Numerical Methods for MAT 120

Name	Email
Irfan Sadat Ameen	irfan.ameen@bracu.ac.bd
Syed Emad Uddin Shubha	syed.shubha@bracu.ac.bd
Fatema Tuz Zohora	fatema.zohora@bracu.ac.bd
Rakibul Alam Shamim	rakibul.alam@bracu.ac.bd
Krity Haque Charu	krity.haque@bracu.ac.bd
Sarah Zabeen	sarah.zabeen@bracu.ac.bd
Reaz Shafqat	reaz.shafqat@bracu.ac.bd
Omer Tahsin	omer.tahsin@bracu.ac.bd

Date: November 4, 2024

Institution: BRAC University

Contents

Chapter 1

Numerical Root-Finding Methods

1.1 Introduction

Roots of a function means the solutions of $f(x) = 0$. Except for few specific functions, it is generally not feasible to obtain an exact analytical expression for the root, thus preventing the precise determination of the solution. We know, for example, that the solution of an equation of the form

$$ax^2 + bx + c = 0 \quad \text{is} \quad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{1.1}$$

A analytic formula like this does not exist for polynomials of all degrees. Such problem can be solved by approximating a solution using an iterative method. We can solve such problems numerically using few algorithms.

1.2 Bisection Method

The bisection method is based on the Bolzano's Theorem which states that

If a continuous function has values of opposite sign inside an interval, then it has a root in that interval.

If a continuous function changes sign over an interval, we can narrow down the approximate solution by using the midpoint of the interval and searching to the left and right of it (similar to a binary search). This can be broken down into a few steps:

1. Confirm that a root lies between the interval by checking if $f(a)f(b) < 0$. Otherwise, choose different interval.

2. Introduce a new variable that is defined as the midpoint of the interval,

$$x_0 = (a + b)/2. \tag{1.2}$$

3. Check if the root lies left or right of the midpoint.

   (a) If $f(a)f(x_0) < 0$, the root lies to the left.

   (b) If $f(b)f(x_0) < 0$, the root lies to the right.

4. Repeat the process from step 2 until the width, $b - a$, is less than a very small number, for example $\epsilon = 0.001$.

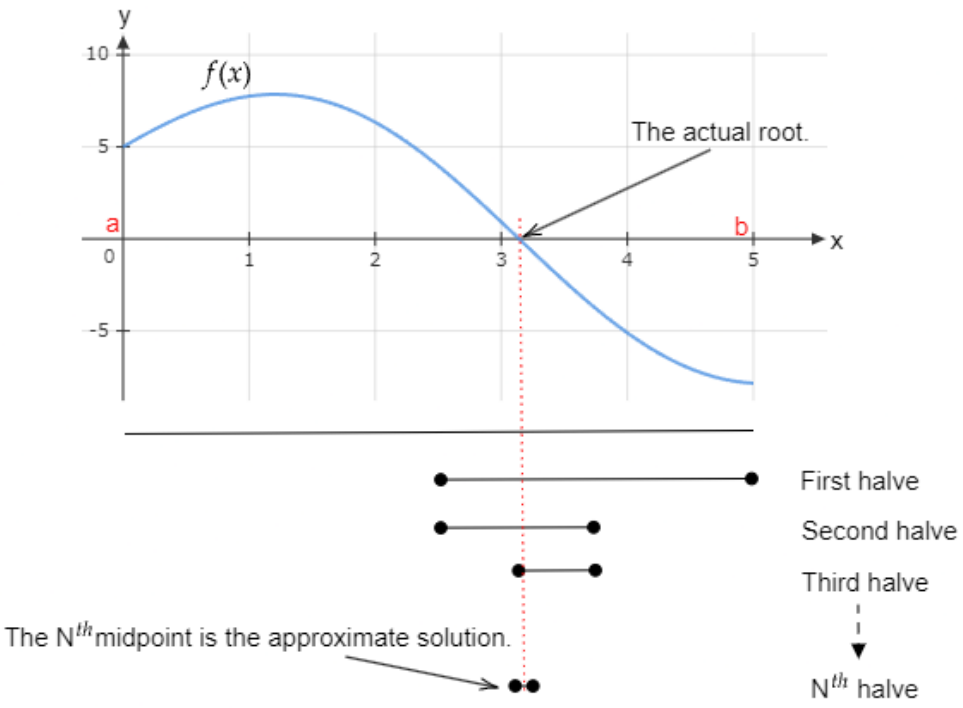

Figure 1.1: Figure: Bisection Method applied to a function $f(x)$ with initial guesses as a and b.

1.2.1 Example

Problem

Find the approximate root of the function

$$f(x) = x^3 + x^2 + x - 8 \tag{1.3}$$

within the interval $[1, 2]$ using the bisection method with an accuracy of $\varepsilon = 0.1$.

Solution

We apply the bisection method step-by-step to narrow down the interval until it is smaller than the required accuracy.

Check if $f(x)$ has opposite signs at $x = 1$ and $x = 2$:

$$f(1) = 1^3 + 1^2 + 1 - 8 = -5$$
$$f(2) = 2^3 + 2^2 + 2 - 8 = 6$$

Since $f(1) = -5$ and $f(2) = 6$ have opposite signs, there is at least one root in $([1, 2]$.

We will iteratively bisect the interval and evaluate $f(x)$ at the midpoint until the interval width is smaller than $\varepsilon = 0.1$.

Iteration 1

$$x_0 = \frac{1+2}{2} = 1.5$$
$$f(1.5) = 1.5^3 + 1.5^2 + 1.5 - 8 = -0.875$$

Since $f(1.5) = -0.875$ and $f(2) = 6$ have opposite signs, the root is in $[1.5, 2]$.

Iteration 2

$$x_0 = \frac{1.5 + 2}{2} = 1.75$$
$$f(1.75) = 1.75^3 + 1.75^2 + 1.75 - 8 = 2.171875$$

Since $f(1.5) = -0.875$ and $f(1.75) = 2.171875$ have opposite signs, the root is in $[1.5, 1.75]$.

Iteration 3

$$x_0 = \frac{1.5 + 1.75}{2} = 1.625$$
$$f(1.625) = 1.625^3 + 1.625^2 + 1.625 - 8 = -0.435546875$$

Since $f(1.625) = -0.435546875$ and $f(1.75) = 2.171875$ have opposite signs, the root is in $[1.625, 1.75]$.

Iteration 4

$$x_0 = \frac{1.625 + 1.75}{2} = 1.6875$$
$$f(1.6875) = 1.6875^3 + 1.6875^2 + 1.6875 - 8 = 0.340576171875$$

Since $f(1.625) = -0.435546875$ and $f(1.6875) = 0.340576171875$ have opposite signs, the root is in $[1.625, 1.6875]$.

Iteration 5

$$x_0 = \frac{1.625 + 1.6875}{2} = 1.65625$$
$$f(1.65625) = 1.65625^3 + 1.65625^2 + 1.65625 - 8 = -0.0494384765625$$

Since $f(1.65625) = -0.0494384765625$ and $f(1.6875) = 0.340576171875$ have opposite signs, the root is in $[1.65625, 1.6875]$.

Approximate Root

The interval $[1.65625, 1.6875]$ has a width of

$$1.6875 - 1.65625 = 0.03125,$$

which is less than the required accuracy of 0.1. Therefore, we stop here and take the midpoint as an approximate root:

$$\text{Approximate root} \approx 1.671875.$$

1.2.2 Python Implementation

The implementation of this algorithm in python is the following,

```
eps = 1e-04  #0.0001

def bisection(expr,a,b):
  #Check for wrong input:
  if expr.subs(x,a)*expr.subs(x,b) >= 0:
    print("Invalid values.")
  else:
    while (b-a) >= eps:
      #Find the midpoint of a and b:
      x0 = (a+b)/2
      #set condition to see if root lies left or right
      if expr.subs(x,a)*expr.subs(x,x0) < 0:
        b = x0
      elif expr.subs(x,b)*expr.subs(x,x0) < 0:
        a = x0
    print("The root is ", x0)
    return
```

1.2.3 Exercises

(a) **Basic Check for Root (Easy)**

Given $f(x) = x^2 - 4$, use the Bisection Method to determine if there's a root between $a = 1$ and $b = 3$. Perform two iterations to find an approximate value for the root.

(b) **Intermediate Iteration (Easy-Medium)**

Apply the Bisection Method to the function $f(x) = x^3 - 6x + 2$ within the interval $[1, 2]$. Find the root to within $\epsilon = 0.1$, completing at least three iterations.

(c) **Precision Increase (Medium)**

Using the Bisection Method, approximate the root of $f(x) = x^2 - 5$ in the interval $[2, 3]$. Continue until the interval width is less than $\epsilon = 0.01$.

(d) **Nonlinear Root Approximation (Medium-Hard)**

For $f(x) = \cos(x) - x$, find an approximate root within the interval $[0, 1]$ using the Bisection Method. Complete enough iterations to reach a precision of $\epsilon = 0.001$.

(e) **High Precision and Multiple Iterations (Hard)**

Apply the Bisection Method to approximate the root of $f(x) = x^3 - x - 1$ within the interval $[1, 2]$ until the interval width is less than $\epsilon = 0.0001$. Determine the number of iterations needed to achieve this precision, showing each step.

1.3 Newton-Raphson Method

One can find the root of a continuous function using another method known as the Newton-Raphson method. This method does not require that the root must lie within an interval, $[a, b]$, rather an arbitrary point, x_a, is chosen and a root (if there is one) that lies close to that point will be discovered. The method starts with the point x_a and a slope at point is taken. The point where the slope crosses the $x - axis$ is found. This point becomes the new value for x_a.

Let x_a be a good estimate of r and let $r = x_a + h$. Since the true root is r, and $h = r - x_a$, the number h measures how far the estimate x_a is from the truth. Since h is 'small',

$$f(r) = f(x_a + h) \approx f(x_a) + hf'(x_a) + h^2 \frac{f''(x_a)}{2!} + ... = 0. \tag{1.4}$$

and therefore, as x_a is an approximate and it is not equal to the root, $f(x_a)$ is close to 0,

$$h \approx -\frac{f(x_a)}{f'(x_a)}. \tag{1.5}$$

As, $r = x_a + h$, we can write

$$r = x_a - \frac{f(x_a)}{f'(x_a)}. \tag{1.6}$$

We can turn this into an iterative formula,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{1.7}$$

Here, $f' = \frac{df}{dx}$. This iterative process will continue until $|f(x_{n+1})| < \epsilon$.

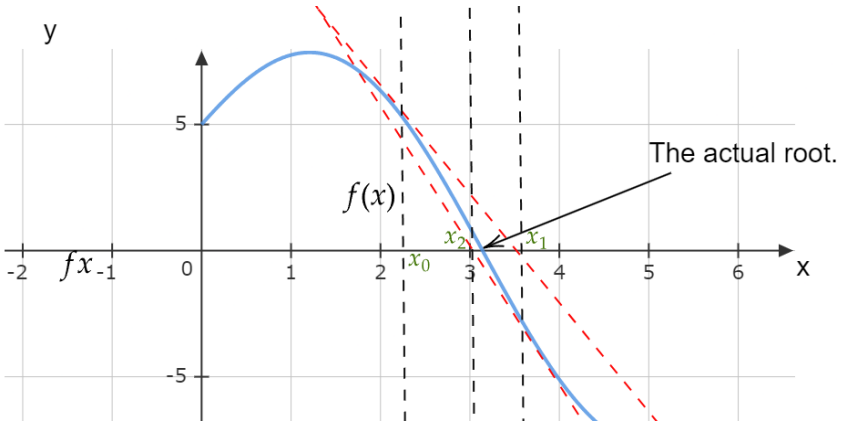

Figure 1.2: Figure: Demonstration of Newton-Raphson method. x_n moves closer to the actual root per iteration.

