QUEEN'S UNIVERSITY BELFAST, SCHOOL OF MATHS & PHYSICS – MTH3023

Integrating functions with infinite derivatives using Romberg integration

Jyoutir 40360629

December 3, 2024

1 INTRODUCTION

Efficient calculation of definite integrals is important in science and engineering. There are many techniques to do this, one of which is Romberg integration, which is an extrapolation method built on the trapezoidal rule. It is often favoured for its rapid convergence on smooth functions. However, its effectiveness diminishes when applied to integrands with singularities. In this report, we examine the performance of Romberg integration on several integrals:

$$I_1 = \int_0^1 x^{1/4}\,\mathrm{d}x, \quad I_2 = \int_0^1 4y^4\,\mathrm{d}y, \quad \text{and} \quad I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,\mathrm{d}x.$$

I_1 poses a challenge due to the singularity at $x = 0$, slowing the convergence rate of Romberg integration. We applied a variable transformation to convert it into the smooth integral I_2, making it straightforward for Romberg integration, which uses the trapezoidal rule. We then extended this to the integral I_3, which has strong singularities at both endpoints. We found that Romberg integration was ineffective for approximating a solution within a reasonable error bound due to extremely slow convergence.

To address this, we used Gauss-Jacobi quadrature, a method designed specifically for integrals with endpoint singularities [1]. By using weight functions that match the integrand's singularities, we achieved highly accurate results for I_3, where Romberg integration struggled, showing importance of selecting the right numerical technique for efficient computation.

In the following Theory and Methods section, we discuss the theory of Romberg integration and the algorithm we implemented from Burden and Faires [2, pp. 211–217], concluding the section with Romberg's limitations and how we addressed slow convergence by using a variable transformation from I_1 into I_2.

In the Results section, we verify the Romberg integration algorithm on I_1 and I_2 and compare its performance on the given integrals, and we extend the work by applying Gauss-Jacobi quadrature to the challenging integral $I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,dx$. This report demonstrates the importance of selecting appropriate numerical techniques based on the characteristics of the integrand for different tasks, ensuring precise and efficient computation.

2 Theory and Methods

This section covers Romberg integration, focusing on its derivation, algorithm, limitations, and efficiency improvements, forming the basis of our findings. Specifically we explore $\int_0^1 x^{1/4}\,dx$, which has an unbounded derivative at $x = 0$, making the trapezoidal rule inefficient. Using the substitution $x = y^4$ we demonstrate that the integral becomes $\int_0^1 4y^3\,dy$ - a smooth integrand with bounded derivatives.

2.1 The Trapezoidal Rule

The trapezoidal rule is a numerical method for approximating definite integrals by approximating the integrand $f(x)$ with piecewise linear functions over the interval $[a, b]$. Specifically, on each subinterval $[x_i, x_{i+1}]$, $f(x)$ is approximated using first-degree Lagrange interpolating polynomial based on values at the endpoints. Integrating this linear interplant over every subinterval gives the composite trapezoidal rule:

$$I = \int_a^b f(x)\,\mathrm{d}x \approx T(h) = \frac{h}{2}\left[f(a) + 2\sum_{i=1}^{n-1} f(a+ih) + f(b)\right], \tag{1}$$

where n is the number of subintervals and $h = \frac{b-a}{n}$ is the width of each subinterval.

The error associated with composite trapezoidal rule [3] can be derived using the remainder term from polynomial interpolation and is given by:

$$E_T(h) = -\frac{(b-a)h^2}{12} f''(\xi), \quad \xi \in [a, b], \tag{2}$$

which shows that the error decreases quadratically with h as $h \to 0$, demonstrating the method's convergence properties.

