UCD College of Engineering & Architecture

CHEN10040
Introduction to Engineering Computing
2020-21

Assignment #2:

Student Name: Alan O'Donnell
Student No.: 21387526
Student CRNs: 19197
Assigned Laboratory Session: *(Wednesday, 11:00)*
Deadline: 6/12/21
Submission Date: 5/12/21
Module Coordinator: Assoc. Prof. Damian A. Mooney

Contents

Plagiarism Declaration Statement

UCD College of Engineering & Architecture

PLAGIARISM DECLARATION

Programme **Engineering (Stage 1)**

Module Code/Title **CHEN10040 / Introduction to Engineering Computing**

Assessment 2 **No./Title**
Wednesday

Date Submitted
5/12/2021

I certify that ALL of the following are true:

1. I have read and fully understand the consequences of plagiarism as discussed in the *College of Engineering & Architecture Plagiarism Protocol* and the *UCD Plagiarism Policy*. These documents were available to me to properly consider.
2. I fully understand the definition of plagiarism.
3. I recognise that plagiarised work (in whole or in part) may be subject to penalties, as outlined in the *College of Engineering & Architecture Plagiarism Protocol* and the *UCD Plagiarism Policy*
4. I have not previously submitted this work, or any version of it, for assessment in any other subject in this, or any other, institution.
5. All of the information I have provided on this Declaration is, to the best of my knowledge, correct.
6. I have not plagiarised any part of this work; it is original and my own.

Student Name: Alan O'Donnell

Student No.: 21387526

Signed: Alan O'Donnell

Date: 1/12/2021

Problem Objective/Description

I have divided the assignment into 5 parts, to make it easier to complete.

Part 1)

1.a) Generate random values, evaluate if they're inside or outside the circle.
1.b) Use these percentages to calculate the approximate area of the circle.
1.c) Calculate the percentage error using the correct formula for the area of the circle.

Part 2)

2.a) Extend the code to integrate any single variable between user inputted limits.
2.b) Take in the function from the user.
2.c) Plot the function
2.d) Calculate the area under the curve
2.e) Write the data into a txt file.

Part 3)

3.a) Calculate how many circles of radius r can fit inside a square of length L
3.b) Calculate how much area is left over.
3.c) Calculate the voidage.
3.d) Display the information.

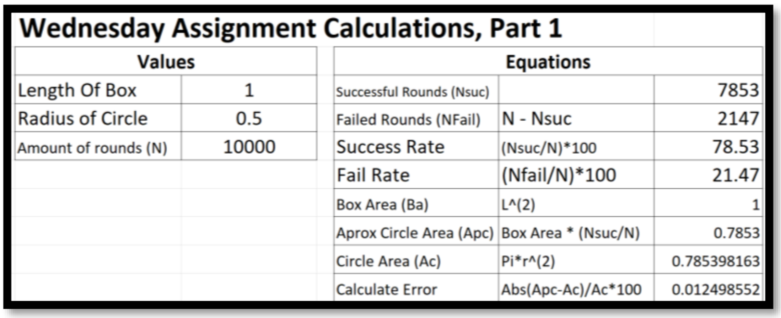
Part 4)

4.a) Calculate how many spheres of radius r can fit inside a cube of length L.
4.b) Calculate how much volume is left over.
4.c) Display the findings to the user.
4.d) Calculate the voidage.

Part 5)

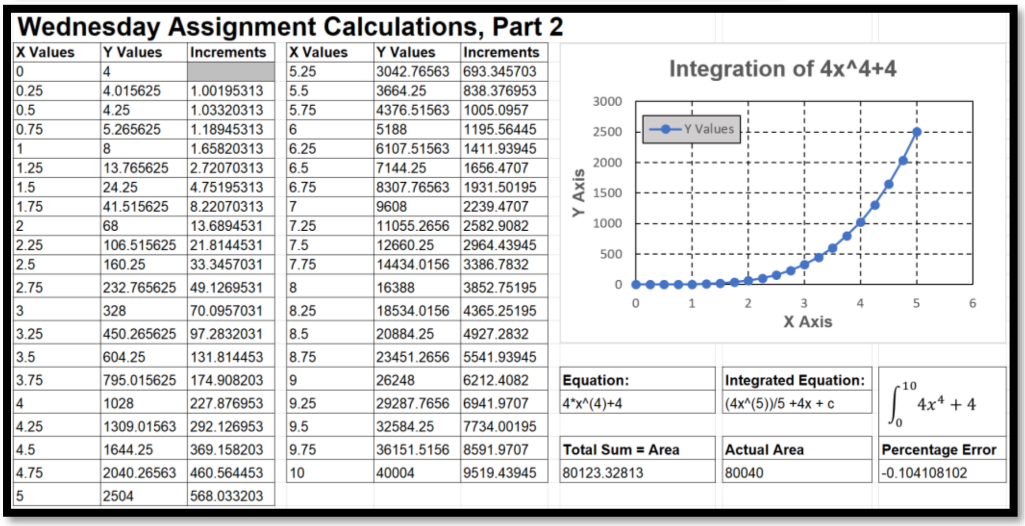
5.a) Calculate the numbers of spheres of radius r that can be randomly generated in a cube of length L.
5.b) Calculate the volume taken up by spheres.
5.c) Subtract the volume from the total volume of the cube to calculate the voidage.
5.d) Compare different factors to increase the number of spheres, volume that can fit into the cube.
5.e) Visually display all this information to the user.

Excel Prep Work & Calculations



Wednesday Assignment Calculations, Part 1

Values		Equations		
Length Of Box	1	Successful Rounds (Nsuc)		7853
Radius of Circle	0.5	Failed Rounds (NFail)	N - Nsuc	2147
Amount of rounds (N)	10000	Success Rate	(Nsuc/N)*100	78.53
		Fail Rate	(Nfail/N)*100	21.47
		Box Area (Ba)	L^(2)	1
		Aprox Circle Area (Apc)	Box Area * (Nsuc/N)	0.7853
		Circle Area (Ac)	Pi*r^(2)	0.785398163
		Calculate Error	Abs(Apc-Ac)/Ac*100	0.012498552

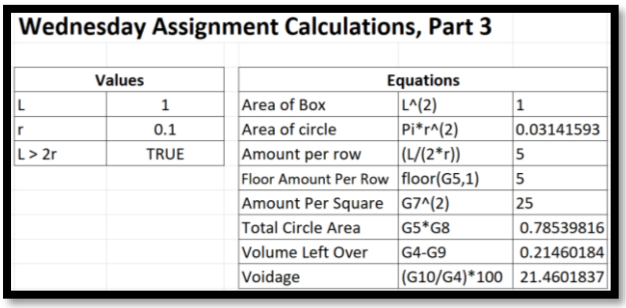


Wednesday Assignment Calculations, Part 2

X Values	Y Values	Increments	X Values	Y Values	Increments
0	4		5.25	3042.76563	693.345703
0.25	4.015625	1.00195313	5.5	3664.25	838.376953
0.5	4.25	1.03320313	5.75	4376.51563	1005.0957
0.75	5.265625	1.18945313	6	5188	1195.56445
1	8	1.65820313	6.25	6107.51563	1411.93945
1.25	13.765625	2.72070313	6.5	7144.25	1656.4707
1.5	24.25	4.75195313	6.75	8307.76563	1931.50195
1.75	41.515625	8.22070313	7	9608	2239.4707
2	68	13.6894531	7.25	11055.2656	2582.9082
2.25	106.515625	21.8144531	7.5	12660.25	2964.43945
2.5	160.25	33.3457031	7.75	14434.0156	3386.7832
2.75	232.765625	49.1269531	8	16388	3852.75195
3	328	70.0957031	8.25	18534.0156	4365.25195
3.25	450.265625	97.2832031	8.5	20884.25	4927.2832
3.5	604.25	131.814453	8.75	23451.2656	5541.93945
3.75	795.015625	174.908203	9	26248	6212.4082
4	1028	227.876953	9.25	29287.7656	6941.9707
4.25	1309.01563	292.126953	9.5	32584.25	7734.00195
4.5	1644.25	369.158203	9.75	36151.5156	8591.9707
4.75	2040.26563	460.564453	10	40004	9519.43945
5	2504	568.033203			

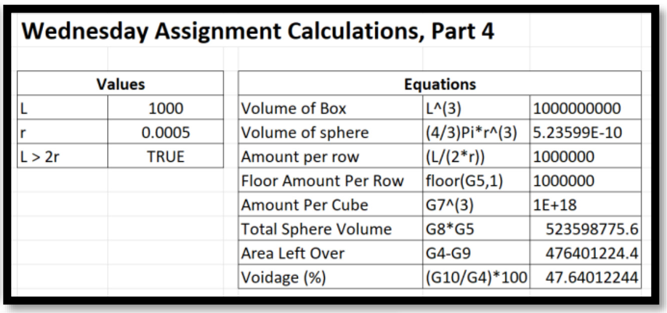
Integration of 4x^4+4

Equation:	Integrated Equation:	
4*x^(4)+4	(4x^(5))/5 +4x + c	$\int_0^{10} 4x^4+4$
Total Sum = Area	**Actual Area**	**Percentage Error**
80123.32813	80040	-0.104108102



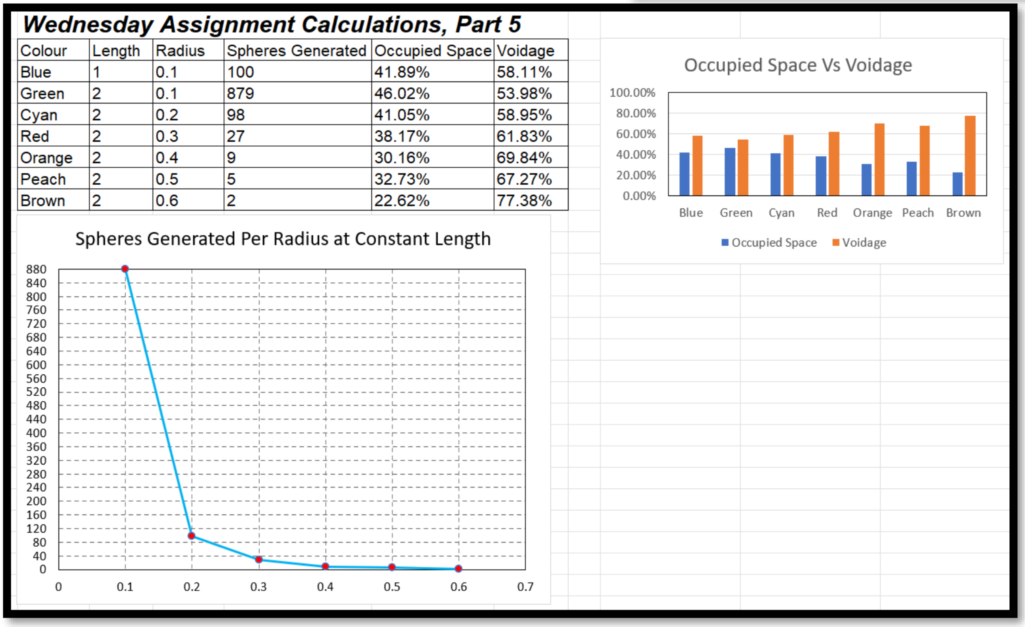
Wednesday Assignment Calculations, Part 3