1.3.1 Example

Problem

Find the approximate root of the function

$$f(x) = x^2 - 4$$

starting from an initial guess of $x_0 = 5$ using Newton's method, with an accuracy of $\varepsilon = 0.1$.

Solution

We will apply Newton's method, which uses the formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1.8}$$

where $f'(x)$ is the derivative of $f(x)$. For the function

$$f(x) = x^2 - 4,$$

, we have

$$f'(x) = 2x.$$

.

Starting with $x_0 = 5$, we will compute each successive approximation x_{n+1} until the difference between consecutive approximations is less than $\varepsilon = 0.1$.

Iteration 1

$$\begin{aligned} x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\ &= 5 - \frac{5^2 - 4}{2 \cdot 5} \\ &= 5 - \frac{21}{10} \\ &= 2.9 \end{aligned}$$

Iteration 2

$$\begin{aligned} x_2 &= x_1 - \frac{f(x_1)}{f'(x_1)} \\ &= 2.9 - \frac{2.9^2 - 4}{2 \cdot 2.9} \\ &= 2.9 - \frac{4.41}{5.8} \\ &\approx 2.1397 \end{aligned}$$

Iteration 3

$$\begin{aligned} x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} \\ &= 2.1397 - \frac{2.1397^2 - 4}{2 \cdot 2.1397} \\ &\approx 2.0062 \end{aligned}$$

Iteration 4

$$\begin{aligned} x_4 &= x_3 - \frac{f(x_3)}{f'(x_3)} \\ &= 2.0062 - \frac{2.0062^2 - 4}{2 \cdot 2.0062} \\ &\approx 2.00002 \end{aligned}$$

Since the difference between $x_3 \approx 2.0062$ and $x_4 \approx 2.00002$ is less than $\varepsilon = 0.1$, we stop here.

Final Answer

The approximate root is:

$$x \approx 2.000$$

1.3.2 Python Implementation

The implementation of this is the following,

```
def NM(expr, x0):
    df = smp.diff(expr, x)
    while abs(f.subs(x,x0)) >= eps:
        r = f.subs(x,x0)/(df.subs(x,x0))
        #As r becomes really small, we eventually approach our solution.
        x0 -= r
    #print(x0)
    #Remove comment to see all the values explicitly
    return(x0)
```

Here, `f.subs(x, obj)` will replace the symbol x in the expression f with the variable obj. This variable can be of any type and will be replaced accordingly. If the substitute variable is an integer or float, they will undergo the arithmetic operations in the expression and give us the value. One can use `sympy.lambdify()` or define a `lambda` function and pass it as the argument instead of an expression. In that case, `.subs()` is not required. The derivative of a symbolic expression is taken, which is done analytically, using `sympy.diff()` where the first argument is the expression and the second is the variable with respect to which the derivative is taken.

1.3.3 Exercise

(a) **Simple Initial Guess (Easy)**
For $f(x) = x^2 - 2$, use Newton-Raphson Method with an initial guess $x_0 = 1.5$. Perform two iterations and calculate the approximate root.

(b) **Polynomial Root with Calculated Derivative (Easy-Medium)**
Given $f(x) = x^3 - 3x + 1$ and an initial guess $x_0 = 1$, use the Newton-Raphson Method to approximate the root. Complete at least three iterations, showing all steps.

(c) **Finding Root of Trigonometric Function (Medium)**
Apply the Newton-Raphson Method to find the root of $f(x) = \sin(x) - 0.5$ with an initial guess of $x_0 = 1$. Perform at least three iterations.

(d) **Higher Degree Polynomial (Medium-Hard)**

Use the Newton-Raphson Method to approximate a root for $f(x) = x^4 - x - 10$ with an initial guess of $x_0 = 2$. Perform four iterations and track the convergence of the solution.

(e) **Precision and Convergence Check (Hard)**

For $f(x) = e^x - 3x$ and initial guess $x_0 = 1$, use the Newton-Raphson Method to find the root. Continue until $|f(x_n)| < 0.0001$, and determine the number of iterations needed to reach this precision.

1.4 Fixed Point Iteration Method

The Fixed Point Iteration method is an iterative process to find the solution of equations of the form:

$$x = f(x)$$

Starting with an initial guess x_0, we then generate a sequence of values:

$$x_{n+1} = f(x_n)$$

This process is repeated until the sequence converges to a fixed point, i.e., when $x_{n+1} \approx x_n$.

1.4.1 Example 1

Consider the equation $x^3 - x + 1 = 0$. We can rearrange this equation in two different ways:

- **Using $x = g(x) = \sqrt[3]{x-1}$:** Starting with an initial guess x_0, the iteration becomes:

$$x_{n+1} = g(x_n) = \sqrt[3]{x_n - 1}$$

Now for $x_0 = -1$, the sequence becomes:

x_0	x_1	x_2	x_3	x_4	...
-1	-1.259921	-1.3122938367	-1.32235382	-1.3242687445515782	...

So this seems to converge at a point. If we iterate it for a few more steps, we will reach $x_{15} = -1.3247179572395291$, which is accurate up to 10 decimal points!

- **Using $x = f(x) = x^3 + 1$:** Starting with an initial guess x_0, the iteration becomes:

$$x_{n+1} = f(x_n) = x_n^3 + 1$$

Now for $x_0 = -1$, the sequence becomes:

x_0	x_1	x_2	x_3	x_4	...
-1	0	1	2	9	...

So this is diverging! Why? We can guess that, since $|x^3| > 1$ for $|x| > 1$, at some point of iteration, $|x|$ exceed 1 and hence the sequence converges.

1.4.2 Example 2

Now say we want to find a real root of $f(x) = x^2 - x - 1$. From our previous experience, we may want to write it in the form $x = \sqrt{x+1}$. And if we start from any $x_0 \geq -1$, we will approach the root $x = \frac{1+\sqrt{5}}{2}$.

However, there is another root $x = \frac{1-\sqrt{5}}{2}$, which can never be obtained using this form (why?). So we proceed with the form $x = -\sqrt{x+1}$ to find the negative root of $f(x)$ and the iteration is given by:

$$x_{n+1} = -\sqrt{x_n + 1}$$

Now for $x_0 = 0$, the sequence becomes:

x_0	x_1	x_2	x_3	x_4	x_5	...
0	-1	0	-1	0	-1	...

So this is neither diverging nor converging. The sequence is dancing between 0 and -1.
What if we start with $x_0 = 0.5$? We will find x_1, but $x_2, x_3, ..$ will be undefined.
However, starting with $x_0 = -0.5$ yields:

x_0	x_1	x_2	x_3	x_4	x_5
-0.5	-0.70710678	-0.5411961	-0.67735	-0.5680223	-0.65725

So the sequence seems approach $x = \frac{1-\sqrt{5}}{2} \approx -0.618034$.
Hence we conclude that $x_{n+1} = f(x_n)$ may generate a converging, diverging, or oscillating sequence (or undefined values) depending on the initial point x_0 and choice of $f(x)$. We will find out the reason soon.

1.4.3 Homework

Using the fixed point iteration method, we want to find the real solution of $x^3 + x^2 - 1 = 0$. The actual solution is $x \approx 0.75487766624669276$.
If we express it as $x = f(x)$, we can have either $f(x) = \sqrt[3]{1 - x^2}$ or $f(x) = \sqrt{1 - x^3}$. Find x_0 for both cases to generate sequences that converge, diverge, or oscillate.

1.4.4 Visualization

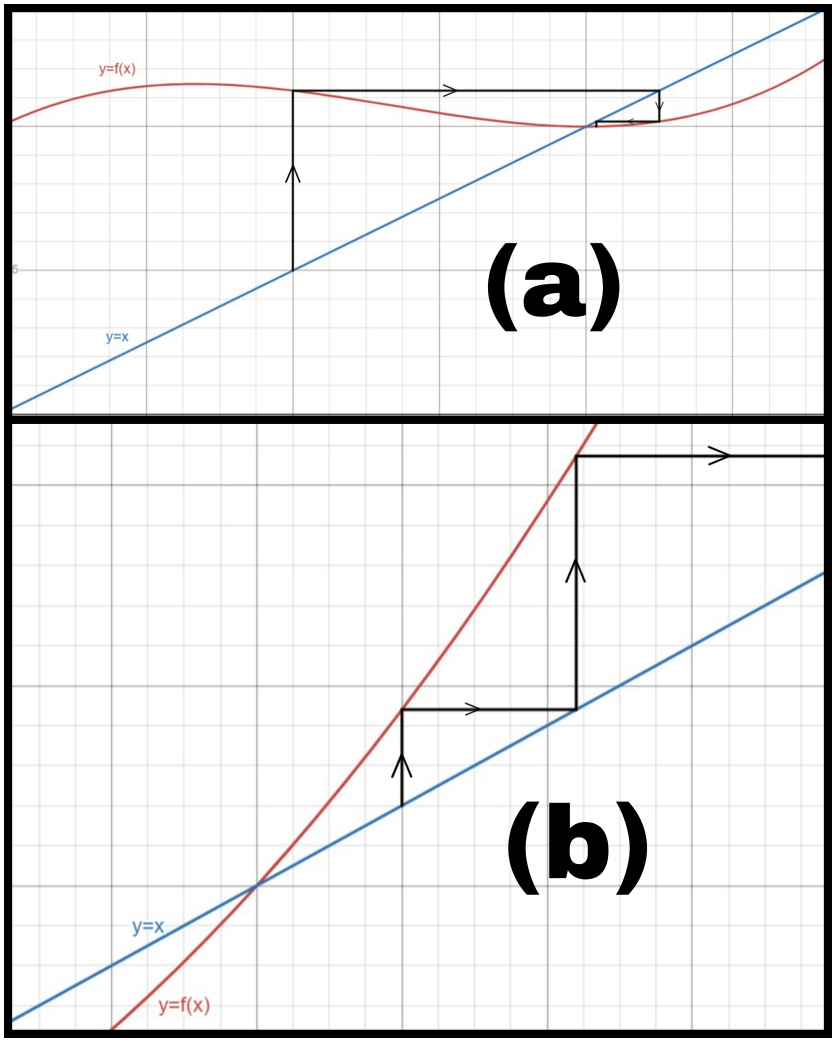

Figure 1.3: (a)Converging (b)Diverging

1.5 Banach Fixed-Point Theorem

From the previous examples, it should be apparent that f(x) plays a crucial role for the convergence of the series. When $f(x) = 1 + x^3$, the sequence diverges, but $f(x) = \sqrt[3]{x-1}$ make the sequence convergence. Why is so? The later seems to "contract" the sequence leading to a fixed point. We call $\phi : [a, b] \to [a, b]$ a contraction map if

$$|\phi(x) - \phi(y)| \leq k|x - y|, \forall x, y \in [a, b]$$

Where $0 \leq k < 1$ is called Lipschitz constant. We call $x = x_*$ a fixed point if $f(x_*) = x_*$.