2.2 Richardson Extrapolation

Richardson extrapolation improves numerical approximations by eliminating leading-order error terms. Applied to the composite trapezoidal rule, we compute approximations $R_{1,1} = T(h)$ and $R_{2,1} = T\left(\frac{h}{2}\right)$ with step sizes h and $\frac{h}{2}$, respectively. An improved estimate is obtained using:

$$R_{2,2} = R_{2,1} + \frac{R_{2,1} - R_{1,1}}{4-1} = \frac{4R_{2,1} - R_{1,1}}{3}. \tag{3}$$

This combination cancels the $O(h^2)$ error term, giving an approximation with error $O(h^4)$. By recursively applying this process, higher-order approximations are achieved via:

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}, \quad j = 2, 3, \dots, k. \tag{4}$$

$R_{k,j}$ represents the approximation at refinement level k and extrapolation order j. This method increases accuracy beyond what the composite trapezoidal rule could achieve on its own, forming foundation of Romberg integration.

2.3 Romberg Integration

Romberg integration combines the trapezoidal rule and Richardson extrapolation to achieve high-order accuracy in numerical integration. By using the trapezoidal rule with successively halved step sizes and recursively applying Richardson extrapolation, Romberg integration removes lower-order error terms, accelerating convergence towards the exact value of the integral and allowing efficient and precise approximations.

2.3.1 Romberg Integration Algorithm

Following Burden and Faires, Algorithm 4.2 [2, pp. 216–217] the Romberg integration algorithm constructs a triangular array of approximations $R_{k,j}$ that converge rapidly to the exact value of the integral $I = \int_a^b f(x)\,\mathrm{d}x$. The structure of the Romberg integration table is illustrated in Table 1.

Table 1: Structure of Romberg Integration Table

$k \backslash j$	1	2	$\cdots$	k
1	$R_{1,1}$			
2	$R_{2,1}$	$R_{2,2}$		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
k	$R_{k,1}$	$R_{k,2}$	$\cdots$	$R_{k,k}$

Where $R_{k,1} = T(h_k)$ are approximations using the composite trapezoidal rule with step size $h_k = \dfrac{b-a}{2^{k-1}}$, and $R_{k,j}$ for $j > 1$ are improved approximations using Richardson extrapolation to remove lower-order errors.

Algorithm Overview

1. Compute $R_{1,1}$ using the trapezoidal rule with one interval ($n_1 = 1$):

$$R_{1,1} = T(h_1) = \frac{h_1}{2}\left[f(a) + f(b)\right],$$

   where $h_1 = b - a$.

2. For $k = 2$ to n:

   - Compute $R_{k,1}$ using the composite trapezoidal rule with $n_k = 2^{k-1}$ subintervals of width $h_k = \dfrac{b-a}{n_k}$:

$$R_{k,1} = T(h_k) = \frac{h_k}{2}\left[f(a) + 2\sum_{i=1}^{n_k-1} f(a + ih_k) + f(b)\right].$$

   - For $j = 2$ to k, refine using Richardson extrapolation:

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}. \tag{5}$$

Notation

- $R_{k,j}$: Approximation at refinement level k, extrapolation order j.
- k: Refinement level ($k = 1, 2, \dots, n$).
- j: Extrapolation order ($j = 1, 2, \dots, k$).
- h_k: Step size at level k, $h_k = \dfrac{b-a}{2^{k-1}}$.
- n_k: Number of subintervals at level k, $n_k = 2^{k-1}$.

2.3.2 Advantages and Limitations of Romberg Integration

Romberg integration efficiently computes integrals by combining the trapezoidal rule with Richardson extrapolation, achieving high accuracy with relatively few function evaluations due to the exponential convergence of the error term $O(h^{2k})$ after k levels of extrapolation. This makes it excellent for smooth functions with bounded derivatives. However, for integrands with unbounded derivatives or singularities at the endpoints, such as $f(x) = x^{1/4}$ at $x = 0$, the error in the trapezoidal rule can become significant (see Equation (2)), reducing Romberg integration's effectiveness. To address this issue, a variable transformation can smooth the integrand, ensuring bounded higher-order derivatives.

