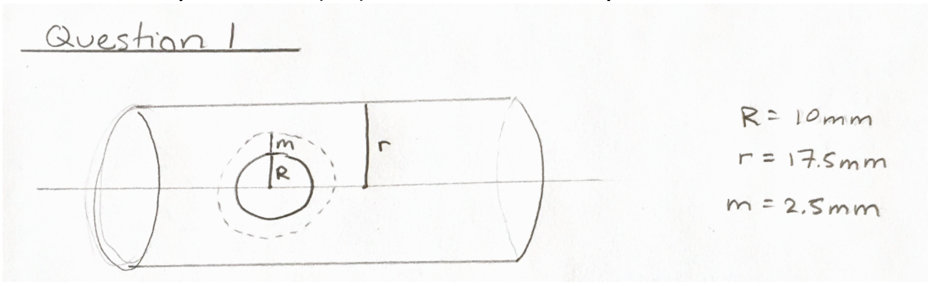
CMPE 330 Assignment 2

Genevieve Hayes
November 3rd, 2020

The script **main_A2.m** will call the testing functions for Assignment 2. All functions and tests are located in the same folder as main_A2.m. To run each question's tests independently within main_A2.m, use the "Run Section" option while the cursor is placed next to the desired test.

Part 1: Compute the Maximum Target Reconstruction Error (TRE)

The tumor is modelled as a sphere with radius R = 10 mm. The irradiation margin around the tumor is m = 5 mm. The Cyber Knife (CK) is modelled as a cylinder with radius r = 17.5 mm.



$$MaxTRE = r - R - m$$
$$MaxTRE = 17.5\ mm - 10\ mm - 5\ mm$$
$$MaxTRE = 2.5\ mm$$

Method

- The following Matlab function was made to test whether the CK cylinder completely covers the tumor sphere:

```
checkCoverage = checkCompleteCoverageOfSphereByCylinder(C,R,r,P,v,margin)
```

- This function returns a 1 if the cylinder completely covers the sphere within the specified margin, a 0 if they don't converge at all and a 2 if they intersect only partially.

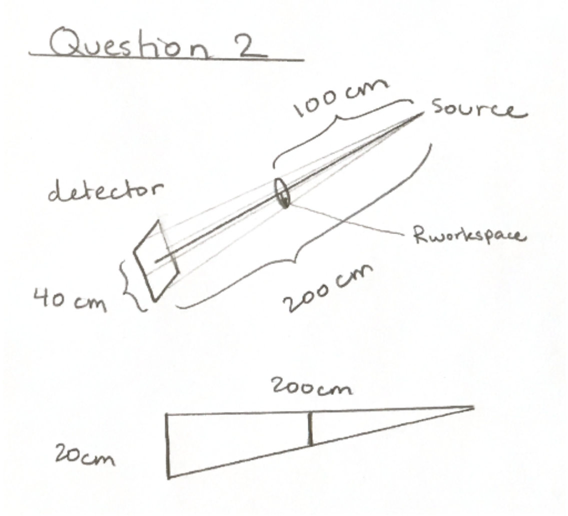
Test Cases

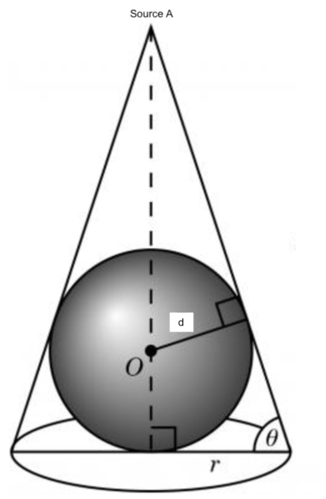
- Test 1: Case that the cylinder completely coverages the sphere with the required margin.
- Test 2: Case that the cylinder partially coverages the sphere with the required margin.
- Test 3: Case that the cylinder does not touch the sphere.

Part 2: Compute the Imaging Workspace

```
function [Rworkspace] = calculateWorkspaceRadius(d_StoD,sideLength_D)
```

The circular workspace is located halfway between the source and the detector. The detector is a square with side lengths of 400 mm and the distance between the source and the detector is 2000 mm. The spherical workspace must be contained in a cone-line shape which is the source projection onto the detector.





