The Trapezoidal Rule

1 Introduction

Numerical integration is a fundamental technique in computational mathematics, enabling the approximation of definite integrals. The trapezoidal rule is one of the most basic methods, providing an efficient way to estimate the integral by approximating the region under the curve as a series of trapezoids.

2 Mathematical Explanation

The trapezoidal rule approximates the definite integral of a function $f(x)$ on the interval $[a, b]$ as follows:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}\left(f(a)+f(b)\right).$$

For subdividing the interval into n subintervals, the approximation becomes:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{n}\left(\frac{f(x_0)}{2}+f(x_1)+f(x_2)+\ldots+f(x_{n-1})+\frac{f(x_n)}{2}\right),$$

where $x_i = a + i\frac{b-a}{n}$ for $i = 0, \ldots, n$.

3 Implementation in Python

Below is an example of how the trapezoidal rule can be implemented in Python:

```python
def trapezoidal_rule(f, a, b, n):
    h = (b - a) / n
    result = 0.5 * (f(a) + f(b))
    for i in range(1, n):
        result += f(a + i * h)
    result *= h
    return result

# Example usage
def func(x):
    return x**2

integral = trapezoidal_rule(func, 0, 1, 100)
print("Approximate integral:", integral)
```

4 Implementation in MATLAB

Below is how the trapezoidal rule can be implemented in MATLAB:

```
function integral = trapezoidal_rule(f, a, b, n)
    h = (b - a) / n;
    result = 0.5 * (f(a) + f(b));
    for i = 1:n-1
        result = result + f(a + i * h);
    end
    integral = h * result;
end

% Example usage
f = @(x) x.^2;
integral = trapezoidal_rule(f, 0, 1, 100);
disp(['Approximate integral: ', num2str(integral)]);
```

5 Computed Example

Let's compute the integral of $f(x) = x^2$ over the interval $[0, 1]$ using the trapezoidal rule with 4 subdivisions.

5.1 Manual Computation

To apply the trapezoidal rule manually to approximate the integral of $f(x) = x^2$ over the interval $[0, 1]$ with 4 subdivisions, we follow these steps:

1. **Determine the subdivision points:**
   The interval $[0, 1]$ is divided into 4 equal subintervals. Thus, each subinterval has a width $h = \frac{1-0}{4} = 0.25$.
   The points are:
$$x_0 = 0, \quad x_1 = 0.25, \quad x_2 = 0.5, \quad x_3 = 0.75, \quad x_4 = 1$$

2. **Calculate function values:**

$$f(x_0) = f(0) = 0^2 = 0$$
$$f(x_1) = f(0.25) = (0.25)^2 = 0.0625$$
$$f(x_2) = f(0.5) = (0.5)^2 = 0.25$$
$$f(x_3) = f(0.75) = (0.75)^2 = 0.5625$$
$$f(x_4) = f(1) = 1^2 = 1$$

3. **Apply the trapezoidal rule formula:**
   The trapezoidal rule approximation is given by:
$$\int_0^1 x^2 \, dx \approx \frac{b-a}{n} \left(\frac{f(x_0)}{2} + f(x_1) + f(x_2) + f(x_3) + \frac{f(x_4)}{2} \right)$$

   Substituting the values:

$$\int_0^1 x^2 \, dx \approx \frac{0.25}{2} \cdot (0 + 2 \times 0.0625 + 2 \times 0.25 + 2 \times 0.5625 + 2 \times 1)$$
$$= 0.125 \cdot (0 + 0.125 + 0.5 + 1.125 + 2)$$
$$= 0.125 \cdot (3.75)$$
$$= 0.46875$$

   Therefore, the approximate value of the integral using the trapezoidal rule with 4 subdivisions is 0.46875.

5.2 Python Implementation

```python
import numpy as np
import matplotlib.pyplot as plt

def trapezoidal_rule(f, a, b, n):
    x = np.linspace(a, b, n+1)
    y = f(x)
    h = (b - a) / n
    integral = h * (np.sum(y) - 0.5 * (y[0] + y[-1]))
    return integral

def func(x):
    return x**2

# Parameters
a, b = 0, 1
n = 4

# Calculate integral
integral = trapezoidal_rule(func, a, b, n)
print("Approximate integral:", integral)
```

5.3 MATLAB Implementation

```matlab
f = @(x) x.^2;
a = 0;
b = 1;
n = 4;
h = (b - a) / n;

x = linspace(a, b, n+1);
y = f(x);
integral = h * (sum(y) - 0.5 * (y(1) + y(end)));

disp(['Approximate integral: ', num2str(integral)]);
```