Values		Equations		
L	1	Area of Box	L^(2)	1
r	0.1	Area of circle	Pi*r^(2)	0.03141593
L > 2r	TRUE	Amount per row	(L/(2*r))	5
		Floor Amount Per Row	floor(G5,1)	5
		Amount Per Square	G7^(2)	25
		Total Circle Area	G5*G8	0.78539816
		Volume Left Over	G4-G9	0.21460184
		Voidage	(G10/G4)*100	21.4601837



Wednesday Assignment Calculations, Part 4

Values		Equations		
L	1000	Volume of Box	L^(3)	1000000000
r	0.0005	Volume of sphere	(4/3)Pi*r^(3)	5.23599E-10
L > 2r	TRUE	Amount per row	(L/(2*r))	1000000
		Floor Amount Per Row	floor(G5,1)	1000000
		Amount Per Cube	G7^(3)	1E+18
		Total Sphere Volume	G8*G5	523598775.6
		Area Left Over	G4-G9	476401224.4
		Voidage (%)	(G10/G4)*100	47.64012244



Wednesday Assignment Calculations, Part 5

Colour	Length	Radius	Spheres Generated	Occupied Space	Voidage
Blue	1	0.1	100	41.89%	58.11%
Green	2	0.1	879	46.02%	53.98%
Cyan	2	0.2	98	41.05%	58.95%
Red	2	0.3	27	38.17%	61.83%
Orange	2	0.4	9	30.16%	69.84%
Peach	2	0.5	5	32.73%	67.27%
Brown	2	0.6	2	22.62%	77.38%

Occupied Space Vs Voidage

Spheres Generated Per Radius at Constant Length

Code Description – Part 1

Part 1 of this paper outlines the code generated in this project, under 8 sections (listed below). The full detail of the code is set out in appendix 1, located on page 15

In Section 1.1:

- I Initialized all my variables that I was going to be using further down my code. I did this by either setting their initial values to 0, or a pre set variable required for initial calculations / restrictions.

In Section 1.2:

- Using a while loop, took in a value for the number of iterations from the user which determines how many times the code will be ran.
- I ensured that this value was a positive whole number, if a negative number was inputted it would be considered invalid, any decimal number was also rounded up to the closest integer. The results were then displayed to the user.

In Section 1.3:

- Using the rand function, I generated a pair of random x and y values between the range of -0.5 and 0.5 for every iteration of the code.

In Section 1.4:

- Using & logic I checked that the generated values were within the range -0.5 to 0.5. I then made use of the sum function to check that both checks were true (i.e., =1), causing the sum to equal to 2

In Section 1.5:

- I Then configured the Y range using distance formulas to find the top & bottom Y Values.
- Next, I Checked that the generated Y value was within the range of Y Values. Depending on the results the Total Success or Total Fail was increased.

In Section 1.6:

- I Then displayed the amount success & failures, converted them into success and fail rate percentages, displayed them.

In Section 1.7:

- Next, I calculated the area of the box and the circle using the given formula, I then calculated the real area of the circle and calculated a percentage error. Then displayed the value to the user.

In Section 1.8:

- Finally, I configured the Pie chart to display all the information to the user. I then customized the pie chart using various functions such as legend, location, renaming, colormap and titles.

Code Description – Part 2

Part 2 of this paper outlines the code generated in this project, under 17 sections (listed below). The full detail of the code is set out in appendix 2, located on page 17

In Section 2.1:

- In section I initialized all my variables that I was going to be using throughout the simulation, I also cleared all existing variables with clear and cleared the command window with clc.

In Section 2.2:

- In this section utilizing a while loop to ensure the inputted number was greater than 1 i.e positive I used the function input, to let the user input the number of iterations the code was going to run. I then rounded this number to ensure it was a positive whole number.

In Section 2.3:

- Next, I had the user Input the lower and higher limits of the interval.

In Section 2.4:

- I then had the user input a single variable function to be integrated. I then converted the string value into a function handle using the function str2func. I also asked the user to confirm that they inputted the correct function to be integrated.

In Section 2.5:

- Next, I calculated the derivative of the function & then solved for the roots. This then allowed me to find the max & min values.

In Section 2.6:

- Using the height of the box as a for loop index, I Set the Max-Min Equation equal to the function with the roots subbed in. Using this I then calculated the Max-Min Values of the function.

In Section 2.7:

- Next, I plotted the X limits P1, P2 on the X Axis as a green line, I then plotted the user defined function ‘Fc’ as a magenta curve. I then customized the graph with titles and legends.

In Section 2.8:

- I then calculated the width of the bow using the range of the X limits, the bow length using the range of Y values.
- I then adjusted these calculations if both the Y Max & Y Min are above the X Axis.
- If the min Y value is greater than the max value, this means the box length is equal to the Y min.
- If the max Y value is greater, then the bow length is equal to the Y max value.

In Section 2.9:

- Continuing with these calculations, I checked if both Y Max & Y Min values were below the X Axis.
- If this is true, using an if statements to check which value was greater, oppositely to the values above the X Axis.

In Section 2.10:

- Using the user defined number of iterations, I then generated random X and Y values using the rand function. Dependent on which was greater the Y max or Y min, different values were used to generate the numbers. (Calculation for when both above the X axis.)

In Section 2.11:

- I then checked if both Y values were negative, reversing the calculations above for both positive numbers.

In Section 2.12:

- I then checked to see if the Y values had opposite signs, Setting the Y values & their limits.

In Section 2.13:

- I then started plotting points in blue if the y value were less than 0, adding to the successful rounds I made use of the feature hold on, to not override the data.

In Section 2.14:

- Next oppositely to the above section, I then plotted the Y values that were greater than 0, added to the successful round's variable.

In Section 2.15:

- I calculated the area of the box and circle (given formula)

In Section 2.16:

- I Then customized the figure with axis labels, grids, and legend titles.

In Section 2.17:

- Finally, I printed off the information to the user and wrote all the data into a txt file.

Code Description – Part 3

Part 3, consists of 5 sections, laid out in appendix 3, located on page: 19

In Section 3.1:

- First, I initialized my variables, used Input to set the length of the square and radius of the circles (disc).

In Section 3.2:

- I then checked that the length was greater than twice the radius, if it's less, then the circle is too big for the square.
- If the circle didn't fit, I asked the user to re-define the measurements.
- Next, I calculated the area of the box and the circle.

In Section 3.3:

- Next, I calculated how many circles could fit in a row, squared it to find out how many could fit in the entire square.
- I then found the total area of circles by multiplying the area of 1 circle, by the total amount of circles.
- Subtracting this value from the total area I was then able to find the voidage percentage using a custom function VoidFunc (See Appendix (F1) on page 23

In Section 3.4:

- I then displayed the information about number of spheres, voidage percentage to the user.

In Section 3.5:

- Finally, using a custom function Custi, (See Appendix (F3) on page 23), I configured the pie chart to display the voidage vs occupied space, customised it.

Code Description – Part 4

Part 4, consists of 6 sections, laid out in appendix 4, located on page: 20

In Section 4.1:

- In this section I configured my variables, took user input for the dimensions of the cube and spheres.

In Section 4.2:

- I then ensured that the user gave a valid response, i.e., there spheres fit inside the cube,

In Section 4.3:

- I then calculated the volume of the box & spheres.

In Section 4.4:

- Next, I found out how many spheres could fit inside a single row, cubed the result. To make sure no partial spheres were included in this calculation I used the MATLAB floor function to round down any decimal number of spheres.
- Next, I calculated the total area of the spheres, then finding the percentage of empty space left over (Voidage) using a custom function SphereVoidage, (See Appendix (F2) on page 23

In Section 4.5:

- I then displayed this information to the user.

In Section 4.6:

- Finally, using a custom function custi (See Appendix (F3) on page 23 I set up the pie chart to visually display Voidage vs Occupied Space.

Code Description – Part 5

Part 5, consists of 19 sections, laid out in appendix 5, located on page: 21

In Section 5.1:

- First, I initialized my values, by taking in a user value for the number of iterations. I then ensured it was a positive whole number.

In Section 5.2:

- User inputted the L of the cube and r of the sphere, which I then checked that L > 2r (So the sphere fits inside)

In Section 5.3:

- I then initialized the colour value for the script by setting it to 0. I then asked the user to pick a colour between 1-7, ensuring that only a value between these would be accepted.

In Section 5.4:

- Next, I Initialized the X, Y, Z & Origin Values. A sphere will then be generated at [X,Y,Z]

In Section 5.5:

- I then plotted the 3D axis using plot 3, ensuring I disabled the legend auto update to prevent it filling the legend with data points. I then adjusted the graphs view with set gca.

In Section 5.6:

- Using linspace & plot3 I plotted the 12 edges of the cube to house the spheres.

In Section 5.7:

- Next, I generated random points within the box based off the user defined iterations.

In Section 5.8:

- I then generated the first sphere in the bottom right corner, that would be the base for the rest, to generate off.
- I made use of surfl to produce a surface plot.

In Section 5.9:

- I then ran a for statement from 1 to infinity until it was broken which would continue to generate locations for spheres until all the available space was taken up. This was done using distance formulas and comparing them to the diameter of the sphere.

In Section 5.10:

- Next, I updated the X,Y,Z values to the values previously generated ones.

In Section 5.11:

- I then located the closest sphere to the origin, using the min & find functions.

In Section 5.12:

- I then re updated the values based off the previous find search for spheres.
- I then broke the loop when no more spheres are located, using the isempty function.

In Section 5.13:

- I then plotted the spheres using plot3 & surfl to help represent all this information.

In Section 5.14:

- Using colormap, I then changed the colour of the spheres based on the user's colour preference.

In Section 5.15:

- I then tidied up the figure, using a tittle, legend, axis, grid, interp shading & using num2str to display values.

In Section 5.16:

- I then displayed how many spheres were generated, depending on if there was only 1, or multiple spheres, a different message was used. I then told the figure to show the legend.

In Section 5.17:

- Next, I calculated the volume of a sphere, total volume of the spheres, volume of the box, volume difference & the voidage. I then displayed the voidage to the user.

In Section 5.18:

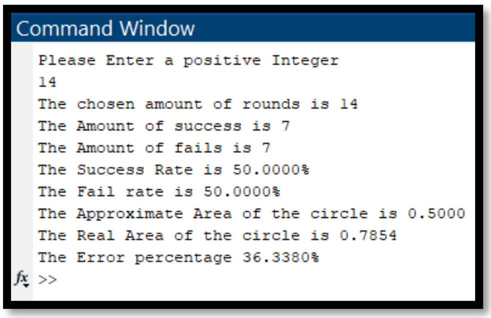
- I then designed a pie chart to display voidage vs occupied space.

In Section 5.19:

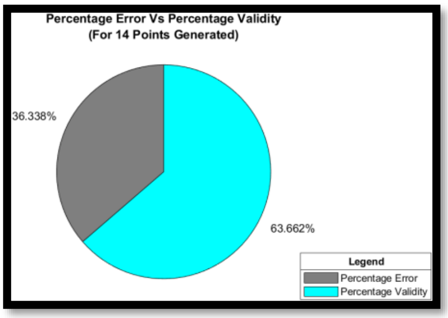
- Finally using RGB values on a 0-1 scale, I colour coded the pie chart to the user colour preference.