Method

- The spherical workspace must be contained in a cone-line shape which is the source projection onto the detector. Note the symmetry between the A and B sources and detectors.
- The shortest distance between the origin and the edge-line of the cone (extending from the to the edge of the detector to the source) using 2 points on a line and a point.

```
function [distance] = distanceBetweenLineAndPoint(L1, L2, P)
```

$$\overrightarrow{d_{PtoL1}} = L_1 - P$$

$$\vec{L} = \frac{L_2 - L_1}{||L_2 - L_1||}$$

$$\vec{d} = \overrightarrow{d_{PtoL1}} - \left(\overrightarrow{d_{PtoL1}} \cdot \vec{L}\right) * \vec{L}$$

$$d = \|\vec{d}\|$$

- Applying this in the CK frame to the edge line of the source beam (L1 = Source A Location, L2 = Detector A edge) passing by the origin (P = (0,0,0)) Rworkspace was found to be 99.504mm.

$$Axis\ Detector\ Distance\ (ADD)\ = 1000mm$$
$$Source\ Axis\ Distance\ (ADD)\ = 1000mm$$

$$\overrightarrow{d_{PtoL1}} = (ADD * cos(45), SAD * \sin(45), 0) - (0,0,0)$$

$$\overrightarrow{d_{PtoL1}} = (0, 1000mm, 0)$$

$$\vec{L} = \frac{(-SAD * cos(45), -SAD * \sin(45), 0) - (ADD * cos(45), ADD * \sin(45), 0)}{||L_2 - L_1||}$$

$$\vec{L} = (0.099504, -0.99504, 0)$$

$$\vec{d} = (0, 1000mm, 0) - ((0, 1000mm, 0) \cdot (0.099504, -0.99504, 0)) * (0.099504, -0.99504, 0)$$

$$\vec{d} = (99.01mm, 9.901mm, 0)$$

$$d = 99.504mm$$

Test Cases

- Case where d_StoD = 200 cm and sideLength_D = 40 cm. The result is Rworkspace = 99.504mm as expected.

Part 3: Generate the Frame Transformations

```
function [TransCKtoA,TransCKtoB] = generateFrameTransformsForCKToDetectors()
```

This function calculates the transformation matrices to take a point from the Cyber Knife frame to the Detector A and B frames respectively. Detectors A and B are located at 100 cm away from the CK origin at -45 degrees from the x-axis about the +z-axis and +45 degrees from the x-axis about the +z-axis respectively.

Method

- First the Cyber Knife orthogonal basis vectors were calculated in both the Detector A and the Detector B frames by rotating the standard e1 = (1,0,0), e2 = (0,1,0) and e3 = (0,0,1) home basis by +45 degrees and -45degrees about the +z-axis respectively. This rotation was achieved using the following function (methods for this function are explained in the Appendix):

```
function [R3by3,R4by4] = rotationMatrixAboutFrameAxis(axis,angle)
```

- The origins of the CK relative the Detector A and B frames were both taken to be OCK = (0,100,0). The frame transformations for the Cyber Knife to the detector frames were achieved by using these origin points and the CK orthogonal basis vectors calculated above in the following function (methods for this function are explained in the Appendix):

```
function [T_e2h] = generateFrameTransformationToHome(Oe,e1,e2,e3)
```

Test Cases

- Test 1: Case where the point to transform is located at the Cyber Knife origin, P_CK1 = (0,0,0). The expected output for Detector A is P_A1 = (0,1000,0). The expected output for Detector B is P_B1 = (0,1000,0).
- Test 2: Case where the point to transform is located an extra y_B = 100 cm beyond the source relative to Detector B, P_CK2 = ($50\sqrt{2}$,$50\sqrt{2}$,0). The expected output for Detector A is P_A2 = (100,100,0). The expected output for Detector B is P_B2 = (0,200,0).
- Test 3: Case where the point to transform is located at the Detector A origin, P_CK3 = ($50\sqrt{2}$,-$50\sqrt{2}$,0). The expected output for Detector A is P_A3 = (0,0,0). The expected output for Detector B is P_B3 = (100,100,0).