From the definitions, it should be clear that the fixed point is a solution to $x = f(x)$. Now the **Banach fixed-point theorem** states that every contraction mapping on a non-empty complete metric space has a unique fixed point, and for any x_0, the sequence $x_{n+1} = f(x_n)$ converges to that fixed point, i.e.,

$$x_* = \phi\left(\lim_{n\to\infty} x_n\right) = \phi(x_*)$$

I won't bore you with the proof but let's define $\epsilon_k = x_k - x_*$, where x_*. Now,

$$x_{k+1} = \epsilon_{k+1} + x_* = \phi(x_k) = \phi(\epsilon_k + x_*)$$

Using Taylor expansion, $\phi(\epsilon_k + x_*) = \phi(x_*) + \phi'(x_*)\epsilon_k + O(\epsilon_k^2)$, and since $\phi'(x_*) = x_*$, we have,

$$\epsilon_{k+1} = \phi'(x_*)\epsilon_k + O(\epsilon_k^2) \approx \phi'(x_*)\epsilon_k$$

Note that ϵ_k is the error in each step, which will approach 0 if $|\phi'(x_*)| < 1$.
The Newton's method is given by:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Let, $g(x) = x - \frac{f(x)}{f'(x)}$, then $x_{k+1} = g(x_k)$, which is just fixed point iteration!

Chapter 2

Numerical Differentiation

2.1 Introduction

Numerical differentiation is a mathematical technique used to approximate the derivative of a function using discrete data points. It's particularly useful in situations where:

- The function is known only at specific data points (e.g., experimental measurements).
- An analytical expression of the derivative is difficult or impossible to obtain.
- The function is too complex to differentiate analytically.

2.1.1 Examples When Numerical Differentiation is Needed

1. **Engineering Applications**: Estimating the rate of heat transfer in a material when temperature measurements are taken at discrete points.
2. **Physics Experiments**: Calculating acceleration from position-time data collected during an experiment.
3. **Financial Modeling**: Determining the rate of change of stock prices using historical data.
4. **Biological Systems**: Estimating population growth rates from periodic census data.

2.2 Key Aspects and Intuition on Taylor Expansion

2.2.1 Taylor Expansion

The Taylor expansion is a powerful mathematical tool that expresses a function as an infinite sum of terms calculated from the derivatives of the function at a single point. For a function $f(x)$ that is infinitely differentiable at a point $x = x_0$, the Taylor series around x_0 is:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \frac{f'''(x_0)}{3!}(x - x_0)^3 + \cdots$$

2.2.2 Intuition Behind Taylor Expansion

- **Local Approximation**: The Taylor series provides a polynomial approximation of a function around a specific point. The more terms included, the closer the approximation to the actual function.

- **Foundation for Numerical Methods**: Many numerical techniques, including finite difference methods for differentiation, are derived from the Taylor expansion.

- **Error Estimation**: By truncating the series after a finite number of terms, we introduce a truncation error. The remainder term gives us an estimate of this error.

2.3 Deriving Finite Difference Formulas Using Taylor Expansion

Finite difference methods approximate derivatives by combining function values at specific points. The accuracy of these approximations depends on the order of the truncation error, commonly expressed using Big O notation (e.g., $O(h)$, $O(h^2)$).

2.3.1 First-Order Derivative Approximations

Forward Difference Scheme

First-Order Accuracy ($O(h)$) Using Taylor expansion around x:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(x)}{6}h^3 + \cdots$$

Rearranged to solve for $f'(x)$:

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x)}{2}h - \frac{f'''(x)}{6}h^2 + \cdots$$

Approximation:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad \text{with an error of} \quad O(h)$$

Second-Order Accuracy ($O(h^2)$) Consider the Taylor expansions of $f(x+h)$ and $f(x+2h)$:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(x)}{6}h^3 + \cdots$$
$$f(x+2h) = f(x) + 2f'(x)h + 2f''(x)h^2 + \frac{4f'''(x)}{3}h^3 + \cdots$$

Form a linear combination to eliminate $f''(x)$:

$$-4f(x+h) + 3f(x) + f(x+2h) = -2f'(x)h + O(h^3)$$

Solve for $f'(x)$:

$$f'(x) = \frac{-4f(x+h) + 3f(x) + f(x+2h)}{2h} + O(h^2)$$

Approximation:

$$f'(x) \approx \frac{-4f(x+h) + 3f(x) + f(x+2h)}{2h} \quad \text{with an error of} \quad O(h^2)$$

Backward Difference Scheme

First-Order Accuracy ($O(h)$) Using Taylor expansion around x:

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(x)}{6}h^3 + \cdots$$

Rearranged to solve for $f'(x)$:

$$f'(x) = \frac{f(x) - f(x-h)}{h} - \frac{f''(x)}{2}h + \frac{f'''(x)}{6}h^2 + \cdots$$

Approximation:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \quad \text{with an error of} \quad O(h)$$

Second-Order Accuracy ($O(h^2)$) Consider the Taylor expansions of $f(x-h)$ and $f(x-2h)$:

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(x)}{6}h^3 + \cdots$$
$$f(x-2h) = f(x) - 2f'(x)h + 2f''(x)h^2 - \frac{4f'''(x)}{3}h^3 + \cdots$$

Form a linear combination to eliminate $f''(x)$:

$$4f(x-h) - 3f(x) - f(x-2h) = 2f'(x)h + O(h^3)$$

Solve for $f'(x)$:

$$f'(x) = \frac{4f(x-h) - 3f(x) - f(x-2h)}{2h} + O(h^2)$$

Approximation:

$$f'(x) \approx \frac{4f(x-h) - 3f(x) - f(x-2h)}{2h} \quad \text{with an error of} \quad O(h^2)$$

Central Difference Scheme

Second-Order Accuracy ($O(h^2)$) Using Taylor expansions:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f^{(3)}(x)}{6}h^3 + \cdots$$
$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f^{(3)}(x)}{6}h^3 + \cdots$$

Subtract the second from the first:

$$f(x+h) - f(x-h) = 2f'(x)h + \frac{f^{(3)}(x)}{3}h^3 + \cdots$$

Rearranged to solve for $f'(x)$:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{f^{(3)}(x)}{6}h^2 + \cdots$$

Approximation:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \quad \text{with an error of} \quad O(h^2)$$

2.3.2 Second-Order Derivative Approximations

Forward Difference Scheme

First-Order Accuracy ($O(h)$) Using Taylor expansions:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f^{(3)}(x)}{6}h^3 + \cdots$$
$$f(x+2h) = f(x) + 2f'(x)h + 2f''(x)h^2 + \frac{4f^{(3)}(x)}{3}h^3 + \cdots$$

Form a combination:

$$f(x+2h) - 2f(x+h) + f(x) = 2f''(x)h^2 + O(h^3)$$

Solve for $f''(x)$:

$$f''(x) = \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} - \frac{f^{(3)}(x)}{3}h + \cdots$$

Approximation:

$$f''(x) \approx \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2} \quad \text{with an error of} \quad O(h)$$

Backward Difference Scheme

First-Order Accuracy ($O(h)$) Using Taylor expansions:

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f^{(3)}(x)}{6}h^3 + \cdots$$
$$f(x-2h) = f(x) - 2f'(x)h + 2f''(x)h^2 - \frac{4f^{(3)}(x)}{3}h^3 + \cdots$$

Form a combination:

$$f(x) - 2f(x-h) + f(x-2h) = 2f''(x)h^2 + O(h^3)$$

Solve for $f''(x)$:

$$f''(x) = \frac{f(x) - 2f(x-h) + f(x-2h)}{h^2} - \frac{f^{(3)}(x)}{3}h + \cdots$$

Approximation:

$$f''(x) \approx \frac{f(x) - 2f(x-h) + f(x-2h)}{h^2} \quad \text{with an error of} \quad O(h)$$

Central Difference Scheme

Second-Order Accuracy ($O(h^2)$) Using Taylor expansions:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f^{(3)}(x)}{6}h^3 + \frac{f^{(4)}(x)}{24}h^4 + \cdots$$
$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f^{(3)}(x)}{6}h^3 + \frac{f^{(4)}(x)}{24}h^4 - \cdots$$

Add the two equations:

$$f(x+h) + f(x-h) = 2f(x) + f''(x)h^2 + \frac{f^{(4)}(x)}{12}h^4 + \cdots$$

Rearranged to solve for $f''(x)$:

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{f^{(4)}(x)}{12}h^2 + \cdots$$

Approximation:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad \text{with an error of} \quad O(h^2)$$

2.4 Summary of Formulas

Derivative	**Scheme**	**Approximation Formula**	**Error Order**
First-order	Forward Difference	$f'(x) \approx \frac{f(x+h) - f(x)}{h}$	$O(h)$
First-order	Backward Difference	$f'(x) \approx \frac{f(x) - f(x-h)}{h}$	$O(h)$
First-order	Central Difference	$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$	$O(h^2)$
Second-order	Forward Difference	$f''(x) \approx \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2}$	$O(h)$
Second-order	Backward Difference	$f''(x) \approx \frac{f(x) - 2f(x-h) + f(x-2h)}{h^2}$	$O(h)$
Second-order	Central Difference	$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$	$O(h^2)$

2.5 Conclusion

Numerical differentiation is an essential tool in computational mathematics, enabling the approximation of derivatives using discrete data points. The finite difference methods derived from Taylor expansions provide a systematic approach to approximating both first and second-order derivatives with known error bounds.

Understanding the truncation errors associated with each scheme allows you to choose the most appropriate method for your specific application. The central difference scheme generally offers higher accuracy ($O(h^2)$) compared to forward and backward schemes ($O(h)$) for the same step size h.

By mastering these techniques, you can effectively tackle problems in engineering, physics, finance, and other fields where analytical differentiation is impractical.

2.6 References for Further Study

1. **"Numerical Methods for Engineers" by Steven C. Chapra and Raymond P. Canale**: This book provides a comprehensive introduction to numerical methods, including differentiation and integration techniques.
2. **"Applied Numerical Methods with MATLAB for Engineers and Scientists" by Steven C. Chapra**: This text offers practical insights into implementing numerical methods using MATLAB.
3. **Online Resources**:
   - *https://ocw.mit.edu/courses/find-by-topic/cat=engineering&subcat=civilengineering&spec=computationalmethods* MIT OpenCourseWare: Numerical Methods
   - *http://numerical.recipes/*Numerical Recipes: Numerical Differentiation

Feel free to delve into these resources to deepen your understanding of numerical differentiation and its applications.

Chapter 3

Numerical Methods for Integration

3.1 Trapezoidal Rule: Numerical Integration

The trapezoidal rule is a method used to approximate the definite integral of a function by summing the areas of trapezoids under the curve. This is particularly useful when the function is complex or its integral cannot be determined analytically. In such cases, numerical integration methods like the trapezoidal rule provide a practical solution.

3.1.1 Derivation of the Trapezoidal Rule

Consider a continuous function $f(x)$ defined over the interval $[a, b]$. To approximate the integral $\int_a^b f(x)\,dx$, we divide the interval $[a, b]$ into n subintervals of equal width $h = \frac{b-a}{n}$.

Let the points dividing the interval be $x_0 = a, x_1 = a + h, x_2 = a + 2h, ..., x_n = b$. For each subinterval, we approximate the area under the curve by the area of a trapezoid, where the height of the trapezoid is h, and the two parallel sides are the function values $f(x_i)$ and $f(x_{i+1})$.

