Computational Physics: Integral Calculator

Ujwal Kumar

February 3, 2021

Introduction

The area under a curve is calculated using the integral of that function defined on some boundary: $\int_a^b f(x)dx$. While this is a relatively easy operation to evaluate for simple functions, this can quickly get out of hand. To mitigate this, we can evaluate the area under a curve by splitting the area under the curve into approximate rectangles with a tiny enough width such that the top right and top left edges of the rectangles closely follow the curve.

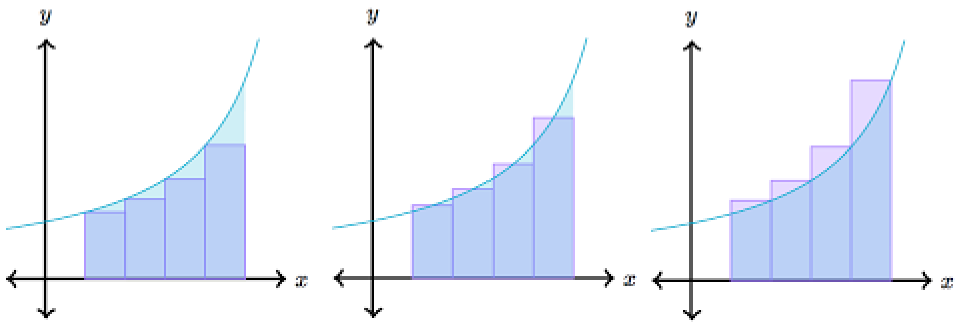

Figure 1: Function split into rectangles and approximated using left-hand, mid-point, and right-hand points of rectangles. *Image source: Khan Academy*

Consider each rectangle with width dx and heights evaluated using the function at the left-hand point, the mid-point and the right-hand point. We call them f(a), f(m) and f(b). If we used the left-hand point of the rectangle to evaluate the area of the rectangle, we would find the area $= f(a)dx$. Likewise, with the mid-point, the area would be $f(m) \times dx$.

The next important consideration to make is the number of rectangles you want to split the curve into. Herein lies the conundrum of using such a method to evaluate integrals of functions. The larger the number of rectangles (greater number of bins, N), the closer the approximation is to the precision value. However, bins are costly. Computationally.

This report explores the accuracy of such integration methods by evaluating functions using the mid-point or left-hand point method. We also study the computational efficiency of increasing the number of bins and the effects on precision.

Problem 1

The integral $\int_0^x e^y dy$ can be computed using the midpoint and left-hand method by evaluating the integral at bin widths of 0.25, 0.1, and 0.02. We plot the analytic answer below for these bin widths. I was unable to get the plot title, labels, or axes titles to work in Mathematica but the blue dot represents the largest bin widths and the green dots represent the smallest bin widths. The blue function is is the function to be analyzed, e^y.

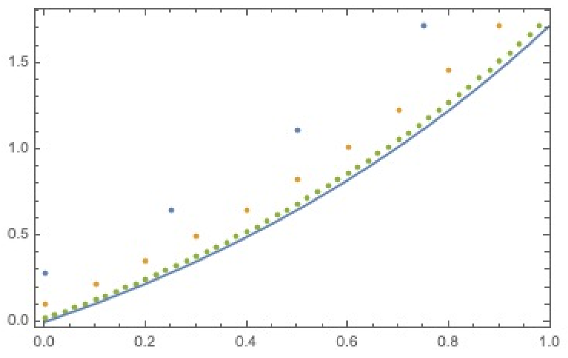

Figure 2: Numerical integration using left-hand rule integrator. Plotted is the analytic result against the bin width.

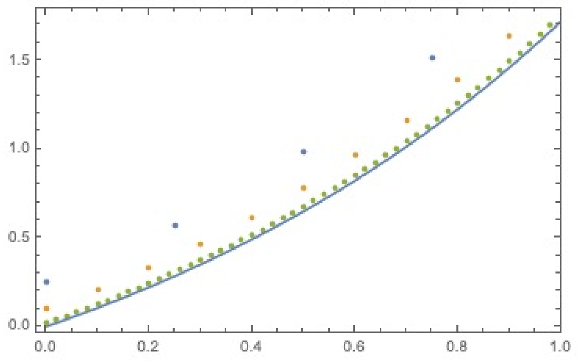

Figure 3: Numerical integration using midpoint integrator. Plotted is the analytic result against the bin width.

As can be seen above, the midpoint integrator converges to the true values much quicker and to higher precision. And lastly, the greater the bin width, the slower the convergence and vice-versa.

Problem 2

The definite integral of $\int_0^1 e^y dy$ can be computed using the left-hand approximation integral calculator. Each bin width is of length $dy = \frac{b-a}{Nbins}$ with the area of the bins calculated as $f(a + idy)dy$.

Using this naïve integral calculator, we compute the result of $\int_0^1 e^y dy$ using with the number of bins ranging from 10^1 to 10^9.

