MAU11601:Introduction To Programming - Tutorial 3
Numerical Integration

Liam Burke
School Of Mathematics, Trinity College Dublin

December 5, 2022

Introduction - Numerical Integration

Numerical integration is a means of evaluating a numerical approximation to a definite integral of some function $f(x)$ on an interval $[a, b]$

$$\int_a^b f(x)dx.$$

This can be done using the fact that the value of the integral is equal to the area under the curve of $f(x)$ in the interval $[a, b]$. We can thus make an approximation to the integral by approximating this area.

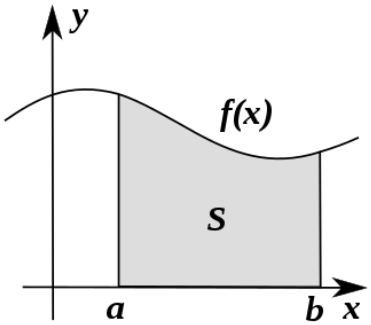

Figure 1: The integral of a function $f(x)$ in the interval $[a, b]$ is equal to the area S under the curve. (picture source: Wikipedia)

One approach is to approximate the area under the curve using the area of the red rectangle shown in Figure 2.

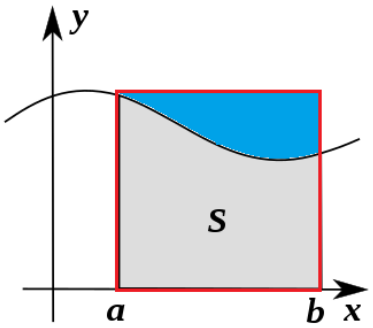

Figure 2:

The area of this rectangle is simply

$$f(a) \times (b - a),$$

and the error of this approximation is represented by the area of the region shaded in blue. Of course, a better approximation can be obtained by dividing the interval $[a, b]$ into more sub intervals, and approximating the area via more rectangles of smaller width, and by summing the area of each. This is known as a *Riemann sum*. The smaller the width of each rectangle, and indeed the more rectangles we take, the more accurate our approximation will be. This can be seen from the diagram below.

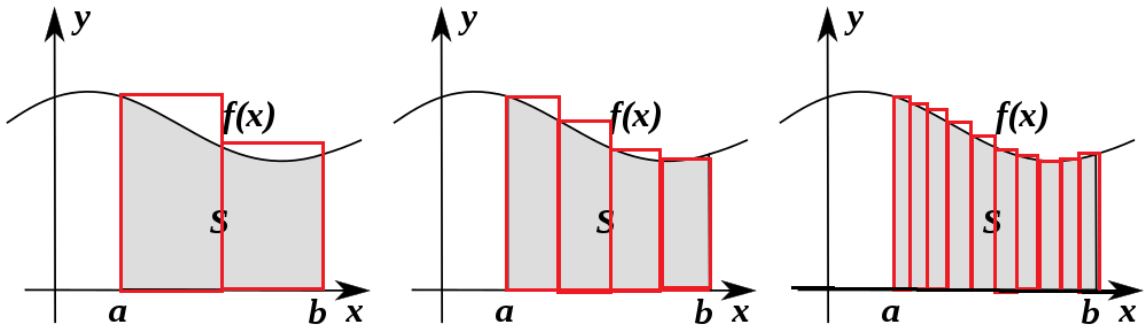

Figure 3: Interval $[a, b]$ partitioned into (a) 2 intervals, (b) 4 intervals, and (c) 10 intervals. For simplicity, we assume the width of each sub interval is the same. (It may not look like this from my Microsoft paint drawing!).

In the diagram above we have chosen each rectangle on the interval $[x_i, x_{i+1}]$ to have height $f(x_i)$, leading to a *left Riemann sum*. We could have also chosen the height of this rectangle to be $f(x_{i+1})$, the *right Riemann sum*. In addition we could also take the height to be $f(\frac{x_{i+1}-x_i}{2})$. This defines the Midpoint method, which we will discuss in the next section.

Midpoint method

The Midpoint method uses the midpoint of the interval to determine the height of its associated rectangle. An example of the resulting rectangles can be seen below.

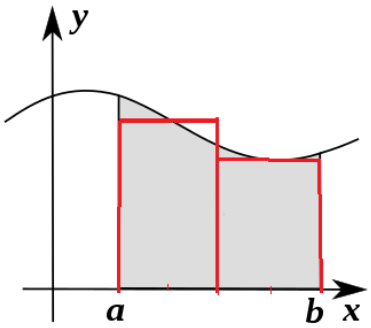

Figure 4: Midpoint rule

Assume we partition the interval $[a, b]$ into n sub intervals of equal length $\Delta = (b - a)/n$, and define the $n + 1$ points as $x_i = a + i\Delta$, given by the set

$$x_0 = a,\ x_1 = a + \Delta,\ x_2 = a + 2\Delta,\ ...,\ x_{n-1} = a + (n - 1)\Delta,\ x_n = b$$

The Midpoint method is then defined by the approximation

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f\left(\frac{x_{i-1} - x_i}{2}\right)\Delta$$

Trapezoidal Rule

Instead of approximating the area using rectangles, one can also approximate using the area of a set of trapezoids. This is seen in the diagram below.

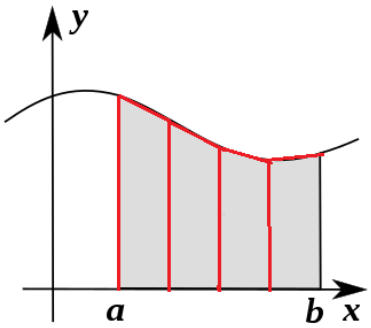

Figure 5: Trapezoidal rule

The Trapezoidal rule can be seen as taking an average of the left and right Riemann sums. ie in the interval $[x_i, x_{i+1}]$ we take the area of the Trapezoid to be

$$\frac{1}{2}(f(x_i)(x_i - x_{i+1}) + f(x_{i+1})(x_i - x_{i+1}))$$

Using the same interval partition scheme as the Midpoint method we can derive the *Trapezoidal rule* approximation as

$$\int_a^b f(x)dx \approx \frac{\Delta}{2}(f(a) + 2f(x_1) + 2f(x_2) + ... + 2f(x_{n-1}) + f(b)) \tag{1}$$

Aim Of Tutorial

Q1: Derive the Trapezoidal rule approximation (equation 1). (We will do this in class).
Q2: Write a MATLAB code to evaluate the definite integral

$$\int_{-3}^{1} 6x^2 - 5x + 2dx$$

Create a run script which defines the end points of your integration interval $a = -3, b = 1$ and a vector containing a list of numbers of interval partitions

$$N = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10].$$

Write two functions to implement the midpoint method and the Trapezoidal rule. These functions should be written in two separate files *midpoint.m* and *trapezoidal.m*. Each function should take as input your two interval end points and an element of the vector N. Both functions should call another function called *test_function* which should take in a value x and return $f(x)$ where

$$f(x) = 6x^2 - 5x + 2$$

This function should be defined in a separate file.

Your run script should contain a for loop to call both functions for each number of sub intervals defined in the vector N. Store the results for each function in a vector named *midpoint_results* and *trapezoidal_results*. Make a plot of the approximation of both methods vs the number of sub intervals used.