

CISC 330

Final Project

Eric Braun 10121660

December 14, 2015

13eb20@queensu.ca

Gamma Knife Radiosurgery

1. Compute Dose Box

The dose box is the smallest cube that we can create which will completely encompass both the PTV and the OAR. To define this cube, we will return the two diagonal corners of the cube.

One corner (LowerLeft) will be the minimum values of the x, y, & z components of the PTV and OAR, while the opposite corner (UpperRight) will be the max values.

Inputs:

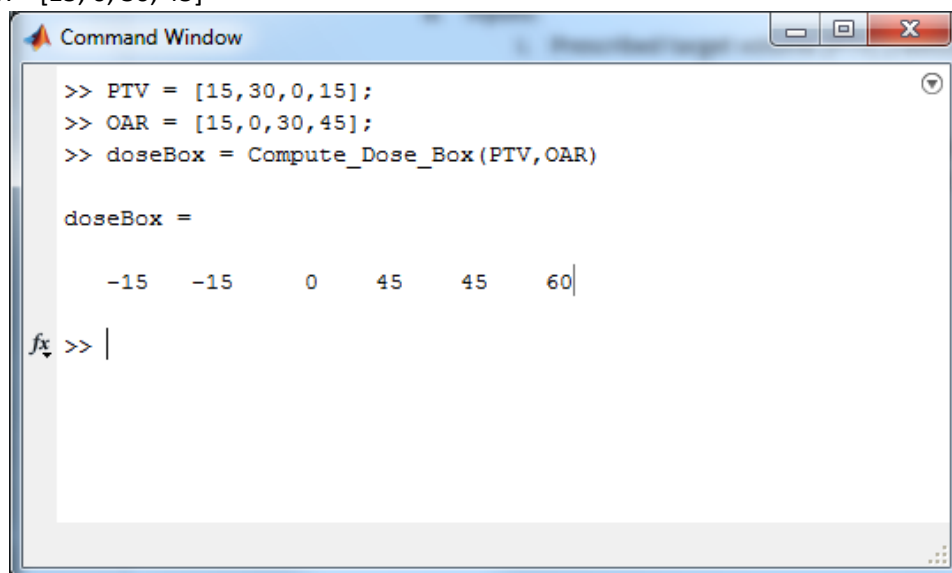
- Prescribed target volume (PTV) [radius, x, y, z]
- Organ at risk (OAR) [radius, x, y, z]

Output:

- Smallest 3D box that could fit around both the PTV & OAR (doseBox)
 - o This consists of the lower-left hand (min-y) and upper-right hand (max-y) corner of the box.

Test:

- PTV = [15, 30, 0, 15]
- OAR = [15, 0, 30, 45]



```
Command Window

>> PTV = [15,30,0,15];
>> OAR = [15,0,30,45];
>> doseBox = Compute_Dose_Box(PTV,OAR)

doseBox =

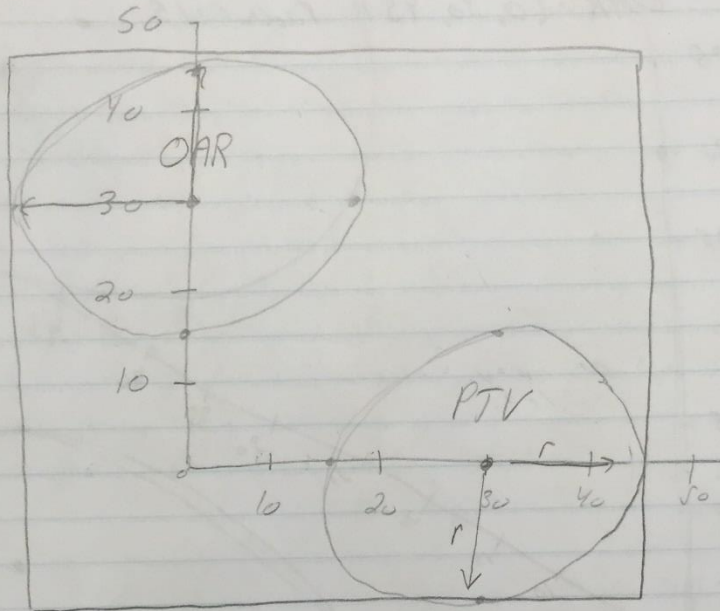
    -15    -15         0     45     45     60

fx >> |
```

Compute Dose Box

$$PTV = (30, 0, 15) \quad r = 15$$

$$OAR = (0, 30, 45) \quad r = 15$$



$$\begin{aligned} \text{Leftmost} &= OAR(x) - \text{radius} \\ &= 0 - 15 = -15 \end{aligned}$$

$$\begin{aligned} \text{Bottom} &= PTV(y) - \text{radius} \\ &= 0 - 15 = -15 \end{aligned}$$

$$\begin{aligned} \text{Lower Left} \\ &= (-15, -15, 0) \end{aligned}$$

$$\begin{aligned} -Z &= PTV(z) - \text{radius} \\ 15 - 15 &= 0 \end{aligned}$$

$$\begin{aligned} \text{Upper Right} \\ &= (45, 45, 60) \end{aligned}$$

$$\begin{aligned} \text{Rightmost} &= PTV(x) + \text{radius} \\ &= 30 + 15 = 45 \end{aligned}$$

$$\text{dose Box} = (-15, -15, 0, 45, 45, 60)$$

$$\begin{aligned} \text{Top} &= OAR(y) + \text{radius} \\ &= 30 + 15 = 45 \end{aligned}$$

$$\begin{aligned} +Z &= OAR(z) + \text{radius} \\ &= 45 + 15 = 60 \end{aligned}$$

- Hand calculated ground truth.

2. Draw 3D Scene

This function will create a 3D rendition of the Gamma-Knife radiosurgery in question. The head will be represented by an ellipsoid, PTV is represented by a green sphere, OAR by a red sphere, isocenter by a blue point, and dose box by a red cube.

Inputs:

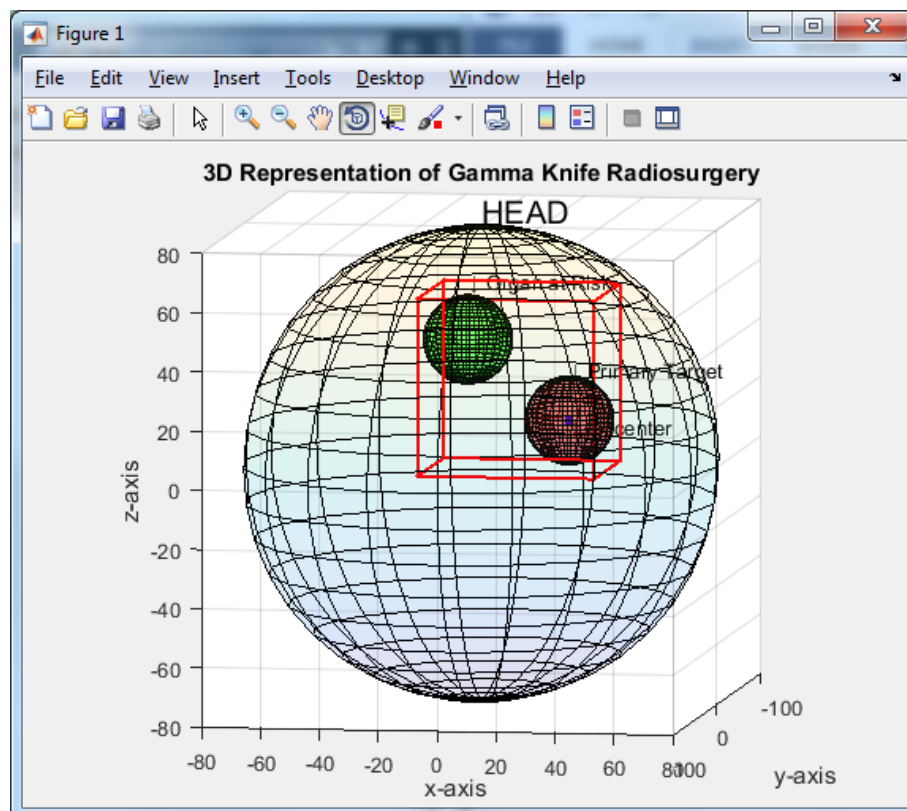
- Head [a, b, c, x, y, z]
- PTV [radius, x, y, z]
- OAR [radius, x, y, z]
- Isocenter Point [x, y, z]