The area of a trapezoid is given by:

$$\text{Area of trapezoid} = \frac{h}{2}[f(x_i) + f(x_{i+1})] \tag{3.1}$$

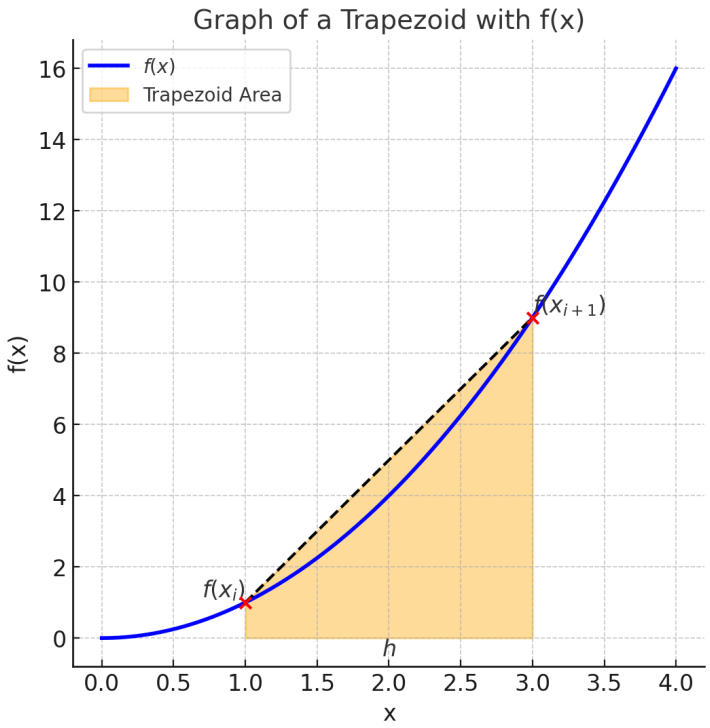

Figure 3.1: Approximating the area under a curve using trapezoids.

Summing the areas of all trapezoids gives us the approximation for the integral. We calculate the area of each trapezoid individually and sum them:

$$\frac{h}{2}\left(f(x_0)+f(x_1)\right)+\frac{h}{2}\left(f(x_1)+f(x_2)\right)+\cdots+\frac{h}{2}\left(f(x_{n-1})+f(x_n)\right)$$

By grouping the terms, we obtain:

$$=\frac{h}{2}\left(f(x_0)+2f(x_1)+2f(x_2)+\cdots+2f(x_{n-1})+f(x_n)\right)$$

This is the general form of the trapezoidal rule formula. The factor of 2 appears for the function values that are shared between adjacent trapezoids (i.e., $f(x_1)$ to $f(x_{n-1})$).

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left(f(x_0)+2f(x_1)+2f(x_2)+\cdots+2f(x_{n-1})+f(x_n)\right) \tag{3.2}$$

So, in advance words, we can write,

$$\int_a^b f(x)\,dx \approx \frac{h}{2}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right) \tag{3.3}$$

Where $h = \frac{b-a}{n}$ and $x_i = a + i \cdot h$. The use of smaller sub-intervals improves the accuracy of the approximation.

3.1.2 Steps for Applying the Trapezoidal Rule

1. **Determine the interval:** Choose the interval $[a, b]$ over which the integration is to be performed.
2. **Choose the number of sub-intervals n:** Increasing n improves accuracy.
3. **Calculate the step size h:** The width of each sub-interval is given by $h = \frac{b-a}{n}$.
4. **Evaluate the function at the endpoints and intermediate points:**
   - Calculate $f(a)$ and $f(b)$.
   - Compute the function at intermediate points $f(x_i)$ for $i = 1, 2, ..., n-1$.
5. **Apply the composite trapezoidal formula:** Plug in the values into the trapezoidal rule formula.

3.1.3 Example 1

Consider the function $f(x) = x^2$, and we want to approximate the integral $\int_0^1 x^2\,dx$ using 4 sub-intervals.

- $a = 0$, $b = 1$, $n = 4$
- $h = \frac{1-0}{4} = 0.25$
- Points: $x_0 = 0$, $x_1 = 0.25$, $x_2 = 0.5$, $x_3 = 0.75$, $x_4 = 1$
- Evaluate function: $f(0) = 0^2 = 0$, $f(0.25) = 0.0625$, $f(0.5) = 0.25$, $f(0.75) = 0.5625$, $f(1) = 1$
- Apply formula:

$$\int_0^1 x^2\,dx \approx \frac{0.25}{2}\,(0 + 2(0.0625 + 0.25 + 0.5625) + 1) = 0.34375$$

3.1.4 Example 2

Let's consider the function $f(x) = \sin(x)$ over the interval $[0, \pi]$. Using 6 sub-intervals, the width h is given by:

$$h = \frac{\pi - 0}{6} = \frac{\pi}{6} \tag{3.4}$$

Explanation of Sub-intervals: In this example, we are dividing the interval $[0, \pi]$ into 6 equal sub-intervals. The width of each sub-interval is $h = \frac{\pi}{6}$. This means that the points where the function is evaluated (i.e., $x_0, x_1, \ldots, x_6$) are:

$$x_0 = 0, \quad x_1 = \frac{\pi}{6}, \quad x_2 = \frac{2\pi}{6} = \frac{\pi}{3}, \quad x_3 = \frac{3\pi}{6} = \frac{\pi}{2}, \quad \ldots, \quad x_6 = \pi$$

These points correspond to the function values at which we apply the trapezoidal rule. For example, $f(x_1) = \sin\left(\frac{\pi}{6}\right)$, $f(x_2) = \sin\left(\frac{\pi}{3}\right)$, and so on.

Applying the composite trapezoidal rule, we calculate:

$$\int_0^{\pi} \sin(x)\, dx \approx \frac{\frac{\pi}{6}}{2} \left[\sin(0) + 2(\sin(\frac{\pi}{6}) + \sin(\frac{\pi}{3}) + \ldots) + \sin(\pi)\right] = 2 \tag{3.5}$$

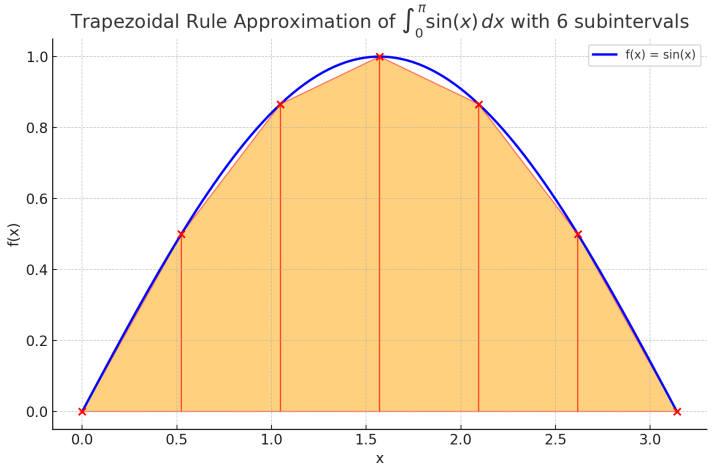

Figure 3.2: Trapezoidal Rule Approximation of $\int_0^{\pi} \sin(x)\, dx$ with 6 sub-intervals.

In the graph above, the blue curve represents the function $f(x) = \sin(x)$, and the orange trapezoids represent the areas approximated by the trapezoidal rule. By summing the areas of the trapezoids, we obtain a numerical approximation of the integral.

3.1.5 Exercises

1. Use the trapezoidal rule to approximate the integral of $f(x) = e^x$ from 0 to 1 using 4 sub-intervals.
2. Approximate $\int_0^2 (x^3 - 3x + 2)\,dx$ using 5 sub-intervals.
3. Approximate $\int_1^4 \ln(x)\,dx$ using 6 sub-intervals.
4. A car's velocity $v(t)$ (in m/s) is measured at several intervals over a 10-second period. The values are provided below. Estimate the total distance traveled by the car over 10 seconds using the trapezoidal rule with 5 sub-intervals.

Time t (s)	0	2	4	6	8	10
Velocity $v(t)$ (m/s)	0	4	8	7	3	0

5. The cross-sectional area $A(d)$ (in m^2) of a reservoir is recorded at different depths d. The data is provided below. Approximate the total volume of water in the reservoir using the trapezoidal rule with 5 sub-intervals.

Depth d (m)	0	2	4	6	8	10
Area $A(d)$ (m^2)	200	180	140	100	50	10

3.1.6 Conclusion

The trapezoidal rule provides a simple yet powerful tool for approximating definite integrals, particularly when an analytical solution is difficult to obtain. By dividing the interval into smaller sub-intervals and applying the trapezoidal formula, we can achieve higher precision in our approximations.

3.2 Simpson's Rule

3.2.1 What is Simpson's Rule?

Simpson's Rule is another numerical approach for finding the approximate value of a definite integral, or in much simpler terms, evaluating the approximate area under the curve for a

given range. The value obtained will be an approximation, and therefore, is likely to contain a small percentage of error.

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

Let us assume we have the definite integral - given above - where the function, $f(x)$ is divided into n equal parts over its interval (a, b). Please note that n will always be even! Note that each interval with have a width of Δx, where:

$$\Delta x = \frac{b-a}{n}$$

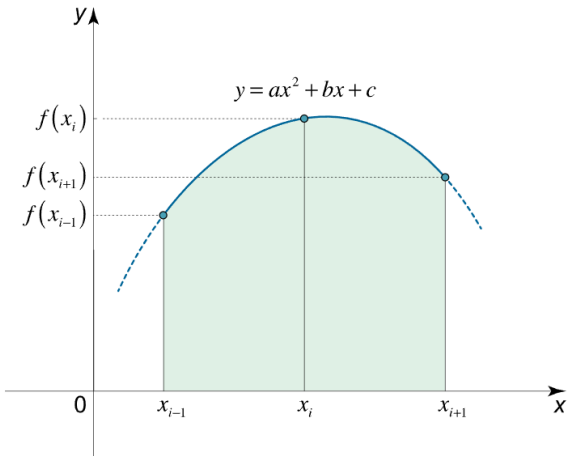

Figure 3.3: The area under a curve by divided into n equal sub-intervals

Unlike Riemann Sums and the Trapezoidal Rule, the method of approximating area relies on using parabolas. Simpson's Rule assumes that small sections of the curve can be well-approximated by parabolas. So, it uses parabolas instead of straight lines to better capture the curve's shape. This is done by taking three or more points at a time and fitting a parabola through them. These points are coordinates / observations, using which we will fit a parabola over these points in order to create a continuous function. In the figure above, on each pair of consecutive sub-intervals $[x_{i-1}, x_i]$, $[x_i, x_{i+1}]$, we consider a quadratic function $y = ax^2 + bx + c$ such that it passes through the points $(x_{i-1}, f(x_{i-1}))$, $(x_i, f(x_i))$, $(x_{i+1}, f(x_{i+1}))$. Our parabola is now fitted as this continuous function $f(x)$ over the range $[a, b]$.

3.2.2 Why Do We Need Simpson's Rule When Simple Formulas for Evaluating Integrals Exist?

We can solve definite integrals using integration rules and formulas, substituting the limits at the end. However, sometimes this isn't possible with basic steps, and we only have specific observed values of the function instead of a clear formula. In such cases, we use numerical methods to approximate the integral. For definite integrals, we are calculating the area under the curve within set limits. Simpson's Rule is a useful method for approximating this area when exact integration is difficult.

3.2.3 Variations & Formulas

Simpson's 1/3 Rule

In Simpson's rule, we use three equally spaced points for finding a fitting polynomial and the endpoints are two of them. Thus Simpson's rule is also called the 3-point closed rule. Let us see the derivation of Simpson's $\frac{1}{3}$ rule.

