**5Th Semester Project Report**

**NUMERICAL COMPUTATION**



**By**

**Talha Aslam (SP20-BSM-038)**

**Haider Rasool Qadri (SP20-BSM-014)**

**Ikram Mehboob (SP20-BSM-017)**

**DEPARTMENT OF MATHEMATICS**

**COMSATS UNIVERSITY, ISLAMABAD**

**JUNE, 2022**

**Table of Content**

Table of Contents

About the members………………………………………………..4

Introduction …………………………………………………….....5

Chapter 1 Non-Linear Equations ………………………………....6

* 1. Bisection Method ………………………………………6
  2. False position Method ………………………………….10
  3. Secant Method …………………………………………14
  4. Newton-Raphson Method ……………………………..17
  5. Fixed-Point Iteration Method …………………………21

Chapter 2 System of Linear Equation …………………………..25

2.1 Gauss Jacobi Method ……………………………………25

2.2 Gauss-Seidel Method……………………………………28

2.3 SOR (Successive-Over Relaxation)…………………….31

Chapter 3 Interpolation……………………………………..…34

3.1 Newton’s Forward Interpolation…………………………34

3.2 Newton’s Backward Interpolation………………………37

3.3 Newton’s Central Interpolation…………………………..39

3.4 Lagrange interpolation…………………………………..44

3.5 Newton’s Divided Difference…………………………...47

3.6 Linear Spline Interpolation………………………………50

3.7 Quadratic Spline Interpolation…………………………..50

3.8 Cubic Spline Interpolation………………………………56

Chapter 4 Integration……………………….………….……….59

4.1 Trapezoidal Rule ……......................................................60

4.2 Simpson’s ( 1/3 ) rule……………………………………62

4.3 Simpson’s ( 3/8 ) rule………………………………..….65

CODING…………………………………………………………67

Reffernces…………………………………………………..…….81

**About the Members**

Dr. Umair Umar, who was the instructor of the course Numerical Computations in the Department of Mathematics at CUI and Islamabad, oversaw this Project. The project includes all the course materials covered during the spring semester of 2022.

The Working of project was done by Talha Aslam alongside Haider Rasool Qadri and Ikram Mehboob. Talha Aslam has done his Higher Education From Punjab Group of Colleges, Rawal Pindi. Haider Rasool Qadari did his Higher Education from HIT Degree College, Taxila. Ikram Mehboob Completed his education from PAK-TURK College, Chak Shehzad. Now we have been enrolled in the Undergraduate program of Mathematics at COMSATS, University, Islamabad.

**Introduction:**

Numerical Analysis or Scientific Computing: The study of approximation techniques for numerically solving mathematical problems.

Numerical Computation is necessary for problem solving in that not all the (in fact, very few) mathematical problems have closed-form solutions. We can only obtain numerical solutions.

**For Example:**

* The root of a polynomial with degree of 5 or more
* The eigenvalues of a matrix
* Most ODEs and PDEs

**Main branches of Numerical Computation:**

The main branches of numerical computation are:

* **Linear Equations:** Solve a system of n linear equations in n unknowns.
* **Linear programming:**  minimize a linear function subject to m linear constraints.
* **Optimization:** Minimize a function of several variables.
* **Numerical PDE:** Numerically solve a Partial Differential Equation.

**What is the difference from pure Mathematics?**

**Pure Focus On:**

* Concept
* Theory
* Reasoning

**Applied Mathematics Focus On:**

* Problem solving
* Algorithm
* Stability, Convergence

**Chapter No 01**

**Non-Linear Equations**

In this chapter we are going to find the roots of non-linear equations.

**Methods For Finding the Roots of Non-Linear Equations**

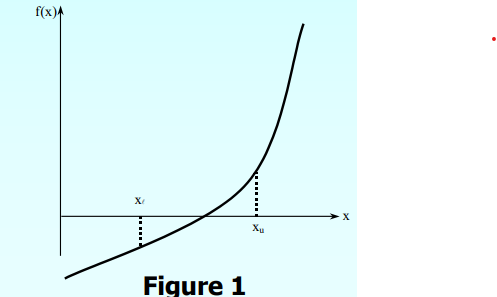
Following are the methods for estimating the roots of Non-Linear Equation:

* **Bisection method**
* **False Position Method**
* **Secant Method**
* **Newton Raphson Method**
* **Fixed Point Method**

**Bisection Method:**

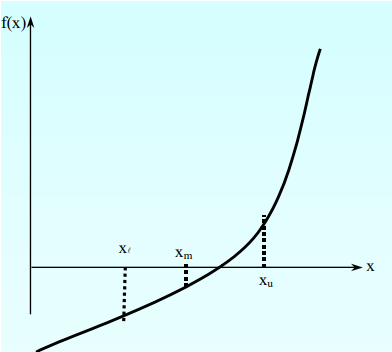
The Algorithm for finding the root using bisection method is as follow:

1. Choose and as two guesses for the root such that f () f ( ) < 0, or in other words, f(x) changes sign between and . As shown in figure 1



1. Estimate the root, of the equation f (x) = 0 as the mid point between and as

=



1. Now check the following conditions:

a) If f () f () < 0, then the root lies between and ; then = ; = .

b) If f () f () > 0, then the root lies between and ; then = ; =

c) If f () f () = 0, then the root is

d) Stop the algorithm if this is true.

1. Find the new estimate of the root

=

1. Find the relative absolute approximate error

=|

Where,

= New estimated root

= Old estimated root

Compare the relative absolute error with pre-specified error. If absolute error is greater than pre-specified error then go to Step ii, using new upper and lower guesses else stop the algorithm.

**Convergence:**

If the given function satisfies the following conditions:

1. It is real
2. It is continuous in an interval which is bounded by two initial guesses between which the function changes sign.

**Drawbacks:**

1. Slow convergence
2. If one of the initial guesses is close to the root the convergence gets tricky and is way slower than ever.

**Advantages:**

1. Always convergent
2. After each iteration the root bracket gets halved.
3. If a function is f(x) is of the sort that it touches the x axis then it will not be possible to find the upper and lower guess values.
4. For some function the function may change the sign between the interval but it is not necessary that root will exist because it may not exist.

**Example N0. 01**

**Find the root of the equation f(x) = -x -1 between 1 and 2, using bisection method**

**Solution:**

Here -x -1 = 0

Let f(x) = -x -1, [1, 2]

**1st iteration**

Here f (1) = -1

f (2) = 5

f (1) \* f(2) = -5 < 0

Now the root lies between 1 and 2

= = 1.5

f () = f (1.5) = – 1.5 -1 = 0.8750 > 0

**2nd iteration:**

Here f (1) = -1

f (1.5) = 0.8750

f (1) \* f(1.5) = -0.8750 < 0

So the root lies between 1 and 1.5

= = 1.25

f () = f (1.25) = – 1.25 -1 = -0.2969 < 0

**Table for bisection method**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N |  | f () |  | f () | = | f () | update |
| 1 | 1.0000 | -1.0000 | 2.0000 | 5.0000 | 1.5000 | 0.8750 | = |
| 2 | 1.0000 | -1.0000 | 1.5000 | 0.8750 | 1.2500 | -0.2969 | = |
| 3 | 1.2500 | -0.2969 | 1.5000 | 0.8720 | 1.3750 | 0.2246 | = |
| 4 | 1.2500 | -0.2969 | 1.3750 | 0.2246 | 1.3125 | -0.0515 | = |
| 5 | 1.3125 | -0.0515 | 1.3750 | 0.2246 | 1.3428 | 0.0826 | = |
| 6 | 1.3125 | -0.0515 | 1.3438 | 0.0826 | 1.3281 | 0.0146 | = |
| 7 | 1.3125 | -0.0515 | 1.3281 | 0.0146 | 1.3203 | -0.0187 | = |
| 8 | 1.3203 | -0.0187 | 1.3281 | 0.0146 | 1.3242 | -0.0021 | = |
| 9 | 1.3242 | -0.0021 | 1.3281 | 0.0146 | 1.3262 | 0.0062 | = |
| 10 | 1.3242 | -0.0021 | 1.3262 | 0.0062 | 1.3252 | 0.0020 | = |
| 11 | 1.3242 | -0.0021 | 1.3252 | 0.0020 | 1.3247 | 0.0000 | = |

**Result:**

Approximate root of the equation -x -1 = 0 using bisection method is 1.3247 after 11 iterations.

**Example No. 02**

**Find the root of equation f(x) = 2 – 2x -5 using bisection method.**

**Solution:**

Here 2 -2x -5 = 0

Let f(x) =2 -2x -5

Here

|  |  |  |  |
| --- | --- | --- | --- |
| x | 0 | 1 | 2 |
| f(x) | -5 | -5 | 7 |

**1st iteration:**

Here f (1) = -5

f (2) = 7

f (1) \* f(2) = -35 < 0

Now the root lies between 1 and 2

= = 1.5

f () = f (1.5) =2\* – 2\*1.5 -5 = -1.2500 < 0

**2nd iteration:**

Here f (1.5) = -1.2500

f (2) = 7

f (1.5) \* f(2) = -8.7500 < 0

So the root lies between 1.5 and 2

= = 1.7500

f () = f (1.75) =2\* – 2\*1.75 -5 = 2.2186 > 0

**Table for bisection method**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N |  |  |  |  | = | f () | update |
| 1 | 1.0000 | -5.0000 | 2.0000 | 7.0000 | 1.5000 | -1.2500 | = |
| 2 | 1.5000 | -1.2500 | 2.0000 | 7.0000 | 1.7500 | 2.2187 | = |
| 3 | 1.5000 | -1.2500 | 1.7500 | 2.2188 | 1.6250 | 0.3320 | = |
| 4 | 1.5000 | -1.2500 | 1.6250 | 0.3320 | 1.5625 | -0.4956 | = |
| 5 | 1.5625 | -0.4956 | 1.6250 | 0.3320 | 1.5938 | -0.0911 | = |
| 6 | 1.5938 | -0.0911 | 1.6250 | 0.3320 | 1.6094 | 0.1181 | = |
| 7 | 1.5938 | -0.0911 | 1.6094 | 0.1181 | 1.6016 | 0.0129 | = |
| 8 | 1.5938 | -0.0911 | 1.6016 | 0.0129 | 1.5977 | -0.0392 | = |
| 9 | 1.5977 | -0.0392 | 1.6016 | 0.0129 | 1.5997 | -0.0132 | = |
| 10 | 1.5997 | -0.0132 | 1.6016 | 0.0129 | 1.6006 | -0.0001 | = |

**Result:**

Approximate root of the equation 2 -2x -5 = 0 using bisection method is 1.6006 after 10 iterations.