Part 4: Develop a Digital Radiography Projector

```
function [ImgPtsA, ImgPtsB] = projectCKpointOntoDetectorImages(MarkerPtsCK,
SDD, ADD, angleA,angleB)
```

This “digital radiography projector” takes a marker point in 3D space and projects it onto Detectors A and B, returning the image points relative to their respective detectors.

Method

- Transform the point from Cyber Knife frame to Detector A and Detector B frames.
- Use intersect line with plane function to see where point relative to Detector A intersections with the Detector A “plane”. This intersection was achieved using the following function (methods for this function are explained in the Appendix):

```
function Int = intersectionOfVectorWithPlane(A,n,P,v)
```

- This is repeated for Detector B to reprieve ImgPtsA and ImgPtsB.

The input to the DR projector can also be a 2D array of vectors where each column represents one point in 3D space in the Cyber Knife frame.

Test Cases

- Test 1: Case where the point to transform is located at the Cyber Knife origin, P_CK1 = (0,0,0). The expected output for the Detector A image is ImgAPt = (0,0,0). The expected output for Detector B image is ImgBPt = ($50\sqrt{2}$,- $50\sqrt{2}$,0).
- Test 2: Case where the point to transform is located at the edge of the workspace when Y_CK = 0 and Z_CK = 0. Using the geometry of the situation, one edge of the workspace will appear at 100 mm rotated +45 degrees, P_CK2 = (,0,0). The expected output for the Detector A Image is ImgAPt = (0,0,0). The expected output for the Detector B Image is at the edge of the detector, which relative to its center is ImgBPt = (200,0,0).

Part 5: Reconstruct Target Points

```
function [TargetPtsCK3D,REMs] =
reconstructCKTargetMatrixFromDetectorImages(ImgPtsA,ImgPtsB, ADD, angleA,
angleB)
```

The target reconstructor takes Detector A and B image points relative to their respective detectors and reconstructs the marker in 3D space relative to the Cyber Knife.

Method

- First the image points are transformed from the detector frames into the CK frame using the inverse of the transformation matrices made above.
- The image points are then back projected toward that workspace by correlating the points from image A and image B. This correlation was achieved using the following function (methods for this function are explained in the Appendix):

```
function [M,dM] = closestIntersectionOf2Vectors(P1,v1,P2,v2)
```

- The Euclidean error in the distance between the two lines was taken as the Residual Error Metric (REM) for that point.
- Each pair of points are reconstructed in the following function

```
function [TargetPtCK,REM] =
generateCKTargetPointAndREMfromABImagePoint(ImgPtA, ImgPtB, ADD, angleA,
angleB)
```

- The full `reconstructCKTargetMatrixFromDetectorImages` function loops through multiple points within an input MarkerPtsCK matrix and returns all possible combinations of reconstructed points as well as the REMs matrix for all of those combinations.

Test Cases

- Test 1: Case where the point to transform is located at the Cyber Knife origin, MarkerPtCK = (0,0,0). The expected output is to get back the TargetPtCK = (0,0,0) after the marker was transformed to the Detector A image, ImgAPt = (0,0,0) and Detector B image, ImgBPt = ($50\sqrt{2}$,- $50\sqrt{2}$,0) with a REM of ~0.
- Test 2: Case where the point to transform is located at the center of Detector A relative to the Cyber Knife, MarkerPtCK = ($50\sqrt{2}$,- $50\sqrt{2}$,0). The expected output is to get back the TargetPtCK = ($50\sqrt{2}$,- $50\sqrt{2}$,0) after the marker was transformed to the Detector A image, ImgAPt = (0,0,0) and Detector B image, ImgBPt = (2000,0,0) with a REM of ~0.
- Test 3: Case where the point is located in the workspace at MarkerPtCK = (10,10,0). The expected output is to get back the TargetPtCK = (10,10,0) after the marker was transformed to the Detector A and B images with a REM of ~0.
- Test 4: Case where a matrix of multiple points (three here) are used as inputs which are DR projected and then all possible target locations are reconstructed. The diagonal of these matrices returns the same target locations as the input locations along with REM's of ~0.