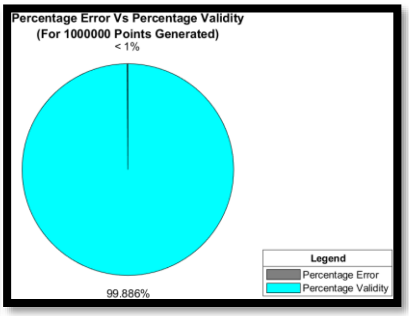
Results – Part 1

a) Upon completing Part 1, I found that, for the more points plotted, a more accurate result can be obtained, this can be seen from the two pie charts below, in the first example 14 points were plotted resulting in a voidage percentage of 36.34%, and in the second example, 1,000,000 points were used, resulting in a voidage percentage of 0.0522%



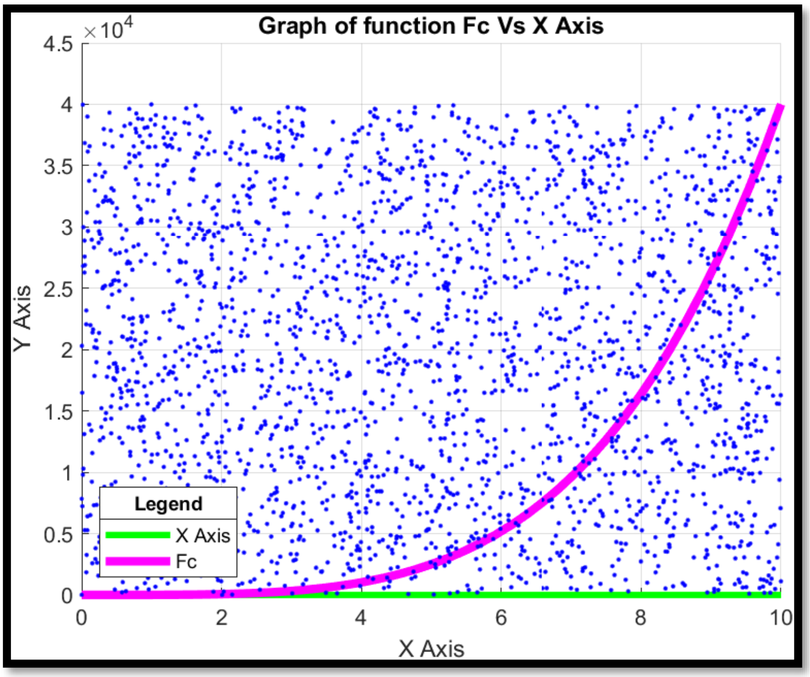
```
Please Enter a positive Integer
14
The chosen amount of rounds is 14
The Amount of success is 7
The Amount of fails is 7
The Success Rate is 50.0000%
The Fail rate is 50.0000%
The Approximate Area of the circle is 0.5000
The Real Area of the circle is 0.7854
The Error percentage 36.3380%
fx >>
```

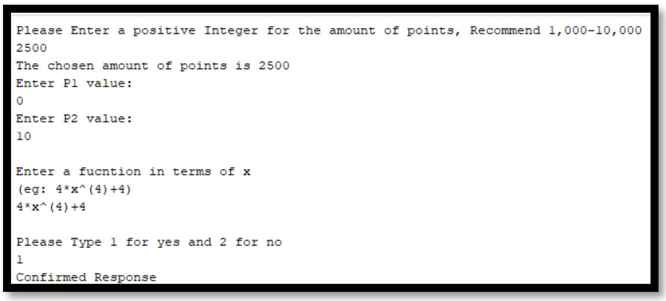




Results - Part 2

a) Using points to calculate the area under the curve, i.e., to integrate the function between the curve Fc (magenta) and the X axis (light green). I was able to achieve a very accurate answer for the area under the curve. I was able to then compare this answer to the value I calculated on excel, I found that it was off by only 608 units^2. Compared to the trapezoidal method which was off by 337.25 units^2. If I was to increase the number of points generated, this error percentage will decrease.

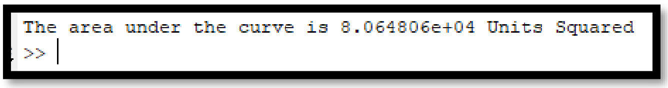




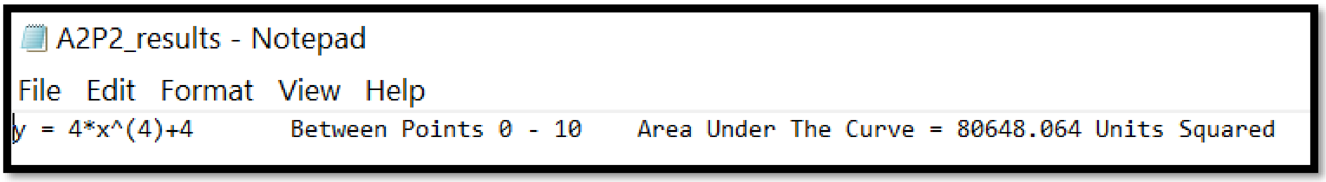
```
Please Enter a positive Integer for the amount of points, Recommend 1,000-10,000
2500
The chosen amount of points is 2500
Enter P1 value:
0
Enter P2 value:
10

Enter a fucntion in terms of x
(eg: 4*x^(4)+4)
4*x^(4)+4

Please Type 1 for yes and 2 for no
1
Confirmed Response
```



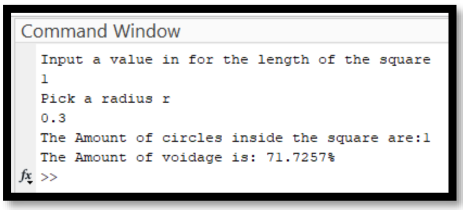
```
The area under the curve is 8.064806e+04 Units Squared
>>
```



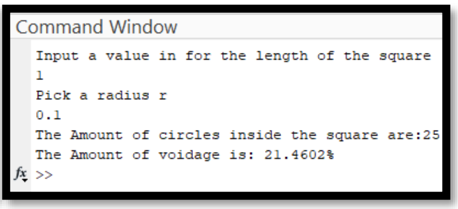
```
y = 4*x^(4)+4      Between Points 0 - 10     Area Under The Curve = 80648.064 Units Squared
```

Results – Part 3

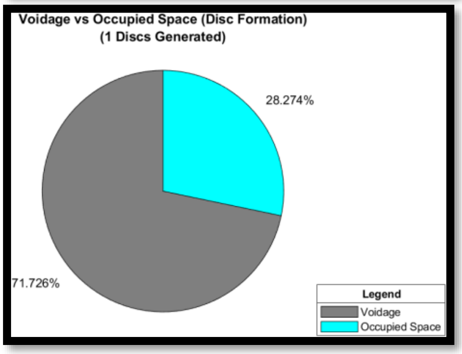
a)

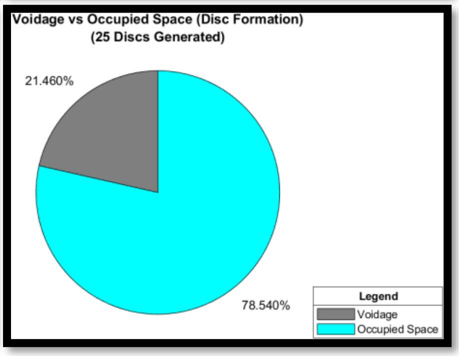


```
Command Window
Input a value in for the length of the square
1
Pick a radius r
0.3
The Amount of circles inside the square are:1
The Amount of voidage is: 71.7257%
fx >>
```



```
Command Window
Input a value in for the length of the square
1
Pick a radius r
0.1
The Amount of circles inside the square are:25
The Amount of voidage is: 21.4602%
fx >>
```

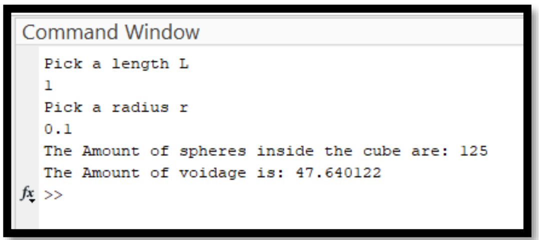




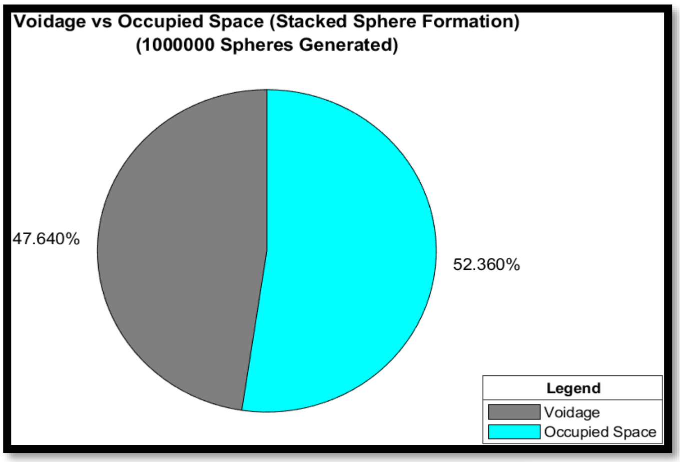
For generating discs inside of a square, the smaller the radius of the discs, the more discs can be fitted in, this can be demonstrated with the two pie charts, the example on the left has a higher radius, therefor less discs can fit inside the square (1), the right pie chart has a smaller radius, allowing it to house more discs.(25)

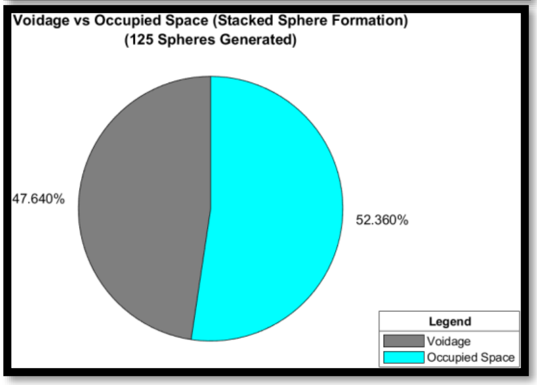
Results – Part 4

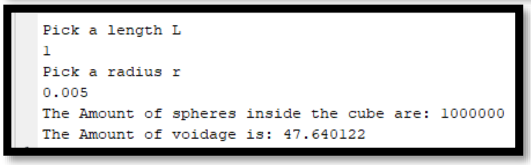
a)



```
Command Window
Pick a length L
1
Pick a radius r
0.1
The Amount of spheres inside the cube are: 125
The Amount of voidage is: 47.640122
fx >>
```





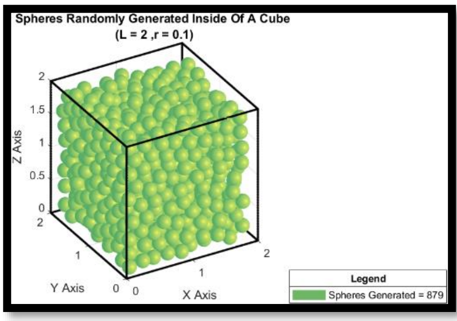


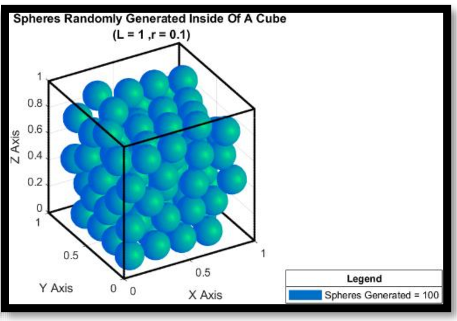
```
Pick a length L
1
Pick a radius r
0.005
The Amount of spheres inside the cube are: 1000000
The Amount of voidage is: 47.640122
```

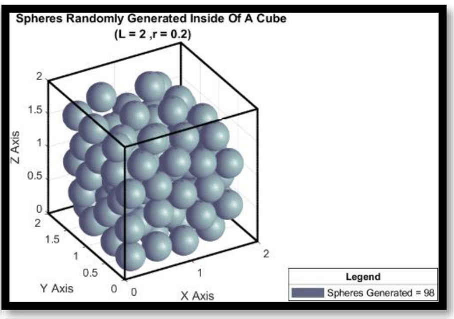
Just like with the discs, when forming the spheres with the stacked square formation, the smaller the radius, results in more spheres being generated, which can be seen from the comparison of radius 0.1 creating 125 spheres and a radius of 0.005 creating 1000000 spheres.