**False Position Method:**

In mathematics, the false position method is a very old method of solving an equation with one unknown variable.

The algorithm of false position method is as follow:

1. Choose and as two guesses for the root such that

f () f () < 0

1. Estimated root is, =
2. Now check the following condition:

a) If f () f () < 0, then the root lies between and ; then = ; = .

b) If f () f () > 0, then the root lies between and ; then = ; =

c) If f () f () = 0, then the root is

d) Stop the algorithm if this is true.

1. Find the new estimated of the root

=

1. Find the relative absolute approximate error

=|

Where

= New estimated root

= Old estimated root

Say s = 10-3 =0.001. If | s| > s, then go to step ii, else stop the algorithm.

**Convergence:**

If [a,b] is an interval for a function f(x) such that a root exists between a and b then either a or b is fixed and the other one varies with p. if a is fixed , then the function f(x) is approximated by the straight line passing through points ) and ) where p=1,2, …

C

Where C= and a- µ is in independent of k

Therefore we can write

where is the asymptotic error constant. Hence the method has linear rate of convergence.

**Drawbacks:**

1. It does not require derivative to be calculated.
2. Method has better linear convergence.
3. Converges better near simple root.

**Advantages:**

1. No practical error bound.
2. Iteration may diverge.
3. It can only calculate one unknown in the given problem.

**Example No 01:**

**Find the root of the equation f(x) = -x -1 between 1 and 2, using false position method**

**Solution:**

Here -x -1 = 0

Let f(x) = -x -1, [1, 2]

**1st iteration:**

Here f (1) = -1

f (2) = 5

f (1) \* f(2) = -5 < 0

Now the root lies between 1 and 2

=

= 1.1667

f () = f (1.1667) = – 1.1667 -1 = -0.5787 < 0

**2nd iteration:**

Here f (1.1667) = -0.5787

f (2) = 5

So, the root lies between =1.1667 and 2

=

= 1.2531

f () = f (1.2531) = – 1.2531 -1 = -0.2854 < 0

**Table for false position Method**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| n |  |  | Xu | f(XU) | Xm | f(Xm) | Update |
| 1 | 1.0000 | -1.0000 | 2.0000 | 5.0000 | 1.1667 | -0.5787 | x0=x2 |
| 2 | 1.1667 | -0.5787 | 2.0000 | 5.0000 | 1.2531 | -0.2854 | x0=x2 |
| 3 | 1.2531 | -0.2854 | 2.0000 | 5.0000 | 1.2934 | -0.1295 | x0=x2 |
| 4 | 1.2934 | -0.1295 | 2.0000 | 5.0000 | 1.3113 | -0.0566 | x0=x2 |
| 5 | 1.3113 | -0.0566 | 2.0000 | 5.0000 | 1.3190 | -0.0243 | x0=x2 |
| 6 | 1.3190 | -0.0243 | 2.0000 | 5.0000 | 1.3223 | -0.0104 | x0=x2 |
| 7 | 1.3223 | -0.0104 | 2.0000 | 5.0000 | 1.3237 | -0.0044 | x0=x2 |
| 8 | 1.3237 | -0.0044 | 2.0000 | 5.0000 | 1.3243 | -0.0019 | x0=x2 |
| 9 | 1.3243 | -0.0019 | 2.0000 | 5.0000 | 1.3245 | -0.0008 | x0=x2 |
| 10 | 1.3245 | -0.0008 | 2.0000 | 5.0000 | 1.3246 | -0.0003 | x0=x2 |
| 11 | 1.3246 | -0.0003 | 2.0000 | 5.0000 | 1.3247 | -0.0001 | x0=x2 |
| 12 | 1.3247 | -0.0001 | 2.0000 | 5.0000 | 1.3247 | 0.0000 | x0=x2 |
| 13 | 1.3247 | 0.0000 | 2.0000 | 5.0000 | 1.3247 | 0.0000 | x0=x2 |

**Result:**

Approximate root of the equation x3-x-1=0 using False Position method is 1.3247 after 13 iterations.

**Example No. 02:**

**Find the root of equation f(x) = 2 – 2x -5 using false position method.**

Here 2-2x-5=0

Let f (x) =2-2x-5

Here

|  |  |  |  |
| --- | --- | --- | --- |
| X | 0 | 1 | 2 |
| f(x) | -5 | -5 | 7 |

**1st Iteration:**  
Here f(1)=-5<0 and f(2)=7>0

Now, Root lies between =1 and =2

=

= 1.1416

f ()= f (1.41667) =2⋅1.4166-2⋅1.4166-5 = -2.1469 < 0

**2nd Iteration:**

Now, Root lies between  =1.41667 and =2

=

= 1.5535

f () = f (1.5535) = 2⋅1.5535 - 2⋅1.5535 – 5 = -0.60759 < 0

**Table for false position Method**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| n |  |  | Xu | f(XU) | Xm | f(Xm) | Update |
| 1 | 1 | -5 | 2 | 7 | 1.41667 | -2.14699 | x0=x2 |
| 2 | 1.41667 | -2.14699 | 2 | 7 | 1.55359 | -0.60759 | x0=x2 |
| 3 | 1.55359 | -0.60759 | 2 | 7 | 1.58924 | -0.15063 | x0=x2 |
| 4 | 1.58924 | -0.15063 | 2 | 7 | 1.59789 | -0.0361 | x0=x2 |
| 5 | 1.59789 | -0.0361 | 2 | 7 | 1.59996 | -0.00858 | x0=x2 |
| 6 | 1.59996 | -0.00858 | 2 | 7 | 1.60045 | -0.00203 | x0=x2 |
| 7 | 1.60045 | -0.00203 | 2 | 7 | 1.60056 | -0.00048 | x0=x2 |

**Result:**

Approximate root of the equation 2-2x-5=0 using False Position method is 1.6005.

**Secant method:**

The algorithm for Secant method is as follows:

1. Take two initial guesses
2. Calculate the next estimate of the root from two initial guesses

Xi+1= Xi -

1. Find the absolute relative approximate error

=|

Find if the absolute relative approximate error is greater than the pre specified relative error tolerance. If so, go back to step 1, else stop the algorithm. Also check if the number of iterations has exceeded the maximum number of iterations.

**Drawbacks:**

1. It may not always converge.
2. Division by zero
3. Root jumping

**Advantages:**

1. Better convergence as compared to bisection method.
2. The Convergence of the method is even is faster than linear convergence.
3. Doesn’t not require derivative of the function.
4. The function is only evaluated once per iteration not like newton’s method that requires two evaluations.

**Example No. 01:**

**Find the root of the equation f(x) = -x -1 between 1 and 2, using Secant method**

Here  - x - 1 = 0  
Let f (x) = – x - 1

**1st iteration:**

= 1 and = 2  
  
f () = f (1) = -1 and f () = f (2) = 5

=

=

= 1.1667

f () = f (1.1667) = - 1.1667 -1 = -0.5787

**2nd iteration:**

= 2 and = 1.1667

f () = f(2) = 5 and f () = f (1.1667) = - 0.5787

=

=

= 1.2531

f () = f (1.2531) = - 1.2531 -1 = -0.2854

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N |  | f () |  | f () |  | f () | update |
| 1 | 1.0000 | -1.0000 | 2.0000 | 5.0000 | 1.1667 | -0.5787 | = = |
| 2 | 2.0000 | 5.0000 | 1.1667 | -0.5787 | 1.2531 | -0.2854 | = = |
| 3 | 1.1667 | -0.5787 | 1.2531 | -0.2854 | 1.3372 | 0.0539 | = = |
| 4 | 1.2531 | -0.2854 | 1.3372 | 0.0539 | 1.3239 | -0.0037 | = = |
| 5 | 1.3372 | 0.0539 | 1.3239 | -0.0037 | 1.3247 | 0.0000 | = = |

**Result:**

Approximate root of the equation – x – 1 =0 using Secant method is 1.3247 after 5 iterations.

**Example No 02:**

**Find a root of an equation f(x)=2x3-2x-5 using Secant method**  
  
**Solution:**  
Here 2x3-2x-5=0  
  
Let f(x)=2x3-2x-5  
  
Here

|  |  |  |  |
| --- | --- | --- | --- |
| X | 0 | 1 | 2 |
| f(x) | -5 | -5 | 7 |

**1st iteration:**

x0 = 1 and x1=2  
  
f (x0) = f (1) = -5  and  f (x1) = f ( 2) = 7  
=

x2= 1 - ( -5)   
x2 =1.41667  
  
∴ f (x2) = f (1.4166) = 2⋅1.4166 – 2 \* 1.4166 – 5 = -2.1469  
  
**2nd iteration:**  
  
x1 =2 and x­2 =1.41667  
  
f (x1) = f (2) =7 and f (x2 ) = f (1.4166) = -2.1469  
  
x3 = 2 – 7 \*   
  
x3 =1.55359

f (x3) = f (1.5535) = 2 \* 1.5535 – 2\* 1.5535 – 5 = -0.6075

Approximate root of the equation 2x3-2x-5=0 using Secant method is 1.6006

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| n | X0 | F (x0) | X1 | F (x1) | X2 | f (x2) | Update |
| 1 | 1.0000 | -5.0000 | 2.0000 | 7.0000 | 1.4167 | -2.1470 | X0 = x1 x1 = x2 |
| 2 | 2.0000 | 7.0000 | 1.4167 | -2.1470 | 1.5536 | -0.6076 | X0 = x1 x1 = x2 |
| 3 | 1.4167 | -2.1470 | 1.5536 | -0.6076 | 1.6076 | 0.0945 | X0 = x1 x1 = x2 |
| 4 | 1.5536 | -0.6076 | 1.6076 | 0.0945 | 1.6004 | -0.0032 | X0 = x1 x1 = x2 |
| 5 | 1.6076 | 0.0945 | 1.6004 | -0.0032 | 1.6006 | -0.0000 | X0 = x1 x1 = x2 |

**Newton-Raphson Method:**

1. Evaluate f(x)
2. Use an initial guess of the root, , to estimate the new value of the root, , as
3. Find the absolute relative approximate error as

|\*100

1. Compare the absolute relative approximate error with the pre-specified relative error tolerance, if relative is error is greater than pre-specified error then go to step ii, otherwise stop the algorithm.

**Convergence:**

Let f:[p,q]→R be any function which is differentiable two time in the interval (a,b) and there is a single root β in (a,b). Let f′(x) and f″(x) be the first and second order derivatives of f(x) with respect to x. If β is a simple root and is calculated by the Newton-Raphson method, then the condition of convergence is

||<

**Drawbacks:**

1. It may not always converge.
2. Division by zero.
3. Root jumping.
4. Issue may occur at inflection points.
5. Derivative must exist and is required.
6. Convergence is slower if multiple roots exist.