Part 6: Generate Target Matrix of Correspondences

```
function [correspondence] = buildCorrespondenceMatrixFromREMs(REMs)
```

The correspondence matric is built from the REMs matrix developed from reconstructing the Cyber Knife target points from Detector A and B images. The correspondence matrix consists of two columns containing the indices of the target number in the Detector A and B image points respectively. If the image points are fed in already in order where the first imgPtA corresponds to the first imgPtB and so on, the correspondence matrix will be as follows:

Correspondence A	Correspondence B
1	**1**
2	**2**
3	**3**

Method

- First check is the REMs matrix has ambiguity by finding all values that are essentially zero and sending an error is more than one zero value is located in a single column.
- If the matrix is not ambiguous, the index of smallest REM in each column is found.
- The target points are number with respect to image A points and then the corresponding image B is the index of the smallest REM in each column as found above.

Test Cases

- Test 1: Case where the 3 markers are located at M1 = (0,0,-50), M2 = (0,0,0), M3 = (0,0,50). These points are projected back to the detector images and then reconstructed. They were reconstructed successfully with a TRE of effectively 0.
- Test 2: Case where the 3 markers are located at M1 = (0,0,0), M2 = (50,0,0), M3 = (0,0,50). These points are projected back to the detector images and then reconstructed. They were not reconstructed successfully since the REMs matrix was ambiguous as some of these points lie in the same plane.

Part 7: Target Reconstruction Error Simulation

Initialize Simulation

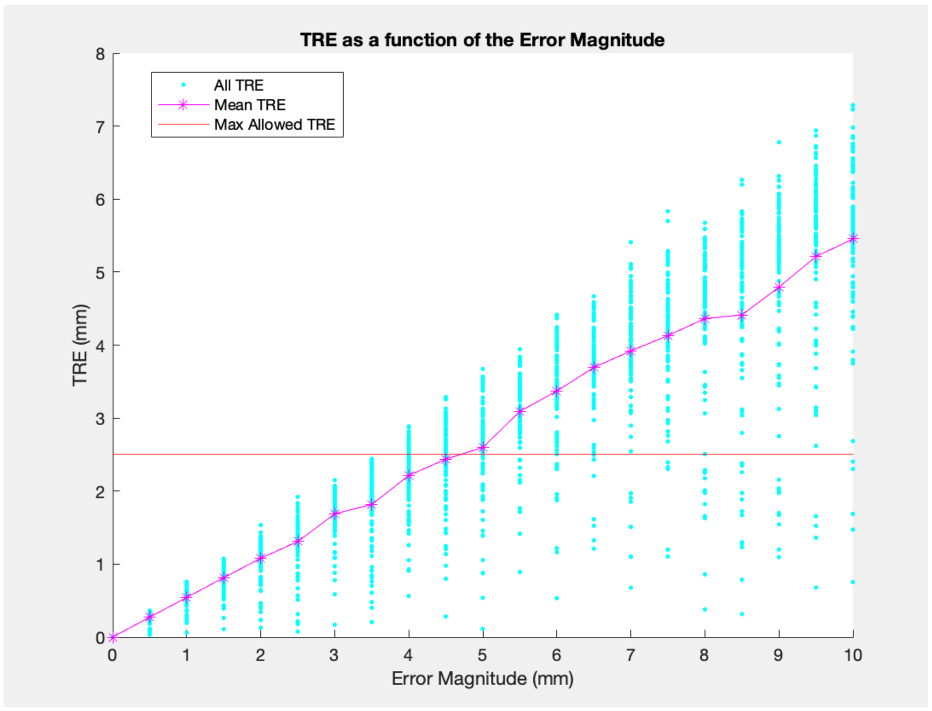
- Estimate Point of Failure: the point of failure is expected to be around **5mm**. Since the maximum allowed TRE is 2.5 mm and the detector receives the workspace projection at a factor of 2 magnification because the source is double the distance from the detector as the workspace, the maximum error will be approximately double the MaxTRE.
- 100 random markers were generated within the workspace using the Rworkspace calculated above as the limit to the magnitude of a random 3D vector between 0 and Rworkspace.
- Each marker is projected back the detector images to get 2D arrays of points for Detector A and B respectively.