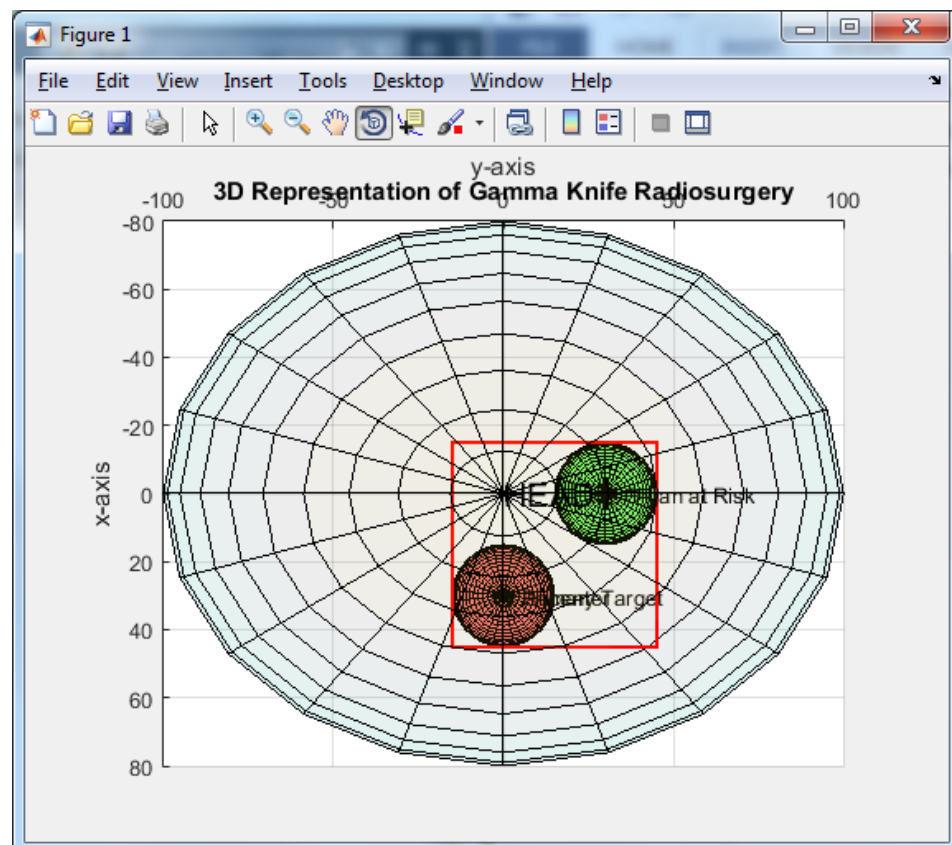
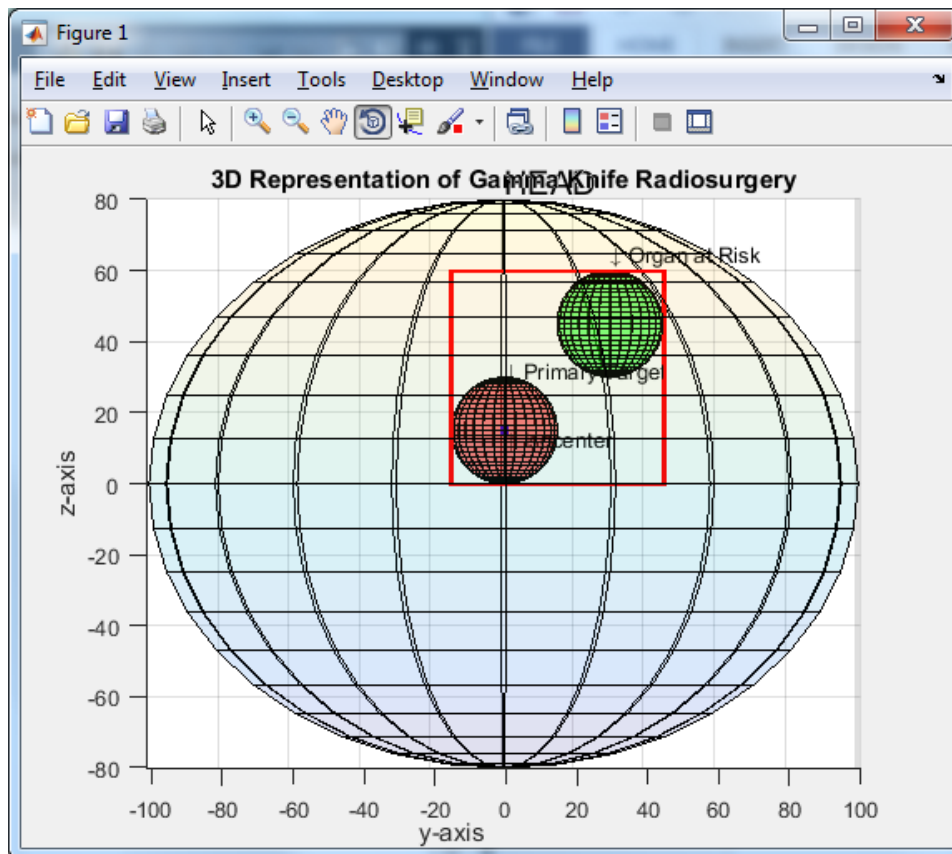
Output: N/A

- Displays a 3D plot

Test:

- Draw_3D_Scene([80,100,80,0,0,0],[15,30,0,15],[15,0,30,45],[30,0,15],doseBox)
- doseBox computed using Compute_Dose_Box
- isocenter located in center of PTV





3. Compute Linear Function

Function is used to compute the y value given two 2D points and an x value. We can do this by creating a line between the two points given. We calculated slope as rise/run, or using point-slope formula: $y - y_1 = m (x - x_1)$. Once we have found the slope, we need to calculate the y-intercept. We will do this by using question 2 from the first assignment, finding the intercept of two lines, each defined by two points. We define the first line as our two lines and the second as the origin [0,0,0] and [0,1,0] (y-axis). We now have the components to define the line: $y = mx + b$. We can plug in any x value to calculate the y value.

Inputs:

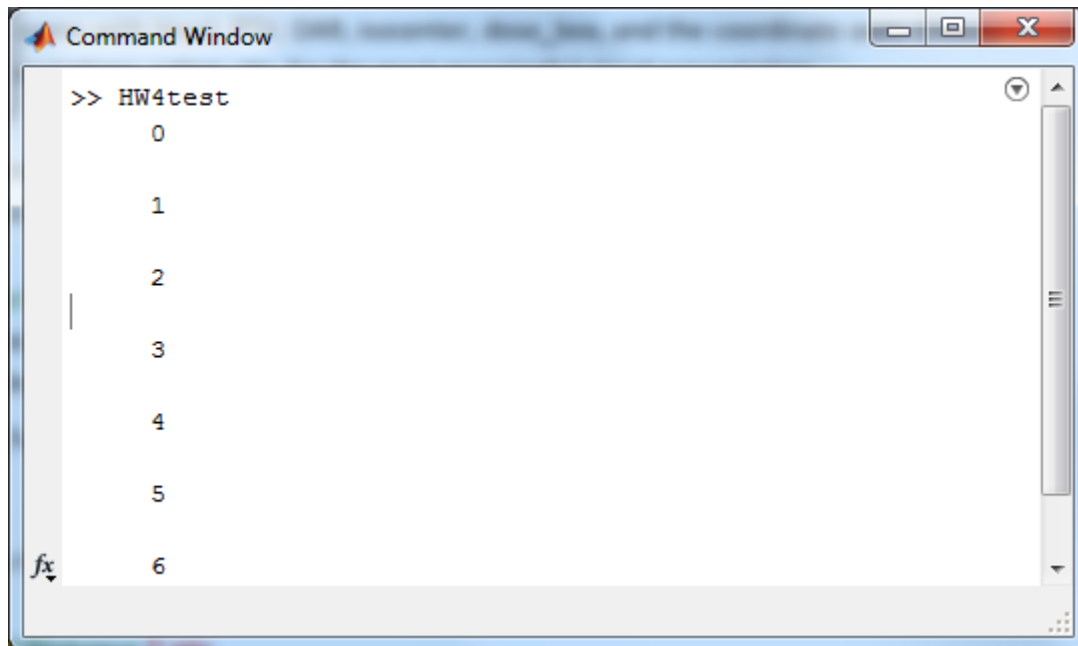
- Point1 [x, y]
- Point2 [x, y]
- X-value

Output:

- Y-value

Test:

- Point1 = [1,1]
- Point2 = [5,5]
- X = {0, 1, 2, 3, 4, 5, 6}



- Y values for computing linear function using P1 = [1,1] P2 = [5,5] x = {1,2,3,4,6}
- The two points create line: $y = x$ slope of 1. Any x value plugged in will yield the same y value

4. Compute Radial Distance

Function is used to compute the distance between a line, defined by a point on the line and the direction vector, and an arbitrary point in space. This is used in radiosurgery to find how far away certain point is from the center line of a beam. To do this we will compute the direction vector between the point on the line and the given point. For the point to be the closest to the line, the line created between it and the line must be perpendicular to the line, so we take the cross product of the two direction vectors. Taking the norm of this, will give us the scalar distance.

