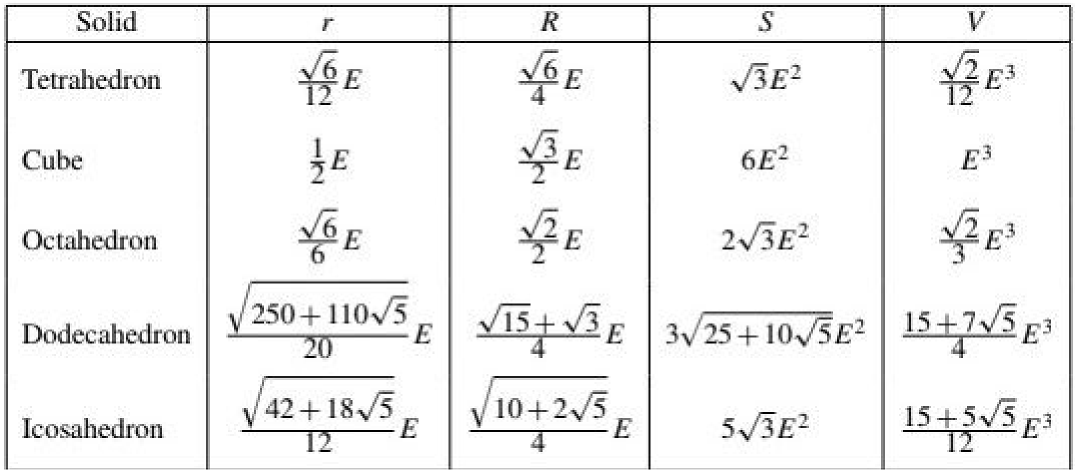
1.1 Introduction

This problem is on platonic solids: solid geometric figures which have uniform geometric faces. The goal of this problem is to come up with a solution to the problem of nesting each of the 5 platonic solid within each other. The sequence is that an Icosahedron is nested with in a Dodecahedron, with in an octahedron with in a cube that's with in a tetrahedron which is nest inside a unit sphere. Each solid must be the largest size that can fit with in its parent solid. The program will solve for the dimensions, inradius, outradius, and edge length of each solid. It will then display the results in a table

1.2 Model and Methods

Al relevant quantities which define the platonic solids can be related to their respective face edge lengths. A table provided in the problem statement gives the formulas which give this relation. By using them we can find each solids inradius and outradius once the edge length is found. Below is an image of this table.



Solid	r	R	S	V
Tetrahedron	$\frac{\sqrt{6}}{12}E$	$\frac{\sqrt{6}}{4}E$	$\sqrt{3}E^2$	$\frac{\sqrt{2}}{12}E^3$
Cube	$\frac{1}{2}E$	$\frac{\sqrt{3}}{2}E$	$6E^2$	E^3
Octahedron	$\frac{\sqrt{6}}{6}E$	$\frac{\sqrt{2}}{2}E$	$2\sqrt{3}E^2$	$\frac{\sqrt{2}}{3}E^3$
Dodecahedron	$\frac{\sqrt{250+110\sqrt{5}}}{20}E$	$\frac{\sqrt{15}+\sqrt{3}}{4}E$	$3\sqrt{25+10\sqrt{5}}E^2$	$\frac{15+7\sqrt{5}}{4}E^3$
Icosahedron	$\frac{\sqrt{42+18\sqrt{5}}}{12}E$	$\frac{\sqrt{10+2\sqrt{5}}}{4}E$	$5\sqrt{3}E^2$	$\frac{15+5\sqrt{5}}{12}E^3$

Here r is the inradius, R is the outradius, and E is the edge length. The outradius is the widest part of the solid and the inradius is the smallest. If the nested solid has an outradius equal to the inradius of its parent solid, then without considering orientation, this is corresponds to the largest solid which can be nested with a particular parent solid. Starting with the unit sphere radius, the edge of each solid is calculated using data from its parent. The next section will show some sample calculations from the code which should clarify.