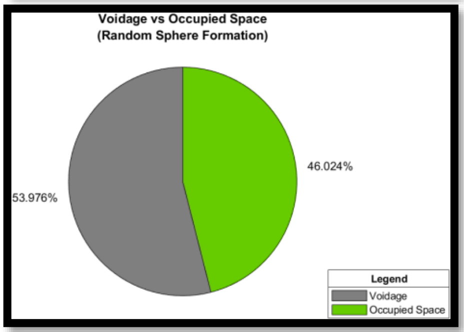
Results – Part 5

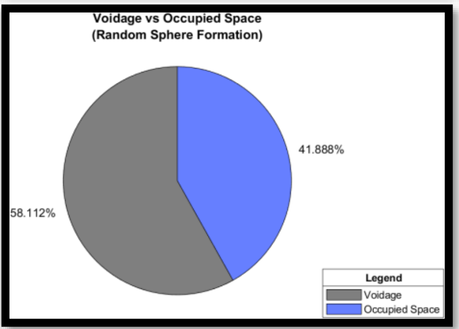
a)

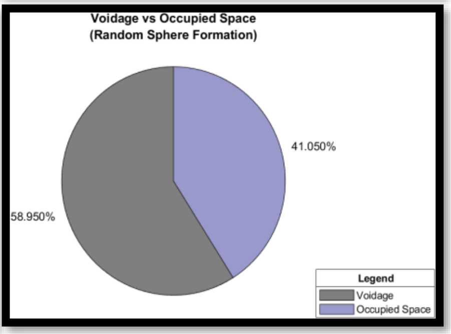


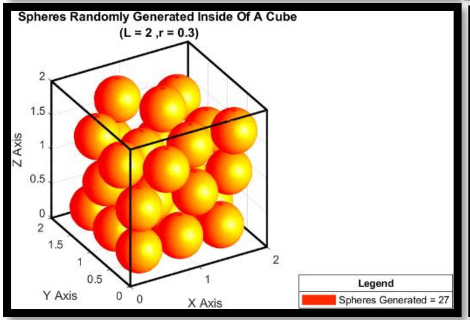


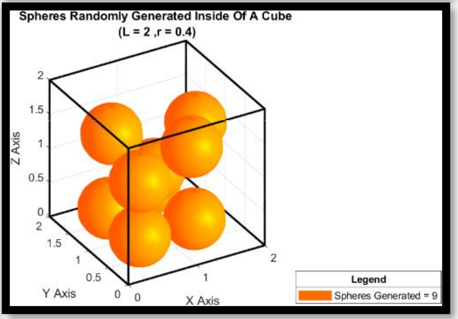


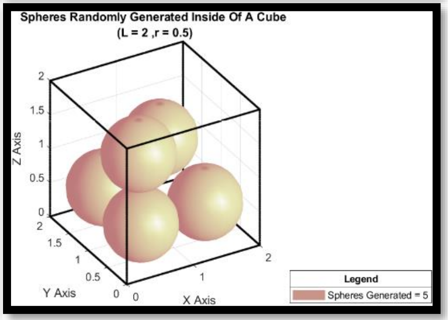


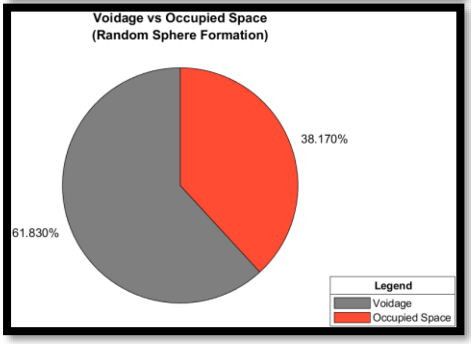


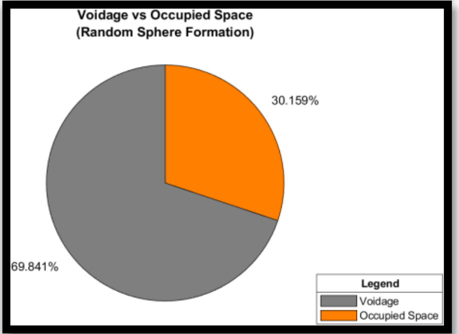


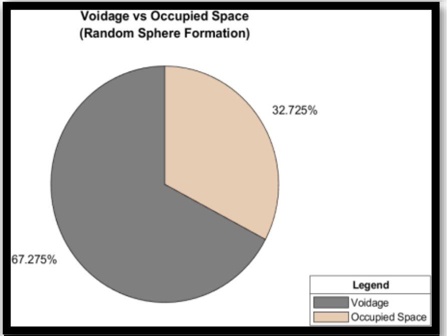


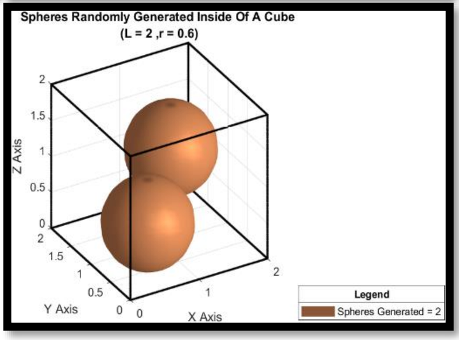


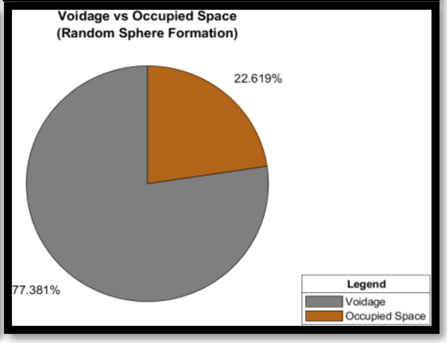












After completing my code, running simulations, I discovered that, when length is kept constant, the smaller the radius, generates more spheres. (Displayed by the red and green graphics 879 vs 27 spheres generated). Vice versa, if radius is constant but length is increased, it will also generate more spheres. This can be seen from comparing the green graph with the cyan graph, both are radius 0.1 but with different lengths.

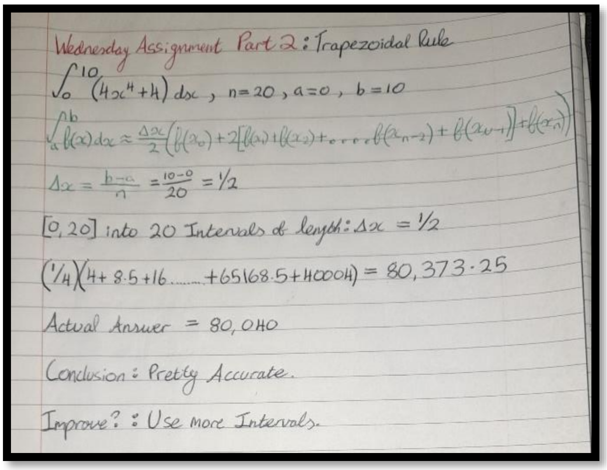
Discussion – MATLAB Concepts

Throughout this assignment I made use of a lot of different MATLAB concepts and functions that include:

- Input → Displays text in the prompt window, allows user to input a value which will then be stored as a variable.
- If → Statement, that if something is true, it will go on and do something, if not true it will do something else.
- While → While an expression is true it will loop code.
- For → Loops a specified number of times using an index e.g. I = 1: N, it will loop from 1 to N
- Rand → Randomly generate numbers.
- Isempty → Returns true if a variable is empty, faster than doing length == 0.
- Break → Terminates a for or while loop.
- Custom functions → Custom functions to perform multiple tasks, can drastically cut down on length of code being re-used.
- Round → Rounds a decimal to the closest whole number. Important for Monte Carlo simulations as you can't run a iteration for a fraction of a round.
- Floor → Rounds any decimal down, important for calculating how many full spheres can fit inside the cube, preventing any partial spheres from fitting inside.
- || → Or logic, comes back true if statement A or B is true, can be used to evaluate if a colour was picked out of a selection.
- && → and logic is used to confirm that multiple conditions were met, for example that a generated value was above limit 1, but below limit 2.
- Plot → Creates a 2D line plot using inputted data, Useful when comparing data to each other, represent it easily.
- Plot3 → 3D plotting for more complicated simulations, such as generating random spheres inside of a cube.
- Surfl → Surface plot, used when generating surfaces for the 3D spheres.
- Set → Sets specified values such as colour , "color".
- Disp → used to simply display a values and variables.
- Fprintf → Used to print more complicated data to text files, more control over how the output is formatted such as scientific notation, controlled decimal places and whole numbers.
- Find → Used to find specific data, output its corresponding values.
- Colormap → Matrix value that defines colours for graphic objects/surfaces, utilized to customise the spheres.
- Zlabel → Apply a Z Axis Label.
- Ylabel → Apply a Y Axis Label.
- Xlabel → Apply a X Axis Label.
- Shading → Colour shading of a surface, technique used, eg interp.
- Title → Applying a title to the top of a figure.
- Legend → useful for labelling the different data in a graph.
- Grid → Applies the major grid lines to the graph.
- Pie → Generates a pie chart using the inputted data.
- Writematrix → Writes a matrix to a file, used when exporting data from part 2, into a txt file.
- Solve → Solves inputted equations, e.g., to find the roots of an equation.
- Length → Calculate the length of a vector, i.e., its size. Acts like a max size function.
- Hold on → Retains the previous plot without over-writing it, important when want to plot multiple pieces of data such as points and spheres.
- Clc → Clear the command window, useful to stop the window becoming stuffed,
- Clear → Clear previous variables, prevents calculations from setting up incorrectly.
- Syms → Generated a symbolic scalar variable, to be used in anonymous functions.
- Abs → calculate the absolute value, used when calculating percentage error.
- Sum → Calculate the sum of all inputted values, used when checking multiple logic checks returned true.
- Sqrt → Calculate the square root of anything in the brackets.