Inputs:

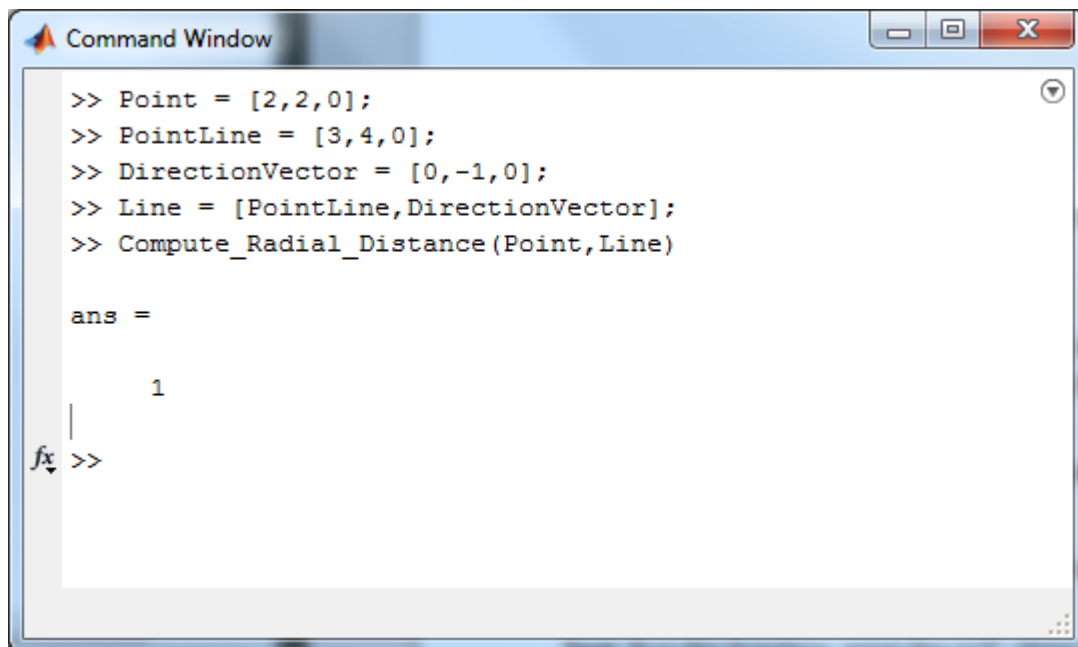
- Arbitrary point $[x, y, z]$
- Line
 - o Point on line $[x, y, z]$
 - o Direction vector $[x, y, z]$

Output:

- Distance between point and line

Tests: Two ground truths

- Case one: Point and a vertical line.
 - o Because the line is vertical, the distance will just be the difference in x-values.

A screenshot of a MATLAB Command Window. The window has a title bar with the text "Command Window" and standard minimize, maximize, and close buttons. The command history shows the following code:

```
>> Point = [2,2,0];  
>> PointLine = [3,4,0];  
>> DirectionVector = [0,-1,0];  
>> Line = [PointLine,DirectionVector];  
>> Compute_Radial_Distance(Point,Line)
```

The output shows

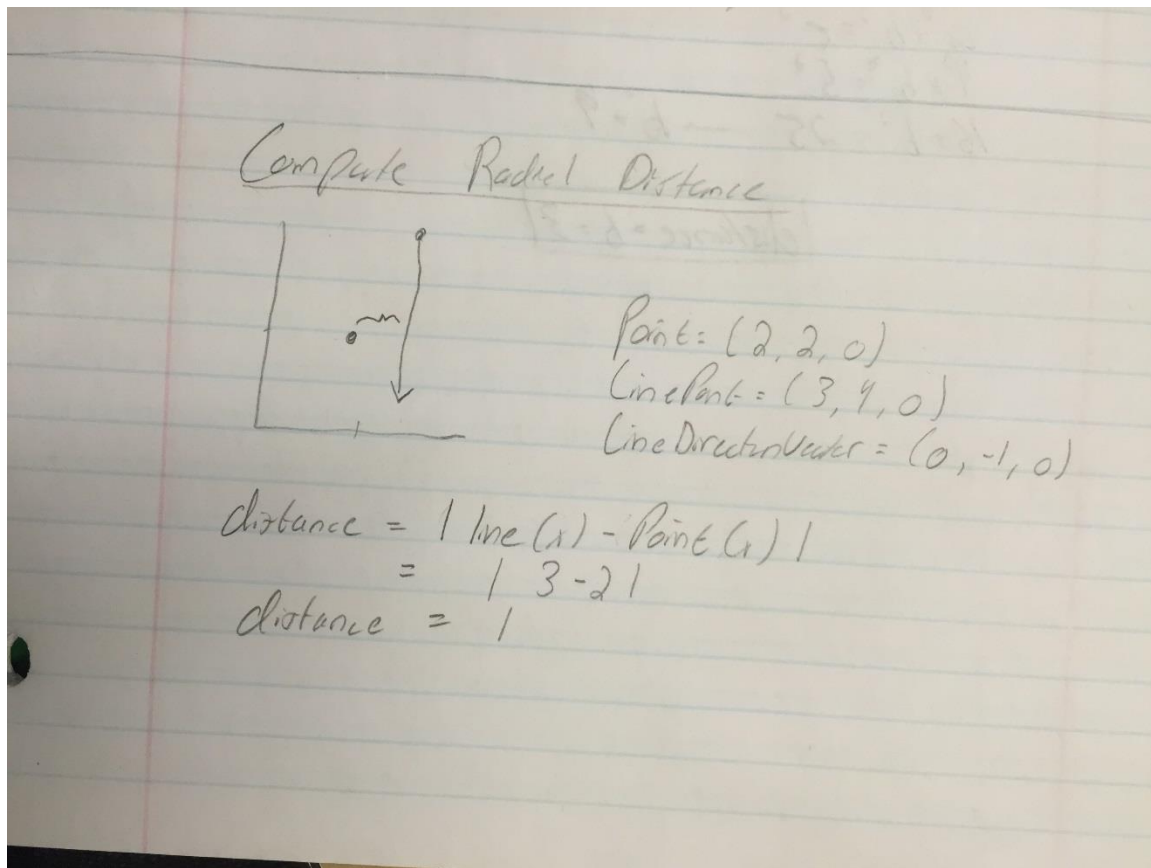
```
ans =  
  
1
```

 followed by a cursor and the prompt

```
fx >>
```

.

```
Command Window  
-- -- X  
  
>> Point = [2,2,0];  
>> PointLine = [3,4,0];  
>> DirectionVector = [0,-1,0];  
>> Line = [PointLine,DirectionVector];  
>> Compute_Radial_Distance(Point,Line)  
  
ans =  
  
1  
fx >>
```



- Hand calculated ground truth for case 1.
- Case two: Point & line creating a right triangle.

```

Command Window

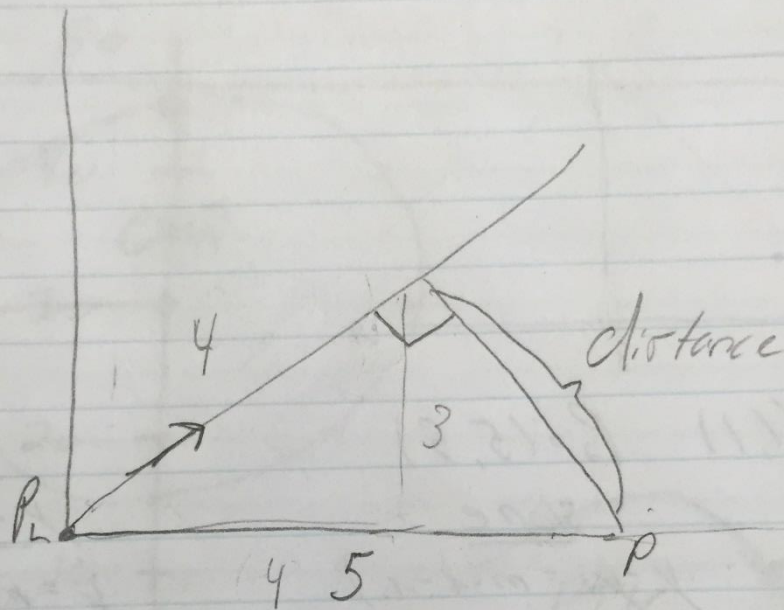
>> Point = [5,0,0];
>> PointLine = [0,0,0];
>> DirectionVector = [4,3,0];
>> Line = [PointLine,DirectionVector];
>> Compute_Radial_Distance(Point,Line)

ans =

     3

fx >>
  
```


Compute Radial Distance



$$\begin{aligned} \text{Point} &= (5, 0, 0) \\ \text{Line Point} &= (0, 0, 0) \quad \text{Direction Vector} = (4, 3, 0) \end{aligned}$$

$$\begin{aligned} a^2 + b^2 &= c^2 \\ 4^2 + b^2 &= 5^2 \\ 16 + b^2 &= 25 \rightarrow b^2 = 9 \end{aligned}$$

$$\boxed{\text{distance} = b = 3}$$

- Hand calculated ground truth for case 2.

