Applied Computational Science

Homework 4

Anand Kamble
amk23j@fsu.edu

1st Dec 2023

Problem 1

Python script to perform trapezoid integration using Romberg method with successive refinement to evaluate the following integral to an accuracy of 10^{-6}

$$\int_0^2 x^4 log_{10}(x + \sqrt{x^2+1})dx$$

In [1]:
```
import numpy as np

def func(x):
    # Define the function to be integrated
    return x**4*np.log(x + np.sqrt(x**2 + 1))

def trapezoidal_rule(f, a, b, n):
    # Trapezoidal rule for integration
    h = (b - a) / n
    integral = 0.5 * (f(a) + f(b))
    for i in range(1, n):
        integral += f(a + i * h)
    return h * integral

def romberg_integration(f, a, b, tol=1e-6, max_iters=20):
    # Romberg integration with successive refinement
    matrix = np.zeros((max_iters, max_iters))

    for i in range(max_iters):
        matrix[i, 0] = trapezoidal_rule(f, a, b, 2**i)

        for k in range(1, i + 1):
            matrix[i, k] = (4**k * matrix[i, k - 1] - matrix[i - 1, k - 1]) / (4**k

        if i > 0 and np.abs(matrix[i, i] - matrix[i - 1, i - 1]) < tol:
            break

    return matrix[i, i]

# Define the integration interval
a = 0
b = 2

# Perform Romberg integration for the given function
result = romberg_integration(func, a, b)

print(f"The result of the integration is: {result:.6f}")
```

```
The result of the integration is: 8.153364
```

Problem 2

Evaluating the following integral using Hermite integration for n = 2, 4, 8, and 16 points.

$$\int_{-\infty}^{\infty} sinx^2 e^{-x^2+x} dx$$

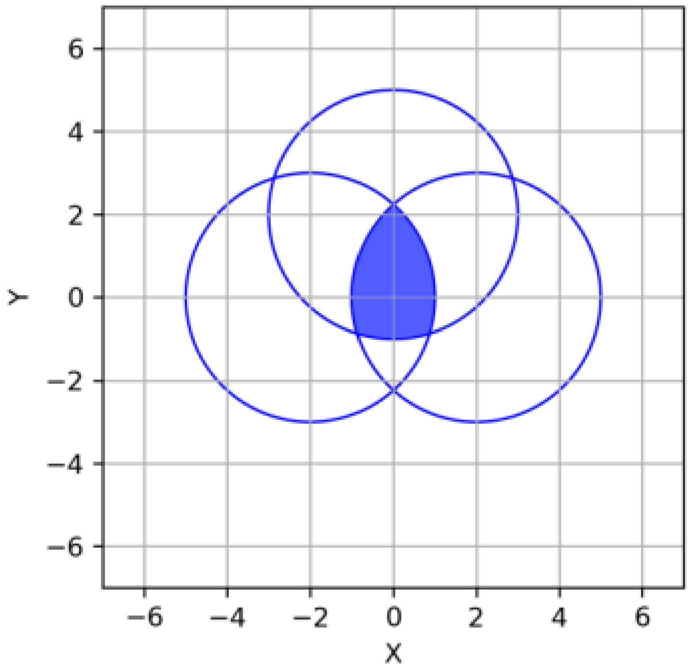
In [29]:

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.polynomial.hermite import hermgauss as hermite

f = lambda x: np.sin(x**2) * np.exp(-x**2 + x)  # define function

def hermite_integration(n):
    xi, wi = hermite(n)  # get nodes and weights for Hermite integration
    print(f"Hermite integration using n = {n}")
    print(f"----------------------------------------------------------------")
    print('nodes:', xi)
    print('weights:', wi)
    y0 = np.zeros([n, 1])

    x = np.arange(xi[0]-10, xi[n-1]+10, 0.01)  # Adjust the range for Hermite nodes
    y = np.exp(-x**2) * f(x)
    intg = np.sum(wi * f(xi) * np.exp(xi**2))
    print('Integral: ', intg)