Discussion – Mathematical Concepts

- The Trapezoidal Rule is used as an approximate method for integration. Isn't the most accurate method but gives a good idea.
  $\int_a^b f(x)dx \approx \frac{dx}{2}[f(x_0) + 2\big(f(x_1) + f(x_2) + \cdots .. + f(x_{n-1})\big) + f(x_n)]$



Wednesday Assignment Part 2: Trapezoidal Rule

$\int_0^{10} (4x^4+4)\,dx$, $n=20$, $a=0$, $b=10$

$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2}\Big(f(x_0)+2[f(x_1)+f(x_2)+\dots f(x_{n-2})+f(x_{n-1})]+f(x_n)\Big)$

$\Delta x = \frac{b-a}{n} = \frac{10-0}{20} = 1/2$

[0, 20] into 20 Intervals of length: $\Delta x = 1/2$

$(1/4)(4+8.5+16 \dots\dots +65168.5+40004) = 80{,}373.25$

Actual Answer = 80, 040

Conclusion: Pretty Accurate.

Improve?: Use more Intervals.

- Distance formula is used to get the distance between 2 points.
  $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
- Differentiation is used to obtain the max & min values of a function.
- Integration is used to complete the question.
- Approx. area of the circle can be calculated from the given formula:
  $Area_{Circle} = Area_{Box} * \frac{NSuc}{N}$
- Actual area of a circle can be calculated from the formula:
  $A_{Circle} = \pi r^2$
- Area of a square can be found from the equation:
  $A_{Square} = L^2$
- The Volume of a cube can be found from the equation:
  $V_{Cube} = L^3$
- The volume of a sphere can be found from the equation:
  $V_{Sphere} = \frac{4}{3}\pi r^3$
- The number of circles that can fit inside a square can be found from the formula:
  $[floor\left(\frac{L}{2r}\right)]^2$
- The number of spheres that can fit inside a cube can be found from the formula:
  $[floor\left(\frac{L}{2r}\right)]^3$
- The percentage error in calculating the area can be found from the equation:
  $\left(|(\frac{Ap_c - A_c}{A_{Circle}})|\right) * 100$
- Percentage Voidage for a cube can be found from the formula:
  $\left(\frac{Cube_{\ Volume} - \ Sphere\ Total_{Volume}}{Cube_{Volume}}\right) * 100$

Conclusions

- Throughout the process of completing this assignment, I have developed my problem-solving skills, MATLAB proficiency and graphical representation skills. I'm very happy with the results of this assignment. The end results look very nice, explaining all the necessary information. It is also very satisfying to watch the sphere being randomly generated into the cube.
- As I progressed through these questions, I began forming a routine that I followed to complete them, Which I will continue to hone in on, to develop these skills further.
- I found that the Monte Carlo method for writing code, very interesting and efficient for solving these types of problems, it's a technique I will come back to in the future.
- If I was to attempt this type of question again in the future, Id be curious what other shapes would be more efficient in using the space, such as hexagons or diamonds.

Recommendations

- If I was to start this assignment all over again, I would have gone with the coordinate method that I used to generate the random sphere, for the disc question, as it gave a much nicer result being able to show the objects, generate, where their located, rather than just pie charts displaying voidage.
- To Improve my problem-solving skills, I would continue solving similar types of problems, expanding on my MATLAB skills to become more efficient and effective.
- I recommend reading through the rest of the MATLAB document, to further familiarize myself with the different types of inbuilt functions that MATLAB has to offer.
- As MATLAB is a very important software to gain a proficiency at, I want to continue learning how to use the software to the best of my abilities.
- In the future I would prefer to make greater use out of custom functions to reduce the amount of code that is used to complete the assignment.
- Before completing more questions, I recommend re-looking over previous notes taken to, re-familiarise myself with the materials, formatting.

References

Throughout the process of developing my skills within MATLAB and Excel I made use of various resources I had at my disposal.

- MathWorks (2021) *MathWorks*, available at: https://uk.mathworks.com/help/matlab/ (Accessed 9th November 2021).

- MATLAB tutorial, Author Derek Banas, available at: https://www.youtube.com/watch?v=NSSTkkKRabI&ab_channel=DerekBanas, (Accessed 14th November 2021).

- Webpage Author Emasher, Engineers Toolbox, available at: https://www.engineeringtoolbox.com/circles-within-rectangle-d_1905.htmlH, (Accessed 28th November 2021).

- Webpage Author Wolf Ram, Circle Packing, available at: https://mathworld.wolfram.com/CirclePacking.htmlH, (Accessed 26th November 2021).

- The Beginner's Guide to Excel - Excel Basics Tutorial, Author Technology for Teachers and Students, available at: https://www.youtube.com/watch?v=rwbho0CgEAE&ab_channel=TechnologyforTeachersandStudents, (Accessed 14th November 2021).

Appendix (1)

```
% MATLAB Wednesday Assignment Part 1
%
% By: Alan O'Donnell
% SN:21387526
% Date: 20/11/21
% Version: 1.3
%
% Aim:
%    1) Generate random values, evaluate if they're inside or outside the circle.
%    2) Use these percentages to calculate the approximate area of the circle.
%    3) Then calculate percentage error using the correct formula for area of the circle.


%-< Section 1.1 >-----<<<< Initializing Variables >>>>--------------------------------------------------------------

clc                                                 % Clear Command Window
clear                                               % Clear Variables
n = 0;                                              % For Loop Variants.
N = 0;                                              % Inputted number of iterations of code.
NSuc = 0;                                           % Total Number of successes
NFail = 0;                                          % Total Number of failures.
L = 1;                                              % Length of the box (Square)
r = L/2;                                            % Radius of the circle

%-< Section 1.2 >-----<<<< Take User Input >>>>-------------------------------------------------------------------

while N < 1                                         % Maintain a while function until the constraint is met.
 N = input('Please Enter a positive Integer \n');   % Take in a value for the number of times the code will run.
end                                                 % The While Loop ensures that the inputted number is positive.
N = round(N);                                       % Round this number to the closest whole number,
                                                    % As we can't run the code a fraction of a time.
fprintf('The chosen number of rounds is %d\n',N)    %Print off how many rounds were chosen.

%-< Section 1.3 >-----<<<< Generate Random Numbers in the range -0.5 to 0.5 >>>>----------------------------------

for  n = 1:N                                        %Run the lines of code N amount of times
x = rand() -0.5;                                    %Generate a random number between -0.5 and 0.5
y = rand() -0.5;                                    %Generate a random number between -0.5 and 0.5

%-< Section 1.4 >-----<<<< Testing the Interval >>>>--------------------------------------------------------------

bx = x <= 0.5 & x>= -0.5;                           % Check that the X values are within the range -0.5 to 0.5
by = y <= 0.5 & y>= -0.5;                           % Check that the Y values are within the range -0.5 to 0.5
bt = sum(bx+by);                                    % Calculate the sum of the Boolean checks,
                                                    % If both is true it will return a value of 2

%-< Section 1.5 >-----<<<< Configuring the Y Range >>>>-----------------------------------------------------------

% To ensure that the generated point is inside the circle, it must be within this Y range.

if bt ==2                                          % If Values are set up correctly it will equal 2
                                                   % Testing Purposes: disp('Values are in the correct interval')
    y1 = sqrt(r^(2) - x^(2));                      % Setting up the Top Y Value
    y2 = -sqrt(r^(2) - x^(2));                     % Setting up the Bottom Y Value

   if y >= y2                                      % Check if the Y value is greater or Equal to the 2nd Y value.
       if y <= y1                                  % Then check if the Y value is Less than or equal to the first Y value,
       NSuc = NSuc + 1;                            % This means it will be between the values Y2 and Y1. Then count it as a success.
                                                   % Testing Purposes: disp('The point is inside the circle')

       else
       NFail = NFail + 1;
                                                   % Testing Purposes: disp('The point is not inside the circle')

       end
     else
       NFail = NFail + 1;                          %This Point is not inside the circle
                                                   % Testing Purposes: disp('The point is not inside the circle')
   end
else
    disp('Generated Values are invalid')           %The Generated values are invalid, Display an error message to the user.
end
end

%-< Section 1.6 >-----<<<< Configure Success & Fail Rates >>>>------------------------------------------------------

fprintf('The Number of successes is %d\n',NSuc)                      % Display the number of successful rounds.
fprintf('The Number of fails is %d\n',NFail)                         % Display the number of failed rounds.
SR = round((NSuc / (N)) *100);                                       % Calculate the Success Rate.
FR = round((NFail / (N)) *100);                                      % Calculate the Fail Rate.
fprintf('The Success Rate is %.4f%%\n',SR)                           % Display the Success rate to the user.
fprintf('The Fail rate is %.4f%% \n',FR)                             % Display the Fail rate to the user.


%-< Section 1.7 >-----<<<< Calculate the Area, Approx. Area & Percentage Error of the circle >>>>-----------------

Ab = 1;                                                        % Area of the box = 1^2
ApC = Ab * (NSuc/N);                                           % Formula Given in the Question
fprintf('The Approximate Area of the circle is %.4f\n',ApC)    % Display the approx. area of the circle.

Ac = (pi)*(r)^2;                                               % Calculate the real area of the circle.
fprintf('The Real Area of the circle is %.4f\n',Ac)            % Display the real area of the circle.

Er = (abs(ApC - Ac ))/Ac * 100;                                % Calculate the Percentage error as an absolute value.
fprintf('The Error percentage %.4f%%\n',Er)                    % Display the Percentage Error.
```

```
%-< Section 1.8 >-----<<<< Configuring the Pie Chart >>>>--------------------------------------------------------

Pc = [Er (100-Er)];                                         % Setting up the Variables for the Pie Chart
PT = pie(Pc,'%.3f%%');                                      % Create the Pie Chart
txt = ['(For ' ,num2str(N) ' ' 'Points Generated)'];       % Defining the additional title
title({'Percentage Error Vs Percentage Validity';txt})      % Apply a title to the Pie Chart
legend('Percentage Error','Percentage Validity')            % Apply a legend to the Pie Chart
legend(Location="southeastoutside")                         % Define a location for the pie chart.
colormap([.5 .5 .5;0 1 1])                                  % Apply a colourmap to the pie chart,
                                                            % To help the user visualize the data.
title(legend,'Legend ')                                     % Title the legend of the pie chart.
%----------------------------------------------------------------------------------------------------------------
```