Table 1: Integral Approximations for Nbins using Naïve Integrator

Nbins	Approximation
10^1	1.63380
10^2	1.70970
10^3	1.71742
10^4	1.71820
10^5	1.71827
10^6	1.71828
10^7	1.71828
10^8	1.71828
10^9	1.71828

To test the accuracy and precision of these approximations, and to test the computational efficiency, we can study a plot of the error from the actual value (computed using Mathematica) against the number of bins. This error is calculated using:

$$\epsilon = \frac{|\text{computed value - analytic value}|}{|\text{analytic value}|}$$

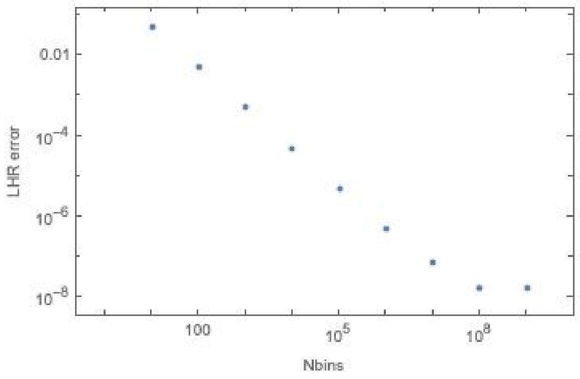

Figure 4: Error using Left-Hand Approximation versus Number of Bins

We can then compute the slope of this error curve between two values of bin widths (bw) – from 10^{-5} to 10^{-3} using the following function :

$$\frac{log(\epsilon_2) - log(\epsilon_1)}{log(bw_1) - log(bw_2)}$$

This yields a slope of $\frac{0.0004494913}{10^{-3}-10^{-5}} \approx 0.49991 \approx 0.50$. This slope gives you the increase in precision for every extra order of bin width and allows users to estimate the number of bins they would need to use to achieve the precision they require while minimizing computing costs.

Problem 3

We can now repeat the above problem using the mid-point rule approximation that simply estimates the value of the function in the middle of the bin. Thus, the areas of the bins are given by $f(a + idy + 0.5dy)dy$. We hope to study the slope of the new error curve produced by the mid-point approximations and study the behaviour for small stepsizes. Using this new approximation, we find the approximations using our mid-point integrator to be:

Table 1: Integral Approximations for Nbins using Mid-Point Method

Nbins	Approximation
10^1	1.71757
10^2	1.71828
10^3	1.71828
10^4	1.71828
10^5	1.71828
10^6	1.71828
10^7	1.71828
10^8	1.71828
10^9	1.71828

This consequently produces a new error function curve and new error slope (evaluated at the same bin widths):

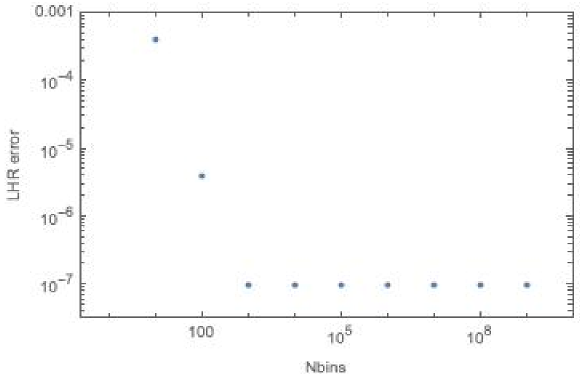

Figure 5: Error using Mid-Point Approximation versus Number of Bins

We notice two immediate facts about this plot in comparison to Figure 2. Firstly, the y-range is shorter. Since the number of bins (or domain is the same), this immediately tells us the slope will be smaller. Secondly, the behaviour of the points at small bin widths (high Nbins) acts such that you get no greater precision after 10^4 bins. Therefore, these extra bins cost you computational power but achieve no greater precision. This suggests that 10,000 bins using the mid-point approximator maximises the precision (to 5 decimal places) and minimizes the computing cost to achieve this precision.

We find the slope of the plot from bw = 10^{-1} to 10^{-3} to be $\approx 0.0486669 \approx 0.05$. The slope then sharply drops to near 0 for smaller bin widths. This is 1 order of magnitude lower than the slope from Problem 2 for large bin widths implying lower return on precision for greater bin width. However, higher precision is achieved using this method at larger bin widths.

Problem 4

Consider a pendulum made of a massless rod with length L and a bob with mass m at the end. In this problem, we aim to study the period and swing angles, specifically or small starting angles, θ_0. This relationship can be expressed as

$$\tau = 2\pi\sqrt{\frac{L}{g}} \tag{1}$$

We study the numerical integration techniques used to represent τ for several angles. We keep constant the gravitational constant $g = 9.81$ m/s^2 with L = 10 m. We will compute these results using 10^5 bins. For these values, we estimate the period to be 6.343739 seconds. The integral calculator computes a value at $\theta = 15°$ to be 6.366 s and a value at $\theta = 30°$ to be 6.391 s. For $\theta = 30°$, this is an error of 0.35% or $\approx 300 seconds per day$.

Conclusions

During the course of this report, I struggled a fair bit with using Mathematica, specifically running arguments through echo, and I wanted to make a note of that. These problems allowed me to study the importance of computational efficiency and also the need to consider errors accumulating over time.