5. Compute Depth Dose

To create this function I analyzed the graph given in the assignment (shown below). We are given d_0 : the distance off the skin entry point that will receive the highest dose of radiation. From the graph, we can see that $6 (6 * d_0) = \text{half the dose}$.

I created conditional statements: if the distance was between 0 and d_0 , I would use the Compute_Linear_Function seen in question 3, to find the y value given an x. If the distance was greater than d_0 , I would use the line created between $[d_0, 1]$ and $[6 * d_0, 0.5]$ and the distance to find the dose value.

In my actual code, when using the head, I would use the maxDepth as $2 * \max(a,b,c)$, which will be the largest diameter the head could be (furthest a beam would have to go).

Inputs:

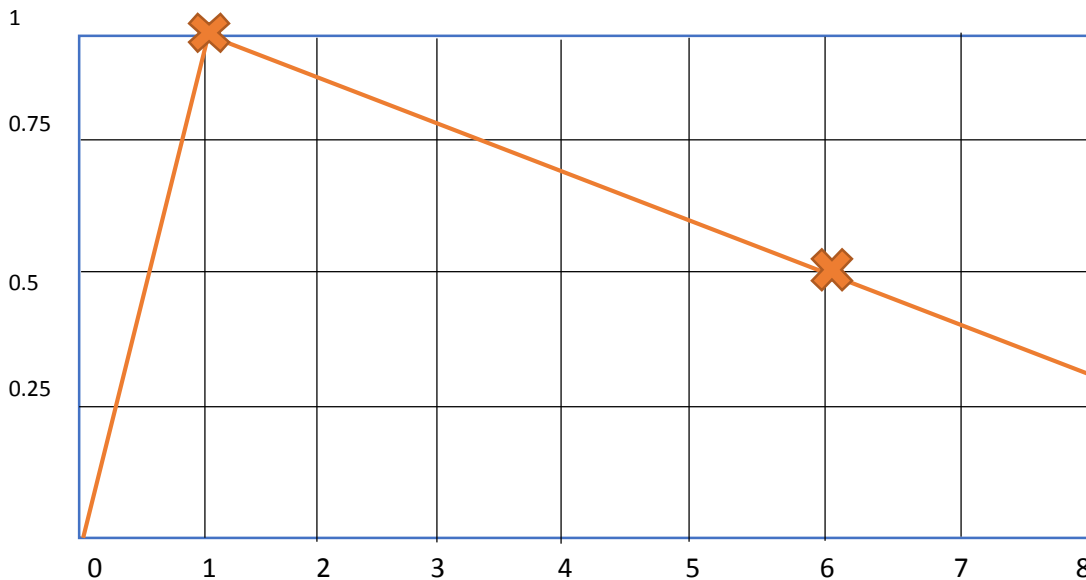
- Max depth the x-ray can radiate
- Distance which gets the greatest amount of radiation from skin entry
- Increment (skin resolution)

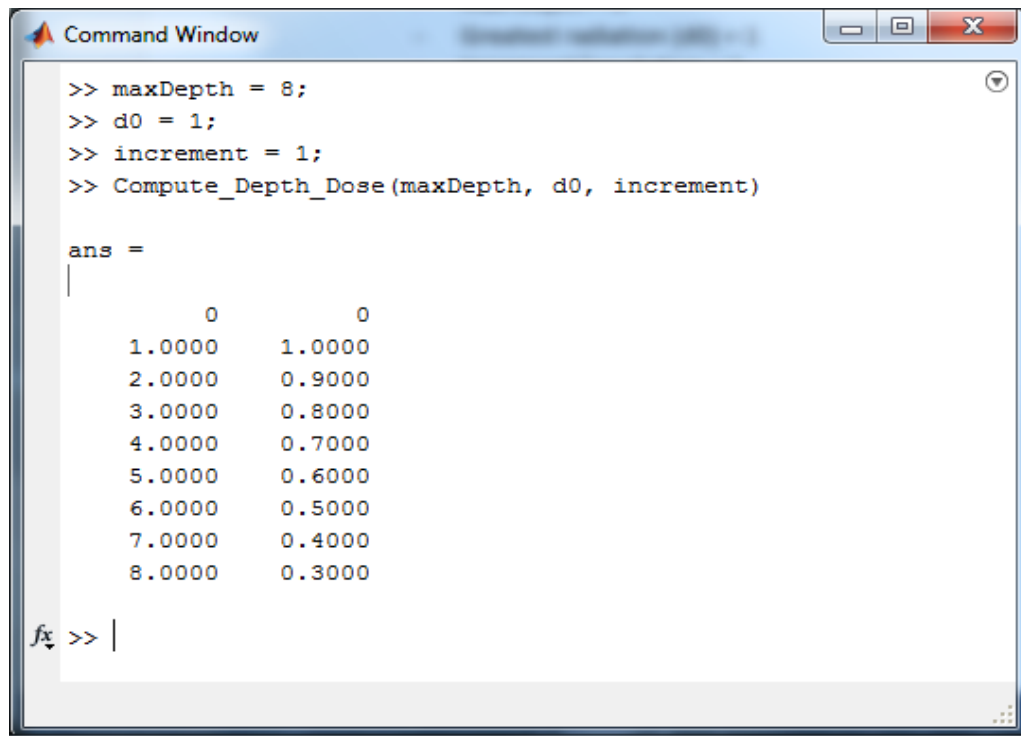
Output:

- Depth dose look up table $[(\text{Max depth} / \text{increment}) \times 2]$

Test:

- Replicate the graph below: $\text{maxDist} = 8$; $d_0 = 1$; $\text{increment} = 1$;





The image shows a MATLAB Command Window with the following content:

```
>> maxDepth = 8;  
>> d0 = 1;  
>> increment = 1;  
>> Compute_Depth_Dose(maxDepth, d0, increment)  
  
ans =  
      0      0  
1.0000  1.0000  
2.0000  0.9000  
3.0000  0.8000  
4.0000  0.7000  
5.0000  0.6000  
6.0000  0.5000  
7.0000  0.4000  
8.0000  0.3000  
  
fx >> |
```

	0	0
1.0000	1.0000	
2.0000	0.9000	
3.0000	0.8000	
4.0000	0.7000	
5.0000	0.6000	
6.0000	0.5000	
7.0000	0.4000	
8.0000	0.3000	

- Data points match the graph.

6. Compute Radial Dose

To create this function I analyzed the graph given in the assignment (shown below). The two heavy blue lines represent the beam radius in either direction from the center line. As we can see, any point within half the radius distance from the middle is getting the highest intensity radiation. Any point within $(-\text{radius}/2 \text{ to } \text{radius}/2)$ will have radiation of 1. Any point from $(\text{radius}/2 \text{ to } 1.5 * \text{radius})$ on either side will have a y value determined using linear function between $[\text{radius}/2, 1]$ and $[\text{radius} * 1.5, 0]$. Any point beyond that range will be 0.

Inputs:

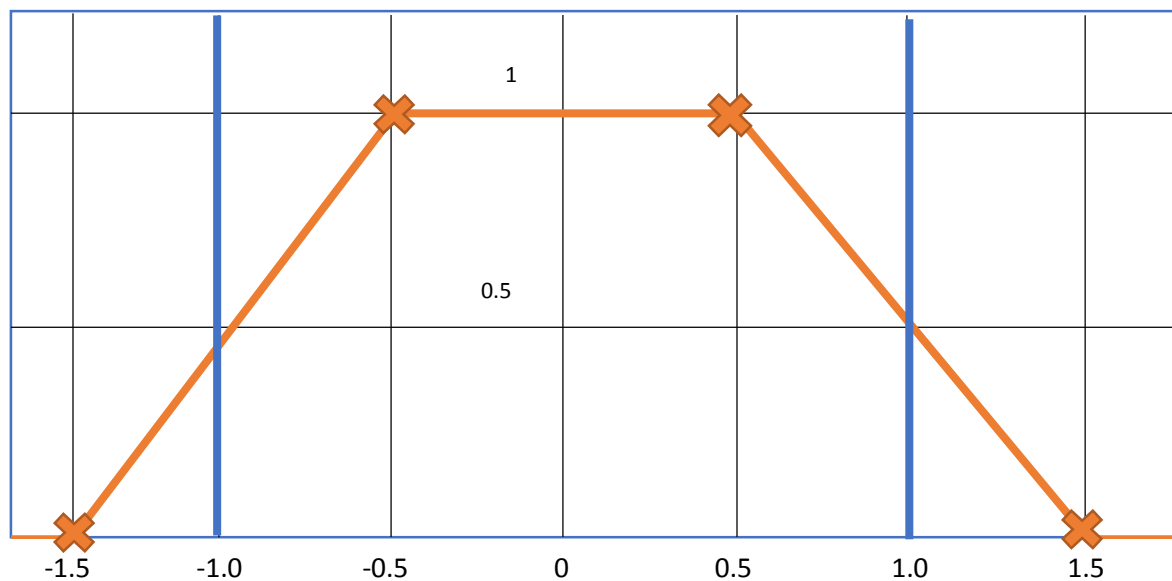
- Max distance from beam center to point in table
- Beam radius of radiation
- Increment (radial distance resolution)

