Lab 04 - Numerical Integration Using General Quadratic Formula

Objective

1. Numerical Integration using Trapezoidal rule.
2. Numerical Integration using Simpson $\frac{1}{3}$ rule.
3. Numerical Integration using Simpson $\frac{3}{8}$ rule.

Theory

Trapezoidal Rule

The Trapezoidal Rule approximates the area under a curve by dividing it into n equal subintervals and summing the areas of the resulting trapezoids. For a function $f(x)$ over $[a, b]$:

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right]$$

where $h = \frac{b-a}{n}$ and $x_i = a + ih$.

Simpson's $\frac{1}{3}$ Rule

Simpson's $\frac{1}{3}$ Rule uses parabolic arcs to approximate the area under a curve. It requires an even number of subintervals (n even):

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4\sum_{\text{odd } i=1}^{n-1} f(x_i) + 2\sum_{\text{even } i=2}^{n-2} f(x_i) + f(x_n)\right]$$

where $h = \frac{b-a}{n}$.

Simpson's $\frac{3}{8}$ Rule

Simpson's $\frac{3}{8}$ Rule uses cubic interpolation and requires the number of subintervals n to be a multiple of 3:

$$\int_a^b f(x)\,dx \approx \frac{3h}{8}\left[f(x_0) + 3\sum_{\substack{i=1 \\ i\not\equiv 0 \pmod 3}}^{n-1} f(x_i) + 2\sum_{\substack{i=3 \\ i\equiv 0 \pmod 3}}^{n-3} f(x_i) + f(x_n)\right]$$

where $h = \frac{b-a}{n}$.

Algorithms

Trapezoidal Rule Algorithm

1. Input: function $f(x)$, limits a, b, and number of subintervals n.
2. Compute $h = \frac{b-a}{n}$.
3. Initialize sum: $S = f(a) + f(b)$.
4. For $i = 1$ to $n - 1$:
   - Compute $x_i = a + i \cdot h$.
   - Update $S = S + 2 \cdot f(x_i)$.
5. Compute result: $I = \frac{h}{2} \cdot S$.
6. Output I.

Simpson's $\frac{1}{3}$ Rule Algorithm

1. Input: function $f(x)$, limits a, b, and even number of subintervals n.
2. Compute $h = \frac{b-a}{n}$.
3. Initialize sum: $S = f(a) + f(b)$.
4. For $i = 1$ to $n - 1$:
   - Compute $x_i = a + i \cdot h$.
   - If i is odd, update $S = S + 4 \cdot f(x_i)$.
   - If i is even, update $S = S + 2 \cdot f(x_i)$.
5. Compute result: $I = \frac{h}{3} \cdot S$.
6. Output I.

Simpson's $\frac{3}{8}$ Rule Algorithm

1. Input: function $f(x)$, limits a, b, and number of subintervals n (multiple of 3).
2. Compute $h = \frac{b-a}{n}$.
3. Initialize sum: $S = f(a) + f(b)$.
4. For $i = 1$ to $n - 1$:
   - Compute $x_i = a + i \cdot h$.
   - If $i \bmod 3 \neq 0$, update $S = S + 3 \cdot f(x_i)$.
   - If $i \bmod 3 = 0$, update $S = S + 2 \cdot f(x_i)$.
5. Compute result: $I = \frac{3h}{8} \cdot S$.
6. Output I.

Source Code

```c
#include <stdio.h>
#include <math.h>

# define N 1000 // Number of subintervals for numerical integration

// Function to calculate the integral of a function using the trapezoidal rule
double trapezoidal_rule(double (*f)(double), double a, double b, int n) {
    double h = (b - a) / n;
    double sum = 0.5 * (f(a) + f(b));

    for (int i = 1; i < n; i++) {
        sum += f(a + i * h);
    }

    return sum * h;
}

// Function to calculate the integral of a function using Simpson's 1/3 rule
double simpsons_13_rule(double (*f)(double), double a, double b, int n) {
    if (n % 2 != 0) {
        n++; // n must be even for Simpson's rule
    }
    double h = (b - a) / n;
    double sum = f(a) + f(b);

    for (int i = 1; i < n; i++) {
        if (i % 2 == 0) {
            sum += 2 * f(a + i * h);
        } else {
            sum += 4 * f(a + i * h);
        }
    }

    return sum * h / 3;
}

// Function to calculate the integral of a function using Simpson's 3/8 rule
double simpsons_38_rule(double (*f)(double), double a, double b, int n) {
    if (n % 3 != 0) {
        n += 3 - (n % 3); // n must be a multiple of 3 for Simpson's 3/8 rule
    }
    double h = (b - a) / n;
    double sum = f(a) + f(b);

    for (int i = 1; i < n; i++) {
        if (i % 3 == 0) {
            sum += 2 * f(a + i * h);
        } else {
            sum += 3 * f(a + i * h);
        }
    }

    return sum * h * 3 / 8;
}

// Example function to integrate
double example_function(double x) {
    return 1 / x; // Example: f(x) = 1/x, note this function is not defined at x = 0
}

// Main function
int main() {
    double a = 1.0; // Lower limit of integration
    double b = 2.0; // Upper limit of integration
    int n = N;    // Number of subintervals
```

```c
    // Calculate the integral using the all rules
    double trapezoidal_result = trapezoidal_rule(example_function, a, b, n);
    double simpson_13_result = simpsons_13_rule(example_function, a, b, n);
    double simpson_38_result = simpsons_38_rule(example_function, a, b, n);

    // Print the result
    printf("The integral of f(x) from %.2f to %.2f is approximately:\n", a, b);
    printf("1. Trapezoidal Rule: %.6f\n", trapezoidal_result);
    printf("2. Simpson 1/3 Rule: %.6f\n", trapezoidal_result);
    printf("3. Simpson 3/8 Rule: %.6f\n", trapezoidal_result);

    return 0;
}
```

Output:

```
1. Trapezoidal Rule: 0.693147
2. Simpson 1/3 Rule: 0.693147
3. Simpson 3/8 Rule: 0.693147
```

Discussion

The results obtained from the three numerical integration methods: Trapezoidal Rule, Simpson's $\frac{1}{3}$ Rule, and Simpson's $\frac{3}{8}$ Rule are nearly identical for the given example function $f(x) = 1/x$ over the interval $[1, 2]$. This is expected because the function is smooth and the number of subintervals ($n = 1000$) is sufficiently large, reducing the approximation error for all methods.

Simpson's rules generally provide higher accuracy than the Trapezoidal Rule for the same number of subintervals, especially for functions that can be well-approximated by polynomials of degree two or three over small intervals. However, as n increases, the difference in accuracy between the methods diminishes. The choice of method may depend on the required precision and computational resources.

Conclusion

This lab demonstrated the implementation and application of three common numerical integration techniques. All methods produced consistent results for the test case, validating their correctness. Understanding the strengths and limitations of each method is essential for selecting the appropriate approach in practical scenarios, particularly when dealing with functions that are not easily integrable analytically.

Name: Kushal Prasad Joshi
Roll No: 230345
Faculty: Science and Technology
Program: BE Computer
Group: B