1.3 Calculations and Results

Here are a few sample calculations which show the process of finding each platonic solid's edge length and radii. The fist is a sample which takes the radius of the sphere and uses an inverse of the table's function for the tetrahedron outradius, with the sphere's radius as a radius dimension, to calculate the value.

```
% R_U is the radius of the unit sphere
               R_U = 1;

 % tetrahedron properties initialized
        E_T = (4/sqrt(6))*R_U;
```

The program progresses through each nested solid, storing the values inside arrays for organization.

```
T = [(sqrt(6)/12)*E_T , (sqrt(6)/4)*E_T, E_T];
```

The array values and inradius, outradius , and edge length respectively. Each value is calculated at the time of initialization using the function provided in the table. Below is a sample of the output when the program is run.

```
solid           inradius    outradius   edge

Tetrahedron     0.333333    1.000000    1.632993
Cube            0.192450    0.333333    0.384900
Octahedron      0.111111    0.192450    0.272166
Dodecahedron    0.088295    0.111111    0.079294
Icosahedron     0.070164    0.088295    0.092839
```

The next section will discuss some of the results.

1.4 Discussion

The purpose of this problem is to create a series of platonic solids which each fit with in each other: in the most efficient way possible. As a result it is expected that the dimensions of each successive solid will be smaller than those of the previous. The results of the program reveal this to be true with one exception. The edge length of the nested icosahedron is actually a bit bigger than the edge of its parent dodecahedron. This exception can be explained by the difference in face shape between the two solids. The icosahedron face is that of an equilateral triangle and the face structure of the dodecahedron is a regular pentagon.

2.1 Introduction

The goal of this problem is to compare 8 methods of approximating the perimeter length of an ellipse and to determine a method, if available, that will show which the perimeter most accurately. The code written will prompt the user to specify the size or the semi-major and semi-minor axis, a and b, and will then display the value found by each method out to the 6th decimal point. A comparison of each method's value and accuracy will be discussed at the end of this section.

2.2 Model and Methods

The the ellipse has well known formula which describes the shape, but as mentioned in the problem statement, there is no easy formula for its perimeter if its semi-major axis is not equal to its semi-minor axis. However there are a number of formulas which approximate the perimeters. Below are the formulas provided which will be tested.

$$\begin{aligned} P_1 &= \pi(a+b) & P_5 &= \pi(a+b)\left(1+\frac{3h}{10+\sqrt{4-3h}}\right) \\ P_2 &= \pi\sqrt{2(a^2+b^2)} & P_6 &= \pi(a+b)\frac{64-3h^2}{64-16h} \\ P_3 &= \pi\sqrt{2(a^2+b^2)-\frac{(a-b)^2}{2}} & P_7 &= \pi(a+b)\frac{256-48h-21h^2}{256-112h+3h^2} \\ P_4 &= \pi(a+b)\left(1+\frac{h}{8}\right)^2 & P_8 &= \pi(a+b)\left(\frac{3-\sqrt{1-h}}{2}\right). \end{aligned}$$

The program is designed to be used multiple times to test how each method differs. At the beginning, the program prompts the user to input values for the major and minor axis, a and b. The fringe case which will be explored will be those for which the ellipse is a circle as well as those approaching a flat line. The next section will reveal some sample results.

2.3 Calculations and Results

Here are a few sample results from the program. The first is under the condition that a = b: that is that the ellipse is a circle.

```
  please input a: 1
  please input b: 1

P1 =        6.283185
P2 =        6.283185
P3 =        6.283185
P4 =        6.283185
P5 =        6.283185
P6 =        6.283185
P7 =        6.283185
P8 =        6.283185
 h =       0.000000
```

Under this condition, each method yields the same answer. Next we test the case with a = b , but still the same order of magnitude.

```
  please input a: 1
  please input b: 5

P1 =       18.849556
P2 =       22.654347
P3 =       20.838968
P4 =       21.002129
P5 =       21.010027
P6 =       21.009401
P7 =       21.009963
P8 =       21.249519
 h =        0.444444
```

Under this condition, there are a variety of results which differ by as much as 11%. Next the case when b >> a is tested.

```
please input a: 1
please input b: 1000

P1 =     3144.734246
P2 =     4442.885160
P3 =     3848.933750
P4 =     3976.524302
P5 =     3998.501894
P6 =     3992.686249
P7 =     3996.617160
P8 =     4617.755487
h =        0.996008
```

Again the results vary as you might expect. Conclusions about these results will be discussed in the following section.