Formula:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right]$$

Here, $\Delta x = \frac{b-a}{n}$, and n is the number of sub-intervals that must be even.

In simple words, we can write the formula as,

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left[F + L + (4 \times (\text{Odd Observations})) + 2 \times (\text{Even Observations})\right]$$

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left[(y_0 + y_n) + 4(y_1 + y_3 + y_5 + \cdots + y_{n-1}) + 2(y_2 + y_4 + y_6 + \cdots + y_{n-2})\right]$$

where, $F = \text{First term} = y_0,\ L = \text{Last term} = y_n$.

Here is a step-by-step explanation on how to apply Simpson's $1/3$ rule for approximating the integral $\int_a^b f(x)\,dx$:

- **Step 1:** Identify the values of 'a' and 'b' from the interval $[a, b]$, and identify the value of 'n' which is the number of sub-intervals.
- **Step 2:** Use the formula $\Delta x = \frac{b-a}{n}$ to calculate the width of each sub-interval.
- **Step 3:** Divide the interval $[a, b]$ into 'n' sub-intervals $[x_0, x_1], [x_1, x_2], [x_2, x_3], \ldots, [x_{n-2}, x_{n-1}], [x_{n-1}, x_n]$ using the interval width Δx.
- **Step 4:** Find the corresponding y values $[y_0, y_1, y_2, y_3], \ldots, [y_{n-2}, y_{n-1}, y_n]$ using the given formula or set of coordinates.
- **Step 5:** Carefully separate the first and last terms, and place the rest of the terms in the appropriate positions in the formula.
- **Step 6:** Substitute all these values in Simpson's 1/3 rule formula and simplify.

Formula Derivation Video: https://www.youtube.com/watch?v=IGC_tEOsdQ8

3.2.4 Simpson's 3/8 Rule

Another approximation method is known as Simpson's 3/8 rule. This rule is more accurate than Simpson's 1/3 rule because it uses cubic interpolation (3rd order polynomials) instead of quadratic interpolation. It incorporates an additional functional value, allowing it to provide more precise results than other methods.

Formula:

$$\int_a^b f(x)\,dx \approx \frac{3\Delta x}{8}\left[f(x_0) + f(x_n) + 2 \times (f(x_3) + f(x_6) + \ldots) + 3 \times (f(x_1) + f(x_2) + f(x_4) + \ldots)\right] \tag{3.6}$$

Here, $\Delta x = \frac{b-a}{n}$, and n is the number of sub-intervals but it has to be a multiple of 3.

In simple words, we can write the formula as,

$$\int_a^b f(x)\,dx \approx \frac{3\Delta x}{8}\left[F + L + 2 \times (\text{Multiples of 3}) + 3 \times (\text{Non multiples of 3})\right]$$

$$\int_a^b f(x)\,dx \approx \frac{3\Delta x}{8}\left[(y_0 + y_n) + 2(y_3 + y_6 + y_9 \cdots + y_{n-3}) + 3(y_1 + y_2 + y_4 + \cdots + y_{n-1})\right]$$

where, F = First term = y_0, L = Last term = y_n.

Here is a step-by-step explanation on how to apply Simpson's 3/8 rule for approximating the integral $\int_a^b f(x)\,dx$:

- **Step 1:** Identify the values of 'a' and 'b' from the interval $[a, b]$, and identify the value of 'n' which is the number of sub-intervals.
- **Step 2:** Use the formula $\Delta x = \frac{b-a}{n}$ to calculate the width of each sub-interval.
- **Step 3:** Divide the interval $[a, b]$ into 'n' sub-intervals $[x_0, x_1], [x_1, x_2], [x_2, x_3], \ldots, [x_{n-2}, x_{n-1}], [x_{n-1}, x_n]$ using the interval width Δx.
- **Step 4:** Separate the first and last terms, and group the rest of the f(x) / y values which are ordered as multiples of 3 (y_3, y_6 , y_9...) and separately group all other terms, ordered as non-multiples of 3 (y_1, y_2 , y_4, y_5, y_7 , y_8....)
- **Step 5:** Substitute all these values in Simpson's 3/8 rule formula and simplify.

Formula Derivation Video: https://www.youtube.com/watch?v=VldXegup5d0

3.2.5 Error in Simpson's Rule

Numerical approximations inherently have errors, as they are estimates, not exact values. Although Simpson's rule provides a very accurate result with minimal error, we can calculate the error using the formula below.

$$\text{Error in Simpson's 1/3 Rule: } E \leq \frac{(b-a)^5}{180n^4}|f^{(4)}(x)|$$

Where:

E : Error in the approximation

a, b : Limits of integration

n : Number of sub-intervals (must be even)

$|f^{(4)}(x)|$: Maximum value of 4th derivative at some point 'x' over the interval $[a, b]$

$$\text{Error in Simpson's 3/8 Rule: } E \leq \frac{(b-a)^5}{6480n^4}|f^{(4)}(x)|$$

Where:

E : Error in the approximation

a, b : Limits of integration

n : Number of sub-intervals (must be a multiple of 3)

$|f^{(4)}(x)|$: Maximum value of 4th derivative at some point 'x' over the interval $[a, b]$

3.2.6 Comparing 1/3 Rule to 3/8 Rule:

The difference between Simpson's $\frac{1}{3}$ rule and Simpson's $\frac{3}{8}$ rule is given below:

Simpson's $\frac{1}{3}$ Rule	**Simpson's $\frac{3}{8}$ Rule**
Approximates a function by a 2nd order polynomial.	Approximates a function by a 3rd order polynomial.
Number of sub-intervals (n) must be a multiple of 2.	Number of sub-intervals (n) must be a multiple of 3.
Less accurate than Simpson's $\frac{3}{8}$ rule.	More accurate results.

3.2.7 Advantages & Disadvantages:

Advantages	**Disadvantages**
Gives a better approximation than Trapezoidal and Midpoint rules.	Not fully accurate; there is always an error between Simpson's rule and the actual integral.
It uses quadratic approximations instead of linear, giving more accurate results.	Integrals provide exact answers for constants, which isn't always possible with Simpson's rule.

3.2.8 Supplementary Videos

- **Simpson's Rule & Numerical Integration**: https://www.youtube.com/watch?v=7EqRRuh-5Lk

- **Simpsons Rule - Approximate Integration**: https://www.youtube.com/watch?v=ns3k-Lz7qWU

3.2.9 Worked Examples

Simpson's 1/3 Rule:

Example 1

Use Simpson's Rule with $n = 4$ to approximate the integral

$$\int_0^8 \sqrt{x}\, dx.$$

Solution:

Let us first calculate the width of each sub-interval:

$$\Delta x = \frac{b-a}{n} = \frac{8-0}{4} = 2,$$

and the endpoints x_i have coordinates

$$x_i = \{0, 2, 4, 6, 8\}.$$

Calculate the function values at the points x_i:

$$\begin{aligned} f(x_0) &= f(0) = \sqrt{0} = 0; \\ f(x_1) &= f(2) = \sqrt{2}; \\ f(x_2) &= f(4) = \sqrt{4} = 2; \\ f(x_3) &= f(6) = \sqrt{6}; \\ f(x_4) &= f(8) = \sqrt{8} = 2\sqrt{2}. \end{aligned}$$

Substitute all these values into the Simpson's Rule formula:

$$\begin{aligned}\int_0^8 \sqrt{x}\,dx &\approx \frac{\Delta x}{3}\left[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+f(x_4)\right]\\ &\approx \frac{2}{3}\left[0+4\cdot\sqrt{2}+2\cdot 2+4\cdot\sqrt{6}+2\sqrt{2}\right]\\ &\approx \frac{2}{3}\left[0+6\cdot\sqrt{2}+4+4\cdot\sqrt{6}\right]\\ &\approx 14.86\end{aligned}$$

The true solution for the integral is

$$\int_0^8 \sqrt{x}\,dx = \int_0^8 x^{\frac{1}{2}}\,dx = \frac{2}{3}x^{\frac{3}{2}}\Big|_0^8 = \frac{2}{3}\left[\sqrt{8}^3\right] = \frac{2}{3}\cdot 8\sqrt{2} = \frac{2}{3}\cdot 16\sqrt{2} = \frac{32\sqrt{2}}{3} \approx 15.08.$$

Hence, the error in approximating the integral is

$$|\varepsilon| = \frac{|15.08-14.86|}{15.08} \approx 0.015 = 1.5\%.$$

Example 1

A function $f(x)$ is given by the table of values. Approximate the area under the curve $y=f(x)$ between $x=0$ and $x=4$ using Simpson's Rule with $n=4$ sub-intervals.

x	0	1	2	3	4
$f(x)$	2	7	12	10	5

Solution:

For $n=4$ sub-intervals, Simpson's rule is given by the following equation:

$$S_4 = \frac{\Delta x}{3}\left[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+f(x_4)\right].$$

The width of the sub-interval is : $\Delta x = \frac{b-a}{n} = \frac{4-0}{4} = 1$.

Substitute the values of the function from the table and calculate the approximate value of the area under the curve:

$$
\begin{aligned}
A &\approx \frac{1}{3}\left[2 + 4 \cdot 7 + 2 \cdot 12 + 4 \cdot 10 + 5\right] \\
&\approx \frac{1}{3}\left[2 + 28 + 24 + 40 + 5\right] \\
&\approx \frac{1}{3} \cdot 99 \\
&\approx 33
\end{aligned}
$$

To calculate the error using the error formula:

$$
Error = \frac{(b-a)^5}{180n^4} f^{(4)}(c)
$$

The limits are $b = 4$ and $a = 0$

The step size is $n = 4$

$f^{(4)}(c)$ is the fourth derivative of the function $f(x)$ evaluated at some point c in the interval $[a, b]$.

So plugging these into the formula:

$$
\begin{aligned}
\text{Error} &\approx \frac{(4)^5}{180 \cdot (4)^4} f^{(4)}(c) \\
&\approx \frac{1024}{180 \cdot 256} f^{(4)}(c) \\
&\approx \frac{4}{180} f^{(4)}(c) \\
&\approx \frac{1}{45} f^{(4)}(c)
\end{aligned}
$$

The value of the error depends on the fourth derivative $f^{(4)}(c)$ of $f(x)$ at some point c in $[0, 4]$. Without an explicit form of $f(x)$, we cannot calculate $f^{(4)}(c)$ exactly.

Example 3

Approximate the area under the curve $y = f(x)$ between $x = -1$ and $x = 5$ using Simpson's Rule with $n = 6$ sub-intervals.

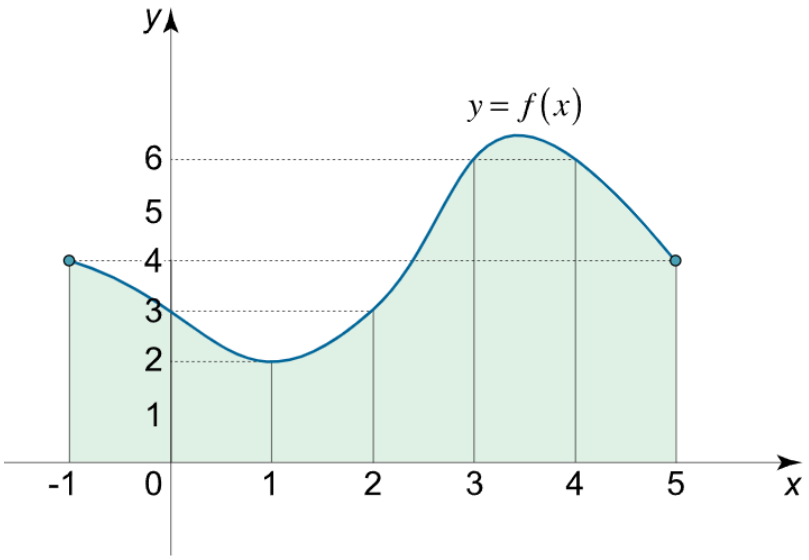

Figure 3.4: The function $y = f(x)$ with shaded area under the curve.