Simulation Cycle

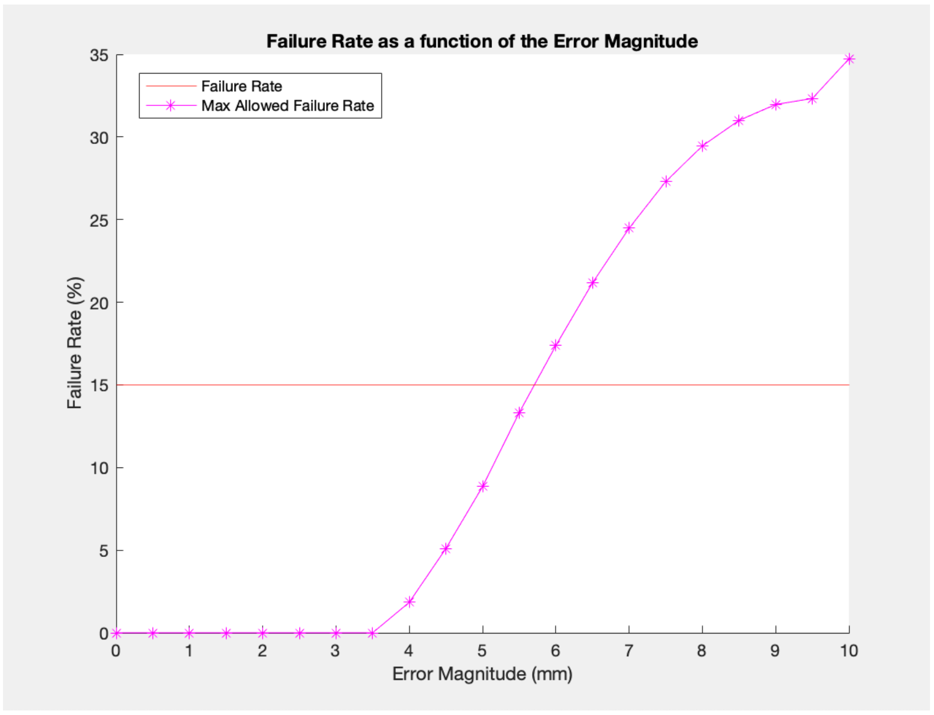
A random error was applied to the Y and Z components of the detector A and B images. The magnitude of the error was incremented by 0.5 mm from 0 mm to 10mm each applied to all 100 randomly generated markers. The reconstructor was applied the new “jittered” detector image points to get the target. The magnitude of the difference between the original marker and the “jittered” reconstructed target was defined as the TRE. The failure rate was also calculated within the 100 markers at each error magnitude. The diagonals of the REMs matrix were taken to illustrate the correlation between the TRE and the magnitude of the error as well as the TRE and the REM as illustrated below.