n = [2,4,8,16]

for ni in n:
    hermite_integration(ni)
```

```
Hermite integration using n = 2
----------------------------------------------------------------
nodes: [-0.70710678  0.70710678]
weights: [0.88622693 0.88622693]
Integral:  1.0712000678636346
Hermite integration using n = 4
----------------------------------------------------------------
nodes: [-1.65068012 -0.52464762  0.52464762  1.65068012]
weights: [0.08131284 0.80491409 0.80491409 0.08131284]
Integral:  0.677009573984422
Hermite integration using n = 8
----------------------------------------------------------------
nodes: [-2.93063742 -1.98165676 -1.15719371 -0.38118699  0.38118699  1.15719371
  1.98165676  2.93063742]
weights: [1.99604072e-04 1.70779830e-02 2.07802326e-01 6.61147013e-01
 6.61147013e-01 2.07802326e-01 1.70779830e-02 1.99604072e-04]
Integral:  0.8259855772785805
Hermite integration using n = 16
----------------------------------------------------------------
nodes: [-4.68873894 -3.8694479  -3.17699916 -2.54620216 -1.95178799 -1.38025854
 -0.82295145 -0.27348105  0.27348105  0.82295145  1.38025854  1.95178799
  2.54620216  3.17699916  3.8694479   4.68873894]
weights: [2.65480747e-10 2.32098084e-07 2.71186009e-05 9.32284009e-04
 1.28803115e-02 8.38100414e-02 2.80647459e-01 5.07929479e-01
 5.07929479e-01 2.80647459e-01 8.38100414e-02 1.28803115e-02
 9.32284009e-04 2.71186009e-05 2.32098084e-07 2.65480747e-10]
Integral:  0.8358472579699888
```

Problem 3

Monte Carlo Method for Calculating Common Area of Three Circles



In [10]:
```
import matplotlib.pyplot as plt
import numpy as np

def mc(n):
    center1, center2, center3= (-2, 0),(2, 0),(0, 2)
    radius = 3
    inCommonArea = 0
    x_inside, y_inside = [], []
    x_outside, y_outside = [], []

    for _ in range(n):
        x = np.random.uniform(-5, 5)
        y = np.random.uniform(-5, 5)
        c1 = (x - center1[0])**2 + (y - center1[1])**2 <= radius**2
        c2 = (x - center2[0])**2 + (y - center2[1])**2 <= radius**2
        c3 = (x - center3[0])**2 + (y - center3[1])**2 <= radius**2

        x_inside.append(x) if (c1 and c2 and c3) else x_outside.append(x)
        y_inside.append(y) if (c1 and c2 and c3) else y_outside.append(y)

        if c1 and c2 and c3:
            inCommonArea += 1

    CommonArea = inCommonArea / n * 100
    return CommonArea, x_inside, y_inside, x_outside, y_outside

num_samples_list = [100, 500, 1000, 5000, 10000, 20000]

num_rows, num_cols = 2, 3
fig, axes = plt.subplots(num_rows, num_cols, figsize=(12, 8))

for i, num_samples in enumerate(num_samples_list):
    CommonArea, x_inside, y_inside, x_outside, y_outside = mc(num_samples)
```

```python
    print(f"Number of samples: {num_samples}, Commmon area estimate: {CommonArea:.2

    row = i // num_cols
    col = i % num_cols

    axes[row, col].scatter(x_inside, y_inside, color='blue', label='Inside Common A
    axes[row, col].scatter(x_outside, y_outside, color='red', label='Outside Commor
    axes[row, col].set_title(f'{num_samples} Samples')
    axes[row, col].legend()

plt.tight_layout()
plt.show()
```

```
Number of samples: 100, Commmon area estimate: 4.00%
Number of samples: 500, Commmon area estimate: 4.60%
Number of samples: 1000, Commmon area estimate: 5.50%
Number of samples: 5000, Commmon area estimate: 4.78%
Number of samples: 10000, Commmon area estimate: 4.66%
Number of samples: 20000, Commmon area estimate: 5.04%
```