Solution:

The Simpson's Rule formula for $n = 6$ sub-intervals is given by

$$S_6 = \frac{\Delta x}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + 4f(x_5) + f(x_6)\right].$$

The function values at the endpoints of the intervals are

$$f(x_0) = f(-1) = 4;$$
$$f(x_1) = f(0) = 3;$$
$$f(x_2) = f(1) = 2;$$
$$f(x_3) = f(2) = 3;$$
$$f(x_4) = f(3) = 6;$$
$$f(x_5) = f(4) = 6;$$
$$f(x_6) = f(5) = 4.$$

So, the approximate value of the area under the curve is

$$A \approx \frac{1}{3}[4 + 4 \cdot 3 + 2 \cdot 2 + 4 \cdot 3 + 2 \cdot 6 + 4 \cdot 6 + 4]$$

$$A \approx \frac{1}{3}[4 + 12 + 4 + 12 + 12 + 24 + 4]$$

$$A \approx \frac{1}{3} \cdot 72$$

$$A \approx 24$$

Simpson's 3/8 Rule:

Example 1

Approximate the area under the curve $y = f(x)$ using the given observations of x and $f(x)$ using Simpson's 3/8 Rule, where $\Delta x = 0.2$

Given values of x and $f(x)$:

x	$f(x)$
1.4	4.0552
1.6	4.9530
1.8	6.0436
2.0	7.3891
2.2	9.0250

Solution:

Applying Simpson's $\frac{3}{8}$ Rule:

$$\int y\,dx \approx \frac{3 \cdot \Delta x}{8}[(y_0 + y_4) + 2(y_3) + 3(y_1 + y_2)]$$

Substituting the values:

$$\int y\,dx \approx \frac{3 \cdot 0.2}{8}[(4.0552 + 9.025) + 2 \cdot (7.3891) + 3 \cdot (4.953 + 6.0436)]$$

$$\approx \frac{3 \cdot 0.2}{8}[(13.0802) + 2 \cdot (7.3891) + 3 \cdot (10.9966)]$$

$$\approx 4.5636$$

Let us also calculate the error:

$$Error \approx \frac{(b-a)^5}{6840n^4}|f^{(4)}(c)|$$

Interval : $a = 1.4$, $b = 2.2$

Step size: $\Delta x = \frac{b-a}{n} = \frac{0.8}{4} = 0.2$

Substitute values into the Error Formula:

$$\begin{aligned}
\text{Error} &\approx \frac{(2.2-1.4)^5}{6480 \cdot (0.2^4)} f^{(4)}(c) \\
&\approx \frac{0.8^5}{6480 \cdot 0.0016} f^{(4)}(c) \\
&\approx \frac{0.32768}{6480 \cdot 0.0016} f^{(4)}(c) \\
&\approx \frac{0.32768}{10.368} f^{(4)}(c) \\
&\approx 0.0316\, f^{(4)}(c)
\end{aligned}$$

To obtain a numerical value for the error, you would need the value of $f^{(4)}(c)$, the fourth derivative of $f(x)$ at some point c within $[1.4, 2.2]$.

Example 2

Solve the integral $\int_1^2 \frac{1}{x}\,dx$ using Simpson's $\frac{3}{8}$ rule where n = 5

Solution:

Our equation is $f(x) = \frac{1}{x}$ and the number of sub-intervals = 5, so let us make a table for x and y

x	1	1.25	1.5	1.75	2
y	1	0.8	0.6667	0.5714	0.5

Applying Simpson's $\frac{3}{8}$ Rule:

$$\int f(x)\,dx = \frac{3 \cdot h}{8}\left[(y_0 + y_4) + 2 \cdot (y_3) + 3 \cdot (y_1 + y_2)\right]$$

Substitute the values:

$$\begin{aligned}\int f(x)\,dx &= \frac{3 \cdot 0.25}{8}\left[(1+0.5)+2\cdot(0.5714)+3\cdot(0.8+0.6667)\right] \\ &= \frac{3 \cdot 0.25}{8}\left[(1+0.5)+2\cdot(0.5714)+3\cdot(1.4667)\right] \\ &= 0.6603\end{aligned}$$

Let us calculate the error, where c = 1, as $f^{(4)}(c)$ will be maximum at that point:

Since $f(x) = \frac{1}{x}$, we calculate its fourth derivative:

$$\begin{aligned}f^{(1)}(x) &= -\frac{1}{x^2} \\ f^{(2)}(x) &= \frac{2}{x^3} \\ f^{(3)}(x) &= -\frac{6}{x^4} \\ f^{(4)}(x) &= \frac{24}{x^5}\end{aligned}$$

Substitute the values into the error equation:

$$\begin{aligned}\text{Error} &\approx \frac{(b-a)^5}{6480 \cdot n^4} \cdot f^{(4)}(c) \\ &\approx \frac{(2-1)^5}{6480 \cdot 5^4} \cdot \frac{24}{1^5} \\ &\approx \frac{24}{4050000} = 0.00000593\end{aligned}$$

3.2.10 Python codes for Simpson's rule

Simpson's 1/3 Rule:

Evaluate $\int logx\,dx$ within limit 4 to 5.2, n = 6

```
# Python code for simpson's 1 / 3 rule
import math

# Function to calculate f(x)
def func( x ):
    return math.log(x)

# Function for approximate integral
def simpsons_( ll, ul, n ):

    # Calculating the value of h
    h = ( ul - ll )/n

    # List for storing value of x and f(x)
    x = list()
    fx = list()

    # Calculating values of x and f(x)
    i = 0
    while i<= n:
        x.append(ll + i * h)
        fx.append(func(x[i]))
        i += 1

    # Calculating result
    res = 0
    i = 0
    while i<= n:
        if i == 0 or i == n:
            res+= fx[i]
        elif i % 2 != 0:
            res+= 4 * fx[i]
        else:
            res+= 2 * fx[i]
        i+= 1
    res = res * (h / 3)
    return res

# Driver code
lower_limit = 4   # Lower limit
upper_limit = 5.2 # Upper limit
n = 6 # Number of interval
print("%.6f"% simpsons_(lower_limit, upper_limit, n))
```

```
1.827847
```

Figure 3.5: Solution in Code

Simpson's 3/8 Rule:

Evaluate $\int \frac{1.0}{1+x^2}\,dx$ within limit 1 to 10, n = 10

```
def func(x):
  return (float(1) / ( 1 + x * x ))

# Function to perform calculations
def calculate(lower_limit, upper_limit, interval_limit ):

  interval_size = (float(upper_limit - lower_limit) / interval_limit)
  sum = func(lower_limit) + func(upper_limit);

  # Calculates value till integral limit
  for i in range(1, interval_limit ):
    if (i % 3 == 0):
      sum = sum + 2 * func(lower_limit + i * interval_size)
    else:
      sum = sum + 3 * func(lower_limit + i * interval_size)

  return ((float( 3 * interval_size) / 8 ) * sum )

# driver function
interval_limit = 10
lower_limit = 1
upper_limit = 10
integral_res = calculate(lower_limit, upper_limit, interval_limit)

# rounding the final answer to 6 decimal places
print (round(integral_res, 6))
```

```
0.687927
```

Figure 3.6: Solution in Code

3.3 Monte Carlo Integration: Methods and Applications

Monte Carlo integration is a technique for approximating the value of definite integrals using random sampling. It is particularly useful when traditional numerical methods (like the trapezoidal or Simpson's rule) are difficult to apply, especially in higher dimensions.

Monte Carlo integration is based on the principle of simulating random variables to estimate the area under a curve. It can be applied to a wide variety of problems, particularly in physics, statistics, and computer science.

In this section, we will discuss two popular methods for Monte Carlo integration:

- The averaging method
- The sampling method

3.3.1 Averaging Method

In the averaging method, the idea is to randomly sample points within the interval of interest, evaluate the function at those points, and then take the average of these function values to estimate the integral.

Steps for the Averaging Method

Suppose we want to approximate the integral of a function $f(x)$ over the interval $[a, b]$:

$$I = \int_a^b f(x)\,dx$$

The steps for the averaging method are as follows:

1. **Generate random points:** Randomly generate N points $x_1, x_2, \ldots, x_N$ uniformly over the interval $[a, b]$.

2. **Evaluate the function:** Evaluate the function $f(x)$ at each of the sampled points.

3. **Compute the average:** Compute the average of these values:

$$\frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

4. **Estimate the integral:** Multiply the average by the length of the interval $(b - a)$ to estimate the integral:

$$I \approx (b - a) \cdot \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

Why It Works

The averaging method is based on the law of large numbers, which states that as the number of random points increases, the average of the values approaches the expected value of the function. Since the integral is essentially the expected value of the function over the interval, this method provides an accurate estimate of the integral as the number of points, N, increases.

Example 1: Estimating the Integral of $f(x) = e^x$

Let's use the averaging method to estimate the integral of $f(x) = e^x$ over the interval $[0, 1]$. Follow the steps outlined above:

$$I = \int_0^1 e^x \, dx$$

1. Generate random points: - We randomly generate $N = 1000$ points uniformly in the interval $[0, 1]$. These points are denoted as $x_1, x_2, \ldots, x_N$.

2. Evaluate the function: - For each of the sampled points x_i, evaluate $f(x_i) = e^{x_i}$.

3. Compute the average: - Compute the average value of $f(x_i)$ for all the randomly sampled points:

   $$\frac{1}{N} \sum_{i=1}^{N} e^{x_i}$$

   After computing this sum, we find that the average value of $f(x)$ at these random points is approximately 1.718.

4. Estimate the integral: - Multiply the average value by the length of the interval $(b - a)$, which is 1, to estimate the integral:

   $$I \approx (1 - 0) \cdot 1.718 = 1.718$$

The exact value of the integral is:

$$\int_0^1 e^x \, dx = e - 1 = 1.718$$

The estimated value is very close to the true value, illustrating the accuracy of the Monte Carlo averaging method.

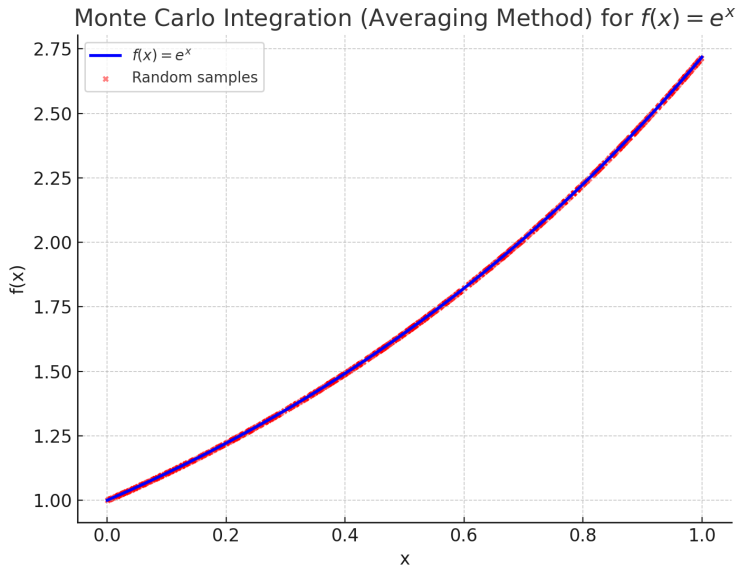

Figure 3.7: Monte Carlo Integration for $f(x) = e^x$ using the averaging method.