Analysis

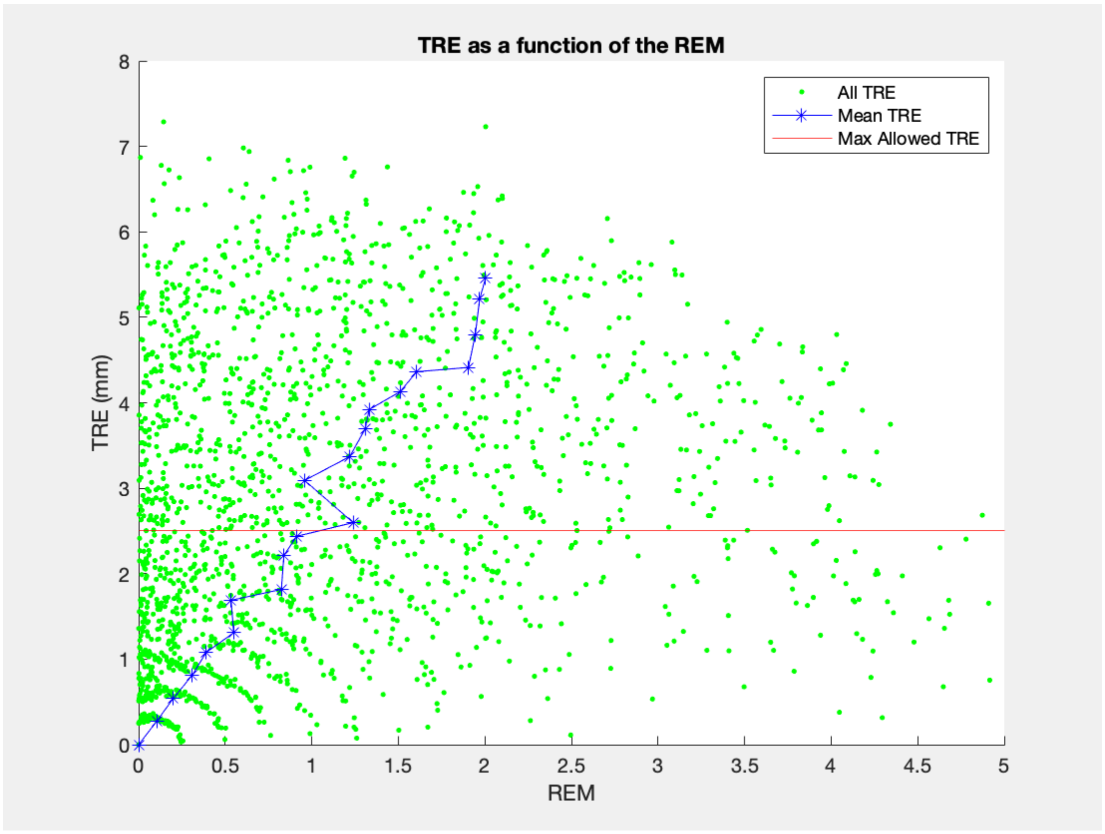
The TRE was plotted as a function of the error magnitude as shown below. The range of TRE values increases as the error magnitude increases. As expected, the mean TRE reaches the MaxTRE just before the error magnitude reaches 5mm.



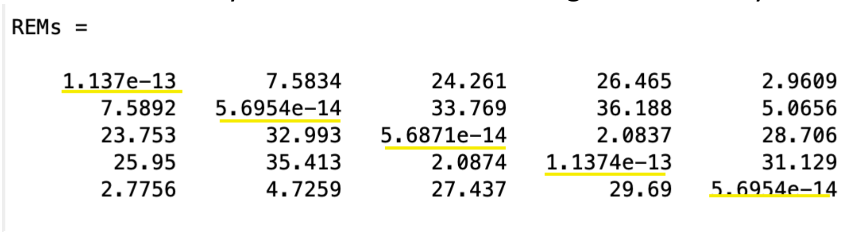
The failure rate was plotted as a function of the error magnitude as shown below. Since the random error magnitude was applied to both the Detector A and B images, it can be expected that in some cases the errors will converge to more failures. The failure rate exceeds an appropriate level right around 5mm again, as expected.



The TRE was plotted as a function of REM as shown below. High REMs appear to mostly correlate with high TRE values, though there are some large REMs that corresponds to low TRE values. However, low REMs do not necessarily correspond to low TREs and should not be trusted to predict an accurate reconstruction.

