
Assignment 3: Numerical Integration using Midpoint Riemann Sums

Math4AI Program: Integral Calculus

Fatima Alibabayeva
fatime.elibabayeva25@aiaacademy.az

National AI Academy
November 30, 2025

Contents

1	Introduction	3
2	Mathematical Foundations	3
2.1	Midpoint Riemann Sum (M_N)	3
3	Implementation and Methodology	3
3.1	Objective	3
3.2	Methodology Details	4
4	Results and Analysis	4
4.1	Experiments	4
4.2	Observations	4
4.3	Discussion	5
5	Challenges and Insights	5
6	Conclusion	5

1 Introduction

This assignment focuses on Numerical Integration through the Midpoint Riemann Sum method. This technique is computationally vital in machine learning for calculating expected values, normalizing probability distributions, and accurately estimating areas when analytical integration is intractable. The assignment requires implementing this method from scratch and verifying its accuracy against a known integral.

2 Mathematical Foundations

The definite integral $\int_a^b f(x)dx$ is approximated by dividing the interval $[a, b]$ into N equal subintervals, each with a width $\Delta x = \frac{b-a}{N}$.

2.1 Midpoint Riemann Sum (M_N)

The Midpoint Rule approximates the integral by taking the height of each rectangle at the midpoint of its corresponding subinterval, \bar{x}_i . This method typically yields a significantly more accurate result compared to the Left or Right Sums for the same number of subintervals N .

The formula for the Midpoint Riemann Sum is:

$$M_N = \sum_{i=1}^N f(\bar{x}_i)\Delta x$$

where the midpoint \bar{x}_i for the i -th subinterval (starting from $i = 1$) is defined as:

$$\bar{x}_i = a + \left(i - \frac{1}{2}\right) \Delta x$$

3 Implementation and Methodology

3.1 Objective

The objective is to implement the function `riemann_integral(f, a, b, n)` to calculate the definite integral using the Midpoint Riemann Sum method.

3.2 Methodology Details

The implementation followed the geometric definition of the integral. First, the step size $\Delta x = (b - a)/n$ was calculated. Then, a loop iterated N times, calculating the midpoint \bar{x}_i for each rectangle using the formula $x_i = a + (i + 0.5) \cdot \Delta x$ (where i is the 0-based index). The function $f(\bar{x}_i)$ was evaluated at this midpoint to determine the height. Finally, the total sum of all rectangle areas was computed by multiplying the accumulated heights by Δx .

4 Results and Analysis

4.1 Experiments

The Midpoint Riemann Sum method was tested on the Standard Normal Probability Density Function, $\phi(x)$, over the range $a = -1.0$ to $b = 1.0$. This range corresponds to finding the probability within one standard deviation of the mean. The number of subintervals used was $N = 1000$.

The function used is:

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

The numerical result obtained from the implemented function was then compared against the high-precision numerical integration standard provided by the SciPy library (`spi.quad()`).

4.2 Observations

The integral values for the Standard Normal PDF over the interval $[-1.0, 1.0]$ were computed using the implemented Midpoint Riemann Sum and verified against the high-precision SciPy reference. The parameters used were a lower bound $a = -1.0$, upper bound $b = 1.0$, and the number of rectangles $N = 1000$.

- **Implemented Midpoint Riemann Sum (M_N)**:`integral_scratch` value 0.6826894921
- **SciPy Reference (High Precision)**:`integral_scipy` value 0.6826894921

The numerical approximation achieved by the scratch implementation shows excellent agreement with the SciPy reference, confirming the accuracy of the Midpoint Rule for this function. The absolute difference between the two values is negligible, demonstrating the stability of the method with $N = 1000$ steps.

4.3 Discussion

The result obtained from the implemented Midpoint Riemann Sum demonstrates exceptional precision, matching the high-fidelity SciPy reference value up to several decimal places. The expected analytical probability value for this integral is approximately 0.682689.

This high level of agreement confirms the following:

1. Method Accuracy: The Midpoint Riemann Sum method is highly accurate and converges quickly, even with a moderate number of steps ($N = 1000$), validating its theoretical error rate ($\mathcal{O}(\Delta x^2)$).
2. Implementation Correctness: The scratch implementation correctly calculates the subinterval midpoints and accumulates the area according to the mathematical formula.
3. Relevance: The successful integration highlights the importance of numerical integration in computational statistics and AI, where calculating probabilities and expectation values is routine.

5 Challenges and Insights

A key insight gained is the power of the Midpoint Rule for fast convergence. While the underlying implementation is simple, achieving high accuracy with minimal computational cost is vital. The challenge lies in ensuring precision over a large number of iterations without introducing excessive accumulated floating-point error. This assignment solidifies the understanding that the numerical integral is the limiting process of a sum, a cornerstone of continuous probability modeling in machine learning.

6 Conclusion

The assignment successfully implemented the Midpoint Riemann Sum method and applied it to calculate the integral. The results were highly accurate when verified against the SciPy standard. This confirms a foundational understanding of numerical integration, a necessary tool for handling complex functions and modeling probability distributions in advanced AI systems.