2.3.3 Variable Transformation

Consider the function $f(x) = x^{1/4}$ on the interval $[0, 1]$. Direct application of Romberg integration may yield poor results due to the infinite derivative at $x = 0$. Indeed, the first derivative tends to infinity as $x \to 0^+$:

$$f'(x) = \frac{1}{4}x^{-3/4} = \frac{1}{4x^{3/4}} \qquad \Longrightarrow \qquad \lim_{x\to 0^+} f'(x) = \infty.$$

Similarly, the second derivative tends to negative infinity as $x \to 0^+$:

$$f''(x) = -\frac{3}{16}x^{-7/4} = -\frac{3}{16x^{7/4}} \quad \Longrightarrow \quad \lim_{x\to 0^+} f''(x) = -\infty.$$

Since the second derivative diverges near $x = 0$, the error bound in the trapezoidal rule, which depends on $f''(x)$, becomes unbounded, as stated in Equation (2). To address this issue, we perform the substitution $x = y^4$, mapping $y \in [0, 1]$ onto $x \in [0, 1]$. The integral transforms as:

$$\begin{aligned} I &= \int_0^1 x^{1/4}\, dx = \int_0^1 (y^4)^{1/4} \cdot \frac{dx}{dy}\, dy && (6) \\ &= \int_0^1 y \cdot 4y^3\, dy = \int_0^1 4y^4\, dy. && (7) \end{aligned}$$

The transformed integrand $g(y) = 4y^4$ is smooth with continuous derivatives on $[0, 1]$. Its second derivative is:

$$g''(y) = \frac{\mathrm{d}^2}{\mathrm{d}y^2}\left(4y^4\right) = 48y^2,$$

which is finite and bounded for $y \in [0, 1]$. Specifically, as $y \to 0^+$ and $y \to 1^-$:

$$\lim_{y\to 0^+} g''(y) = 0, \qquad \lim_{y\to 1^-} g''(y) = 48.$$

Therefore, $g''(y)$ remains bounded on the interval $[0, 1]$. According to Equation (2), the error in the trapezoidal rule depends on $g''(\xi)$. By applying this transformation, we mitigate the issues caused by the unbounded derivatives of the original function at $x = 0$, allowing accurate and efficient computation of I using Romberg integration.

3 Results and Discussion

This section presents numerical results obtained from implementing the algorithm, theory and methods detailed in the pervious section. We use Romberg integration algorithm then analyse convergence behaviour. Lastly, comparing with Guass-Jacobi quadrature to show limitations on specific integrals.

3.1 Verification of Algorithm

To validate accuracy of our Romberg integration algorithm, implemented based on Burden and Faires as detailed in Algorithm 4.2 , we applied it to the integral $\int_0^{\pi} \sin(x)\,dx$ [2, pp. 215–216], replicating Example 4.3. Our computed results closely match those from the textbook, confirming the correctness of our algorithm's implementation. Specifically, our final approximation $R_{4,4} = 2.00000555$ agrees with the exact value 2 to within 5.55×10^{-6}, showing the reliability of the implemented Romberg integration algorithm.

3.2 Convergence and Error Analysis for $\int_0^1 x^{1/4}\,dx$ and $\int_0^1 4y^4\,dy$

I applied the Romberg integration algorithm to compute the approximations for $\int_0^1 x^{1/4}\,dx$ and $\int_0^1 4y^4\,dy$, with each approximation $R_{k,k}$ representing the most accurate estimate at iteration step k, as explained in Section 2.3.1. The approximations are obtained from the diagonal elements of the Romberg integration table. To analyse the convergence of Romberg integration and provide visual insight for these functions, we plotted the absolute error $|I - R_{k,k}|$ against the iteration steps k in Figure 1.

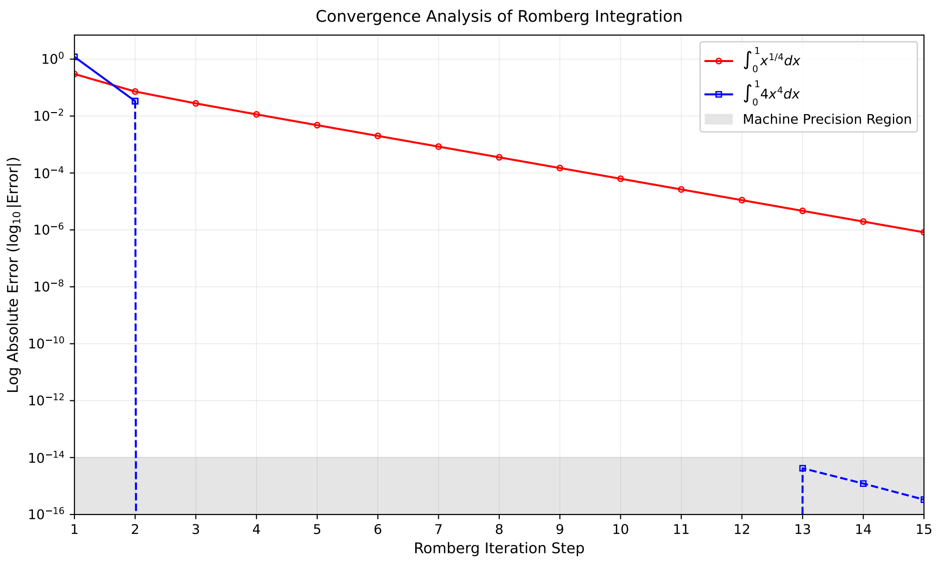

Figure 1: Absolute error $|I - R_{k,k}|$ against Romberg iteration steps for $I_1 = \int_0^1 x^{1/4}\,dx$ (red) and $I_2 = \int_0^1 4y^4\,dy$ (blue), along with machine precision region (grey)

In Figure 1, we observe that for I_2, the error decreases exponentially, reaching the exact value $I = 0.8$ by step 2 ($R_{2,2}$), due to the smoothness of $f(y) = 4y^4$, which allows affective extrapolation. But, for I_1, convergence is much slower because of the singularity at $x = 0$ in $f(x) = x^{1/4}$, where the derivative becomes infinite. Although the algorithm was run up to step 15, the approximation $R_{14,14} = 0.79999918$ still had not achieved the same level of accuracy as I_2 did at step 2, showing limitations of Romberg integration for functions with endpoint singularities. Fluctuations beyond step 13 for I_2 arise from round-off errors near machine precision [4].

To further quantify convergence behaviour, the empirical order of convergence p_k was calculated to measure the rate at which errors decrease across iterations:

$$p_k = \frac{\log|R_{k,k} - R_{k-1,k-1}| - \log|R_{k+1,k+1} - R_{k,k}|}{\log 2}, \tag{8}$$

where $R_{k,k}$ is the Romberg estimate at iteration k. This equation is derived by assuming the error behaves like $E_k \propto h_k^p$ with $h_k = \dfrac{h_0}{2^{k-1}}$; taking logarithms and using errors at successive steps allows us to solve for p.

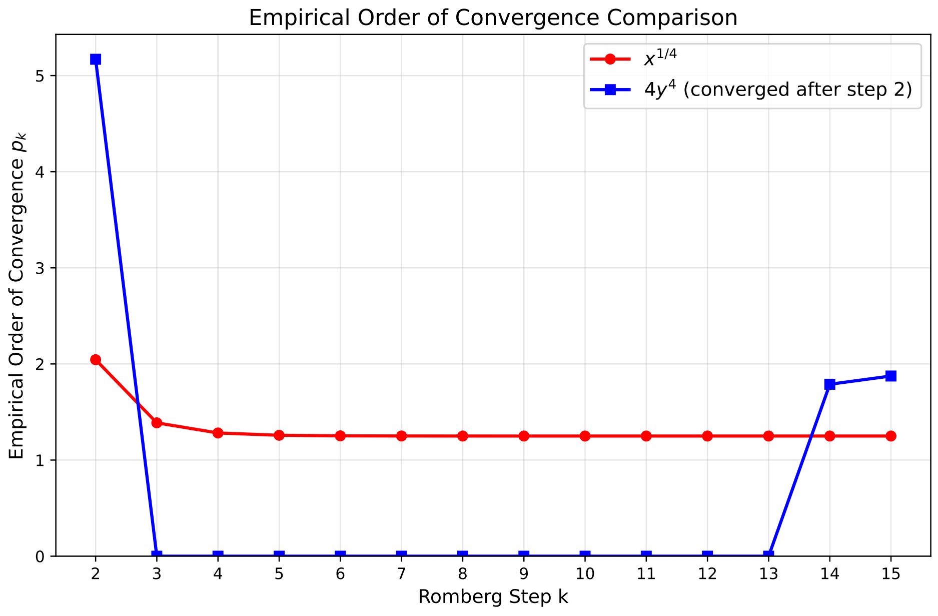

Figure 2: Empirical order of convergence (p_k) vs. Romberg iteration steps for $I_1 = \int_0^1 x^{1/4}\,dx$ (red) and $I_2 = \int_0^1 4y^4\,dy$ (blue).

- From Figure 2, we observe that for I_1, the empirical order converges slowly due to the singularity at $x = 0$ in $f(x) = x^{1/4}$, reducing the efficiency of the extrapolation process. With more Romberg steps, the order of convergence decreases by increasingly smaller increments.

- For I_2, empirical order drops rapidly to 0 by Romberg step 3, showing high efficiency of Romberg integration for smooth functions. Higher-order extrapolation terms are effectively used, showing benefits of variable transformation in this case.

3.3 Limitations of Romberg Integration

Although Romberg integration performs well for smooth functions, it struggles with integrands that exhibit singularities at the endpoints. Consider the integral:

$$I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,dx. \tag{9}$$

The integrand in Equation (9) has integrable singularities at both $x = 0$ and $x = 1$, where $x^{-0.9}$ and $(1-x)^{-0.9}$ diverge. While these singularities are integrable, they make the function unbounded near the endpoints, posing challenges for numerical integration methods.

Figure 3 shows Romberg integration's inefficiency in computing I_3 (Equation 9) effectively. The error for I_3 remains significant due to the singularities, even after running 36 iterations, which took 690 minutes, the error is still greater than 0.1, showing the impracticality of using Romberg integration for I_3

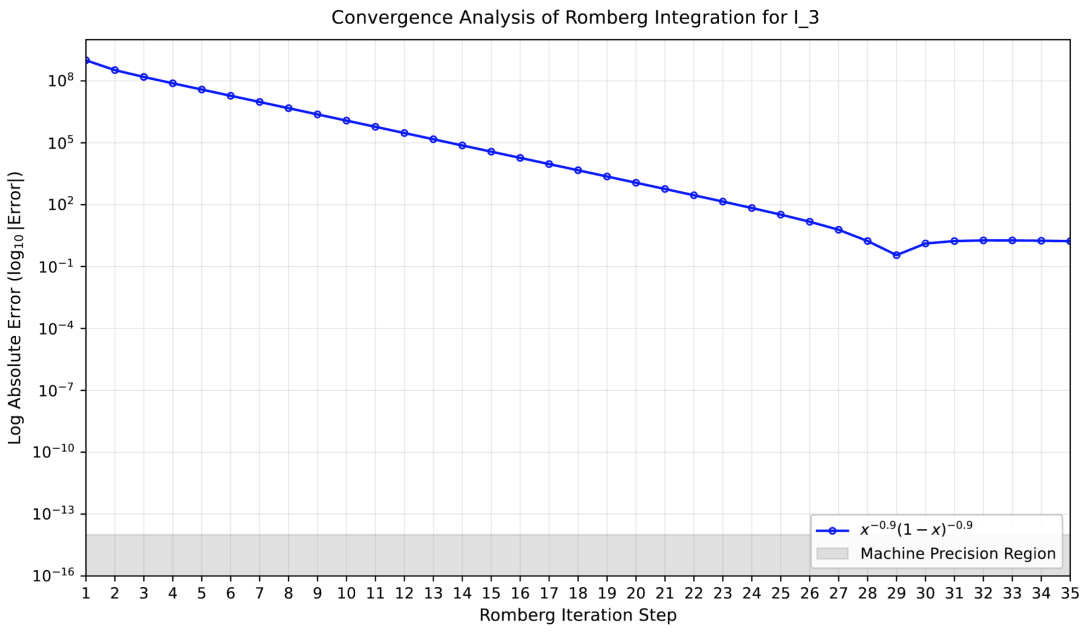

Figure 3: Absolute error vs. Romberg iteration steps for $I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,dx$.

3.4 Gauss-Jacobi Quadrature

To address this limitation of Romberg integration for functions with singularities, I explored Gauss-Jacobi quadrature. Romberg integration uses closed Newton-Cotes formulas (which include the endpoints and struggle with singularities), but Gauss-Jacobi quadrature is specifically designed for such cases. It uses orthogonal polynomials and weight functions to accurately integrate functions with endpoint singularities of the form:

$$I = \int_{-1}^{1} (1-x)^{\beta}(1+x)^{\alpha} f(x)\,\mathrm{d}x, \quad \text{where } \alpha, \beta > -1, \tag{10}$$

where the weight function $w(x) = (1-x)^{\beta}(1+x)^{\alpha}$ accounts for singular behaviour at the endpoints $x = \pm 1$. In this method, the nodes x_i are the roots of the Jacobi polynomials $P_n^{(\alpha,\beta)}(x)$, which are orthogonal with respect to $w(x)$ on $[-1, 1]$. By choosing α and β to match the integrand's singular exponents, Gauss-Jacobi quadrature provides accurate results.

Our integral is defined over $[0, 1]$, so to apply Gauss-Jacobi quadrature, we transformed the integral to the standard interval $[-1, 1]$ using the substitution:

$$x = \frac{1+t}{2}, \quad \text{so that} \quad \mathrm{d}x = \frac{\mathrm{d}t}{2}. \tag{11}$$

This substitution transforms the integral as follows:

$$\begin{aligned} I &= \int_0^1 x^{\alpha}(1-x)^{\beta} f(x)\,\mathrm{d}x \\ &= \int_{-1}^{1} \left(\frac{1+t}{2}\right)^{\alpha} \left(\frac{1-t}{2}\right)^{\beta} f\left(\frac{1+t}{2}\right) \frac{\mathrm{d}t}{2} \\ &= \frac{1}{2^{\alpha+\beta+1}} \int_{-1}^{1} (1+t)^{\alpha}(1-t)^{\beta} f\left(\frac{1+t}{2}\right) \mathrm{d}t. \end{aligned} \tag{12}$$

The weight function for Gauss-Jacobi quadrature becomes $w(t) = (1-t)^{\beta}(1+t)^{\alpha}$, matching the form in Equation (10), allowing us to use the method efficiently. Thus, the application to our specific integral was:

- For $I = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,\mathrm{d}x$, the weight function is $w(x) = x^{-0.9}(1-x)^{-0.9}$, corresponding to $\alpha = -0.9$ and $\beta = -0.9$. The transformed integral is given by:

$$\begin{aligned} I &= \int_0^1 x^{-0.9}(1-x)^{-0.9}\,\mathrm{d}x \\ &= 2^{0.8} \int_{-1}^{1} (1+t)^{-0.9}(1-t)^{-0.9}\,\mathrm{d}t. \end{aligned} \tag{13}$$

Here, $f\left(\frac{1+t}{2}\right) = 1$, as the integrand is entirely represented by the weight function.

To compute this integral, we used the `roots_jacobi` function from the `scipy.special` library [5] to determine the quadrature nodes t_i and weights w_i corresponding to the specified α and β. The integral was then evaluated as:

$$I \approx 2^{0.8} \sum_{i=1}^{n} w_i, \tag{14}$$

where n is the number of quadrature points. This method addresses endpoint singularities and provides accurate results for the transformed integral.

Moreover, the integral I can be evaluated analytically using the Beta function $B(a, b)$:

$$I = \int_0^1 x^{-0.9}(1-x)^{-0.9}\, \mathrm{d}x = B(0.1, 0.1) = \frac{\Gamma(0.1)\Gamma(0.1)}{\Gamma(0.2)}, \tag{15}$$

where Γ denotes the Gamma function [6, Ch. 6, pp. 255–258].

By comparing the numerical result from Gauss-Jacobi quadrature with the analytical value from Equation (15) computed using `gamma` from the `scipy.special` library [7], we can assess the exact accuracy of Gauss-Jacobi quadrature method.

3.4.1 Numerical Results, Comparison, and Error Analysis

Applying Gauss-Jacobi quadrature to the integral $I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\, \mathrm{d}x$, we visualised the efficiency by plotting the absolute error against the number of quadrature points in Figure 4.

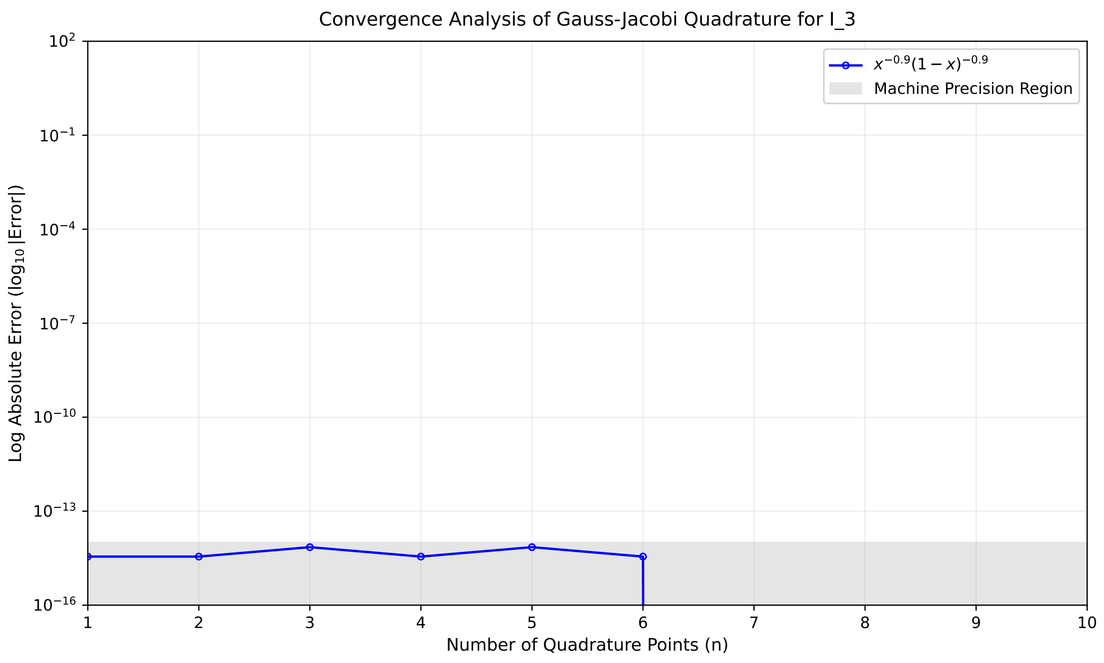

Figure 4: Absolute error vs. number of quadrature points n for $I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,dx$ using Gauss-Jacobi quadrature.

In Figure 4, the error immediately hits machine precision for all values of n, showing Gauss-Jacobi quadrature's high efficiency and precision for this particular integral. This is likely because the proper choice of Gauss-Jacobi quadrature tailored to the weight function $x^{-0.9}(1-x)^{-0.9}$. For all values n the error is immediately limited by machine precision, as the error values approach the numerical limits of my computing environment.

The convergence behaviour is summarised in Table 2. The numerical results show the computed value and absolute error for different numbers of quadrature points n.

Table 2: Convergence results for $I = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,dx$ using Gauss-Jacobi quadrature.

Quadrature Points (n)	**Computed Value**	**Absolute Error**
1	19.714639489050	3.55×10^{-15}
5	19.714639489050	7.11×10^{-15}
10	19.714639489050	0.00×10^{0}

The exact value of the integral is $I = 19.714639489050$, as calculated using the Beta function in Equation (15). We observe that Gauss-Jacobi quadrature achieved high accuracy almost immediately, with the absolute error reaching machine precision even with a small number of quadrature points. In contrast, Romberg integration failed to compute I_3 effectively. Even after running the algorithm for 690 minutes, Romberg integration did not converge to an accurate result.