Appendix (2)

```
% MATLAB Wednesday Assignment Part 2
% By: Alan O'Donnell
% SN:21387526
% Date: 20/11/21
% Version: 1.5
%
% Aim:
%    1) Extend my code to integrate any single variable function between any two user defined limits.
%    2) Plot the function
%    3) Calculate the area under the curve.
%    4) Write the data into a txt file.
%

%-< Section 2.1 >-----<<<< Initializing Variables >>>>-----------------------------------------------------------------------------

clc                                                                     % Clear Command Window
clear                                                                   % Clear Variables
syms x                                                                  % Create x as a symbolic scalar variable.
N = 0;                                                                  % Initializing the number of points that will be used.
Max_Min_Value = [];                                                     % Initializing the Max Min Values
Y_Values = [];                                                          % Initializing the Y values
NSuc = 0;                                                               % Setting the number of successful rounds to 0
NFail = 0;                                                              % Setting the number of failed rounds to 0

%-< Section 2.2 >-----<<<< Take In User Input (Points) >>>>-----------------------------------------------------------------------

while N < 1         %Ensuring that a positive Integer is inputted
 N = input('Please Enter a positive Integer for the number of points, Recommend 1,000-10,000 \n');
end
N = round(N);       %We can't perform a partial round of code, so every number must be a positive whole number
fprintf('The chosen number of points is %d\n',N)

%-< Section 2.3 >-----<<<< Setting up the limits of Integration >>>>--------------------------------------------------------------

P1 = input('Enter P1 value: \n');                                       % Take in user input for the lower limit.
P2 = input('Enter P2 value: \n');                                       % Take in user input for the Higher limit.
Q = 0;                                                                  % Initialize while restriction variable.

%-< Section 2.4 >-----<<<< Take In User Input (Function) >>>>----------------------------------------------------------------------

while Q ~= 1                                                            % Loop until condition is broken
Eq = input('\nEnter a function in terms of x \n(eg: 4*x^(4)+4)\n','s'); % Taking in user input to set up functions
fc = str2func(['@(x)' char(Eq)]);                                       % Converts input to function handle
Q = input('\nPlease Type 1 for yes and 2 for no \n');                   % Ask the User If they are happy with the inputted function
if Q == 1                                                               % If inputted 1 for Yes:
disp('Confirmed Response')                                              % Display a confirmation response
break                                                                   % Break While Loop.
elseif Q == 2                                                           % If Inputted 2 for No:
    disp('Please Enter in a new function')                              % Ask the User to Input a new function (Repeat the Process).
elseif Q ~= 1 && Q ~= 2                                                 % If the user inputted an invalid response i.e., not 1 or 2.
disp('Unknown Response')                                                % Display an error message.
disp('Please Enter in a new function')                                  % Repeat the process
end                                                                     % End the If Statement.
end                                                                     % End the while loop.
                                                                        % && was used instead of &
                                                                        % (Logical Short-Circuiting Operator, for scalars)

%-< Section 2.5 >-----<<<< Find the Max & Min Values (Differentiation) >>>>---------------------------------------------------------

DS = char(diff(fc(x)));                      % Calculate the derivative of the function fc with respect to x.
dif = str2func(['@(x)' DS]);                 % Convert the string format of the function fc into an anonymous function.
Rts = solve(dif(x));                         % Finding the roots of the derivatives.
MMV = [fc(P1) fc(P2)];                       % Finding the Max Min values by subbing in,
                                             % The interval points (P1 % P2) into the equation.

%-< Section 2.6 >-----<<<< For Statement >>>>-------------------------------------------------------------------------------------

for h = 1:length(Rts)                        % The Height of the box is equal to the length of the root values
    MMV(end + 1) = fc(Rts(h));               % Setting the Max Min Value (end+1) to be equal to the function
end                                          % End the for loop.
      YMax = max(MMV);                       % Using MATLAB’s Max function to find the maximum value of the function fc.
      YMin = min(MMV);                       % Using MATLAB’s Min function to find the minimum value of the function fc.

%-< Section 2.7 >-----<<<< Plotting the function. >>>>----------------------------------------------------------------------------
                                                  % Debugging: plot(P1:P2,0,"Color",'k')
hold on                                           % Hold On
fplot(@(x) 0, [P1 P2],"Color",'g', "LineWidth",3) % Plot 0 from the first limit point to the second showing the length across it.
hold on                                           % Hold On
legend('AutoUpdate','off')                        % Prevents MATLAB from presenting all the points.
Pos = fplot(fc(x),[P1 P2],"LineWidth", 4);        % Plot the function
Pos.Color = ('m');                                % Changing the colour of the line % legend data.
legend('X Axis','Fc',Location= 'southwest')       % Configuring the legend (Labels, Location)
hold on                                           % Hold On
title('Graph of function Fc Vs X Axis')           % Add a title to the graph.

%-< Section 2.8 >-----<<<< Box Length & Width Calculations >>>>--------------------------------------------------------------------

Bw = P2 - P1;    % Calculate Box Width from the range of the Limits.

% Next: To calculate the boxes Length, Bl there is a few different variations to consider
% First if both y values are above the x axis, we can find the Box Length by subtracting the Max Y value from the minimum Y value

Bl = YMax - YMin;                % Bow Length is equal to the range of the Y values.
if YMin >= 0                     % Checking that the minimum value for Y is above the x axis for the previous statement to be true.
    if YMax >= 0                 % Checking that the maximum value for Y is above the x axis for the previous statement to be true.
        if YMin > YMax           % Next: Check that the Minimum Y value is higher than the Max value
            Bl = YMin;           % If the Minimum Value is higher, this means that the Box length is equal to the Minimum Y length
        elseif YMax > YMin       % If the previous line is false, this means that the Y maximum is greater than the Y minimum value
            Bl = YMax;           % Therefore the length of the box is equal to the Maximum value for y.
        end                      % End the Inner If Statement.
    end                          % End the Outer If Statement.


%-< Section 2.9 >-----<<<< Box Calculations (Y Values Below the X Axis) >>>>---------------------------------------------

elseif YMin < 0                       % Check that the Minimum Y value is indeed below the x axis i.e., less than 0
    if YMax < 0                       % Check that the Maximum Y value is indeed below the x axis i.e., less than 0
        if YMin < YMax                % If the Maximum value for Y is greater than the minimum value then:
            Bl = YMin;                % The box length is equal to the Minimum Value for Y
        elseif YMax < YMin            % If the previous statement is not true then we check if,
                                      % The Minimum Y value is greater than the y maximum value.
            Bl = YMax;                % If this is true then the bow length is equal to the Maximum Y value.
        end                           % End the Inner If Statement
    end                               % End the Middle If Statement
end                                   % End the continued Else If Statement.

%-< Section 2.10 >-----<<<< Check if Y Values are both above the X Axis (Same Signs (+)) >>>>------------------------------

for i = 1:N                           % For Loop from 1 to N iterations
    x = (P2-P1).*rand() + P1;         % Calculate the X Value using the limit & rand (built in MATLAB function).
    y = (YMax-YMin).*rand() + YMin;   % Calculate the Y Value using the Y Range & Rand function.
    if YMin >= 0                      % Check if the Minimum Y Value is on or above the X Axis.
        if YMax >= 0                  % Check if the Maximum Y Value is on or above the X Axis.
            if YMin > YMax            % Next Check that the Y Min Value is greater than the Y Max Value.
                y = YMin.*rand();     % If the previous statements are true: calculate a random Y Value from the Min Value
            elseif YMax > YMin        % If the Y Max is greater than the minimum value,
                y = YMax.*rand();     % Calculate a random Y Value using the Y Max value.
            end                       % End Inner If Statement.
        end                           % End Outer If Statement.

%-< Section 2.11 >-----<<<< Check if Y Values are both below the X Axis (Same Signs (-)) >>>>------------------------------

 elseif YMin < 0                      % Else If Statement: Check that the Min Y Value is below the X Axis.
        if YMax < 0                   % If Statement: Check that that the Max Y Value is below the X Axis.
            if YMin < YMax            % If the Y Min Value is Less than the Y Max Value:
                y = YMin.*rand();     % Generate the Y Value using the Y Min & Rand Function.
            elseif YMax < YMin        % If the Y Max Value is less than the Y Min Value:
                y = YMax.*rand();     % Generate the Y Value using the YMax & Rand Function.
            end                       % End Inner If Statement.
        end                           % End Outer If Statement.

%-< Section 2.12 >-----<<<< Y Value Check (Opposite Signs) >>>>-------------------------------------------------------------

                               % Now that I have confirmed if they are either above or below the X Axis.
                               % Next: Check their signage.
    end                        % End the previous ElseIf Statement
    Y_Values(end + 1) = y;     % Set the Y Values of end + 1 equal to the Generated Y Value.
    Y_Limit = fc(x);           % Set the Y Limit to the function Fc with respect to x

%-< Section 2.13 >-----<<<< Plot based on the Generated Y Value being less than 0 >>>>-------------------------------------

   % Some additional Tests are to compare a generated y value,
   % with the original value, then check if it is inside or outside the area.

plot(x,y,"Marker",".","Color",'b')                   % Plot the points in blue with a decimal point as a marker.
    hold on                                          % Hold On --> Don't Reset the plotting.
    if Y_Limit < 0                                   % If Statement: Check that the Y Limit is less than 0.
        if y >= Y_Limit                              % If Statement: Generated Y Value is less than or equal to the Y Limit.
            if y < 0                                 % If Statement: Check that the Generated Y Value is less than 0.
                NSuc = NSuc + 1;                     % Calculate a successful round.
                plot(x,y,"Marker",".","Color",'b')   % Plot the points in blue with a decimal point as a marker.
                hold on                              % Hold On --> Don't Reset the plotting.
            end                                      % End the Inner If Statement.
        end                                          % End the Outer If Statement.

%-< Section 2.14 >-----<<<< Plot based on the Generated Y Value being greater than 0 >>>>----------------------------------------------

    elseif Y_Limit > 0                            % ElseIf Statement: Check if the Y Limit is greater than 0
        if y <= Y_Limit                           % If Statement: Check that the generated Y Value is less than or equal to the Y Limit
            if y > 0                              % If Statement: Check that the generated Y Value is greater than 0
                NSuc = NSuc + 1;                  % Calculate a successful round.
                plot(x,y,"Marker",".","Color",'b')% Plot the points in blue with a decimal point as a marker.
                hold on                           % Hold On --> Don't Reset the plotting.
            end                                   % End the Inner If Statement.
        end                                       % End the Middle If Statement
    end                                           % End the outer If Statement.
end                                               % End the For Loop (1:N)

%-< Section 2.15 >-----<<<< Area Calculations>>>>------------------------------------------------------------------

Ba = Bl*Bw;              % To Calculate the Area of the Box, I multiplied Box Length by Box Width
Ca = Ba*NSuc/N;          % Calculate the Area of the circle with the given formula: Box Area *(Amount of success / Amount of versions)

%-< Section 2.16 >-----<<<< Customize the figure. >>>>---------------------------------------------------------------------------------

ylabel('Y Axis')         % Label Y Axis
xlabel('X Axis')         % Label X Axis
grid                     % Apply a grid to the figure.
title(legend,'Legend ')  % Apply a title to the legend.

%-< Section 2.17 >-----<<<< Displaying & Printing Information >>>>---------------------------------------------------------------------

fprintf('The area under the curve is %d Units Squared\n',Ca)   % Display the area under the curve to the user.
data = [("y = " + char(Eq) + "       Between Points " + P1 +' - '+ P2 +"    Area Under The Curve = " + Ca + ' ' + "Units Squared")];
writematrix(data,'A2P2_results.txt')                            % Write the data into a txt file
%--------------------------------------------------------------------------------------------------------------------------------------
```