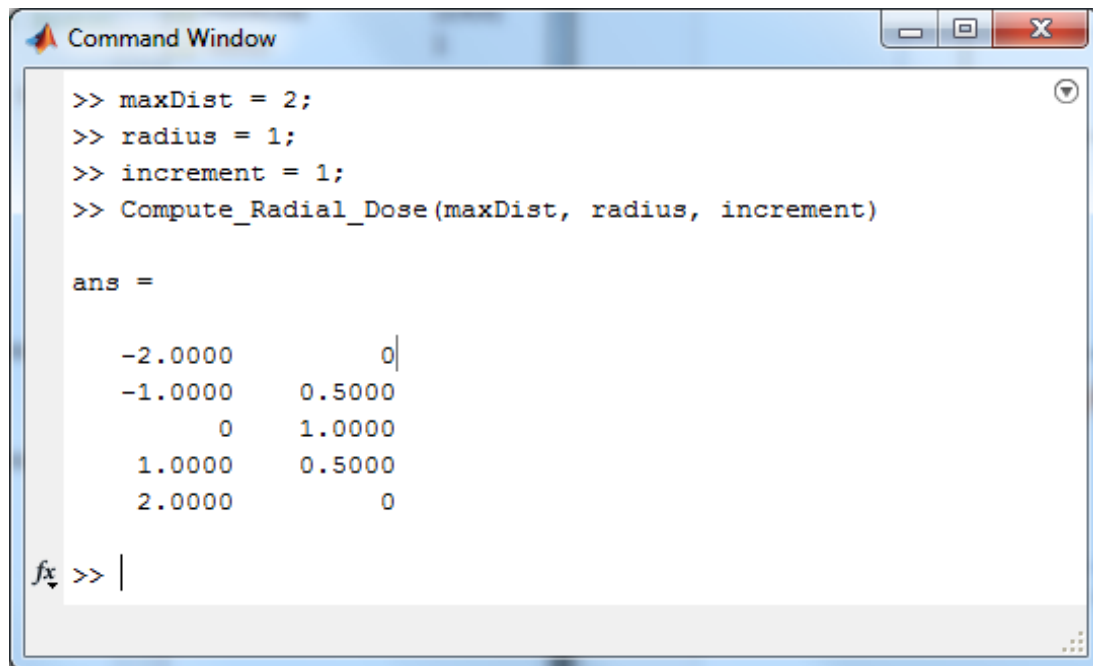
Output:

- Radial dose look up table $[(\text{Max distance} * \text{Increment} * 2) \times 2]$

Tests

- Replicate the table shown below: MaxDist = 3; radius = 1; increment = 1





The image shows a MATLAB Command Window with the following text:

```
>> maxDist = 2;  
>> radius = 1;  
>> increment = 1;  
>> Compute_Radial_Dose(maxDist, radius, increment)
```

The output is:

```
ans =  
  
    -2.0000         0  
    -1.0000    0.5000  
         0    1.0000  
     1.0000    0.5000  
     2.0000         0
```

At the bottom, there is a prompt `fx >> |`.

- Data points match the graph.

7. Compute Beam Direction Vector

This function creates a direction vector for a beam given its source longitude and latitude on the Gamma Knife. We define the points $[x,y,z]$ treating longitude and latitude as azimuth and elevation, respectively, for creating points on a sphere. Latitude is the angle between the “equator” plane of a metaphorical sphere and a line and longitude as the angle between the perpendicular vertical plane and a line.

We define this $[x,y,z]$ coordinate as the first point and $[0,0,0]$ as the second point (the isocenter) in tumor frame. We then simply take $\text{dirVec} = \text{Point2} - \text{Point1}$ and normalize to generate the direction vector.

Inputs:

- Beam source longitude on Gamma Knife (degrees)
- Beam source latitude on Gamma Knife (degrees)

Output:

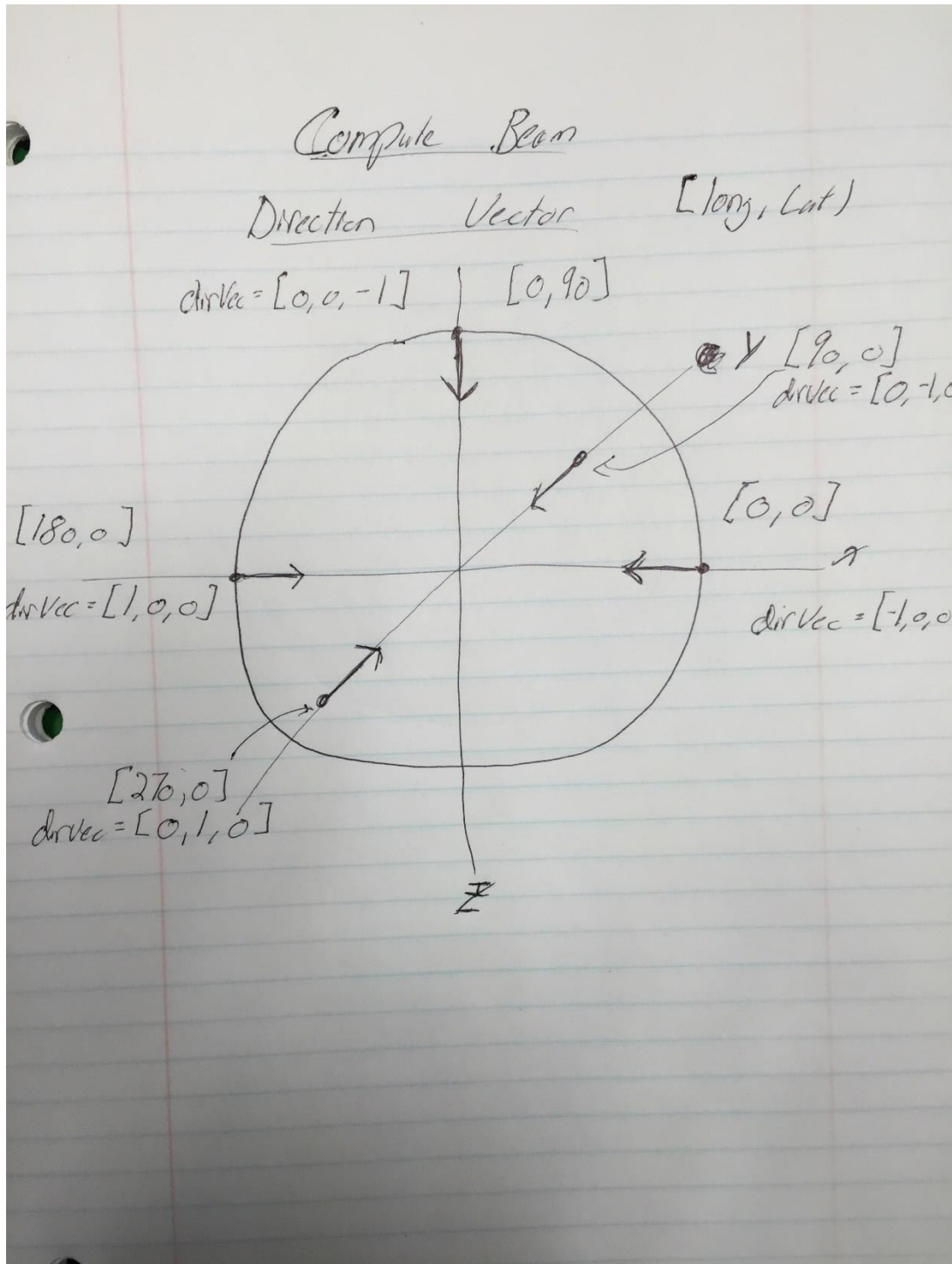
- Normalized beam direction vector $[x, y, z]$

Test:

- Ran in HW4test function
- Longitudes = {0, 90, 180, 270}
- Latitudes = {0, 90}

```
>> HW4test
DirectionVectors =
    0    0    0    0  90.0000  90.0000  90.0000
    0 -1.0000    0    0 -0.0000    0 -1.0000
  90.0000 -0.0000 -1.0000    0 -0.0000 -0.0000 -1.0000
 180.0000  1.0000 -0.0000    0  0.0000 -0.0000 -1.0000
 270.0000  0.0000  1.0000    0  0.0000  0.0000 -1.0000
```

- Longitudes are defined in the first column
- Latitudes are defined in the first row



- Hand calculated / drawn direction vectors:
 - Used a unit sphere (radius = 1)
 - All points at Latitude = 90 will be the same
 - Used Direction Vector = $[0, 0, 0] - [x, y, z]$

8. Compute Skin Entry Point

To compute the skin entry point I computed the direction vector of the beam using the given longitude and latitude. Then I found the point position using equation $\text{DirectionVector} = \text{Point2} - \text{Point1}$

$\text{Point1} = \text{Point2} - \text{DirectionVector}$ where Point2 = the isocenter. Then, I found the intersection points of the line defined by the two points and the ellipsoid (head). I then used conditional statements to find which intersection point was the entry point.