Example 2: Estimating the Integral of $f(x) = \sin(x)$ over $[0, \pi]$

Let's estimate the integral of $f(x) = \sin(x)$ over the interval $[0, \pi]$ using the same steps.

$$I = \int_0^{\pi} \sin(x)\, dx$$

Steps:

1. Generate random points: - Randomly generate $N = 1000$ points uniformly in the interval $[0, \pi]$.

2. Evaluate the function: - Evaluate $f(x_i) = \sin(x_i)$ at each of the randomly sampled points.

3. Compute the average: - Compute the average of $f(x_i)$ over all sampled points:

$$\frac{1}{N} \sum_{i=1}^{N} \sin(x_i)$$

The average value of $f(x) = \sin(x)$ at these random points is approximately 0.637.

4. Estimate the integral: - Multiply the average by the length of the interval $(\pi - 0)$ to estimate the integral:

$$I \approx \pi \cdot 0.637 = 2.001$$

The exact value of the integral is 2, so the Monte Carlo method provides a very close approximation.

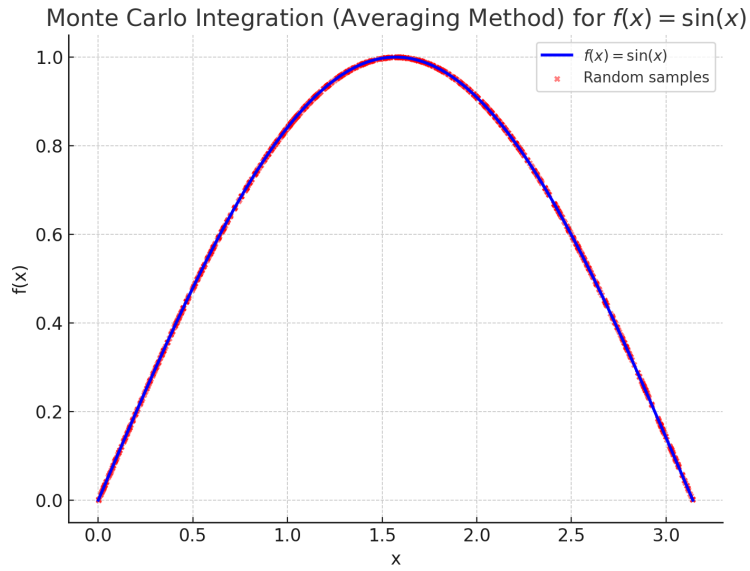

Figure 3.8: Monte Carlo Integration for $f(x) = \sin(x)$ using the averaging method.

3.3.2 Sampling Method

In the sampling method, we take a different approach. Instead of just averaging function values over an interval, we randomly sample points both in the x-axis and the y-axis. We count how many points fall under the curve to estimate the integral.

Steps for the Sampling Method

Let's say we want to approximate the same integral $I = \int_a^b f(x)\,dx$:

1. Randomly generate N points (x_i, y_i) such that x_i is uniformly distributed in $[a, b]$, and y_i is uniformly distributed in $[0, \max(f(x))]$.

2. Count how many points (x_i, y_i) fall under the curve, i.e., where $y_i \leq f(x_i)$.

3. The ratio of points under the curve to the total points approximates the area under the curve:

$$\text{Integral} \approx \frac{\text{Number of points under the curve}}{N} \times (b-a) \times \max(f(x))$$

Example 1: Hypothetical Example for Sampling Method

Let's consider estimating the integral of $f(x) = \sin(x)$ over the interval $[0, \pi]$ using the sampling method.

- Suppose we generate 1000 random points in the 2D plane where x is distributed uniformly in $[0, \pi]$ and y is distributed in $[0, 1]$. - Out of these 1000 points, we find that 700 points lie below the curve $y = \sin(x)$, and 300 points lie above the curve.

The total number of points under the curve is 700. Thus, the ratio of points under the curve to the total points is:

$$\frac{700}{1000} = 0.7$$

The length of the interval is π, and the maximum value of $\sin(x)$ is 1, so the integral is approximated as:

$$I \approx \pi \times 1 \times 0.7 = 2.199$$

The exact value of the integral is 2, so this hypothetical example shows that we can closely estimate the integral using the sampling method.

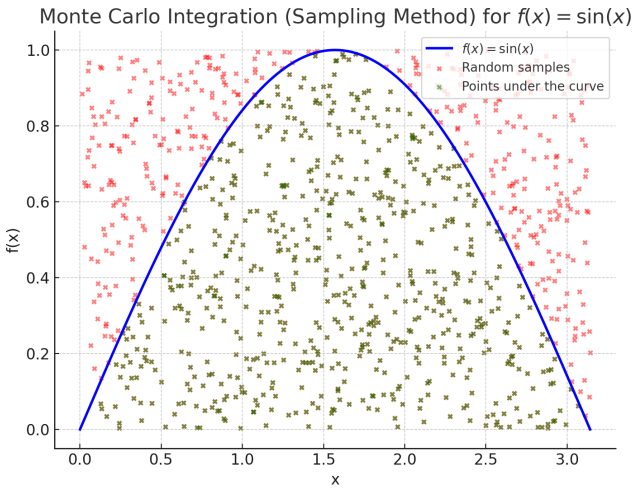

Figure 3.9: Monte Carlo Integration using the sampling method. Random points are sampled, and the fraction of points under the curve is used to estimate the integral.

3.3.3 Comparison of Methods

Both the averaging and sampling methods are powerful, especially in higher dimensions, where other numerical methods struggle. However, they may converge more slowly than other techniques, particularly when the function is highly irregular.

- **Averaging method:** Easy to implement and can handle smooth functions very well. It is generally preferred when we can sample points along the x-axis efficiently.
- **Sampling method:** More versatile, as it can be used to approximate the area under a curve by counting points, even if the function has complex shapes.

3.3.4 Conclusion

Monte Carlo integration provides a flexible, powerful approach to estimating definite integrals, especially in higher dimensions where other methods fall short. While it may not always be the fastest converging method, its simplicity and versatility make it an essential tool in numerical analysis.

Chapter 4

Numerical Methods for Solving Ordinary Differential Equations

An Ordinary Differential Equation or ODE is an equation that relates to a differential equation that contains only one independent variable. Many problems in physics, engineering, biology, and economics are modeled using ODEs such as wave equations, circuit designing, population model, growth model etc.

Many ODEs cannot be solved analytically (i.e., with an exact solution). For such cases, we use numerical methods like Euler's method, Runge-Kutta methods, etc., to approximate the solution.

4.1 Euler's Method

The Euler's method is one of the simplest and most basic numerical methods used to solve ordinary differential equations (ODEs). Despite its simplicity, it is a foundational tool in numerical analysis. It uses a straightforward iterative process to approximate the solution of a differential equation.

Let's say we have a first-order ODE

$$\frac{dy(x)}{dx} = f(x, y) \tag{4.1}$$

with the initial condition: $y(x_0) = y_0$. Where $f(x, y)$ is any given function in the problem. We have to solve for $y(x)$.

We have $y_i \equiv y(x_i)$ at point x_i. At point x_{i+1} we have $y_{i+1} \equiv y(x_{i+1}) = y(x_i + h)$ after a small change $h = x_{i+1} - x_i$. The approximate forward difference formula for the first derivative is given by,

$$\begin{aligned} \frac{dy}{dx} &\approx \frac{y(x_i + h) - y(x_i)}{h} \\ &= \frac{y_{i+1} - y_i}{h} \end{aligned} \tag{4.2}$$

With this, (4.10) becomes,

$$\frac{y_{i+1} - y_i}{h} = f(x_i, y_i) \tag{4.3}$$

Hence, we find formula for subsequent y_{i+1} from the previous y_i:

$$y_{i+1} = y_i + hf(x_i, y_i) \tag{4.4}$$

This is Euler's formula for solving first-order ODE.

To use this formula we start with the initial condition given $y(x_0) = y_0$ at x_0. We can obtain $y_1 \equiv y(x_1)$ using Euler formula given by, $y_1 = y_0 + hf(x_0, y_0)$. And then use this to find $y_2 = y_1 + hf(x_1, y_1)$ and so on. We need to define the step size $h = x_{i+1} - x_i$ which should be a small number. The starting point is x_0 and the end point is x_n.

4.1.1 Example:

To illustrate Euler's method, we consider the differential equation $y' = -y$ with the initial condition $y(0) = 1$. Equation (4) with $h = 0.01$ gives,

$$y(0.01) = 1 + 0.1(-1) = 0.99$$

$$y(0.02) = 0.99 + 0.01(-0.99) = 0.9801$$

$$y(0.03) = 0.9801 + 0.01(-0.9801) = 0.9703$$

$$y(0.04) = 0.9703 + 0.01(-0.9703) = 0.9606$$

The exact solution is $y = e^{-x}$ and from this the value at $x = 0.04$ is $y = 0.9608$

Problem 1: Decay Equation

Consider a sample of radioactive material with an initial N_0 number of nuclei. As time passes, some of the nuclei will decay into counterparts. Our job is to predict the number of nuclei remaining at a given time. The decay equation follows as

$$\frac{dN}{dt} = -0.1N \tag{4.5}$$

Let's solve it for $t = (0, 50)$ and with initial condition $N(0) = 500$. Here, $f(t, N) = -0.1N$ and we choose $h = 1$

$$\begin{aligned} N_{i+1} &= N_i + hf(t_i, N_i) \\ &= N_i - 0.1N_i \\ \text{Or,} \quad N_{i+1} &= 0.9N_i \end{aligned}$$

Also, remember that, $t_{i+1} = t_i + h$.

i	t_i	N_i	$N_{i+1} = 0.9N_i$
0	$t_0 = 0$	$N_0 = 500$	$N_1 = 0.9 \times 500 = 450.00$
1	1	450.00	405.00
2	2	405.00	364.50
3	3	364.50	328.05
...	...	...	...

