given T,S,Theta: Zahin Nambiar
3b. Plotting of the Configurations and Screw Axis: Maxim Gurevich
3c. Screw to qsh: Zahin Nambiar
3d. Qsh to Screw: Maxim Gurevich

Bonus: Maxim