Inputs:

- Head [a, b, c, x, y, z]
- Beam longitude (degree)
- Beam latitude (degree)
- Isocenter point [x, y, z]

Output:

- Skin entry point [x, y, z]

9. Compute Skin Entry Point Table

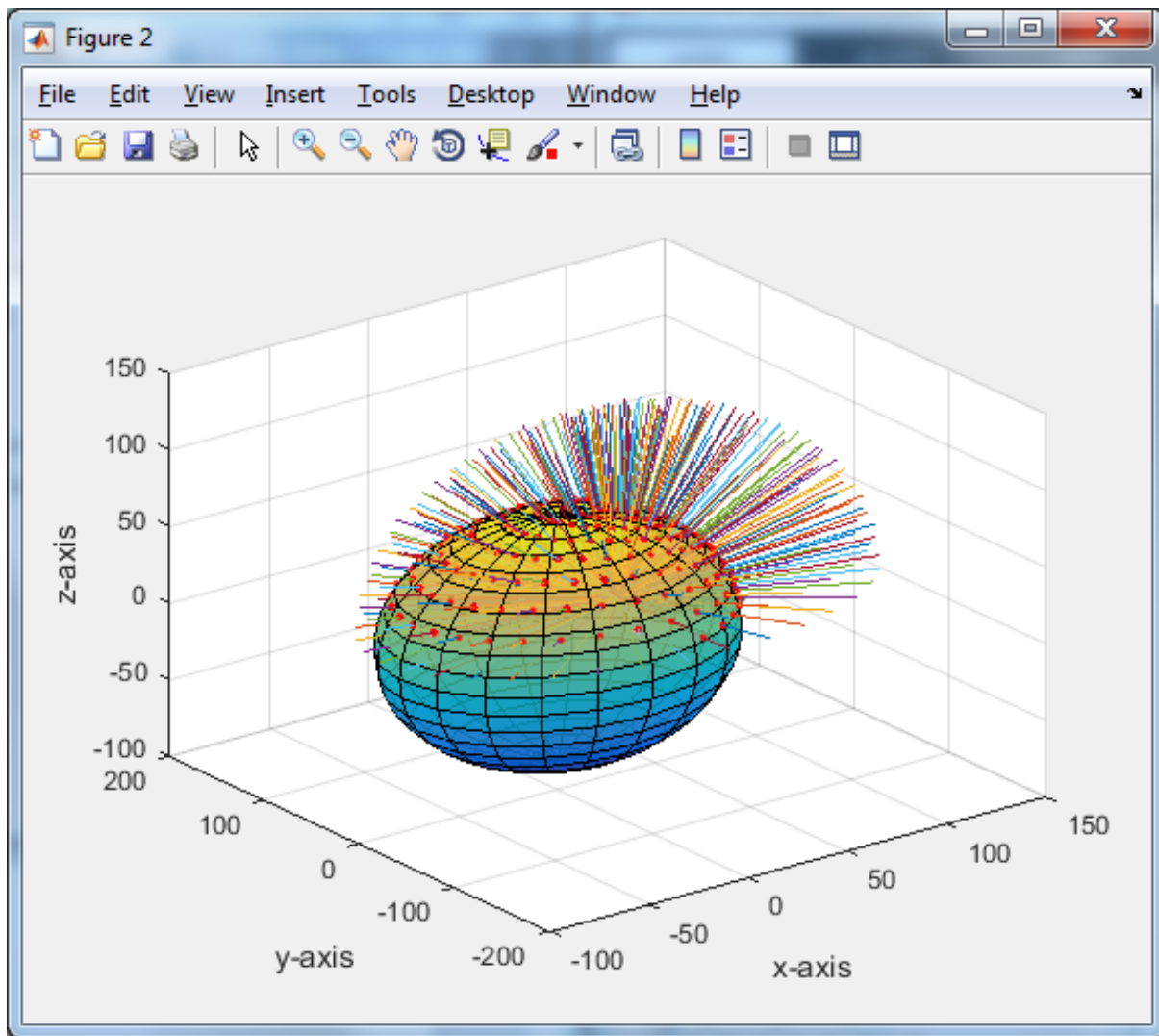
To compute skin entry table, I calculated all the different skin entry points for given helmet difference is source angles. I then filled these values into a matrix based on the size of the inputs.

Inputs:

- Head $[a, b, c, x, y, z]$
- Helmet $[\text{separationAngles}, \text{beamRadius}]$
- Isocenter $[x, y, z]$

Output:

- Skin Entry Point look up table
- 3D scene



- Lines are going through the entry points.

10. Compute Beam Safety

To compute the beam safety, I computed the direction vector of the source using `Compute_Beam_Direction_Vector`, then found the distance between the isocenter point and given beam from the angles using `Compute_Radial_Distance` function.

If the distance is less than or equal to the beam radius + the OAR radius then it would be in contact with the organ at risk, there for `safeCheck = 0`.

If the distance is greater, it is safe; `safeCheck = 1`.

Inputs:

- `Helmet[angles,beamRadius]`
- Longitude (degrees)
- Latitude (degrees)
- Isocenter `[x,y,z]`
- OAR `[radius,x,y,z]`

Output:

- 1 if the beam is safe (does not contact OAR)
- 0 if the beam is unsafe (contacts OAR)

11. Compute Beam Safety Table

I computed the beam safety look up table by running through all the different latitudes and longitudes defined by helmet separation of angles.

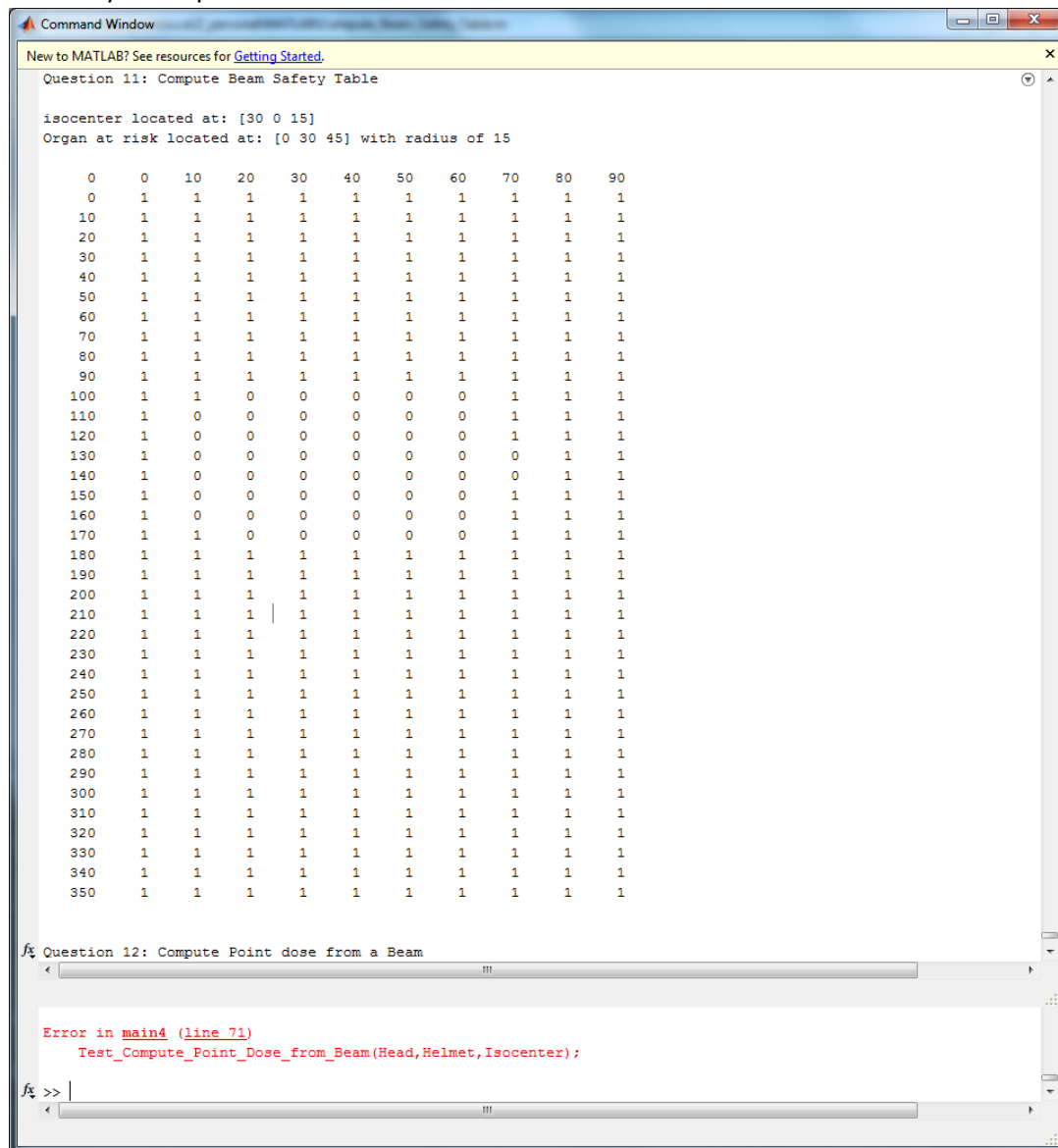
I filed these values into a matrix with the longitudes down the first column and the latitudes across the first row.