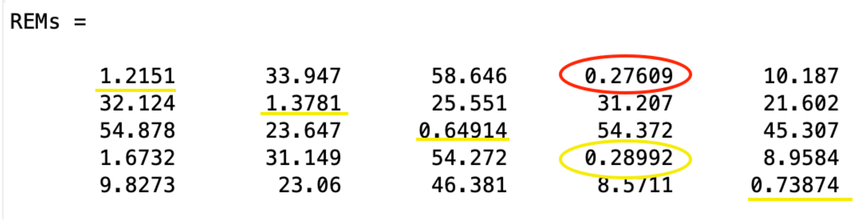


Extra Note: Using the REMs matrix to build a correspondence matrix
Very occasionally (about every hundredth random point generated) some of the REMs indicate that the marker location is best reconstructed at a different intersection correlation than the actual marker location. A correctly correlated REMs matrix for 5 target points is illustrated below followed by a REMs matrix illustrating an incorrectly reconstructed point.



```
REMs =

    1.137e-13       7.5834       24.261       26.465       2.9609
       7.5892   5.6954e-14       33.769       36.188       5.0656
       23.753       32.993   5.6871e-14       2.0837       28.706
        25.95       35.413       2.0874   1.1374e-13       31.129
       2.7756       4.7259       27.437        29.69   5.6954e-14
```



```
REMs =

       1.2151       33.947       58.646      0.27609       10.187
       32.124       1.3781       25.551       31.207       21.602
       54.878       23.647      0.64914       54.372       45.307
       1.6732       31.149       54.272      0.28992       8.9584
       9.8273        23.06       46.381       8.5711      0.73874
```

This illustrates yet again some of the short comings of the REMs matrix to reconstruct target markers from non-perfect image points.

Appendix: Additional Functions

Rotation-About-Frame-Axis (Assignment 1 Question 8)

```
function [R3by3,R4by4] = rotationMatrixAboutFrameAxis(axis,angle)
```

Method

- Given θ, the 3x3 rotation matrices are found using:

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

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

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

- These can be made into the 4x4 homogeneous rotation matrices by padding the 4th column with [0,0,0,1] and the 4th row will be [0,0,0,1] as well.

Frame-Transformation-to-Home (Assignment 1 Question 9)

```
function [T_e2h] = generateFrameTransformationToHome(Oe,e1,e2,e3)
```

Method

- The translation matrix is given by:

$$Trans_{e\ to\ h} = \begin{pmatrix} 1 & 0 & 0 & O_{ex} \\ 0 & 1 & 0 & O_{ey} \\ 0 & 0 & 1 & O_{ez} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The rotation matrix is given by:

$$Rot_{e\ to\ h} = \begin{pmatrix} e1_x & e2_x & e3_x & 0 \\ e1_y & e2_y & e3_y & 0 \\ e1_z & e2_z & e3_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- These transformations can be used to transform a point in the orthonormal e-basis to the orthonormal h-basis (home frame):

$$P_h = Trans_{e\ to\ h} * Rot_{e\ to\ h} * P_e$$

- This can be written in the form of a single transformation matrix operation:

$$P_h = T_{e\ to\ h} * P_e$$

Where

$$T_{e\ to\ h} = Trans_{e\ to\ h} * Rot_{e\ to\ h}$$

Intersect-Line-and-Plane (Assignment 1 Question 2)

```
function Int = intersectionOfVectorWithPlane(A,n,P,v)
```

Method

- Check if line is parallel with plane by taking the cross product of the normal vector and the direction vector.
    - If they are parallel return [NaN, NaN, NaN] for intersection.
    - If they are not parallel, compute their time of intersection, t using:

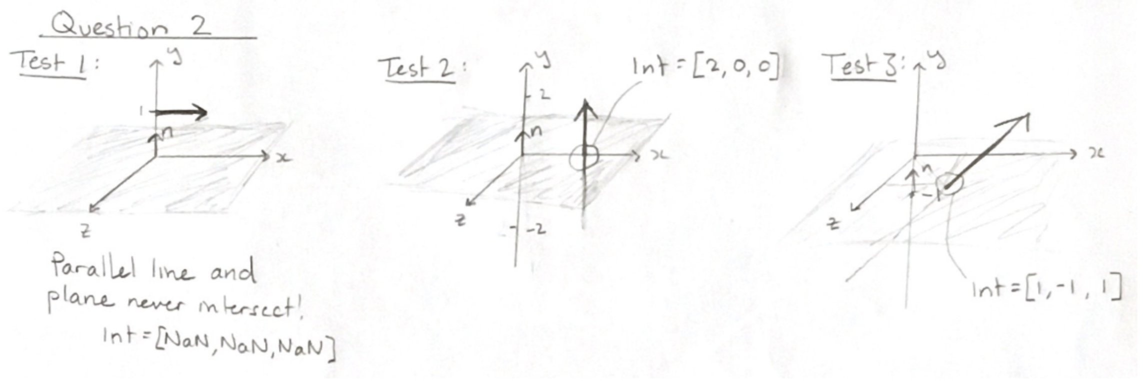
$$t_{int} = \frac{(A - P) \cdot n}{v \cdot n}$$