Appendix (3)

```
% MATLAB Wednesday Assignment Part 3
% Aim:
%    1) Calculate how many circles of radius r can fit inside a square of length L
%    2) Calculate how much area is left over.
%    3) Calculate the voidage.
%    4) Display the information
% By: Alan O'Donnell
% SN: 21387526
% Date: 27/11/21
% Version: 2.1
% Lab Slot: Wednesday

%-< Section 3.1 >-----<<<< Initializing Variables >>>>---------------------------------------------
clc
clear                                                                     % Clear Variables
L = input('Input a value in for the length of the square \n');            % Take Length Input
r = input('Pick a radius r \n');                                          % Take Radius Input
%-< Section 3.2 >-----<<<< Area Calculations) >>>>-------------------------------------------------------------------------------------
--
if L < 2*r                                 % Check if the Length of the Box is less than twice the radius,
                                           % If this is true, the values are invalid, as it means a circle won’t fit inside the square.
    disp('Values in valid')                % Then Display The error to inform the user.
    L = input('Pick a length L \n');       % Re-take Length Input
    r = input('Pick a radius r \n');       % Re-take Radius Input
else
Ba = L^(2);                                % Calculate the box (Square) area, using the formula Length Squared
Ca = pi*r^(2);                             % Calculate the circle area, using the formula Pi*Radius Squared

%-< Section 3.3 >-----<<<< Calculate Quantity of Discs >>>>----------------------------------------------------------------------------
---
N_Row = floor(L / (2*r));                   % Calculate How many circles can fit in one row by dividing,
                                            % the Length of the square by the diameter of a circle,
                                            % Ie.2*Radius. It’s important to floor this value so there isn’t part of a circle inside a
row,
                                            % i.e., ensuring there all full circles fitting in the square.
N_Total = N_Row^(2);                        % To calculate how many circles will fill the entire square,
                                            % Square the number of circles found for 1 row.

Ca_Total = Ca*N_Total;                      % Calculate the total area taken up by the circles by multiplying,
                                            % the area of 1 circle multiplied by the total amount of circles in the square.

A_LeftOver = Ba - Ca_Total;                 % Calculate how much area is left over by subtracting the Box (Square)
                                            % Area by the area taken up by all the circles.

V = VoidFunc(A_LeftOver,Ba);                % Calculate the Percentage left over (Voidage), by dividing the amount left over by the
total
                                            % Area of the box and multiplying it by 100 to make it into a percentage.

end                                         % End If

%-< Section 3.4 >-----<<<< Print off the results of the simulation >>>>----------------------------------------------------------------

fprintf('The Number of circles inside the square are:%d \n', N_Total);       %Print how many circles there are.
fprintf('The Amount of voidage is: %.4f%% \n', V);                           %Print the amount of voidage.

%-< Section 3.5 >-----<<<< Configuring the Pie Chart >>>>------------------------------------------------------------------------------

Pc = [V (100-V)];                                               % Setting up the Variables for the Pie Chart
pie(Pc,'%.3f%%')                                                % Create the Pie Chart
txt = ['(',num2str(N_Total) ' ' 'Discs Generated)'];            % Defining the additional title
title({'Voidage vs Occupied Space (Disc Formation)';txt})       % Apply a title to the Pie Chart
Custi()                                                         % Custom Function for customizing Figure.
%--------------------------------------------------------------------------------------------------------------------------------------
```

Appendix (4)

```matlab
% MATLAB Wednesday Assignment, Part 4 (Spheres inside a cube under stacked formation)
% Aim:
%    1) Calculate how many spheres of radius r can fit inside a cube of length L.
%    2) Calculate how much volume is left over.
%    3) Display the findings to the user.
%    4) Calculate the voidage.
%
% By: Alan O'Donnell
% SN: 21387526
% Date: 27/11/21
% Version: 1.6
% Lab Slot: Wednesday

%-< Section 4.1 >-----<<<< Configure Variables >>>>------------------------------------------------------------------------------------------------

clc                                         % Clear Command Window
clear                                       % Clear Variables
L = input('Pick a length L \n');            % Take Length Input
r = input('Pick a radius r \n');            % Take Radius Input

%-< Section 4.2 >-----<<<< Ensure a valid response to user input >>>>------------------------------------------------------------------------------

if L < 2*r                                  % Check if the Length of the Box is less than twice the radius,
                                            % If this is true, the values are invalid, as it means a sphere won’t fit inside the square.
    disp('Values invalid')                  % Then Display The error to inform the user.
    L = input('Pick a length L \n');        % Re-take Length Input
    r = input('Pick a radius r \n');        % Re-take Radius Input
else

%-< Section 4.3 >-----<<<< Volume Calculations >>>>------------------------------------------------------------------------------------------------

Bv = L^(3);                                 % Calculate the cubes volume, using the formula Length cubed.
Cv = (4/3)*pi*r^(3);                        % Calculate the spheres volume, using the formula (4/3)Pi*Radius cubed.

%-< Section 4.4 >-----<<<< Sphere Calculations >>>>------------------------------------------------------------------------------------------------

N_Row = floor(L / (2*r));                   % Calculate How many spheres can fit in one row by dividing,
                                            % The Length of the square by the diameter of a sphere,
                                            % i.e., 2*Radius. It’s important to floor this value so there isn’t part of a sphere inside a row,
                                            % i.e., ensuring there all full spheres fitting in the cube.
N_Total = N_Row^(3);                        % To calculate how many spheres will fill the entire square,
                                            % Cube the number of spheres found for 1 row.

Cv_Total = Cv*N_Total;                      % Calculate the total volume taken up by the spheres:
                                            % By multiplying the volume of 1 sphere multiplied by the total amount of spheres in the cube.
V_LeftOver = Bv - Cv_Total;                 % Calculate how much volume is left over by subtracting,
                                            % the cube volume by the volume taken up by all the spheres.

V = SphereVoidage(V_LeftOver,Bv);           % Calculate the Percentage left over (Voidage),
                                            % By dividing the amount left over by the total volume of the cube,
                                            % ,and multiplying it by 100 to make it into a percentage.
                                            % Save each point, using distance formula make the distance equal 2R
end                                         % End the If statement

%-< Section 4.5 >-----<<<< Print off the results of the simulation >>>>----------------------------------------------------------------------------

fprintf('The Number of spheres inside the cube are: %d \n', N_Total);      %Print how many Spheres there are.
fprintf('The Number of voidage is: %-10f\n', V);                           %Print the amount of voidage.

%-< Section 4.6 >-----<<< Configuring the Pie Chart >>>>-------------------------------------------------------------------------------------------

Pc = [V (100-V)];                                                           % Setting up the Variables for the Pie Chart
pie(Pc,'%.3f%%')                                                            % Create the Pie Chart
txt = ['(',num2str(N_Total) ' ' 'Spheres Generated)'];                      % Defining the additional title
title({'Voidage vs Occupied Space (Stacked Sphere Formation)';txt})         % Title the Pie Chart
Custi()                                                                     %Custom Function for customizing Figure.
%--------------------------------------------------------------------------------------------------------------------------------------------------
```