Inputs:

- Helmet [separationAngles, radius]
- Isocenter [x,y,z]
- OAR [radius,x,y,z]

Output:

- Beam safety look up table.



Command Window

New to MATLAB? See resources for [Getting Started](#).

Question 11: Compute Beam Safety Table

```
isocenter located at: [30 0 15]
Organ at risk located at: [0 30 45] with radius of 15
```

	0	10	20	30	40	50	60	70	80	90
0	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1	1	1	1
50	1	1	1	1	1	1	1	1	1	1
60	1	1	1	1	1	1	1	1	1	1
70	1	1	1	1	1	1	1	1	1	1
80	1	1	1	1	1	1	1	1	1	1
90	1	1	1	1	1	1	1	1	1	1
100	1	1	0	0	0	0	0	0	1	1
110	1	0	0	0	0	0	0	0	1	1
120	1	0	0	0	0	0	0	0	1	1
130	1	0	0	0	0	0	0	0	0	1
140	1	0	0	0	0	0	0	0	0	1
150	1	0	0	0	0	0	0	0	1	1
160	1	0	0	0	0	0	0	0	1	1
170	1	1	0	0	0	0	0	0	1	1
180	1	1	1	1	1	1	1	1	1	1
190	1	1	1	1	1	1	1	1	1	1
200	1	1	1	1	1	1	1	1	1	1
210	1	1	1	1	1	1	1	1	1	1
220	1	1	1	1	1	1	1	1	1	1
230	1	1	1	1	1	1	1	1	1	1
240	1	1	1	1	1	1	1	1	1	1
250	1	1	1	1	1	1	1	1	1	1
260	1	1	1	1	1	1	1	1	1	1
270	1	1	1	1	1	1	1	1	1	1
280	1	1	1	1	1	1	1	1	1	1
290	1	1	1	1	1	1	1	1	1	1
300	1	1	1	1	1	1	1	1	1	1
310	1	1	1	1	1	1	1	1	1	1
320	1	1	1	1	1	1	1	1	1	1
330	1	1	1	1	1	1	1	1	1	1
340	1	1	1	1	1	1	1	1	1	1
350	1	1	1	1	1	1	1	1	1	1

Question 12: Compute Point dose from a Beam

```
Error in main4 (line 71)
    Test_Compute_Point_Dose_from_Beam(Head,Helmet,Isocenter);
```

>>

12. Compute Point Dose from Beam

To compute the dose from the beam I found the distance the point is from the center of the line using `Compute_Radial_Distance`, then checked the global variable `RadialDoseTable` for the given distance to find its respective radial dose. I also checked the depth of the point, on the line, from the skin entry using a right triangle and pathagoreoms theorm. Here the hypotenuse would be the distance from the point to the skin, `Compute_Skin_Entry_Point`, find distance. The other side of the triangle would be the distance from the beam center. Taking the square root of these values summed squares, gives the distance of the point, projected onto the line, from the skin entry point.

I then ran this depth distance through the global variable `DepthDoseTable`.

The total dose value is the compute radial dose * computed depth dose.

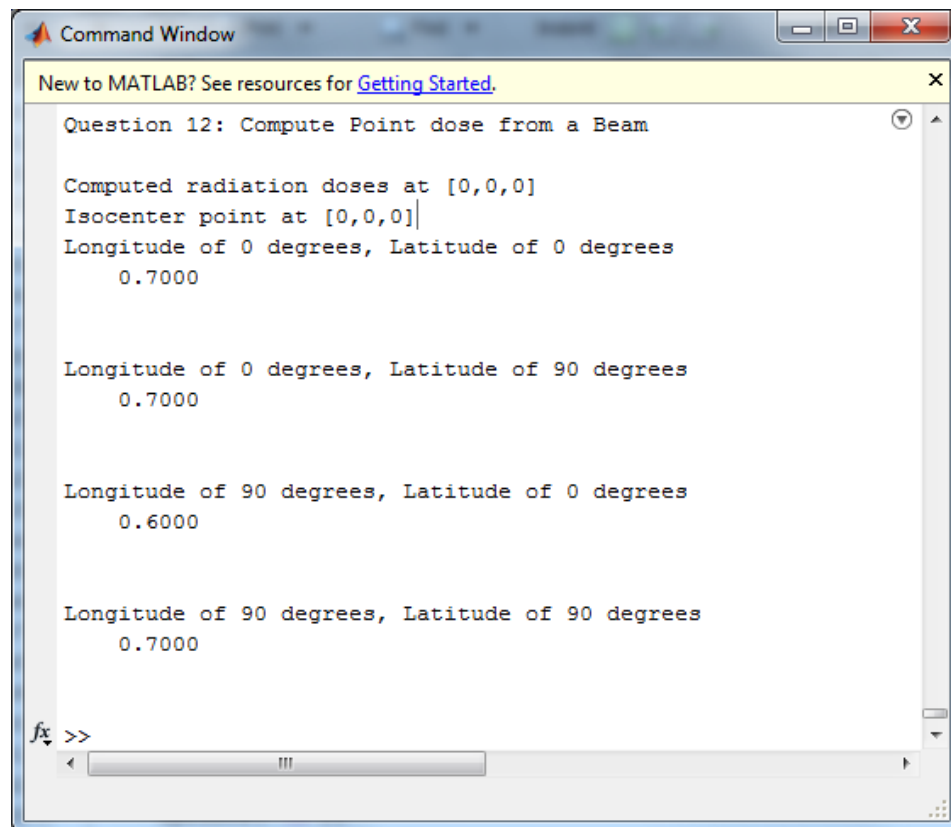
If the point is out of either of the distance, the dose value is 0.

Inputs:

- Head [a,b,c,x,y,z]
- Helmet [angles,radius]
- Longitude (degrees)
- Latitude (degrees)
- Isocenter [x,y,z]
- Point [x,y,z]

Output:

- Radiation Dose Value



```
Command Window
New to MATLAB? See resources for Getting Started.

Question 12: Compute Point dose from a Beam

Computed radiation doses at [0,0,0]
Isocenter point at [0,0,0]
Longitude of 0 degrees, Latitude of 0 degrees
    0.7000

Longitude of 0 degrees, Latitude of 90 degrees
    0.7000

Longitude of 90 degrees, Latitude of 0 degrees
    0.6000

Longitude of 90 degrees, Latitude of 90 degrees
    0.7000

fx >>
```

- Checking these numers in the Depth & Radial dose tables displayed after in `main4.m` shows these values are correct.

13. Compute Point Dose

To compute the point dose at a single point from all the beams, I called the global variable, SafetyTable. I then looked through, and found where ever the table was 1 (safe beam – doesn't contact OAR) and called the Compute_Point_from_Beam at that longitude and latitude (first column & first row). I added these values together in a running radial sum.

Inputs:

- Head [a, b, c, x, y, z]
- Helmet [SeparationAngles, beamRadius]
- OAR [radius, x, y, z]
- Isocenter [x, y, z]
- Point [x, y, z]

Output:

- Point radiation dose value.