Then compute the intersection point:

$$Int = P + tv$$

Testing

- Test 1: Case of vector parallel to plane. The expected output is [NaN, NaN, NaN] for the intersection.
- Test 2: Case of Y direction vector perpendicular to X plane, intersecting at x = 2 (Int = [2,0,0]).
- Test 3: Case of Y-Z direction vector passing through a plane below the X – axis. The expected intersection is at x = 1, y = -1, z = 1 (Int = [1,-1,1]).



Intersect-Two-Lines (Assignment 1 Question 1)

```
function [M,dM] = closestIntersectionOf2Vectors(P1,v1,P2,v2)
```

Method

- Check if lines are parallel by taking the cross product of their direction vectors.
    - If the lines are parallel ($v_2 \times v_1 = 0$)return [NaN, NaN, NaN] for intersection and error.
    - If the lines are not parallel, create v_3, V and P such that:

$$v_3 = v_2 \times v_1$$
$$V = \begin{bmatrix} -v_1 \\ v_2 \\ v_3 \end{bmatrix}$$
$$P = \begin{bmatrix} P_{1x} - P_{2x} \\ P_{1y} - P_{2y} \\ P_{1z} - P_{2z} \end{bmatrix}$$

Using the formula of a line $L = P + tV$, determine t of closest intersection by

$$t_{int} = V^{-1} \times P$$

Then compute the location on the line of closest intersection, $L_1 = P_1 + t_{int}V_1$ and $L_2 = P_2 + t_{int}V_2$. The midpoint between these gives the closest intersection point:

$$M = \frac{(L_2 + L_1)}{2}$$

with error:

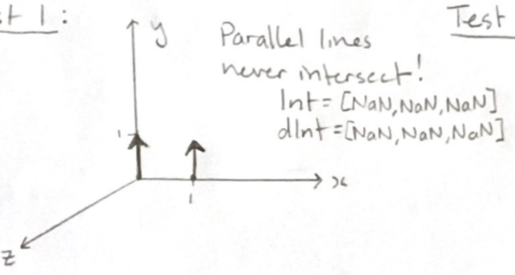
$$dM = abs\left(\frac{(L_2 - L_1)}{2}\right)$$

Testing

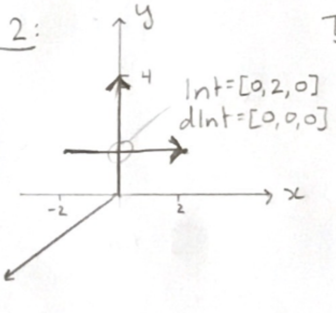
- Test 1: Case of 2 parallel unit vectors. The expected output is [NaN, NaN, NaN] for both M and dM.
- Test 2: Case of perpendicular, intersecting vectors. The expected output is an intersection at y = 2 (M = [0,2,0]) with dy = 0 (dM = [0,0,0]).
- Test 3: Case of vectors that never intersect. Their closest approach occurs when L1 = [0,0,0] and L2 = [1,0,0]. The expected approximate intersection is [0.5,0,0] with error [0.5,0,0].

Question 1

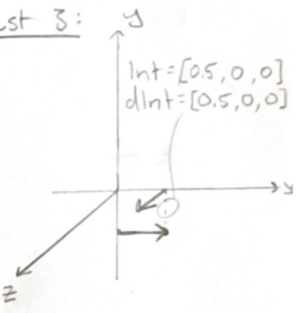
Test 1:



Test 2:



Test 3:


y
lnt=[0.5,0,0]
dlnt=[0.5,0,0]
y
z