Appendix (5)

```
% MATLAB Wednesday Assignment, Part 5 (Spheres inside a cube under random formation)
% By: Alan O'Donnell
% SN:21387526
% Date: 20/11/21
% Version: 1.3
% Aim:
%    1) Calculate the numbers of spheres of radius r that can be randomly generated in a cube of length L.
%    2) Calculate the volume taken up by spheres.
%    3) Subtract the volume from the total volume of the cube to calculate the voidage.
%    4) Compare different factors to increase the number of spheres, volume that can fit into the cube.
%    5) Visually display all this information to the user.
% Note:
% To Calculate how many spheres, we can fit inside a cube in random locations, we must store each of the spheres coordinate points.
clc                 % Clear Command Window
clear               % Clear Variables
N = 0;              % Initializing N
n = 0;              % Initializing n
w = 0;              % Initializing w

%-< Section 5.1 >-----<<<< Initializing Values >>>>------------------------------------------------------------------------------------

while N < 500000                                               % Ensure that the number of iterations is a positive number.
 N = input('Please Enter a positive Integer (Ideally where N >1,000,000) \n');
                                                               % For high levels of accuracy, recommend going over 500,000 iterations.
end                                                            % Select how many iterations to run.
N = round(N);                                                  % Code cannot run a fraction of an iteration,
                                                               % so, all numbers must be whole numbers.
fprintf('The chosen amount of rounds is %d\n',N)

%-< Section 5.2 >-----<<<< User Input (L+r) >>>>---------------------------------------------------------------------------------------
-------------------------------
L = input('Pick a length L (Recommended:1-3) \n');          % Take Length Input
r = input('Pick a radius r (Recommended:0.1-0.5) \n');      % Take Radius Input
if L < 2*r                                                  % Check if the Length of the Box is less than twice the radius,
                                                            % If this is true, the values are invalid, as it means a sphere won't fit
inside the square.
    disp('Values invalid')                                  % Then Display The error to inform the user.
    L = input('Pick a length L (Recommended:1) \n');        % Re-take Length Input
    r = input('Pick a radius r (Recommended:0.1) \n');      % Re-take Radius Input
else

%-< Section 5.3 >-----<<<< User Input (Colour Preference) >>>>-------------------------------------------------------------------------

ICv = 0;                   %Initialize Input Colour Value
while (ICv > 7 || ICv < 1) %Ensure that a colour on the list is selected
ICv =input('Choose a colour by entering its corresponding value, Green (1), Cyan (2), Red (3), Orange (4), Peach (5), Brown(6) or Blue
(7)  \n');
ICv = round(ICv);          % Take in user input, Round to a positive whole number.
end

%-< Section 5.4 >-----<<<< Initializing Values (X,Y,Z,O) >>>>--------------------------------------------------------------------------
%  X Values        Y Values         Z Values        Origin Values
   Xv = [];        Yv = [];         Zv = [];        Ov = [];

   [X,Y,Z] = sphere;    % At Point X,Y,Z a sphere will be located.

%-< Section 5.5 >-----<<<< 3D Plotting >>>>--------------------------------------------------------------------------------------------

legend('AutoUpdate','off')                % Disabling auto-update for the legend so it isn't filled with data points.
hold on                                   % Hold On
plot3(0:L,0,0);                           % Plot X of length L
plot3(0,0:L,0);                           % Plot Y of length L
plot3(0,0,0:L);                           % Plot Z of length L
set(gca,'View',[-30,30]);                 % Adjust the elevation of the plot

%-< Section 5.6 >-----<<<< Plotting The Cube >>>>--------------------------------------------------------------------------------------

% Define x,y,z values as 1xL matrix
L1 = linspace(0,L,100);                                                                            % Generating the edges
%--------------------------------------------------------------------------------------------------------------------------------------
% Right Z Axis            % Close Z Axis             % Back Z Axis            % Left Z Axis
plot3(L,0,L1,'k .');      plot3(0,0,L1,'k .');       plot3(L,L,L1,'k .');     plot3(0,L,L1,'k .');  % k for black edges
%--------------------------------------------------------------------------------------------------------------------------------------
% Bottom Close X Axis     % Top Close X Axis         % Bottom Far X Axis      % Top Far X Axis
plot3(L1,0,0,'k .');      plot3(L1,0,L,'k .');       plot3(L1,L,0,'k .');     plot3(L1,L,L,'k .');
%--------------------------------------------------------------------------------------------------------------------------------------
---------------------------------
% Left Bottom Y Axis      % Left Top Y Axis          % Right Bottom Y Axis    % Right Top Y Axis
plot3(0,L1,0,'k .');      plot3(0,L1,L,'k .');       plot3(L,L1,0,'k .');     plot3(L,L1,L,'k .');

%-< Section 5.7 >-----<<<< Generate Random Points within the box >>>>------------------------------------------------------------------

for n = 1:N                     % For loop, loops for N number of iterations.
    x = ((L-r) -r).*rand() + r; % Generate Random X Value
    Xv(end + 1) = x;            % Configuring X Value

    y = ((L-r) -r).*rand() + r; % Generate Random Y Value
    Yv(end + 1) = y;            % Configuring Y Value

    z = ((L-r) -r).*rand() + r; % Generate Random Z Value
    Zv(end + 1) = z;            % Configuring Z Value
end
```

```
%-< Section 5.8 >-----<<<< Generate The First Sphere >>>>---------------------------------------------------------------------------

%Now Generate the first sphere
%Created first circle in bottom right corner
X1 = r;                                  %Let the X Value = Radius (r)
Y1 = r;                                  %Let the Y Value = Radius (r)
Z1 = r;                                  %Let the Z Value = Radius (r)
plot3(X1,Y1,Z1,"Marker","*")             %Plotting the first sphere (Marker)
legend('AutoUpdate','on')                %Only Want the legend displaying the marker for sphere, not the rest of the calculations.
Surf = surfl(X*r+X1,Y*r+Y1,Z*r+Z1);
legend('AutoUpdate','off')
end
%-< Section 5.9 >-----<<<< For Loop >>>>-----------------------------------------------------------------------------------------

for i = 1:9000000000000000000  %This is approx. (~) 1: infinity as MATLAB has a restriction on for loop iterations.
    % For Bug Fixing disp(i);
    TXv = [];                             %Initializing The True X Value
    TYv = [];                             %Initializing The True Y Value
    TZv = [];                             %Initializing The True V Value
    TOv = [];                             %Initializing The True Distance to the origin.
%Check that it will go through each point
    for n = 1:length(Xv)
        X2 = Xv(n);
        Y2 = Yv(n);
        Z2 = Zv(n);
%By calculating the distance between the circles, if they are two close, the spheres but be discarded.
        if sqrt((X2-X1)^2 + (Y2-Y1)^2 + (Z2-Z1)^2) < 2*r
            Xv(n)= 0;   %Discard the X Value
            Yv(n)= 0;   %Discard the Y Value
            Zv(n)= 0;   %Discard the Z Value
%Save the points that are considered possible sphere locations.
        else
            TOv(end + 1) = sqrt((X2)^2 + (Y2)^2 + (Z2)^2);
            TXv(end + 1) = X2;
            TYv(end + 1) = Y2;
            TZv(end + 1) = Z2;
        end
%This is repeated until all points have been checked
    end
%-< Section 5.10 >-----<<<< Updating Values >>>>-------------------------------------------------------------------------------

%  Origin Value    X Value       Y Value       Z Values
   Ov = TOv;       Xv = TXv;     Yv = TYv;     Zv = TZv;

%-< Section 5.11 >-----<<<< Locate the closest Sphere to the origin >>>>-------------------------------------------------------------

    Dmin = min(Ov);                 %The minimum distance is the minimum distance from the origin
    Search = find(Ov == Dmin);      %Find function is used to find when two values match up,
                                    %In this case, find the next closest sphere.

%-< Section 5.12 >-----<<<< Update the values via find function. >>>>---------------------------------------------------------------

% Y Value              X Value Z Value        Z Value
  Y1 = Yv(Search);     X1 = Xv(Search);       Z1 = Zv(Search);

   if isempty(X1) % If X1 is empty, there is no more points
        break %If no more points are found, break the for loop.
    end

%-< Section 5.13 >-----<<<< Plot The Spheres >>>>-------------------------------------------------------------------------------

    plot3(X1,Y1,Z1,"Marker","*")        % 3 Value Plot
    legend('AutoUpdate','off')          % Disable AutoUpdate
    surfl(X*r+X1,Y*r+Y1,Z*r+Z1)         % Create a 3D Surface plot

%-< Section 5.14 >-----<<<< Change The Colour Map Based On User Input >>>>-----------------------------------------------------------

if ICv == 1
    colormap("summer")   %If the user selected the colour green, the colour map pre-set is summer.
elseif ICv == 2
    colormap("bone")     %If the user selected the colour cyan, the colour map pre-set is bone.
elseif ICv == 3
    colormap("hot")      %If the user selected the colour red, the colour map pre-set is hot.
elseif ICv == 4
    colormap("autumn")   %If the user selected the colour orange, the colour map pre-set is autumn.
elseif ICv == 5
    colormap("pink")     %If the user selected the colour peach, the colour map pre-set is pink.
elseif ICv == 6
    colormap("copper")   %If the user selected the colour brown, the colour map pre-set is copper.
elseif ICv == 7
    colormap("winter")   %If the user selected the colour blue, the colour map pre-set is winter.
else
    disp('Failed to input colour, default will be used instead.') % If the user inputted an invalid response, display an error message.
    colormap("default")                                            % Set the colourmap to default.
end
%-< Section 5.15 >-----<<<< Figure Maintance >>>>-------------------------------------------------------------------------------

figure(1);shading interp                                                % Controls the colour shading of the surface.
txt = ['(L =' ' ',num2str(L),' ,r =' ' ',num2str(r),')'];               % Utilize Num2str to convert Numbers to string values.
figure(1);title({'Spheres Randomly Generated Inside of A Cube ';txt})   % Apply title to the figure
figure(1);title(legend,'Legend ')                                       % Apply title to the legend
figure(1);legend(Location="southeastoutside")                           % Assigning legend location.
figure(1);axis equal                                                    % Ensure the axis are equal
figure(1);grid on                                                       % Enable Grids
end                                                                     % End the loop
fprintf('The Number of spheres generated inside the cube is %d \n',i)   % Display how many spheres are generated.
xlabel('X Axis')                                                        % Label the X Axis
ylabel('Y Axis')                                                        % Label the Y Axis
zlabel('Z Axis')                                                        % Label the Z Axis

%-< Section 5.16 >-----<<<< Display How Many Spheres are generated in the legend >>>>-----------------------------------------------
```

```
     if i > 1
          Surf.DisplayName = ['Spheres Generated = ',num2str(i)];           % If more than 1 sphere is generated use this wording.
     else
          Surf.DisplayName = 'Single Sphere Generated';                     % If only 1 sphere is generated use this wording.
     end
          figure(1);legend("show")                                          % Show the legend

%-< Section 5.17 >-----<<<< Calculations >>>>-----------------------------------------------------------------------------------------

Volume_Sphere = (4/3)*(pi)*(r)^(3);                                  % Calculate the volume of a single sphere.
Total_Vs = Volume_Sphere*i;                                          % Calculate the total volume of all the spheres.
Volume_Box = L^(3);                                                  % Calculate the volume of the box
Volume_Difference = Volume_Box - Total_Vs;                           % Calculate the difference in volume of the box and total spheres.
Voidage = ((Volume_Difference)/(Volume_Box))*100;                    % Calculate the percentage of empty space (Voidage).
fprintf('The Voidage of the cube is %.4f%%\n',Voidage)               % Display the voidage percentage

%-< Section 5.18 >-----<<<< Configuring the Pie Chart >>>>----------------------------------------------------------------------------

Pc = [Voidage (100-Voidage)];                                                % Setting up the Variables for the Pie Chart
figure(2);pie(Pc,'%.3f%%')                                                   % Create the Pie Chart
figure(2);title({'Voidage vs Occupied Space';'(Random Sphere Formation)'})   % Title the Pie Chart
figure(2);legend('Voidage','Occupied Space')                                 % Apply a legend to the pie chart.
figure(2);legend(Location="southeastoutside")                                % Define a location for the pie chart.
figure(2);title(legend,'Legend ')                                            % Title the legend of the pie chart.

%-< Section 5.19 >-----<<<< Using User inputted colour preference, for pie chart via 0-1 RGB Values >>>>------------------------------

if ICv == 1                                                                 %If Green Was Selected for the spheres
    figure(2);colormap([.5 .5 .5;0.4 0.8 0])
elseif ICv == 2                                                             %If Cyan Was Selected for the spheres
    figure(2);colormap([.5 .5 .5; 0.6 0.6 0.8])
elseif ICv == 3                                                             %If Red Was Selected for the spheres
    figure(2);colormap([.5 .5 .5; 1 0.3 0.2])
elseif ICv == 4                                                             %If Orange Was Selected for the spheres
    figure(2);colormap([.5 .5 .5; 1 0.5 0])
elseif ICv == 5                                                             %If Peach Was Selected for the spheres
    figure(2);colormap([.5 .5 .5; 0.9 0.8 0.7])
elseif ICv == 6                                                             %If Brown Was Selected for the spheres
    figure(2);colormap([.5 .5 .5; 0.7 0.4 0.1])
elseif ICv == 7                                                             %If Blue Was Selected for the spheres
    figure(2);colormap([.5 .5 .5; 0.4 0.5 1])
else
end
%------------------------------------------------------------------------------------------------------------------------------------
```

Appendix (F1)

```
function Void = VoidFunc(A_LeftOver,Ba)
% Void Function for circle: Calculates the Voidage Percentage
% Calculate the Percentage left over (Voidage),
% by dividing the amount of left over area by the total area,
% of the square and multiplying it by 100 to make it into a percentage.
Void = ((A_LeftOver / Ba)*100);
end
```

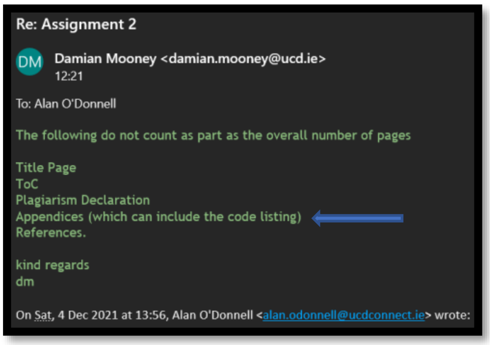
Appendix (F2)

```
function VoidSph = SphereVoidage(V_LeftOver,Bv)
% Void Function for spheres: Calculates the Voidage Percentage
% Calculate the Percentage left over (Voidage),
% by dividing the amount left over volume by the total volume,
% of the cube and multiplying it by 100 to make it into a percentage.
VoidSph = ((V_LeftOver / Bv)*100);
end
```

Appendix (F3)

```
function Cust = Custi()
% Customize the pie chart in one function:
% Includes:)
%          1) Legend Title
%          2) Legend Location
%          3) Legend
%          4) Colour Map
title(legend,'Legend ')               % Title the legend of the pie chart.
legend(Location="southeastoutside")   % Define a location for the pie chart.
legend('Voidage','Occupied Space')    % Apply a legend to the pie chart.
colormap([.5 .5 .5;0 1 1 ])           % Apply a colourmap to the pie chart to help the user visualize the data.
end
```

Correction Note.



Re: Assignment 2

DM Damian Mooney <damian.mooney@ucd.ie>
12:21

To: Alan O'Donnell

The following do not count as part as the overall number of pages

Title Page
ToC
Plagiarism Declaration
Appendices (which can include the code listing)
References.

kind regards
dm

On Sat, 4 Dec 2021 at 13:56, Alan O'Donnell <alan.odonnell@ucdconnect.ie> wrote: