Romberg Integration

Diego Frias, Parker McCoog

June 2024

1 Introduction

Riemann sums can be an accurate way to approximate definite integrals, but they present a few problems in practicality. First, they can be quite slow, especially on complex functions with many segments. Additionally, there is no way to inherently detect when a function is not continuous over the interval being integrated. Further analysis must be done, which can be even slower. To solve both of these problems, our research led us to Romberg integration, which is much a faster way to approximate definite integrals that can also account for discontinuities. This paper will introduce and derive the Romberg formula. We will also implement a simple definite integral calculator in Python using the formula, and show our results on an online website (`https://dzfrias.github.io/romberg/`).

2 Definitions

In this section, we will go over some definitions used throughout this paper.

2.1 Riemann Sums

Riemann sums are a way of estimating an integral of a function by partitioning the region into smaller shapes (rectangles, trapezoids, etc.) and then adding together the sum of those areas. As the number of smaller regions *increase*, the error in the estimate of the integral *decreases*.

More precisely, if f is a function continuous over the closed interval $[a, b]$ and

$x_0, x_1, x_2, \ldots, x_n$ are numbers such that

$$a = x_0 < x_1 < x_2 < \ldots < x_n = b$$

A Riemann sum of f over $[a, b]$ is given by

$$\mathrm{S} = \sum_{i=1}^{n} f(x_i^*)\Delta x_i$$

where $x_i^* \in [x_{i-1}, x_i]$. This definition is largely based on Wikipedia [5].

2.2 Composite Trapezoidal Rule

Given a continuous function f, the area of the trapezoid (given by A) under f from $[a, b]$ is given by

$$\mathrm{A} = \frac{b-a}{2}(f(a) + f(b))$$

We can use this to approximate the area under f, using a Riemann sum (defined in 2.1). With n sub-intervals (each of width $h = \frac{b-a}{n}$), the approximation is given by using n trapezoids:

$$\begin{aligned}
\int_a^b f(x)dx &\approx \sum_{i=0}^{n} \frac{f(ih) + f((i+1)h)}{2} h \\
&\approx \frac{h}{2} \sum_{i=0}^{n} (f(ih) + f((i+1)h)) \\
&\approx \frac{h}{2} \left[f(0) + 2f(h) + 2f(2h) + \ldots + 2f((n-1)h) + f(nh) \right] \\
&\approx \frac{h}{2} (f(0) + 2\sum_{i=1}^{n-1} f(ih) + f(nh))
\end{aligned}$$

This will be referred to as the **Composite Trapezoidal Rule**. We will assign this formula to a function T.

$$T(n) = \frac{h}{2}(f(0) + 2\sum_{i=1}^{n-1} f(ih) + f(nh)) \quad \text{where } h = \frac{b-a}{n}$$

2.3 Composite Trapezoidal Rule Error

The error E_n of a composite trapezoidal sum approximation over the interval $[a, b]$ with n sub-intervals (each of width $h = \frac{b-a}{n}$) is a number such that

$$\int_a^b f(x)dx = T(n) + \mathrm{E}_n$$

where $T(n)$ the composite trapezoidal approximation of the area under f, defined in 2.2. This error is given by the series

$$\mathrm{E}_n = c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots$$

where $c_1, c_2, c_3, ...$ are unknown constants that are based on derivatives of f. Importantly, these constants **do not** depend on n (the number of sub-intervals). This was given by [3].

2.4 Composite Trapezoidal Rule Optimization

Given the definition of $T(n)$ in section 2.2, we will define make an observation about $T(2n)$, that requires less trapezoidal approximations than the naive usage of the composite trapezoidal rule function.

$$\begin{aligned}
T(2n) &= \frac{m}{2n}\left[f(a) + 2\sum_{i=1}^{2n-1} f(a + \frac{im}{2n}) + f(b)\right] \quad \text{where } m = b - a \\
&= \frac{m}{2n}\left[f(a) + 2\sum_{i=1}^{n-1} f(a + \frac{2im}{2n}) + 2\sum_{i=0}^{n-1} f(a + \frac{m \cdot (2i+1)}{2n}) + f(b)\right] \\
&= \frac{m}{2n}\left[f(a) + 2\sum_{i=1}^{n-1} f(a + \frac{im}{n}) + f(b) + 2\sum_{i=0}^{n-1} f(a + \frac{m \cdot (2i+1)}{2n})\right] \\
&= \frac{1}{2}T(n) + \frac{m}{2n}\left(2\sum_{i=0}^{n-1} f(a + \frac{m(2i+1)}{2n})\right)
\end{aligned}$$

Note that usage of this optimization *is not* required for proofs; it is only useful to eliminate the number of calculations (by half).

3 Romberg Integration Formula

The Romberg formula is given by the function $R(i,k)$

$$R(i,k) = \begin{cases} T(2^i) & k = 0,\ i \geq 0 \\ R(i,k-1) + \frac{R(i,k-1) - R(i-1,k-1)}{4^k - 1} & k > 0,\ i \geq k \end{cases}$$

As k increases, the approximation for the integral $\int_a^b f(x)dx$ improves. Note that this function is **recursive**.

3.1 Derivation

We will derive the Romberg integration formula using mathematical induction, and show that approximations improve through more extrapolations. Parts of this proof have been inspired by [1], [2], and [3].

3.1.1 Derivation of Order 1

Let f be a continuous, integrable function. The integral over the interval $[a,b]$ can be given by

$$\begin{aligned} \int_a^b f(x)dx &= T(n) + \mathrm{E}_n \\ &= T(n) + c_1 h^2 + c_2 h^4 + c_3 h^6 + \dots \end{aligned}$$

When we have an initial approximation with n sub-intervals. We can achieve an approximation *slightly better* than $T(n)$ by truncating all but the first term of the error value E_n

$$\begin{aligned} \int_a^b f(x)dx &\approx T(n) + c_1 h^2 \\ &\approx T(n) + \frac{c_1}{n^2} \end{aligned}$$

We can trivially double the number of segments to get a second equation

$$\begin{aligned} \int_a^b f(x)dx &\approx T(2n) + \frac{c_1}{(2n)^2} \\ &\approx T(2n) + \frac{c_1}{4n^2} \end{aligned}$$

With this, we have a system of equations with two unknowns, c_1 and $\int_a^b f(x)dx$. We can find an expression for $\frac{c_1}{4n^2}$ by multiplying the second equation by four and subtracting the two to get

$$\begin{aligned} -3\int_a^b f(x)dx &\approx T(n) - 4T(2n) \\ \int_a^b f(x)dx &\approx \frac{4T(2n)}{3} - \frac{T(n)}{3} \\ &\approx T(2n) + \frac{T(2n) - T(n)}{3} \end{aligned}$$

This is an approximation for $\int_a^b f(x)dx$ that is in terms of two composite trapezoidal approximations. We have already shown that

$$\int_a^b f(x)dx \approx T(2n) + \frac{c_1}{4n^2}$$

Achieves an approximation better than $T(2n)$ alone. We have found an expression for $\frac{c_1}{4n^2}$ involving $T(n)$ and $T(2n)$. Let $m = 2^{i-1}n$. We will now define $A_{i,1}$, where

$$\mathrm{A}_{i,1} = T(2m) + \frac{T(2m) - T(m)}{3}$$

If we let $n = 1$, then we get

$$\mathrm{A}_{i,1} = T(2^i) + \frac{T(2^i) - T(2^{i-1})}{3} \quad \text{where } i \geq 1$$

The i, k represents the **sweep** and **order of extrapolation**, respectively. $\mathrm{A}_{i,1}$ is in the first order of extrapolation.

3.1.2 Induction

We want to show that

$$\mathrm{A}_{i,k} = \mathrm{A}_{i,k-1} + \frac{\mathrm{A}_{i,k-1} - \mathrm{A}_{i-1,k-1}}{4^k - 1}$$

for any $k \geq 1$ and any $i \geq k$. We also want to show that for larger values of k, the approximation improves. We will prove it using mathematical induction. We've proved the base case $k = 1$ (in section 3.1.1). If we assume that the above

statement is true, we can show that the statement is true for $k+1$ and $i+1$

$$\mathrm{A}_{i+1,k+1} = \mathrm{A}_{i+1,k} + \frac{\mathrm{A}_{i+1,k} - \mathrm{A}_{i,k}}{4^{k+1} - 1}$$

Given that the the $\mathrm{A}_{i,k}$ equation holds, we know that

$$\int_a^b f(x)dx \approx \mathrm{A}_{i,k} + c_{k+1}h^{2(k+1)} + c_{k+2}h^{2(k+2)} + \dots$$

So, we can truncate all but the $k+1$ error term to get an approximation better than $\mathrm{A}_{i,k}$

$$\begin{aligned}\int_a^b f(x)dx &\approx \mathrm{A}_{i,k} + c_{k+1}h^{2(k+1)} \\ &\approx \mathrm{A}_{i,k} + \frac{c_{k+1}}{n^{2(k+1)}}\end{aligned}$$

We also know $\mathrm{A}_{i+1,k}$ is valid because of our assumption, so

$$\int_a^b f(x)dx \approx \mathrm{A}_{i+1,k} + \frac{c_{k+1}}{4^{k+1}n^{2(k+1)}}$$

which is a better approximation than $\mathrm{A}_{i+1,k}$. As with before, we have a system of equations with two unknowns, c_{k+1} and $\int_a^b f(x)dx$. To eliminate c_{k+1}, we can multiply the second equation by 4^{k+1} and subtract the two

$$\begin{aligned}(-4^{k+1}+1)\int_a^b f(x)dx &\approx \mathrm{A}_{i,k} - 4^{k+1}\mathrm{A}_{i+1,k} \\ \int_a^b f(x)dx &\approx \frac{4^{k+1}\mathrm{A}_{i+1,k}}{4^{k+1}-1} - \frac{\mathrm{A}_{i,k}}{4^{k+1}-1} \\ &\approx \mathrm{A}_{i+1,k} + \frac{\mathrm{A}_{i+1,k} - \mathrm{A}_{i,k}}{4^{k+1}-1}\end{aligned}$$

So, we get a formula for $\mathrm{A}_{i+1,k+1}$

$$\mathrm{A}_{i+1,k+1} = \mathrm{A}_{i+1,k} + \frac{\mathrm{A}_{i+1,k} - \mathrm{A}_{i,k}}{4^{k+1} - 1}$$

Since both the base case ($k = 1$) and the induction step hold true, by mathematical induction, we have shown that

$$\mathrm{A}_{i,k} = \mathrm{A}_{i,k-1} + \frac{\mathrm{A}_{i,k-1} - \mathrm{A}_{i-1,k-1}}{4^k - 1}$$

3.1.3 The Romberg Function

We have shown in section 3.1.2 that

$$\mathrm{A}_{i,k} = \mathrm{A}_{i,k-1} + \frac{\mathrm{A}_{i,k-1} - \mathrm{A}_{i-1,k-1}}{4^k - 1}$$

for $k \geq 1$ and $i \geq k$. We will define a function $R(i,k)$ (the **Romberg function**) that is defined for $k \geq 0$ and $i \geq k$.

When $k = 0$, there are no extrapolations. So,

$$\mathrm{A}_{i,0} = T(2^i)$$

for all $i \geq 0$, which is what we used to get the first order extrapolation in section 3.1.1. So we will define the Romberg function **recursively**, as follows

$$R(i,k) = \begin{cases} T(2^i) & k = 0,\ i \geq 0 \\ R(i,k-1) + \frac{R(i,k-1) - R(i-1,k-1)}{4^k - 1} & k > 0,\ i \geq k \end{cases}$$

This is concludes the derivation of the function proposed in section 3. As k gets larger, the approximation improves.

4 Calculator

We will use the Python language to implement a simple definite integral calculator using Romberg. In our tests, it achieved significantly better performance and better accuracy when compared with rectangular Riemann sums.

Some of this code was taken from [4].

4.1 Basic Definitions

```
def f(x):
    return x**2

# Defining the range of the integral
a = 0
b = 1
```

We will be integrating the function $f(x) = x^2$ over the closed interval $[0, 1]$. Note that we are using the numpy package (aliased as np). We will be using it in other parts of the program.

4.2 Composite Trapezoidal Rule

```
# Calculate a Riemann sum using trapezoids of the function f from a to b with
# n subintervals
def riemann_trapezoid(a, b, f, n):
    h = (b - a) / n
    total = 0
    for i in range(1, n, 2):
        x = a + (h * i)
        total += 2 * f(x)
    return (total) * (h / 2)
```

This function implements the composite trapezoidal approximation formula as defined in section 2.2 (referred to as $T(n)$).

4.3 Romberg

```
# An upper triangular matrix that will be used to store our calculations
I = np.zeros((22, 22))
I[0, 0] = (f(a) + f(b)) * (b-a)/2

i = 0
error = 1
# The loop will go for 15 iterations, or until the error value is greater than
# 0.0001 (chosen arbitrarily)
while i < 15 or error > 0.0001:
    # If it has gone through 25 iterations but still has not met the error
    # threshold, our code determines that the integral does not converge

    if i > 20:
        print("Does not converge")
        break
```

```
    # Calculating the next item in the first column (trapezoidal sums)
    n = 2**(i+1)
    I[i + 1, 0] = I[i, 0]/2 + riemann_trapezoid(a, b, f, n)

    # The main romberg formula
    for k in range(1, i + 2):
        j = 1 + i - k
        I[j, k] = ((4 ** k * I[j+1, k - 1]) -  I[j, k - 1]) / (4**k - 1)

    # Error as: (previous_estimate - new_estimate) / (previous_estimate + new_estimate)
    error = abs((I[1, i] - I[1, i - 1]) / (I[1, i] + I[1, i - 1]))
    i += 1
    print(i, I[0, i])

# If the error theshold was met, we will print the best estimate
if i < 21:
    print(I[0, i])
```

There are a few subtle differences between the $R(i, k)$ and the method of Romberg integration in the code. Firstly, we calculate the best approximation iteratively, using a matrix, instead of recursively. In the Python implementation, we store our values in an upper-left triangular matrix, meaning the best approximation is $\mathrm{I}_{0,k}$. The best approximation in the recursive formula is the largest i and k.

We detect when two the function does not converge with the `error` variable. This compares our current estimate and the previous estimate. If they are not getting closer to the same value, we make the assertion that the function does not converge over $[a, b]$.

Also note that we make usage of the optimization described in section 2.4 to calculate the first row of Riemann sums.

5 Summary of Results

In this paper, we have introduced and derived the Romberg formula for approximating definite integrals, defined as follows:

$$R(i,k) = \begin{cases} T(2^i) & k = 0,\ i \geq 0 \\ R(i,k-1) + \frac{R(i,k-1) - R(i-1,k-1)}{4^k - 1} & k > 0,\ i \geq k \end{cases}$$

where $T(n)$ is the composite trapezoidal approximation formula defined in section 2.2. We know that as the order of extrapolation gets larger, the approximation improves.

We have also shared code for a simple definite integral calculator written in Python (see section 4). The calculator is *much faster* than the naive rectangular Riemann sum implementation, and can detect discontinuities.

We put the calculator on a website, available at this address:
`https://dzfrias.github.io/romberg/`

On the website, the core Romberg algorithm (described in section 4) was rewritten in Rust and compiled to WebAssembly for maximum performance on the browser. The code is distributed under the MIT license and is viewable at this address:
`https://github.com/dzfrias/romberg`

6 Applications

As mentioned in section 1, the primary application for Romberg integration is with calculators. Because old calculations can be re-used to get better approximations, it is much faster than using Riemann sums to approximate an integral.

It is worth noting that the Romberg integration formula is sometimes practical to do by hand, with three or four orders of extrapolation, which can still give an accurate estimate for some functions.

References

[1] Autar Kaw. *Chapter 07.04: Lesson: Richardson's Extrapolation of Trapezoidal Rule: Theory*. https://www.youtube.com/watch?v=i9xxITSH0_w. Accessed: June 6, 2024. 2010.

[2] Autar Kaw. *Chapter 07.04: Lesson: Romberg Integration: Theory: Part 1 of 2*. https://www.youtube.com/watch?v=DZ8y6o3bUrM. Accessed: June 6, 2024. 2010.

[3] Autar Kaw. *Chapter 07.04: Lesson: Romberg Integration: Theory: Part 2 of 2*. https://www.youtube.com/watch?v=RpNbjT4DFkQ. Accessed: June 6, 2024. 2010.

[4] Shams ElFouly. *Romberg Integration - Python Code*. https://www.youtube.com/watch?v=2BxLDODvnQA. Accessed: May 31, 2024. 2021.

[5] Wikipedia contributors. *Riemann sum — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Riemann_sum&oldid=1222233544. Accessed: May 31, 2024. 2024.