**Advantages:**

1. Convergence is quadratic.
2. Requires only one guess.
3. Formula is simple so calculations are easy.

**Example No 01:**

**Find a root of an equation f(x)=x3-x-1 initial solution x0=1, using Newton Raphson method.**

**Solution:**

Here   
  
Let f(x)=  
  
∴f′(x)=   
  
Take initial guess =1

**1st iteration:**  
  
f()=f(1)==-1.0000  
  
f′()=f′(1)= -1=2.0000  
  
=-   
  
=1-   
  
=1.5000  
  
**2nd iteration:**  
  
f()=f(1.5)=-1.5-1=0.8750  
  
f′()=f′(1.5)=3\*-1=5.7500  
  
=  
  
=1.5000-  
  
=1.3478

**Table for Newton Raphson Method:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| n |  | f() | f′() |  | Update |
| 1 | 1.0000 | -1.0000 | 2.0000 | 1.5000 |  |
| 2 | 1.5000 | 0.8750 | 5.7500 | 1.3478 |  |
| 3 | 1.3478 | 0.1007 | 4.4499 | 1.3252 |  |
| 4 | 1.3252 | 0.0021 | 4.2685 | 1.3247 |  |
| 5 | 1.3247 | 0.0000 | 4.2646 | 1.3247 |  |

**Result:**

Approximate root of the equation  using Newton Raphson mehtod is 1.3247 after 5 iterations.

**Example No 02:**

**Find a root of an equation f(x) = 2-2x-5 using Newton Raphson method**

**Solution:**

Here 2-2x-5=0

Let f(x) = 2-2x-5

∴f′(x) = 6-2

Here

|  |  |  |  |
| --- | --- | --- | --- |
| x | 0 | 1 | 2 |
| f(x) | -5 | -5 | 7 |

Here f(1)=-5 < 0 and f(2)=7 >0

Root lies between 1 and 2

000

**1st iteration:**  
  
f()=f(000)==-1.2500  
  
f′()=f′(000)= -1=11.5000  
  
=-   
  
=1.5000-   
  
=1.6087  
  
**2nd iteration:**  
  
f()=f(1.6087)=-1.5-1=0.1089  
  
f′()=f′(1.6087)=3\*-1=13.5274  
  
=  
  
=1.6087-  
  
=1.6007

**Table for Newton Raphson Method:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N |  | f() | f′() |  | Update |
| 1 | 1.0000 | -1.2500 | 11.5000 | 1.6087 |  |
| 2 | 1.6087 | 0.1089 | 13.5274 | 1.6007 |  |
| 3 | 1.6007 | 0.0006 | 13.3724 | 1.6006 |  |
| 4 | 1.6006 | 0.0000 | 13.3715 | 1.6006 |  |

**Result:**

Approximate root of the equation 2-2x-5=0 using Newton Raphson mehtod is 1.6006 after 4 iterations.

**Fixed Point Iterations:**

The Algorithm of Fixed point Iterations Method is as follows:

1. Rearranging the function

f (X) = 0

So that x is on the left-hand side of the equation:

X= g (X)

For example:

X2 – 2X + 3 = 0

Can be simply manipulated to yield

X =

X = g (X)

1. Given an initial guess at the root pi , X= g (X) can be used to compute a new estimate pi+1 as expressed by the iterative formula

Pi+1 = g (pi)

**When does Fixed-Point iteration Converge?**

**Existence and Uniqueness Theorem**

1. If g C [a, b] and g (x) [a, b] for all x [a, b], then g has a fixed-point in [a, b].
2. If, in addition, g’ (x) exists on (a , b) and a positive constant k < 1 exists with

|g’ (x)| k, for all x (a, b)

Then there is exactly one fixed point in [a, b].

**Note:**

1. G C [a , b] – g is continuous in [a , b]
2. g (x) C [a, b] – g takes values in [a, b].

**Convergence**

Let g be such that g(x) , for all x. Suppose, in addition, that g’(x) exists on (a, b) and that a constant 0<k<1 exists with

g’(x)|for all X.

Then, for any p0 in [a, b], the sequence defined by

Pn = g(pn-a)

**Drawbacks:**

1. The method does not always converge.
2. There are infinite many ways of finding so this step usually takes time.
3. Only few of the arrangements of the main function can converge only if it satisfies the convergence criteria.
4. The derivative of the chosen function must also be continuous on the interval under investigation.

**Advantages:**

1. Converges fast if it convergent.
2. It requires only one guess.
3. Formula is simple so no lengthy calculation.
4. When g'(x) is close to zero the method converges faster.

**Example No 01:**

**Find a root of an equation f (x) = x3 – x -1 between 1 and 2, using fixed Point Iteration method.**

Let f (x) = x3 – x -1

Here x3 – x -1 = 0

x3– x = 1

x (x2 -1) = 1

x =

(x) =

Here f (1) = -1 < 0 and f (2) = 5 > 0

Root lies between 1 and 2

Xo = = 1.5

x1 = (xo) = (1.5) = 0.8

x2 = (x1) = (0.8) = -2.7778

**Table for Fixed point method**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **Xo** | **X1 = ϕ (x0)** | **Update** | **Difference x1-x0** |
| 2 | 1.5000 | 0.8000 | x0 = x1 | 0.7000 |
| 3 | 0.8000 | -2.7778 | x0 = x1 | 3.5778 |
| 4 | -2.7778 | 0.1489 | x0 = x1 | 2.9267 |
| 5 | 0.1489 | -1.0227 | x0 = x1 | 1.1716 |
| 6 | -1.0227 | 21.8055 | x0 = x1 | 22.8281 |
| 7 | 21.8055 | 0.0021 | x0 = x1 | 21.8034 |
| 8 | 0.0021 | -1.0000 | x0 = x1 | 1.0021 |
| 9 | -1.0000 | 112564.0186 | x0 = x1 | 112565.0186 |

**Result:**

The method leads us away from the solution, so it is divergent.

**Example No. 02:**

**Find a root of an equation f (x) = 2x3 – 2x -5**

**Solution:**

Let f (x) = 2x3 – 2x -5

2x3 – 2x -5 = 0

2x3 = 2x + 5

x3 =

x =

(x) =

Here

|  |  |  |  |
| --- | --- | --- | --- |
| X | 0 | 1 | 2 |
| f(x) | -5 | -5 | 7 |

Here f (1) = -5 < 0 and f (2) = 7 > 0

xo = = 1.5

x1 = (xo) = (1.5) = 1.5874

x2 = (x1) = (1.5874) = 1.5988

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | Xo | X1 = ϕ (xo) | Update | Difference x1-x0 |
| 2 | 1.5 | 1.5874 | x0 = x1 | 0.0874 |
| 3 | 1.5874 | 1.59888 | x0 = x1 | 0.01148 |
| 4 | 1.59888 | 1.60037 | x0 = x1 | 0.0015 |
| 5 | 1.60037 | 1.60057 | x0 = x1 | 0.00019 |

**Result:**

Approximate root of the equation 2x3 – 2x -5 = 0 using iteration method is 1.6005.

**Chapter N0. 02**

**System of Linear Equations**

Jacobi method is an iterative method use to solve system of linear equations and is the simplest method. Gaussian elimination is an uncommon numerical technique since it is direct. That is, after a single application of Gaussian elimination, a solution is produced. Gaussian elimination does not allow for refinement once a "solution" has been found. Because Gaussian elimination is sensitive to rounding error, as seen in the preceding section, the absence of refinements can be an issue.

This method is named after the mathematician Carl Gustav Jacob Jacobi (1804-1851). This method is for the convergent system of equation. If the system is non-convergent, we make it by interchanging the equations. Jacobi method is applicable if the system has non zeros diagonal entries and it is assumed that the system has a unique solution.

For the linear system of equations **Ax=B,**

the general formula of the Jacobi method is

Jacobi method retains the current iteration value in the next iteration.

Error in the iterative methods is calculated by the formula:

**Error =**

Where k is the iteration and k-1 is the previous iteration.

To find the convergence of the system of linear equations, we either check that the system is diagonally dominant or we use the spectral radius.

These are the following iterative methods used for estimating the roots:

* **Gauss Jacobi Method**
* **Gauss Seidel Method**
* **Successive Over Relaxation Method**

**Gauss Jacobi Method:**

**Convergence criteria of Gauss-Jacobi method:**

Jacobi method has two ways to check the convergence of the system. Following are the ways:

1. **Diagonally dominant:**

The system is diagonal dominant if the absolute value of the diagonal entries is greater than or equal to the sum of the absolute values of the other row entries of the matrix.

**+**

1. **Spectral radius:**

This is the second method through which we can check the convergence of the Jacobi method.

**ρ (A) = <1**

Where D, L and U are the diagonal, lower and upper matrix.

Two assumptions made on Jacobi Method:

The system given by:

a11x1 + a12x2 + …a1nxn = b1

a21x1 + a22x2 + …a2nxn = b2

an1x1 + an2x2 + …anmxn = bn

Has a unique solution.

The coefficient matrix has no zeros on its main diagonal, namely, a11, a22…, ann are non-zeros

**Example No. 01:**

**Solution:**

Total Equations are 2  
5x+7y=5  
2x+11y=7  
From the above equations  
xk+1 = ( 5 – 7yk )  
yk+1 = ( 7 – 2xk)  
Initial guess (x(0) ,y(0)) = (0,0)  
Solution steps are  
**1st  Approximation:**  
x(1) = [ 5 - 7(0) ] = [5] = 1   
  
y(1) = [ 7 - 2(0) ] = [ 7 ] = 0.6364 **2nd  Approximation:**  
  
x(2) = [ 5 - 7(0.6364) ] = [0.5455] = 0.1091   
y(2) = [ 7 - 2(1) ] = [ 5 ] = 0.4545