We can conclude that Gauss-Jacobi quadrature outperforms Romberg integration for $I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,\mathrm{d}x$, providing very accurate results with minimal computational effort. This shows Gauss-Jacobi quadrature is more effective for integrals with singularities at both endpoints when the integrand is properly aligned with the weight function.

4 Conclusions and Future Work

The objective was to observe the performance of Romberg integration on different integrals. We demonstrated that the Romberg integration algorithm can struggle with endpoint singularities due to the nature of the trapezoidal rule. For instance, in $\int_0^1 x^{1/4}\,\mathrm{d}x$, convergence was initially slow. However, when we applied the variable transformation $x = y^4$, converting the integral to $\int_0^1 4y^3\,\mathrm{d}y$, the function was smoothed, and the algorithm converged rapidly to the exact value. For $I_3 = \int_0^1 x^{-0.9}(1-x)^{-0.9}\,\mathrm{d}x$, Romberg integration failed to provide an accurate result due to the strong singularities at the endpoints, even after 690 minutes of computation time the error was large.

Thus we explored the Gauss-Jacobi quadrature to handle integrals with endpoint singularities effectively. It achieved high accuracy where Romberg integration failed, showing the importance of selecting a numerical method that aligns with the integrand's specific characteristics. Gauss-Jacobi quadrature incorporated the integrand's behaviour at the endpoints through weight functions of the form $(1-x)^{\beta}(1+x)^{\alpha}$, which made it well-suited for our integral I_3

Romberg integration is effective for smooth functions but may not converge when dealing with strong singularities, even with variable transformations. Gauss-Jacobi quadrature, while more specialised, excels in handling singularities directly and provides higher accuracy with fewer function evaluations, making it a good choice when computational resources are limited.

Future work could look at exploring other numerical integration methods that use closed Newton-Cotes formulas and can handle functions with discontinuities or singularities, such Gaussian quadrature with specialised weight functions similar to Guass-Jacobian. Also, investigating performance of Gauss-Jacobi quadrature on integrals with other type of singularities or in higher dimensions could give further information into limitations.

In conclusion, understanding an integrand's behaviour is very important for choosing appropriate numerical integration methods to ensure accurate and reliable results. As with the No Free Lunch Theorem [8, pp. 67–82], there is no universally optimal method; the challenge lies in optimising the approach to suit the problem's demands and our computational resources.

References

[1] *Gauss–Jacobi quadrature - Wikipedia.* https://en.wikipedia.org/wiki/Gauss%E2%80%93Jacobi_quadrature, Accessed 21st November 2024.

[2] Burden, R. L., Faires, J. D., & Burden, A. M. (2015). *Numerical Analysis* (10th ed.). Cengage Learning. https://books.google.co.uk/books?id=9DV-BAAAQBAJ, Accessed 12th November 2024.

[3] *Trapezoidal Rule - Wikipedia.* https://en.wikipedia.org/wiki/Trapezoidal_rule, Accessed 17st November 2024.

[4] *Machine Epsilon - Wikipedia.* https://en.wikipedia.org/wiki/Machine_epsilon, Accessed 29th November 2024.

[5] SciPy Documentation: scipy.special.roots_jacobi. (n.d.). *SciPy v1.14.1 Manual*, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.roots_jacobi.html. Accessed November 21st 2024.

[6] Abramowitz, M., and Stegun, I. A. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards. Chapter 6. Accessed November 29th 2024.

[7] Virtanen, P., et al. (2020). *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17, 261–272. DOI: 10.1038/s41592-019-0686-2. Documentation available at: https://docs.scipy.org/doc/scipy/reference/generated/scipy.special.gamma.html. Accessed: November 29, 2024.

[8] Wolpert, D. H., & Macready, W. G. (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82. https://www.cs.ubc.ca/~hutter/earg/papers07/00585893.pdf, accessed 25th November 2024.