Lab 06 - Romberg's Integration

Objective

1. Implement and test Romberg's Integration in C

Theory

Romberg integration is a numerical technique for approximating definite integrals with high accuracy. It builds upon the trapezoidal rule by applying Richardson extrapolation recursively to eliminate error terms and accelerate convergence.

The process begins by computing a sequence of trapezoidal approximations with successively halved step sizes. These approximations are organized in a triangular array, where each new row refines the previous estimates. Richardson extrapolation is then used to combine these values, systematically reducing the error at each level.

Mathematically, the Romberg method constructs a table $R_{i,j}$, where:

- $R_{i,0}$ is the trapezoidal rule estimate with 2^i intervals,
- $R_{i,j} = R_{i,j-1} + \frac{R_{i,j-1} - R_{i-1,j-1}}{4^j - 1}$ for $j > 0$.

The final value $R_{n,n}$ provides a highly accurate estimate of the integral, often converging much faster than using the trapezoidal or Simpson's rule alone.

Algorithm

1. **Initialize**: Set the integration limits a and b, and choose the number of levels n.
2. **Compute Trapezoidal Approximations**:
   - For each level i from 0 to $n-1$:
     - Divide the interval $[a, b]$ into 2^i subintervals.
     - Calculate $R_{i,0}$ using the trapezoidal rule with these subintervals.
3. **Apply Richardson Extrapolation**:
   - For each i from 1 to $n-1$:
     - For each j from 1 to i:
       - Compute $R_{i,j} = R_{i,j-1} + \frac{R_{i,j-1} - R_{i-1,j-1}}{4^j - 1}$.
4. **Result**: The most accurate estimate is $R_{n-1,n-1}$.

Implementation

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

// Define the function to integrate here
double f(double x) {
    // Example: integrate sin(x)
    return sin(x);
}

// Romberg integration function
double romberg(double (*func)(double), double a, double b, int n) {
    double R[n][n];
    int i, j, k;
    double h = b - a;

    // R[i][0]
    for (i = 0; i < n; i++) {
        int N = 1 << i; // 2^i
        double sum = 0.0;
        double step = h / N;
        for (k = 1; k < N; k += 2) {
            sum += func(a + k * step);
        }
        if (i == 0)
            R[i][0] = (func(a) + func(b)) * h / 2.0;
        else
            R[i][0] = 0.5 * R[i-1][0] + sum * step;
    }

    // Richardson extrapolation
    for (i = 1; i < n; i++) {
        for (j = 1; j <= i; j++) {
            R[i][j] = R[i][j-1] + (R[i][j-1] - R[i-1][j-1]) / (pow(4, j) - 1);
        }
    }

    return R[n-1][n-1];
}

int main() {
    double a, b;
    int n;

    printf("Enter lower limit a: ");
    scanf("%lf", &a);
    printf("Enter upper limit b: ");
    scanf("%lf", &b);
    printf("Enter number of levels (e.g., 4 or 5): ");
    scanf("%d", &n);

    double result = romberg(f, a, b, n);
    printf("Romberg integration result: %.10lf\n", result);

    return 0;
}
```

Output

```
Enter lower limit a: 0
Enter upper limit b: 1
Enter number of levels (e.g., 4 or 5): 5
Romberg integration result: 0.4596976941
```

Discussion

Romberg's integration is a powerful method for numerical integration, especially when high accuracy is required. By systematically applying Richardson extrapolation to the trapezoidal rule, it accelerates the convergence of the integral's approximation. This means that, for smooth functions, Romberg's method can achieve very precise results with relatively few function evaluations compared to basic methods like the trapezoidal or Simpson's rule.

However, the method assumes that the function being integrated is sufficiently smooth over the interval. If the function has discontinuities or is not well-behaved, the convergence may be slow or the results may be inaccurate. Additionally, since Romberg's method requires storing a triangular array of intermediate results, its memory usage grows with the number of levels chosen.

In practice, Romberg integration is well-suited for problems where the integrand is smooth and the cost of function evaluation is not prohibitive. It is widely used in scientific computing and engineering applications where reliable and accurate integration is essential.

Conclusion

Romberg's integration offers a systematic and efficient approach to numerical integration, leveraging the strengths of the trapezoidal rule and Richardson extrapolation. Its rapid convergence and high accuracy make it a valuable tool for evaluating definite integrals, particularly when the integrand is smooth. Understanding and implementing Romberg's method equips students and practitioners with a robust technique for tackling a wide range of integration problems in computational mathematics.

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