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Solution By Gauss Jacobi Method. x = 0.1463 ≅ 0.15 y = 0.6096 ≅ 0.61 Iterations are tabulated as below   |  |  |  | | --- | --- | --- | | n | **X** | **y** | | 1 | 1.0000 | 0.6364 | | 2 | 0.1091 | 0.4545 | | 3 | 0.3636 | 0.6165 | | 4 | 0.1369 | 0.5702 | | 5 | 0.2017 | 0.6115 | | 6 | 0.1439 | 0.5997 | | 7 | 0.1604 | 0.6102 | | 8 | 0.1457 | 0.6072 | | 9 | 0.1499 | 0.6099 | | 10 | 0.1462 | 0.6091 | | 11 | 0.1473 | 0.6098 | | 12 | 0.1463 | 0.6096 |   **Example No. 02:**  **Solution:** Total Equations are 2 5x - 25y = 2 13x + 11y = 3 The coefficient matrix of the given system is not diagonally dominant. Hence, we re-arrange the equations as follows, such that the elements in the coefficient matrix are diagonally dominant.  13x + 11y = 3 5x - 25y = 2  From the above equations: xk+1 = ( 3 – 11yk )  yk+1 = ( 7 – 5xk ) Initial guess (x(0) ,y(0)) = (0,0)  Solution steps are **1st  Approximation**  x(1) = [ 3 - 11(0) ] = [3] = 0.2308   y(1) = [ 2 - 5(0) ] = [ 2 ] = -0.08  **2nd  Approximation**  X(2) = [ 3 - 11(-0.08) ] = [3.88] = 0.2985  y(2) = [ 2 - 5(0.2308) ] = [ 0.8462 ] = -0.0338Solution By Gauss Jacobi Method. x = 0.2551 ≅ 0.26  y = - 0.0289 ≅ -0.03  **Iterations are tabulated as below:**     |  |  |  | | --- | --- | --- | | **N** | **x** | **y** | | 1 | 0.2308 | -0.0800 | | 2 | 0.2985 | -0.0338 | | 3 | 0.2594 | -0.0203 | | 4 | 0.2480 | -0.0281 | | 5 | 0.2546 | -0.0304 | | 6 | 0.2565 | -0.0291 | | 7 | 0.2554 | -0.0287 | | 8 | 0.2551 | -0.0289 | |

**The Gauss-Seidel Method:**

Gauss Jacobi method is more time consuming and it takes more time to converge. Gauss-Seidel method is the improved version of the gauss Jacobi method. It was named after the Carl Friedrich Gauss (Apr. 1777–Feb. 1855) and Philipp Ludwig von Seidel (Oct. 1821–Aug. 1896).

Gauss Sediel method unlike gauss Jacobi uses updated value from the previous step in the same iteration. First unknown is calculated by the first unknown value. Gauss Seidel method is applicable on the system that are strictly dominant.

For the linear systems of equations **Ax=B**

**(D-L) xk = Uxk-1 + B**

**x(k) = (D-L)-1 Uxk + (D-L)-1B**

**Tg = (D-L)-1 U**

**For convergence, we can also check the spectral radius**

**ρ (Tg) < 1**

**The general form of the Gauss-Seidel method is**

**Now, Solve the system of linear equation by Gauss-Seidel method**

**Solve Equations 5x + 7y = 5, 2x + 11y = 7 using Gauss Seidel method  
Solution:**

Total Equations are 2  
5x + 7y = 5

2x + 11y = 7  
From the above equations  
xk+1 = ( 5 – 7yk )  
yk+1 = ( 7 – 2xk+1 )  
Initial guess (x(0) ,y(0)) = (0,0)  
  
Solution steps are  
**1st  Approximation**  
x(1) = [ 5 - 7(0) ] = [5] = 1   
  
y(1) = [ 7 - 2(1) ] = [ 5 ] = 0.4545  
**2nd  Approximation**  
x(2) = [ 5 - 7(0.4545) ] = [1.8182] = 0.1091   
y(2) = [ 7 - 2(0.3636) ] = [ 6.2727 ] = 0.5702s

Solution by Gauss Seidel Method:  
x = 0.1466 ≅ 0.15  
y = 0.6097 ≅ 0.61  
**Iterations are tabulated as below:**

|  |  |  |
| --- | --- | --- |
| n | **X** | **y** |
| 1 | 1.0000 | 0.4545 |
| 2 | 0.3636 | 0.5702 |
| 3 | 0.2017 | 0.5997 |
| 4 | 0.1604 | 0.6072 |
| 5 | 0.1499 | 0.6091 |
| 6 | 0.1473 | 0.6096 |
| 7 | 0.1466 | 0.6097 |

**Example No. 02:**

**Solve Equations 5x-25y=2, 13x+11y=3 using Gauss Seidel method**  
**Solution:**

Total Equations are 2  
5x - 25y = 2  
13x + 11y = 3

The coefficient matrix of the given system is not diagonally dominant.  
Hence, we re-arrange the equations as follows, such that the elements in the coefficient matrix are diagonally dominant.  
13x + 11y = 3  
5x - 25y = 2  
From the above equations:

xk+1 = (3 – 11yk)  
  
yk+1 = ( 7 – 5xk+1 )

Initial guess (x(0) ,y(0)) = (0,0)

Solution steps are

**1st  Approximation**  
  
x(1) = [ 3 - 11(0) ] = [3] = 0.2308   
  
y(1) = [ 2 - 5(0.2308) ] = [ 0.8462 ] = -0.0338  
  
**2nd  Approximation**  
  
x(2) = [ 3 - 11(-0.0338) ] = [3.3723] = 0.2594  
  
y(2) = [ 2 - 5(0.2594) ] = [ 0.703 ] = -0.0281

Solution By Gauss Jacobi Method.  
x = 0.2551 ≅ 0.26  
y = - 0.0289 ≅ -0.03  
Iterations are tabulated as below:

|  |  |  |
| --- | --- | --- |
| n | **x** | **y** |
| 1 | 0.2308 | -0.0338 |
| 2 | 0.2594 | -0.0281 |
| 3 | 0.2546 | -0.0291 |
| 4 | 0.2554 | -0.0289 |

**SOR (Successive Over-Relaxation) Method:**

Relaxation method is the iterative approach to solve the system of linear equations. The convergence rate is better than gauss Jacobi and Gauss Seidel method.

**Example No :**

**Solve Equations 3x-y+z=-1, -x+3y-z=7, x-y+3z=-7 using SOR (Successive over-relaxation) method**  
  
**Solution:**  
We know that, for symmetric positive definite matrix the SOR method converges for values of the relaxation parameter w from the interval 0<w<2.

The iterations of the SOR method:

Total Equations are 3  
3x - y + z = -1  
-x + 3y – z = 7  
x - y + 3z = - 7  
From the above equations, First write down the equations for Gauss Seidel method  
x k+1 = ( -1 + y k- z k)  
y k+1= ( 7 + x k + 1 + z k)  
z k+1 = ( -7 – xk + 1+ yk+ 1)  
  
Now multiply the right hand side by the parameter w and add to it the vector xk from the previous iteration multiplied by the factor of (1-w)  
xk+1 = (1- w )⋅ xk + w⋅ ( -1 + yk –zk )  
yk+1 = ( 1- w )⋅ yk + w⋅ ( 7 + xk + 1 + zk)  
zk+1 = (1-w)⋅ zk + w⋅ ( -7 - xk + 1 + yk + 1 )

Initial guess ( x , y , z ) = ( 0 , 0 , 0 ) and w=1.25  
  
Solution steps are  
**1st  Approximation**  
  
x1 = (1 - 1.25) ⋅ 0 + 1.25 ⋅ [-1 + (0) - (0) ] = (-0.25) ⋅ 0 + 1.25 ⋅ [ -1 ] = 0 ± 0.41667

= -0.41667  
y1 = (1 - 1.25) ⋅ 0+ 1.25⋅ [7 + (-0.41667) + (0)] = (-0.25) ⋅ 0+ 1.25 ⋅ [6.58333] = 0+2.74306

= 2.74306  
z1= (1-1.25)⋅0 + 1.25⋅ [-7- (-0.41667)+ (2.74306)]= (-0.25)⋅0+1.25⋅ [-3.84028]= 0± 1.6001

= -1.6001  
 **2nd Approximation**:  
x2 = (1 - 1.25) ⋅ -0.4166 +1.25 ⋅ [-1 + (2.7430) - (-1.6001)] = (-0.25) ⋅ -0.4166 + 1.25 ⋅ [ 3.3431] = 0.1041 + 1.3929

= 1.4971  
y2 = (1 - 1.25) ⋅ 2.7430 + 1.25⋅ [7 + (1.4971) + (-1.6001)] = (-0.25) ⋅ 2.7430 + 1.25 ⋅ [6.8970] = 0.6857 + 2.8737

= 2.188  
z1= (1-1.25)⋅ -1.6001 + 1.25⋅ [-7- (1.4971) + (2.188)] = (-0.25)⋅ -1.6001+ 1.25⋅ [-6.3091] = 0.4000 ± 2.6288

= -2.2287

Solution By SOR (successive over-relaxation) method.  
x=1.0002≅1  
  
y=2.00005≅2  
  
z=-2.0001≅-2  
  
Iterations are tabulated as below

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration** | **X** | **Y** | **z** |
| 1 | -0.41667 | 2.74306 | -1.60012 |
| 2 | 1.49715 | 2.188 | -2.22878 |
| 3 | 1.04937 | 1.87824 | -2.01411 |
| 4 | 0.9428 | 2.00073 | -1.97234 |
| 5 | 1.00308 | 2.01263 | -2.00294 |
| 6 | 1.00572 | 1.998 | -2.00248 |
| 7 | 0.99877 | 1.99895 | -1.9993 |
| 8 | 0.99958 | 2.00038 | -1.99984 |
| 9 | 1.0002 | 2.00005 | -2.0001 |

**Chapter No. 03**

**Interpolation**

Interpolation is a method of fitting the data points to represent the value of a function. It has a various number of applications in engineering and science, that are used to construct new data points within the range of a discrete data set of known data points or can be used for determining a formula of the function that will pass from the given set of points (x, y).

**Following are different types of interpolation for equal intervals**

* **Newton’s forward interpolation**
* **Newton’s backward interpolation**
* **Newton’s central interpolation**

Newton’s forward, newton’s backward and newton’s central interpolation are used for finite differences means the difference between points x0, x1, x2 ,..., xn are same for all the values.

1. **Newton’s forward interpolation**

This formula is used for interpolating the values of y near the beginning of a set of tabulated values.

Values of ‘x’ must have equal distance i.e. the value of h must be same for every data point.

Let y=f(x) x0=f(x0)= f0 and  Xn = x 0+ nh xp = x0 +ph

P= where h = Xn- Xn-1

And, fp = f(x0 +ph) = Epf0=(1+∆)pf0

Expanding (1+∆)p by binomial expansion

Fp= {1+p∆+p(p-1) ∆2+p(p-1)(p-2) ∆3+…p(p-1)(p-2)…(p-n+1) ∆n}f0

Fp= f0+p∆ f0+p(p-1) ∆2 f0+p(p-1)(p-2) ∆3 f0+…p(p-1)(p-2)…(p-n+1) ∆nf0

**Difference table:**

To construct a difference table, let us consider a set of data points having equal distances between two consecutive points. The difference between two points is denoted by **h.**

**h = x1-x0 = x2- x1 = xn -xn-1**

**or x1 = x0 + h**

**x2 = x1 + h = x0 + h + h = x0 + 2h**

**Xp = x 0+ ph**

**Xn = x 0+ nh**

**And f(Xp ) = fp = f(x0 +ph)**

In many numerical processes concerned with the set of data points and a functional value arranges in tabular form called finite differences. The standard format of displaying finite differences is called difference table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Xi | fi | 1st order | 2nd order | 3rd order | 4th order |
| X0 | f0 |  |  |  |  |
|  |  | f1 -f0 |  |  |  |
| X1 | f1 |  | f2 -2f1 +f0 |  |  |
|  |  | f2 -f1 |  | f3-3f2 +3f1-f0 |  |
| X2 | f2 |  | f3 -2f2 +f1 |  | f4-4f3+6f2-4f1-f0 |
|  |  | f3 -f2 |  | f4-3f3 +3f2-f1 |  |
| X3 | f3 |  | F4 -2f3 +f2 |  |  |
|  |  | f4 -f3 |  |  |  |
| X4 | f4 |  |  |  |  |

For a constant function all differences are zero.