14. Compute Dose Volume Histogram

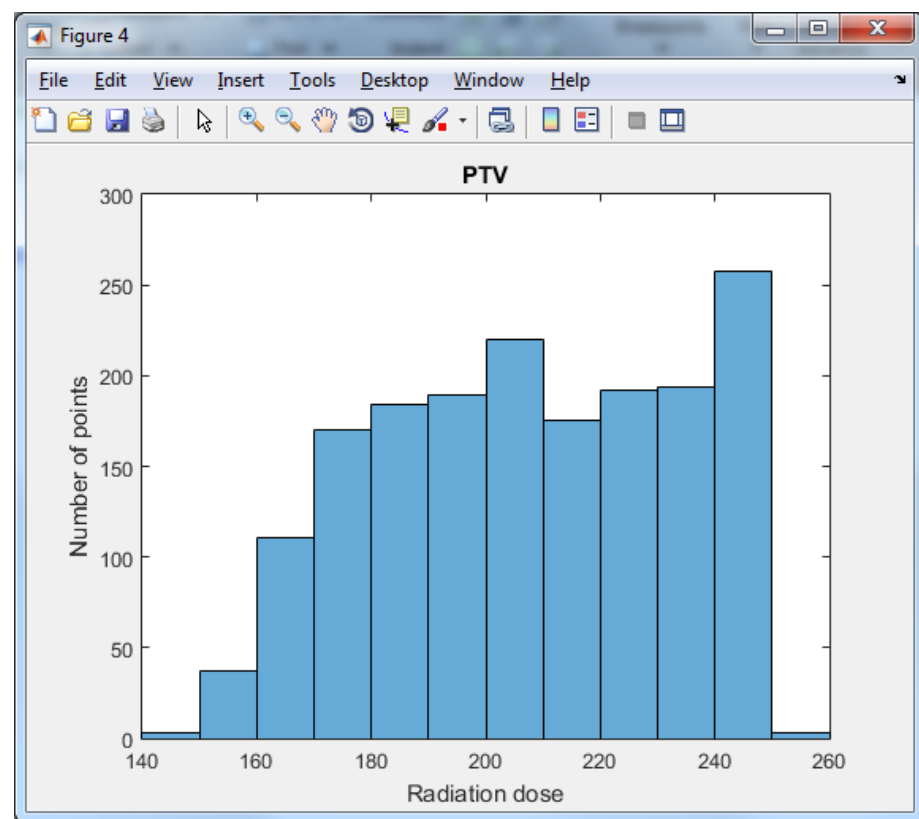
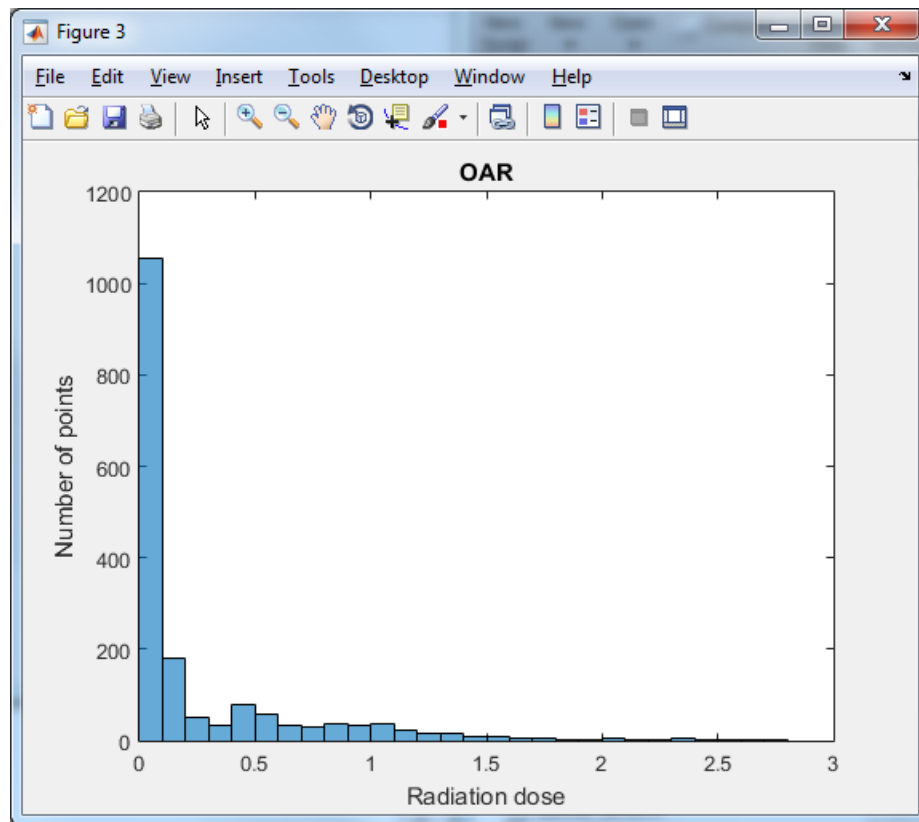
To compute the dose volume histogram for both the PTV and OAR, I called the global variable, doseBox, defined as [the minimum coordinate values : maximum coordinate values] such that the PTV and OAR fit in. I then ran through this doseBox taking every 2 points (voxel size = 2), and called helped functions Is_Point_Inside_OAR(Point,OAR) and Is_Point_Inside_PTV(Point,PTV) to calculate which points within the dose box were inside the spheres.

I then filed these point values into two matrices OARpoints and PTVpoints. Finally, I ran through both of these matrices, computing the point dose at each point using Compute_Point_Dose_from_All_beams to find each points radiation dose. I then filed each into a Histogram chart.

Inputs:

- Head [a, b, c, x, y, z]
- Helmet [SeparationAngle, BeamRadius]
- PTV [radius, x, y, z]
- OAR [radius, x, y, z]
- Isocenter [x, y, z]

Looking at the histograms shown below, it is very obvious that the PVT received significantly more radiation than the OAR (look at x axis values). Also, for the OAR, almost all the radiation is clumped on the left side, so is very minor. Whereas the PVT has a more distributed amount of radiation which is what we were hoping for.



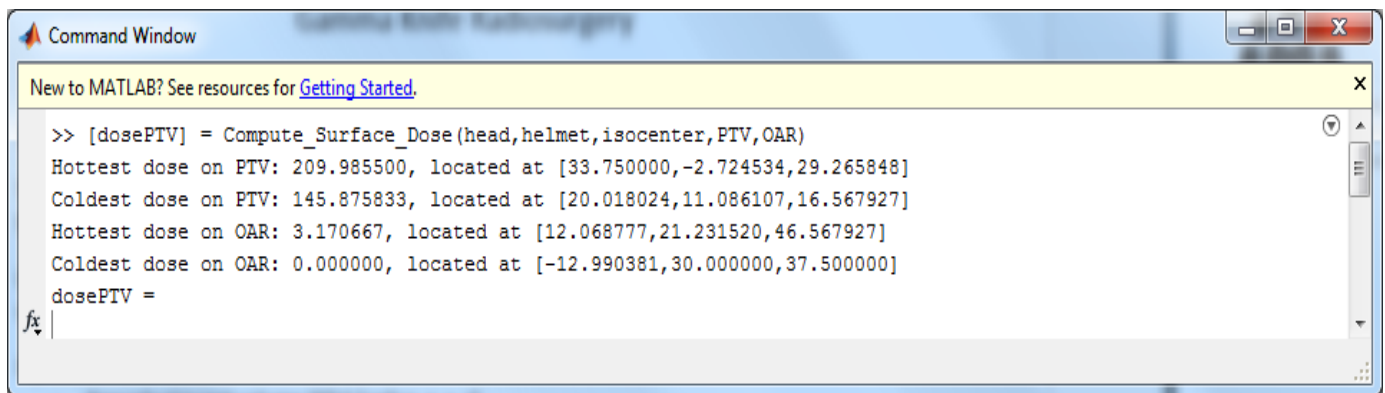
15. Compute Surface Dose

This function computes the dose of radiation on the surface of both the OAR and the PTV. It creates points on the surface of the sphere using the `sphere()` function and adjusting radius & position. It then runs through the points and calculates the maximum dose in PTV OAR and their respective positions. This function will then create a 3D rendition of the spheres with the correct color key.

Inputs:

- Head [a, b, c, x, y, z]
- Helmet[angleSeparation, beamRadius]
- Isocenter [x, y, z]
- PTV [radius, x, y, z]
- OAR [radius, x, y, z]

My graphing code does not work.



```
Command Window
New to MATLAB? See resources for Getting Started.
>> [dosePTV] = Compute_Surface_Dose(head, helmet, isocenter, PTV, OAR)
Hottest dose on PTV: 209.985500, located at [33.750000, -2.724534, 29.265848]
Coldest dose on PTV: 145.875833, located at [20.018024, 11.086107, 16.567927]
Hottest dose on OAR: 3.170667, located at [12.068777, 21.231520, 46.567927]
Coldest dose on OAR: 0.000000, located at [-12.990381, 30.000000, 37.500000]
dosePTV =
```


16. Compute Does Surface Histogram

Function uses number 15 doseValues from points on the surface of the PTV to generate a histogram chart.

Inputs:

- Head [a, b, c, x, y, z]
- Helmet[angleSeparation, beamRadius]
- Isocenter [x, y, z]
- PTV [radius, x, y, z]
- OAR [radius, x, y, z]