Table 4.1: Euler's Method

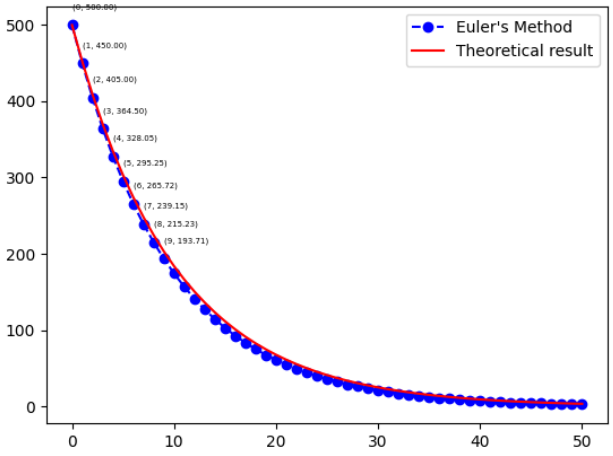

Figure 4.1: Euler's Method

4.1.2 Second-Order ODE

The generic form of a second-order ODE can be written as:

$$\frac{d^2y}{dx^2} + a\frac{dy}{dx} + by + c = 0 \tag{4.6}$$

Now, let's define $\frac{dy}{dx} = v$. With this, we have two first-order equations in our hand

$$\frac{dy}{dx} = v$$

and,

$$\frac{dv}{dx} = -(av + by + c)$$

Euler's method for these two equations gives the following

$$y_{i+1} = y_i + hv_i \tag{4.7}$$

$$f_{i+1} = f_i - h(av_i + by_i + c) \tag{4.8}$$

Problem 2:

Damped Harmonic Oscillator: $F = -kx - 2m\lambda v$

$$\frac{d^2x}{dt^2} + \lambda\frac{dx}{dt} + \omega^2 x = 0$$

Let $\omega = 2$, $\lambda = 0.5$ and the initial conditions: $x(0) = 0$ and $v(0) = \dot{x}(0) = 10$.

$$\frac{d^2x}{dt^2} + \frac{dx}{dt} + 4x = 0$$

Solution:

We choose $h = 0.1$. Given, $x_0 = 0$ and $v_0 = 10$. The two first-order equations

$$\frac{dx}{dt} = v$$

$$\frac{dv}{dt} = -(av + bx + c)$$

Euler's method for these two equations gives the following

$$x_{i+1} = x_i + 0.1v_i$$

$$v_{i+1} = v_i - 0.1(v_i + 4x_i)$$

t_i	x_i	$x_{i+1} = x_i + 0.1v_i$	v_i	$v_{i+1} = v_i - 0.1(v_i + 4x_i)$
0	$x_0 = 0$	$x_1 = x_0 + 0.1 \cdot v_0 = 0 + 0.1 \cdot 10 = 1$	10	$v_1 = v_0 - 0.1(v_0 + 4 \cdot x_0) = 10 - 0.1 \cdot 10 = 9$
0.1	$x_1 = 1$	$x_2 = x_1 + 0.1 \cdot v_1 = 1 + 0.1 \cdot 9 = 1.9$	9	$v_2 = v_1 - 0.1(v_1 + 4 \cdot x_1) = 9 - 0.1 \cdot (9 + 4 \cdot 1) = 7.7$
0.2	$x_2 = 1.9$	$x_3 = x_2 + 0.1 \cdot v_2 = 1.9 + 0.1 \cdot 7.7 = 2.67$	...	...

Table 4.2: Euler's Method for Second Order ODE

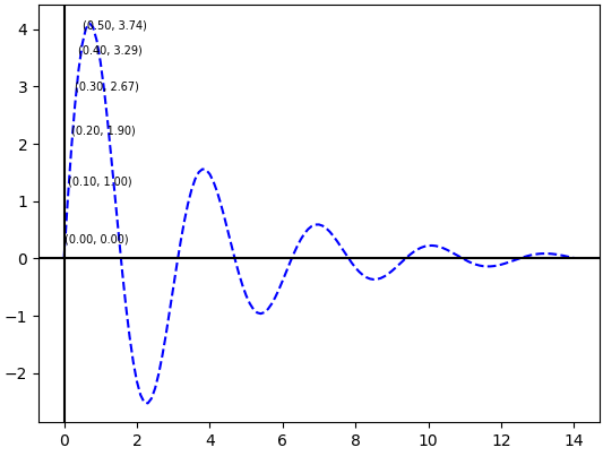

Figure 4.2: Euler's Method for Second Order ODE

4.1.3 Error Analysis for Euler's Method

To derive the local error for the Euler method, we start with the Taylor series expansion of the true solution $y(x + h)$:

$$y(x + h) = y(x) + hy'(x) + \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(x) + O(h^4) \tag{4.9}$$

In the Euler method given by (4.4), we approximate $y(x+h)$ using:

$$y(x+h) \approx y(x) + hf(x, y(x)).$$

With $f(x,y) = y'$, we can express it as:

$$y(x+h) \approx y(x) + hy'(x).$$

As we can see this only involves up to the second term in (4.9) which is the first order in h leaving out the second and higher order terms. This means the error is given by,

$$\tau = \frac{h^2}{2}y''(x) + \frac{h^3}{6}y'''(x) + O(h^4)$$

This shows that the local error in each step is proportional to h^2.

On the other hand, the global error is the cumulative effect of the local errors committed in each step. But the number of steps $n = \frac{t_n - t_0}{h}$, which is proportional to $\frac{1}{h}$, and the error committed in each step is proportional to h^2. Thus, it is to be expected that the global error will be proportional to h. The following figure depicts how the accuracy of this method improves as we make h smaller.

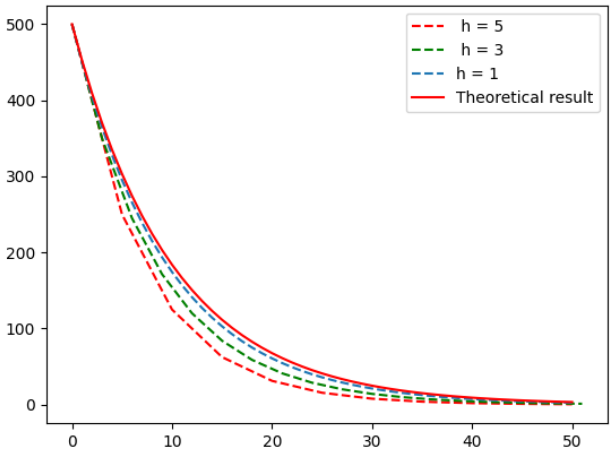

Figure 4.3: Euler Method for different values of h.

4.1.4 Algorithm and Code Template for Euler's Method

Algorithm:

- Initial values x_0, y_0, step size h and number of steps N (or the end value for x)
- Set $x = x_0, y = y_0$
- Create a loop for each step n from 0 to $N - 1$, where
    1. Calculate the slope at the current point: $f(x, y)$
    2. Update y using $y_{n+1} = y_n + hf(x_n, y_n)$
    3. Update x using $x_{n+1} = x_n + h$
- The result will be the final (x, y) values or the entire sequence of (x, y) pairs.

Python Template:

```python
def euler_method(f, x0, y0, h, N):
    x, y = x0, y0
    results = [(x, y)]  # Store initial condition
    for _ in range(N):
        y = y + h * f(x, y)  # Update y based on Euler's formula
        x = x + h            # Update x to the next point
        results.append((x, y))
    return results
```

4.2 Runge-Kutta Method

The goal is to approximate the value of the unknown function $y(x)$ at discrete points $x_0, x_1, x_2, ...$, given an initial condition $y(x_0) = y_0$ and the ODE:

$$\frac{dy(x)}{dx} = f(x, y) \tag{4.10}$$

The Runge-Kutta methods do this by estimating the slope (derivative) at multiple points between x_n and x_{n+1}, and then combining them to give a weighted average slope, which improves the accuracy over simpler methods like Euler's.

The fourth order Runge-Kutta Method (RK4) is the most widely used form of Runge-Kutta methods, because of its higher order of accuracy $(O(h^4))$.

The RK4 method uses four slopes k_1, k_2, k_3, k_4 calculated as follows:

$$k_1 = hf(x_n, y_n) \tag{4.11}$$

$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_1) \tag{4.12}$$

$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{1}{2}k_2) \tag{4.13}$$

$$k_4 = hf(x_n + h, y_n + k_3) \tag{4.14}$$

The next value of y is then computed as:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{4.15}$$

Where,

k_1: The slope at the beginning of the interval (Euler's method approximation).

k_2: The slope at the midpoint using k_1 to estimate y

k_3: Another slope at the midpoint, but now using k_2

k_4: The slope at the end of the interval using k_3

The final estimate for y_{n+1} is a weighted average of these four slopes.

4.2.1 Example:

Given $\frac{dy}{dx} = 1 + y^2$, where $y = 0$ when $x = 0$. Find $y(0.2), y(0.4)$ and $y(0.6)$

Sol: Considering $h = 0.2$, with $x_0 = y_0 = 0$, the coefficients become,

$$k_1 = 0.2$$

$$k_2 = 0.2(1.01) = 0.202$$

$$k_3 = 0.2(1 + 0.010201) = 0.20204$$

$$k_4 = 0.2(1 + 0.040820) = 0.20816$$

and,

$$y(0.2) = 0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 0.2027$$

which is correct to 4-decimal places.

For calculating $y(0.4)$, we can take $x_0 = 0.2$ and $y_0 = 0.2027$. And the result found from the calculation will be used to find the value of $y(0.6)$.

Problem 3: Investment Growth with Interest

Imagine you have an investment that grows continuously with a constant interest rate. The equation for the growth of an investment over time is:

$$\frac{dy}{dt} = r \cdot y$$

Where:

- $y(t)$ is the amount of money at time t,
- r is the constant interest rate.

This differential equation models the exponential growth of money due to interest.

To predict the value of the investment at future points in time, we can apply the RK4 method. Let's say the initial amount of money, y_0, is \$1000, and the interest rate $r = 5\%$ or 0.05. We want to know how the money grows over time with a time step h (e.g., 1 year).

The RK4 method uses four slopes to calculate the next value y_{n+1} from the current value y_n. These slopes are:

$$\begin{aligned}
k_1 &= h \cdot f(y_n) = h \cdot (r \cdot y_n) \\
k_2 &= h \cdot f\left(y_n + \frac{k_1}{2}\right) = h \cdot \left(r \cdot \left(y_n + \frac{k_1}{2}\right)\right) \\
k_3 &= h \cdot f\left(y_n + \frac{k_2}{2}\right) = h \cdot \left(r \cdot \left(y_n + \frac{k_2}{2}\right)\right) \\
k_4 &= h \cdot f(y_n + k_3) = h \cdot (r \cdot (y_n + k_3))
\end{aligned}$$

Then, the next value y_{n+1} is calculated as:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Let's calculate the investment's value after one year using RK4 with a time step $h = 1$ and an initial investment of \$1000.

- Initial $y_0 = 1000$, $r = 0.05$
- $k_1 = 1 \cdot (0.05 \cdot 1000) = 50$
- $k_2 = 1 \cdot (0.05 \cdot (1000 + 50/2)) = 51.25$
- $k_3 = 1 \cdot (0.05 \cdot (1000 + 51.25/2)) = 51.28125$
- $k_4 = 1 \cdot (0.05 \cdot (1000 + 51.28125)) = 52.5640625$

Then:

$$y_1 = 1000 + \frac{1}{6}(50 + 2(51.25) + 2(51.28125) + 52.5640625) \approx 1051.271$$

After one year, the investment grows from \$1000 to approximately \$1051.27 using the RK4 method.

This example shows how RK4 can be used in a straightforward financial problem like investment growth with interest!

4.2.2 Algorithm and Code Template for RK4 Method

Algorithm:

- Initial Values x_0, y_0, step size h and number of steps N.
- Set $x = x_0, y = y_0$
- Create a loop for each step n from 0 to $N - 1$, where
    1. Calculate the coefficients k_1, k_2, k_3, k_4
    2. Update y using $y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
    3. Update x using $x_{n+1} = x_n + h$
- The result will be the final (x, y) values or the entire sequence of (x, y) pairs.

Python Template:

```python
def rk4(f, x0, y0, h, N):
    x, y = x0, y0
    results = [(x, y)]

    for _ in range(N):
        k1 = h * f(x, y)
        k2 = h * f(x + h / 2, y + k1 / 2)
        k3 = h * f(x + h / 2, y + k2 / 2)
        k4 = h * f(x + h, y + k3)

        y += (k1 + 2 * k2 + 2 * k3 + k4) / 6
        x += h

        results.append((x, y))

    return results
```