It helps in determining the behaviour of the derivative of a given function.

It plays an important role in interpolation, numerical differentiation, numerical integration, numerical solution of ordinary and partial differential equations.

The nth- difference of an exact polynomial of degree n are constant.

If the function does not represent an exact polynomial, the above 3 points will not hold.

**Example**

Find Solution using Newton's Forward Difference formula

|  |  |
| --- | --- |
| X | f(x) |
| 1891 | 46 |
| 1901 | 66 |
| 1911 | 81 |
| 1921 | 93 |
| 1931 | 101 |

x = 1895

The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1891 | 1901 | 1911 | 1921 | 1931 |
| y=f(x) | 46 | 66 | 81 | 93 | 101 |

Newton's forward difference table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | Y |  | y | y | y |
| 1891 | 46 |  |  |  |  |
|  |  | 20 |  |  |  |
| 1901 | 66 |  | -5 |  |  |
|  |  | 15 |  | 2 |  |
| 1911 | 81 |  | -3 |  | -3 |
|  |  | 12 |  | -1 |  |
| 1921 | 93 |  | -4 |  |  |
|  |  | 8 |  |  |  |
| 1931 | 101 |  |  |  |  |

The value of x at x=1895

h=x1-x0=1901-1891=10

p===0.4

Newton's forward difference interpolation formula is

y(x)=y0+pΔy0+⋅y0+⋅y0+⋅y0+...

y(1895) =46+0.4×20+×-5+×2+×-3

y(1895)=46+8+0.6+0.128+0.1248

y(1895)=54.8528

Solution of newton's forward interpolation method y(1895)=54.8528

1. **Newton’s backward interpolation**

This formula is used for interpolating the values of y near the end of a set of tabulated values, it may also be applicable in other parts by suitably shifting the origin

Values of ‘x’ must have equal distance i.e., the value of h must be same.

P= where h = Xn- Xn-1

And fp= f(x0 +ph) = Epf0=(1-∇)-pf0

Expanding (1+∆) p by binomial expansion

Fp= {1+p∇ +p(p+1) ∇ 2+p(p+1) (p+2) ∇ 3+…p(p+1) (p+2) …(p+n-1) ∇ n} f0

Fp= f0+ p∇ f0 +p(p+1) ∇ 2 f0+p(p+1) (p+2) ∇ 3 f0+…p(p+1) (p+2) …(p+n-1) ∇ nf0

**Example**

Find Solution using Newton's Backward Difference formula

|  |  |
| --- | --- |
| X | f(x) |
| 1891 | 46 |
| 1901 | 66 |
| 1911 | 81 |
| 1921 | 93 |
| 1931 | 101 |

x = 1925

The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1891 | 1901 | 1911 | 1921 | 1931 |
| y=f(x) | 46 | 66 | 81 | 93 | 101 |

Newton's backward difference table is

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | Y | ∇y | y | y | y |
| 1891 | 46 |  |  |  |  |
|  |  | 20 |  |  |  |
| 1901 | 66 |  | -5 |  |  |
|  |  | 15 |  | 2 |  |
| 1911 | 81 |  | -3 |  | -3 |
|  |  | 12 |  | -1 |  |
| 1921 | 93 |  | -4 |  |  |
|  |  | 8 |  |  |  |
| 1931 | 101 |  |  |  |  |

The value of x at x=1925  
  
h=x1-x0=1901-1891=10  
  
p==1925-193110=-0.6  
  
Newton's backward difference interpolation formula is

y(x)=y0+p∇y0+⋅y0+⋅y0+⋅y0+...

y(1925)=101+(-0.6)×8+×-4+×-1+×-3  
  
y(1925)=101-4.8+0.48+0.056+0.1008

y(1925)=96.8368

Solution of newton's forward interpolation method y(1925)=96.8368

1. **Newton’s central interpolation**

In the previous section we discussed the interpolation methods using the values at the beginning or near the beginning and the points at the end or near the end. Now we will discuss some methods of interpolation using central value.

1. **Gauss forward formula**

The difference operator ∆ is defined by the following relation:

∆fr=fr+1-fr

Where r is an integer, and ∆fr=∆f(xr )

Also , fr+1 = ∆f(xr+h) and ∆fr+1/2 = ∆f(xr + )

Nth order difference is given by,

∆nfr=∆n-1fr+1-∆n-1fr

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Xi | fi | ∆f | ∆2f | ∆3f | ∆4f |
| X0 | f0 |  |  |  |  |
|  |  | ∆ f0 |  |  |  |
| X1 | f1 |  | ∆2 f0 |  |  |
|  |  | ∆ f1 |  | ∆3 f0 |  |
| X2 | f2 |  | ∆2 f1 |  | ∆4 f0 |
|  |  | ∆ f2 |  | ∆3 f1 |  |
| X3 | f3 |  | ∆2 f2 |  |  |
|  |  | ∆ f3 |  |  |  |
| X4 | f4 |  |  |  |  |

**Example**

Find Solution using Gauss Forward formula

|  |  |
| --- | --- |
| X | f(x) |
| 1 | 1 |
| 2 | -1 |
| 3 | 1 |
| 4 | -1 |
| 5 | 1 |

x = 3.5

The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1 | 2 | 3 | 4 | 5 |
| y=f(x) | 1 | -1 | 1 | -1 | 1 |

h=2-1=1  
Taking x0=3 then p==

Now the central difference table is

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | p= | Y | Δy | y-1 | y-1 | y-2 |
| 1 | -2 | 1 |  |  |  |  |
|  |  |  | -2 |  |  |  |
| 2 | -1 | -1 |  | 4 |  |  |
|  |  |  | 2 |  | -8 |  |
| 3 | 0 | 1 |  | -4 |  | 16 |
|  |  |  | -2 |  | 8 |  |
| 4 | 1 | -1 |  | 4 |  |  |
|  |  |  | 2 |  |  |  |
| 5 | 2 | 1 |  |  |  |  |

x=3.5

p===0.5

Gauss's forward interpolation formula is

y(x)=y0+pΔy0+⋅y0-1+⋅y0-1+⋅y0-2+...

y0.5=1+(0.5)(-2)+⋅(-4)+⋅(8)+⋅(16)

y0.5=1-1+0.5-0.5+0.375

0.5=0.3750

Solution of Gauss's forward interpolation is y(3.5)=0.3750

1. **Gauss backward formula:**

The backward difference operator ∇ is defined by the following relation:

∇ fr=fr-fr-1

Nth order difference is given by,

∇nfr=∇n-1fr-∇n-1fr-1  ; for n>=1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Xi | fi | ∇f | ∇2f | ∇ 3f | ∇ 4f |
| X0 | f0 |  |  |  |  |
|  |  | ∇ f1 |  |  |  |
| X1 | f1 |  | ∇ 2 f1 |  |  |
|  |  | ∇ f2 |  | ∇ 3 f1 |  |
| X2 | f2 |  | ∇ 2 f2 |  | ∇ 4 f1 |
|  |  | ∇ f3 |  | ∇ 3 f2 |  |
| X3 | f3 |  | ∇ 2 f3 |  |  |
|  |  | ∇ f4 |  |  |  |
| X4 | f4 |  |  |  |  |

1. **Central Difference Operator:**

The difference operator 𝛿 is defined by the following relation:

𝛿fr=fr+1/2 -fr-1/2

Nth order difference is given by,

𝛿nfr=𝛿n-1 fr+1/2 -𝛿n-1 fr-1/2

1. **Shift operator:**

Efr=fr+1

E-1fr=fr-1

E2fr=fr+2

In general, Enfr=fr+n

1. **Mean operator:**

µfr=(fr+ +fr-)

important relationship between operators:

∆fr=fr+1-fr

∆fr=Efr-fr=(E-1) fr

∆fr=(E-1) fr

∇ fr=fr- E-1 fr

∇ fr=(1- E-1) fr

Therefore,

∆=E-1

∇=1- E-1

Likewise, = E1/2-E-1/2

µ = E1/2+E-1/2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Xi | fi | 𝛿f | 𝛿2f | 𝛿 3f | 𝛿 4f |
| X0 | f0 |  |  |  |  |
|  |  |  |  |  |  |
|  |  | 𝛿 f |  |  |  |
| X1 | f1 |  | 𝛿2 f1 |  |  |
|  |  | 𝛿f |  | 𝛿 3 f |  |
| X2 | f2 |  | 𝛿 2 f2 |  | 𝛿 4 f2 |
|  |  | 𝛿 f |  | 𝛿 3 f |  |
| X3 | f3 |  | 𝛿 2 f3 |  |  |
|  |  | 𝛿f |  |  |  |
| X4 | f4 |  |  |  |  |

1. **Stirling interpolation formula:**

x-1

x0 f0 𝛿2f0 𝛿4f0

x1

it is expressed as follows

fp= f0 +  p( ) +  P2𝛿2f0 + ( + + 𝛿4f0 + (+ ) + 𝛿6f0 + …

**Remarks:**

This formula is suitable for small value of p , for example, -0.25≤p≤0.25

1. **Bessels interpolation method:**

Bessels formula follows the following path through the difference table:

X0 f0  𝛿2f0  𝛿4f0

X1 f1 𝛿2f1 𝛿4f1

Bessels formula can be expressed as follows

fp= f0 + p +( + + + ( + +…

**Remarks:**

This formula is suitable for small values of p not far from 0.5, for example, 0.25≤p≤0.75

**Following are different types of interpolation for unequal intervals**

* **Lagrange interpolation**
* **Newton’s divided difference**

When the values of the independent variable occur with unequal spacing, the formula’s discussed earlier are not applicable to such type of problems. In this situation other formulas called as Newton’s divided difference and Lagrange’s interpolation are used. In simple words, we can say that when the difference between the interval is unequal, we use these formulas.

