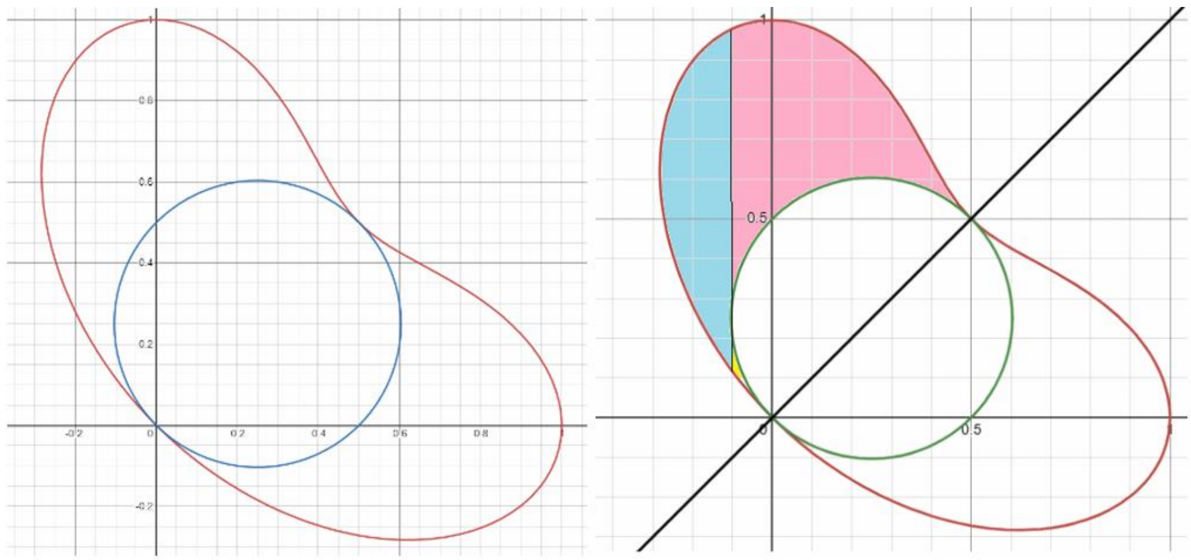
114847849
Kangning Fengwu
AMS326
Homework 2
Source Code Link: https://github.com/kfengwu/AMS326.git

Problem 2.1:

The “kidney” equation $(x^2 + y^2)^2 = x^3 + y^3$ (red curve) can be graphed as



Dig a disc from the kidney. The disc equation is $(x - 0.25) + (y - 0.25) = 0.125$ (blue).

(1) Write a program to use the rectangle method to compute the area of the remaining kidney (4 significant digits).

(2) Write a computer program to use the trapezoidal method to compute the area of the remaining kidney (4 significant digits).

Algorithm :

```
Function rectangle_method(x_min, x_max, y_min, y_max, n):
    set dx = (x_max - x_min) / n  // Width of each rectangle
    set dy = (y_max - y_min) / n  // Height of each rectangle
    set total_area = 0.0

    for i form 0 to n - 1:  // Loop through x subdivisions
        for j from 0 to n - 1:  // Loop through y subdivisions
            compute x = x_min + (i + 0.5) * dx  // Midpoint x-coordinate
            compute y = y_min + (j + 0.5) * dy  // Midpoint y-coordinate

            // Check if the midpoint is inside the kidney and outside the disc
            if is_inside_kidney(x, y) <= 0 and not is_inside_disc(x, y):
                total_area = total_area + (dx * dy)  // Add rectangle area
```

```
    return round(total_area, 4)  // Round to 4 decimal places

Funtion trapezoidal_method(x_min, x_max, y_min, y_max, n):
    set dx = (x_max - x_min) / n  // Grid spacing in x-direction
    set dy = (y_max - y_min) / n  // Grid spacing in y-direction
    set total_area = 0.0

    for i from 0 to n:  // Loop through grid points along x-axis
        for j from 0 to n:  // Loop through grid points along y-axis
            compute x = x_min + i * dx  // Current x-coordinate
            compute y = y_min + j * dy  // Current y-coordinate
            set weight = 1.0  // Default weight (for inner points)

            // Assign different weights for edge and corner points
            if (i == 0 or i == n) and (j == 0 or j == n):
                weight = 0.25  // Corner points get 1/4 weight
            else if i == 0 or i == n or j == 0 or j == n:
                weight = 0.5  // Edge points get 1/2 weight

            // Only count points inside the kidney but outside the disc
            if is_inside_kidney(x, y) AND NOT is_inside_disc(x, y):
                total_area = total_area + (weight * dx * dy)  // Add weighted area

    return round(total_area, 4)
END
```