2.4 Discussion

For the first case tested, where a = b, we get the expected results: that is that the perimeter is equal to that of a circle. This partly validates the effectiveness of all the methods, but it doesn't provide any information on which would be the best approximation to use. The second case reveals differences, but is not helpful as we have no way of knowing the accuracy of any one over the other. This is where the third case becomes a useful tool in distinguishing the best method for approximation. In the final case b = 1000, which is much greater than a = 1. Under this condition the ellipse degenerate into a sudo line, which should have a perimeter of 4b. The method which most accurately calculates this is method 5, which calculates p5 = 3998.5. Thus method 5 is likely to be the most accurate method for use.

3.1 Introduction

The goal of this problem is to write a code which will solve for the area of the lense created by two overlapping circles. The area will be calculated using the coordinates of the centers of each circle as well as their respective radii. I will be solving this using a provided analytical solution. Once the solution is implemented, a few questions will be answered about some of the fringe cases which the program could encounter; such as what happens when the distance of the two circles' centers is exactly the sum of the two radii and what happens if the distance and one of the radii is less than the other radii.

3.2 Model and Methods

The script prompt the user to input the problem parameters using matlab's input function. The quantities required are x1, y1, r1, x2, y2, and r2 where x and y are the coordinates of the centers of the two circles and r is the radius of each respective circle.

```
x1 = input('please input X1: ');
```

The script then breaks the problem up into a few different calculations. Then takes the results of those calculations and uses them as the input to the formula provided for the area of the lense created by the overlapping of two circles. The formula for the area is shown below.

$$Area = r_1^2 cos^{-1}(\frac{d^2 + r_1^2 - r_2^2}{2dr1}) + r_2^2 cos^{-1}(\frac{d^2 - r_1^2 + r_2^2}{2dr1}) - (\frac{d}{2})c \quad \textbf{(1)}$$

Here, d is the straight line distance between the two circle centers, and c is the chord length of the lense. The code for the calculation of each of these quantities is provided below.

```
d = sqrt(dX^2 + dY^2);

c = (1/d)*sqrt((-d + r1 +r2)*(d - r1 + r2)*(d + r1 - r2)*(d + r1 + r2));
```

Once the area is calculated, the "fprintf" function is used to print all the user defined parameters as well as the calculated lense area. Here is a sample.

```
fprintf('\nx1 = %15.2f\n',x1)
```

3.3 Calculations and Results

When the program executes, the user is prompted to input parameters. Here is a sample.

```
please input X1: 0
please input Y1: 0
please input R1: 1
please input X2: 2
please input Y2: 0
please input R2: 1.5
```

The results of the program are then printed below.

```
x1 =           0.00
y1 =           0.00
r1 =           1.00
x1 =           2.00
y1 =           0.00
r1 =           1.50
Area =         0.4974
```

The program is run also with two fringe cases which we are considering: that is the case when the distance is exactly the sum of the two radii and when the distance plus one of the radii is less than the radius of the other. Here is a sample calculation for the first case.

```
x1 =           0.00
y1 =           0.00
r1 =           1.00
x1 =           2.00
y1 =           0.00
r1 =           1.00
Area =         0.0000
```

And the following is the result from a test of the other case.

```
x1 =           0.00
y1 =           0.00
r1 =           1.00
x1 =           0.00
y1 =           5.00
r1 =           7.00
Area =         3.1416
```

The results or these sample calculations will be discussed in the following section.

3.4 Discussion

For the two fringe cases, the expected results are produced when the program is given input parameters that define these cases. When $d = r_1 + r_2$ the geometric interpretation of this case is that the two circle are just touching each other, but do not overlap at all. In this cases we would expect the area of the lense to be 0, which the program does in fact calculate.

The following case: that $d + r_1 < r_2$ but d not less than 0 is geometrically interpreted to mean that the second circle is completely enclosing the first circle. In this case we would expect that the area of the lense would simply be the area of the first circle. The program does in fact calculate this result which agrees with the theory.