**Lagrange interpolation**

**Introduction:**

There many different types of interpolation but Lagrange’s interpolation in particular are useful due to the fact that it is one of the interpolations that work for both equal and unequal interval of points. In this method for different values of input (x-values) we have been given some output (y-values) so that we can investigate what is happening between any two points of the provided dataset.

**Derivation**

Suppose, , ,…, be some given observations for which we have corresponding functional values as ,,,…,, where .

= …

…

…

.

.

.

… (1)

Here are constants

Let

Then (1) becomes

= …

= (2)

Let

Then (1) becomes

= …

= (3)

Similarly

Let

Then (1) becomes

= …

= (4)

Now using in (1)

= … +

… +

.

.

.

… (5)

Let =

=

.

.

.

= (6)

Combining (5) and (6)

= + + …+

**Formula**

=

Where

Lagrange's Interpolation formula

y(x)=×+×+×+...+

×

**Example**

 Find Solution using Lagrange's Interpolation formula

|  |  |
| --- | --- |
| X | f(x) |
| 300 | 2.4771 |
| 304 | 2.4829 |
| 305 | 2.4843 |
| 307 | 2.4871 |

x = 301

The value of table for x and y

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 300 | 304 | 305 | 307 |
| y=f(x) | 2.4771 | 2.4829 | 2.4843 | 2.4871 |

Lagrange's Interpolating Polynomial

The value of x at x=301

Lagrange's formula is

y(x)=×+×+×+...+

×

y(301)=×2.4771+×2.4829+×2.4843+×2.4871  
  
y(301)=×2.4771+×2.4829+×2.4843+×2.4871  
  
y(301)=×2.4771+×2.4829+×2.4843+×2.4871  
  
y(301)=2.4786

Solution of the polynomial at point 301 is y(301)=2.4786

**Newton’s divided difference**

Given a set of data x0 ,x1 ,x2,...,xn [ (n+1) points] that may or may not be equally spaced

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Xi** | **fi** | **1st order divided difference** | **2nd order divided difference** | **3rd order divided difference** |
| **X0** | **f0** |  |  |  |
|  |  | **F[x0,x1]=** |  |  |
| **X1** | **f1** |  | **F[x0,x1,x2]=** |  |
|  |  | **F[x1,x2]=** |  | **F[x0,x1,x2 ,x3]=** |
| **X2** | **f2** |  | **f[x1,x2 ,x3]=** |  |
|  |  | **F[x2,x3]=** |  |  |
| **X3** | **f3** |  |  |  |
| **.**  **.**  **.** | **.**  **.**  **.** | **.**  **.**  **.** |  |  |

Then the polynomial of degree ‘n’ through (x0,y0), (x1,y1),.., (xn,yn), is given by the newton’s Divided difference Interpolation formula

**F(x) = f(**X0)+( )F[x0,x1]+( ) ( ) F[x0,x1,x2]+( )( ) )( ) F[x0,x1,x2 ,x3]+…+ ]+( )( )…( ) F[x0,x1,x2 ,xn]

Gives nth degree polynomial from (n+1) points.

Divided differences are symmetric with respect to the arguments i.e. independent of the order of arguments. So,

f[x0, x1]=f[x1, x0]

f[x0, x1, x2]=f[x2, x1, x0]=f[x1, x2, x0]

**Example**

Find Solution using Newton's Divided Difference Interpolation formula

|  |  |
| --- | --- |
| X | f(x) |
| 300 | 2.4771 |
| 304 | 2.4829 |
| 305 | 2.4843 |
| 307 | 2.4871 |

x = 301

The value of table for x and y

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 300 | 304 | 305 | 307 |
| Y=f(x) | 2.4771 | 2.4829 | 2.4843 | 2.4871 |

Newton's divided difference table is

|  |  |  |  |
| --- | --- | --- | --- |
| X | Y | 1st order | 2nd order |
| 300 | 2.4771 |  |  |
|  |  | 0.00145 |  |
| 304 | 2.4829 |  | 0 |
|  |  | 0.0014 |  |
| 305 | 2.4843 |  | 0 |
|  |  | 0.0014 |  |
| 307 | 2.4871 |  |  |

The value of x at x=301

Newton's divided difference interpolation formula is

y(x)=y0+(x-x0)f[x0,x1]+(x-x0)(x-x1)f[x0,x1,x2]+...

Then the first divided difference is given by

f[x0, x1]=

Then the second divided difference is given by

f[x0, x1, x2]=

and so on.

y(301)=2.4771+(301-300)×0.00145+(301-300)(301-304)×0

y(301)=2.4771+(1)×0.00145+(1)(-3)×0

y(301)=2.4771+0.00145+0

y(301)=2.47858

Solution of divided difference interpolation method y(301)=2.4786

**Spline Interpolation**

It is a form of interpolation that connects two points using different kind of interpolation for example the linear spline interpolating scheme or he quadratic interpolating scheme or the cubic interpolating scheme. The spline generates a polynomial which has to be continuous at the points joined by interpolation and at these points there derivative both first and second must also be continuous.

Thereare the following types of spline interpolation

1. linear interpolation
2. quadratic interpolation
3. cubic interpolation

* **Linear spline**

The simplest piecewise-polynomial approximation is piecewise-linear interpolation, which consists of joining a set of data points { (x0,y0), (x1,y1),.., (xn,yn) } by a series of straight lines ( forming the consecutive data through straight lines)

**Note**: Function must be continuous at the intersection point.

**Formula:**

F(x) = f(x0) + (x-x0) x0 ≤ x≤ x1

F(x) = f(x1) + (x-x1) x1 ≤ x≤ x2

F(x) = f(xn-1) + (x-xn-1)

xi-1 ≤ x≤ xi , i = 1, 2, . . . , n

* **Quadratic spline**

They have the advantage over linear Splines that the sensitivity decreases as more data points are in each section. But if there are too many data points in each section the interpolation worsens. The interpolation can turn into a curve that is far of all the points that’s why we used quadratic interpolation.

a1x2+b1x+c1 x0≤x≤x1 (1)

a2x2+b2x+c2 x1≤x≤x2 (2)

anx2+bnx+cn xn-1≤x≤xn (3)

ai ;1,2,…,n

bi ;1,2,…,n 3n constants/unknown

ci ;1,2,…,n

we have 3n constants so we should have 3n equations, now we will find all this 3n equations

**Each quadratic goes through 2 consecutives data points**

from equation (1) we have

y0= a1x02+b1x0+c1

y1= a1x12+b1x1+c1

from equation (2) we have

y1= a2x12+b2x1+c2

y2= a2x22+b2x2+c2

from equation (3) we have

Yn-1= anxn-12+bnxn-1+cn

Yn= anxn2+bnxn+cn

So we have find “2n” equations, now we will find remaining “n” equation

Removing x0 we will have n points (remaining) and removing x1 we will have n-1 points (remaining) so we will have **n-1 interior points from n+1 points** (2 points x0 , xn are exterior points which doesn’t satisfy the slopes are continuous because there is no quadratic before x0 and after xn)

**First derivative at two consecutive quadratics are continuous at common interior points**

a1x2+b1x+c1) = a2x2+b2x+c2)

a1x+b1) = a2x+b2)

a1x1+b1) = a2x1+b2)

a1x1+b1-a2x1-b2=0

(an-1x2+bn-1x+cn-1) = (anx2+bnx+cn)

an-1x+bn-1) = anx+bn)

an-1xn-1+bn-1= anxn-1+bn

an-1xn-1+bn-1-anxn-1-bn=0

So we will get n-1 equations by equating the slopes at n-1 interior points.

So far, **we have 2n+ (n-1) =3n-1 equation’s** so we are left with only one equation

**Last equation**

Choose a1 = 0 or an = 0

Criteria if |x1-x0|≤|xn-xn-1|

Then choose a1 = 0

Else choose an = 0

So we will have **2n+n-1+1=3n** equations

**Example:**

**The upward velocity of a rocket is given as a function of time as Velocity as a function of time.**

|  |  |
| --- | --- |
| **t (s)** | **V(t) ( (m/s)** |
| **0** | **0** |
| **10** | **227.04** |
| **15** | **362.78** |
| **20** | **517.35** |
| **22.5** | **602.97** |
| **30** | **901.67** |

(**a) Determine the value of the velocity at t =16 seconds using quadratic splines.**

Since there are six data points, five quadratic splines pass through them.

V(t) = a1t2+b1t+c1 0≤t≤10

a2t2+b2t+c2 10≤t≤15

a3t2+b3t+c3 15≤t≤20

a4t2+b4t+c4 20≤t≤22.5

a5t2+b5t+c5 22.5≤t≤30

**1. Each quadratic spline passes through two consecutive data points**.

**a1t2+b1t+c1 passes through t = 0 and t = 10 .**

a1(0)2+b1(0)+c1 =0(1)

a1(10)2+b1t(10)+c1 =227.04(2)

**a2t2+b2t+c2 passes through t = 10 and t = 15** .

a2(10)2+b2(10)+c2 =227.04 (3)

a2(15)2+b2(15)+c2 =362.78(4)

**a3t2+b3t+c3 passes through t = 15 and t = 20 .**

a3(15)2+b3(15)+c3 =517.35 (5)

a3(20)2+b3(20)+c3 =517.35 (6)

**a4t2+b4t+c4 passes through t = 20 and t = 22.5** .

a4(20)2+b4(20)+c4 =602.97 (7)

a4(22.5)2+b4(22.5)+c4 =602.97 (8)

**a5t2+b5t+c5 passes through t = 22.5 and t = 30 .**

a5(22.5)2+b5(22.5)+c5 = 901.67 (9)

a5(30)2+b5(30)+c5 = 901.67 (10)

**2. Quadratic splines have continuous derivatives at the interior data points.**

**At t = 10**

a1(10)+b1-a2(10)-b2=0 (11)

**At t = 15**

a2(15)+b2-a3(15)-b3=0 (12)

**At t = 20**

a3(20)+b3-a4(20)-b4=0 (13)

**At t = 22.5**

a4(22.5)+b4-a5(22.5)-b5=0 (14)

3. for last equation

|t1-t0|≤|t5-t4|

|10-0|≤|30-22.5|

10≤7.5 not true

So we will choose a5=0

a5t2+b5t+c5

b5t+c5 = 901.67 (15)

now we will write matrix from those 15 equations

A = =

|  |  |  |  |
| --- | --- | --- | --- |
| **i** | **ai** | **bi** | **ci** |
| **1** | **0** | **22.704** | **0** |
| **2** | **0.8888** | **4.928** | **88.88** |
| **3** | **–0.1356** | **35.66** | **-141.61** |
| **4** | **1.6048** | **–33.956** | **554.55** |
| **5** | **0.20889** | **28.86** | **–152.13** |

After solving the matrix we have,

Therefore, splines are given by

V(t) = 22.704t0≤t≤10

= 0.8888t2+4.928t+88.8810≤t≤15

=-0.1356t2+35.66t-141.6115≤t≤20

=1.6048t2+-33.956t+554.5520≤t≤22.5

=0.20889t2+28.86t+-152.1322.5≤t≤30

At t =16 s

V (16) =-0.1356(16)2+35.66(16)-141.61

=394.24m/s

* **Cubic spline**

A function ‘S’ is called a spline of degree ‘k’ if it satisfied the following conditions.