Result:

To get a relatively high accurate approximation, the number of subdivisions is set to 100 and the x and y boundaries are set to [-1,1]. The rectangle method yielded the remaining kidney area of 0.5920. The trapezoidal method yielded the remaining area of 0.5884. A smaller number of subdivisions was also attempted, resulting in a bigger difference between both methods, and less accurate area.

```
n: 100
Remaining area using Rectangle Method: 0.592
Remaining area using Trapezoidal Method: 0.5884
```

```
n: 10
Remaining area using Rectangle Method: 0.48
Remaining area using Trapezoidal Method: 0.68
```

Problem 2.2:

Generate an $N \times N$ matrix A with uniformly distributed floating-point random numbers as its elements,

$$a_{ij} \sim U(-1, 1)$$

You are given a N-dimensional vector with "1" as its all elements:

$$b = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

Please write a program to solve the linear system of equations $AX = b$, i.e., find the unknown vector X that satisfies the given linear system of equations for $N = 66$.

Algorithm:

```
Function generate_matrix(N):
   //Create an N × N matrix A
   for each row i from 0 to N-1:
      for each column j from 0 to N-1:
         A[i][j] = Random number from U(-1,1)
   return A

Function generate_vector(N):
   Create a vector b of size N
   Set all elements of b to 1
   return b

Function gaussian_elimination(A, b):
   N = length of A

   //Forward Elimination
   for i from 0 to N-1:
      find max_row where |A[row][i]| is maximum for row >= i
      swap row i with max_row in A and b

      for each row j from i+1 to N-1:
         factor = A[j][i] / A[i][i]
         for each column k from i to N-1:
            A[j][k] = A[j][k] - factor * A[i][k]
         b[j] = b[j] - factor * b[i]

   //Back Substitution
   //Create solution vector x of size N
   For i from N-1 to 0:
      x[i] = (b[i] - Sum(A[i][j] * x[j] for j from i+1 to N-1)) / A[i][i]
   Return x
   END
```

Result:

N was set to 66, and the solution vector X was generated as follows:

```
Solution Vector X:
[2.137037070083878e+17, -4.624716180037401e+17, -1.462301444270577e+17, 2.3051836808525516e+16, 3.
1162673986228148e+16, -5.4921147648874136e+16, -1.952534927680993e+16, -1.7580840304161002e+16,
-1226984505680247.0, 1987282224111824.8, 2123488101066912.0, -1387126652618151.5, -165716608369271.
75, 1414888319941.2488, 3751501079215.346, -206800044906.12308, -3670099832284.608, -872490362057.
3536, 476757877399.0912, -656225323949.2528, -376664612219.13385, -475776247595.3444, -42286125307.
865295, 48890971536.71944, -28969747612.668144, 24011649492.62482, 37025098472.90955, -572997030.
1769811, -15728598715.791958, -321925733.7169647, -205691969.58492285, -499463338.14708936,
916440839.339496, -187232459.37252855, 241374656.2102332, 208470351.75461972, 176425709.67813864,
116323353.37075074, 79909968.85418394, -28171076.40018109, 3303409.0134767005, 7866274.760481741,
3825198.6960413763, -3807695.3886978095, -5777140.947523551, 1004673.1313431521, -666025.
4058327472, -301075.0401022864, -41815.98893381023, -33036.86248719437, -10939.029647961186, -632.
1750980390931, -433.52222704807747, -74.03447073677823, -94.61750473829163, -31.694804567800677,
121.80692330024908, -16.45357408228667, 39.28496298592831, 8.546851167724393, -18.422980410972368,
-2.27775073951504, 2.336708573003369, -1.7901041902170631, -0.4789447786616194, 1.2744500900408522]
```