1. S is defined in the interval ,-
2.  is continuous on , ; 
3. S is polynomial of degree Shape

   Description automatically generated with medium confidence on each subinterval

,

Finding a curve that connects data points with a degree of three is done using cubic spline interpolation. Splines are polynomials that have continuous first and second derivatives at their intersections and are smooth and continuous across a specified plot.

Cubic spline has continuous second derivative whereas quadratic spline only has continuous first derivative so cubic spline is smoother.

a1x3+b1x2+c1x+d1 x0≤x≤x1 (1)

a2x3+b2x2+c2x+d2 x1≤x≤x2 (2)

anx3+bnx2+cnx+dn xn-1≤x≤xn (3)

ai ;1,2,…,n

bi ;1,2,…,n 4n constants/unknown

ci ;1,2,…,n  
di ;1,2,…,n

we have 4n constants so we should have 4n equations, now we will find all this 4n equations

**Each cubic goes through 2 consecutives data points**

from equation (1) we have

y0= a1x03+b1x02+c1x0+d1

y1= a1x13+b1x12+c1x1+d1

from equation (2) we have

y1= a2x13+b2x12+c2x1+d2

y2= a2x23+b2x22+c2x2+d2

from equation (3) we have

Yn-1= anxn-13+bnxn-12+cnxn-1+dn-1

Yn= anxn3+bnxn2+cnxn+dn

So we have find “2n” equations, now we will find remaining “2n” equation

Removing x0 we will have n points (remaining) and removing x1 we will have n-1 points (remaining) so we will have **n-1 interior points from n+1 points** (2 points x0 , xn are exterior points which doesn’t satisfy the slopes are continuous because there is no cubic before x0 and after xn)

**First derivative at two consecutive cubic are continuous at common interior points**

a1x3+b1x2+c1x+d1) = a2x3+b2x2+c2x+d2)

a1x2+2b1x+c1) = a2x2+2b2x+c2)

a1x12+2b1x1+c1) = a2x12+2b2x1+c2)

a1x12+2b1x1+c1 a2x12-2b2x1-c2=0

a1a2)x12+(2b1 -2b2) x1+c1 -c2=0

(an-1x3+bn-1x2+cn-1x+dn-1) = (anx3+bnx2+cnx+dn)

(3an-12+2bn-1+cn-1) = (3an2+2bn+cn)

(3an-1- 3an) 2+(2bn-1- 2bn) + (cn-1- cn) =0

So we will get n-1 equations by equating the slopes at n-1 interior points.

**Second derivative at two consecutive cubic are continuous at common interior points**

a1x3+b1x2+c1x+d1) = a2x3+b2x2+c2x+d2)

a1x2+2b1x+c1) = a2x2+2b2x+c2)

a1x+2b1) = a2x+2b2)

a1x1+2b1) = a2x1+2b2)

(a1a2)x1+2b1-2b2=0

(an-1x3+bn-1x2+cn-1x+dn-1) = (anx3+bnx2+cnx+dn)

(3an-1x2+2bn-1x+cn-1) = (3anx2+2bnx+cn)

(6an-1x+2bn-1) = (6anx+2bn)

(6an-1+2bn-1) = (6an+2bn)

(6an-1-6an) +2bn-1-2bn=0

So we will get n-1 equations by equating the second derivative at n-1 interior points.

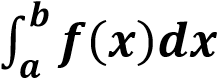
So far, **we have 2n+ (n-1)+(n-1) =4n-2 equation’s** so we are left with 2 equations.

**Chapter No. 03**

**Numerical integration**

Integration is the process of finding Area under the curve. But it is not always possible to find exact value of integration (when function is not continuous) so we find the approximate value of integration.

**Numerical integration:** The process of producing a numerical value for the defining integral



is called Numerical Integration. Numerical Integration is the study of how the numerical value of an integral can be found.

Also called Numerical Quadrature if Shape

Description automatically generated with medium confidence which refers to finding a square whose area is the same as the area under the curve.

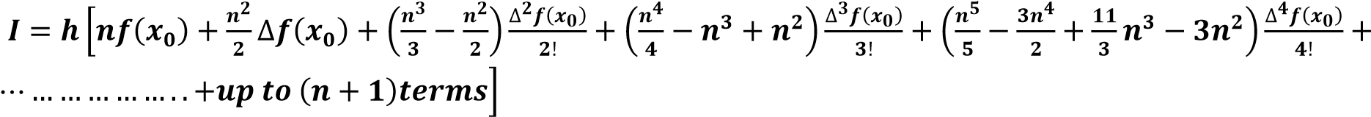
**A general formula for solving numerical integration**

This formula is also called a general quadrature formula.

Suppose f(x) is given for equidistant value of ‘x’ say a=x0, x0+h,x0+2h …. x0+nh = b

Let the range of integration (a,b) is divided into ‘n’ equal parts each of width ‘h’ so that “b-a=nh”.

By using fundamental theorem of numerical analysis It has been proved the general quadrature formula which is as follows



By putting n into different values various formulae is used to solve numerical integration.

That are Trapezoidal Rule, Simpson’s 1/3, Simpson’s 3/8, Boole’s, Weddle’s etc.

IMPORTANCE: Numerical integration is useful when

* Function cannot be integrated analytically.
* Function is defined by a table of values.
* Function can be integrated analytically but resulting expression is so complicated.

**Types of numerical integration:**

Trapezoidal and Simpson’s rules are limited to operating on a single interval. Of course, since definite integrals are additive over subinterval, we can evaluate an integral by dividing the interval up into several subintervals, applying the rule separately on each one and then totalling up. This strategy is called Composite Numerical Integration.

# **Trapezoidal Rule**

Rule is based on approximating   by a piecewise linear polynomial that interpolates  at the nodes 

Trapezoidal Rule defined as follows



And this is called Composite form of Trapezoidal Rule is –

Consider a curve y=f(x) bounded by x0=a and x1=b we have to find i.e.

Area under the curve y=f(x) then for one Trapezium under the area i.e. n = 1



f (x

0

)



f (x

1

)



Y



O



X



a=x

0



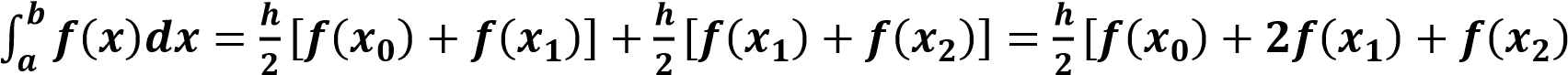
B=x

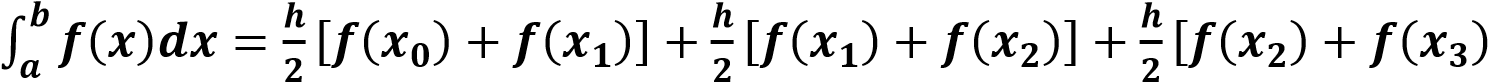
1

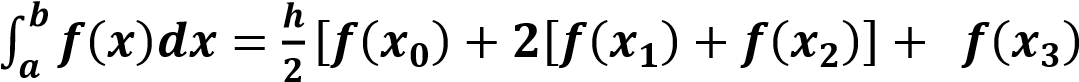
= area of trapezium =



For two trapeziums i. e. n = 2

-

For n = 3 -

-

In general for n – trapezium the points will be  and function will be

Shape

Description automatically generated with medium confidence-

Trapezium rule is valid for n (number of trapezium) is even or odd.

The accuracy will be increase if number of trapeziums will be increased OR step size will be decreased mean number of step size will be increased.

**Example**

**Find Solution using Trapezoidal rule**

|  |  |
| --- | --- |
| x | f(x) |
| 0.0 | 1.0000 |
| 0.1 | 0.9975 |
| 0.2 | 0.9900 |
| 0.3 | 0.9776 |
| 0.4 | 0.8604 |

The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 0 | 0.1 | 0.2 | 0.3 | 0.4 |
| y | 1 | 0.9975 | 0.99 | 0.9776 | 0.8604 |

Using Trapezoidal Rule  
∫ydx =h2 [y0+y4+2(y1+y2+y3)]  
  
∫ydx =0.12 [1+0.8604+2×(0.9975+0.99+0.9776)]  
  
∫ydx =0.12 [1+0.8604+2×(2.9651)]  
  
∫ydx =0.3896  
  
Solution by Trapezoidal Rule is 0.3896

1. **Simpson’s ( 1/3 ) rule**

Rule is based on approximating f(x) by a Quadratic Polynomial that interpolate f(x) at



Simpson’s Rule is defined as for simple case 

While in composite form it is defined as

-

Global error for Simpson’s Rule is defined as 

REMARK

In Simpson Rule number of trapezium must of Even and number of points must of Odd.

1. **Derivation of Simpson’s ( 1/3 ) rule (1st method)**

Consider a curve bounded by x = a and x = b and let ‘c’ is the mid-point between  and  such that  we have to find Shape

Description automatically generated with medium confidence i.e. Area under the curve.

Y

X

0

A

B

C

b

a

c

f(a)

f(c)

f(b)

Consider 

Now 

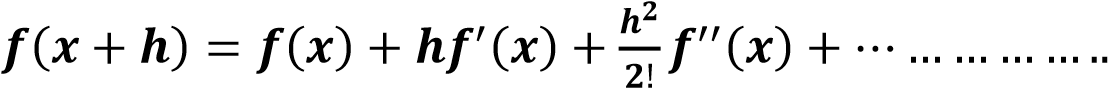
Shape

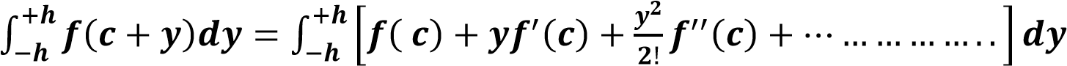
Description automatically generated with medium confidence 



Now Shape

Description automatically generated with medium confidence where y is small change

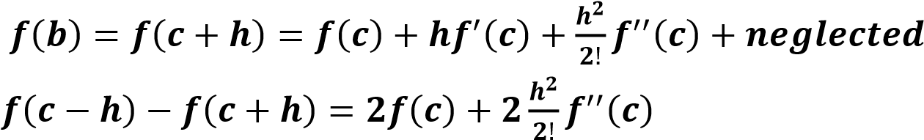
Using Taylor Series Formula 



Neglecting higher derivatives

Shape

Description automatically generated with medium confidence



  Put this value in (i)



Shape

Description automatically generated with medium confidence-

For n = 4

-

-

In General



1. **Derivation of Simpson’s ( 1/3 ) rule (2nd method)**



-

-

This is required formula for Simpson’s (1/3) Rule

**Example**

**Find Solution using Simpson's 1/3 rule**

|  |  |
| --- | --- |
| x | f(x) |
| 0.0 | 1.0000 |
| 0.1 | 0.9975 |
| 0.2 | 0.9900 |
| 0.3 | 0.9776 |
| 0.4 | 0.8604 |

The value of table for x and y

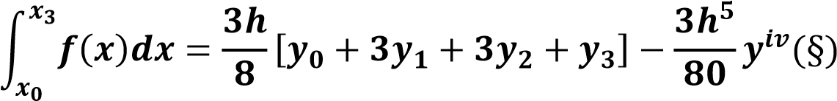
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 0 | 0.1 | 0.2 | 0.3 | 0.4 |
| y | 1 | 0.9975 | 0.99 | 0.9776 | 0.8604 |

Using Simpsons 1/3 Rule  
  
∫ydx=h3[(y0+y4)+4(y1+y3)+2(y2)]  
  
∫ydx=0.13[(1+0.8604)+4×(0.9975+0.9776)+2×(0.99)]  
  
∫ydx=0.13[(1+0.8604)+4×(1.9751)+2×(0.99)]  
  
∫ydx=0.3914  
  
Solution by Simpson's 1/3 Rule is 0.3914

1. **Simpson’s ( 3/8 ) rule**

Rule is based on fitting four points by a cubic.

Simpson’s Rule is defined as for simple case

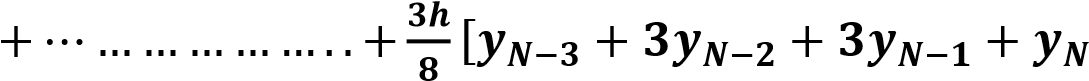


While in composite form (“n” must be divisible by 3) it is defined as



Shape

Description automatically generated with medium confidence

-



**Remarks:** Global error in Simpson’s (1/3) and (3/8) rule are of the same order but if we consider the magnitude of error then Simpson (1/3) rule is superior to Simpson’s (3/8) rule.

**Example**

**Find Solution using Simpson's 3/8 rule**

|  |  |
| --- | --- |
| x | f(x) |
| 0.0 | 1.0000 |
| 0.1 | 0.9975 |
| 0.2 | 0.9900 |
| 0.3 | 0.9776 |
| 0.4 | 0.8604 |

The value of table for x and y

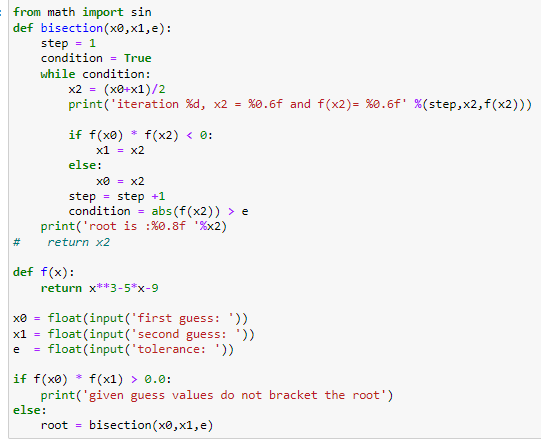
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 0 | 0.1 | 0.2 | 0.3 | 0.4 |
| y | 1 | 0.9975 | 0.99 | 0.9776 | 0.8604 |

Using Simpsons 3/8 Rule

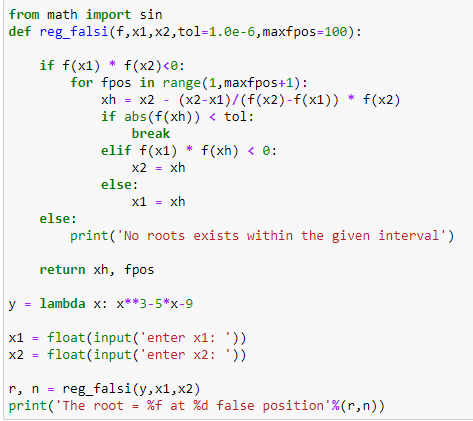
∫*ydx*=3*h*8[(*y*0+*y*4)+2(*y*3)+3(*y*1+*y*2)]  
  
∫*ydx*=3×0.18[(1+0.8604)+2×(0.9776)+3×(0.9975+0.99)]  
  
∫*ydx*=3×0.18[(1+0.8604)+2×(0.9776)+3×(1.9875)]  
  
∫*ydx*=0.3667  
  
Solution by Simpson's 38 Rule is 0.3667

**Codes Algorithm of All Methods:**

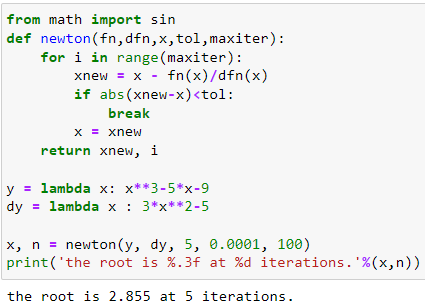
**Bisection Method:**



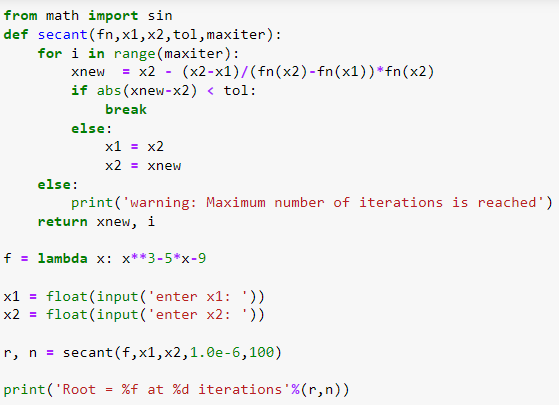
**False Position Method:**

****

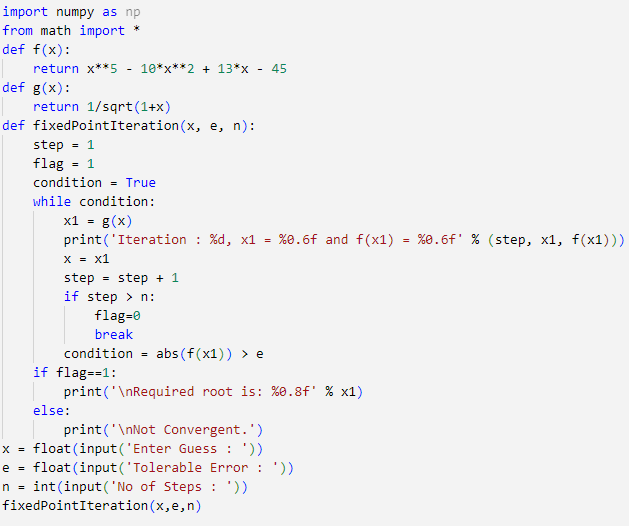
**Newton Raphson Method:**



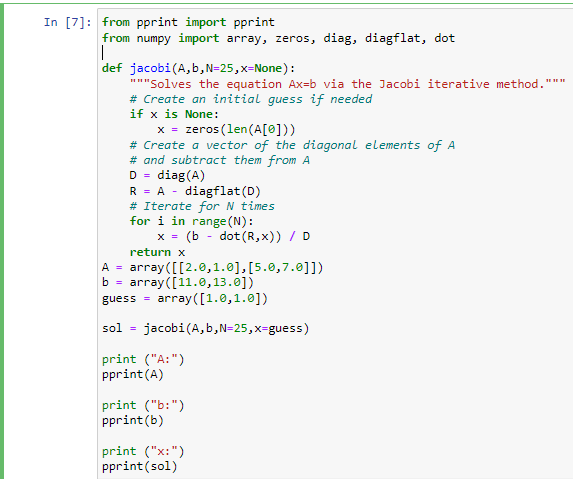
**Secant Method:**



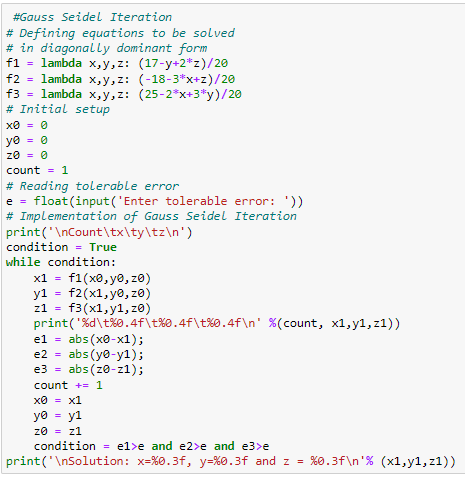
**Fixed Point iteration Method:**

**:**

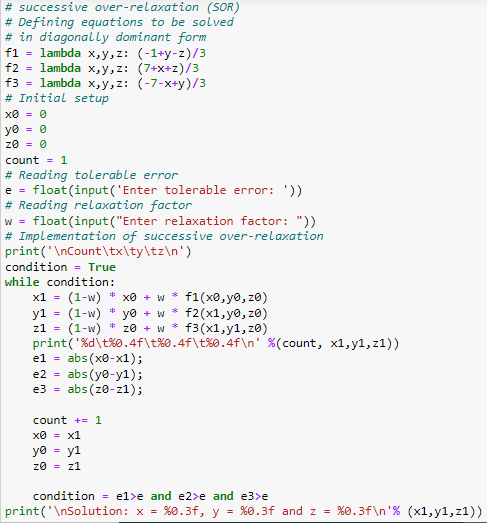
**Jacobi Method:**



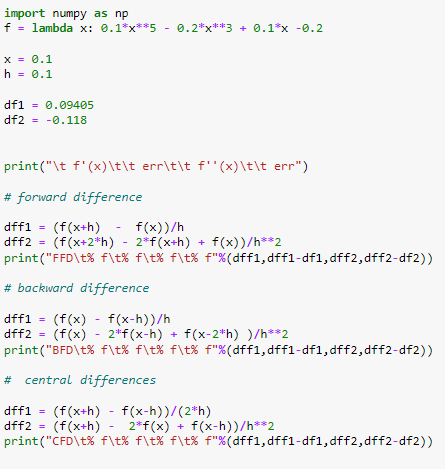
**Gauss-Seidel Method:**



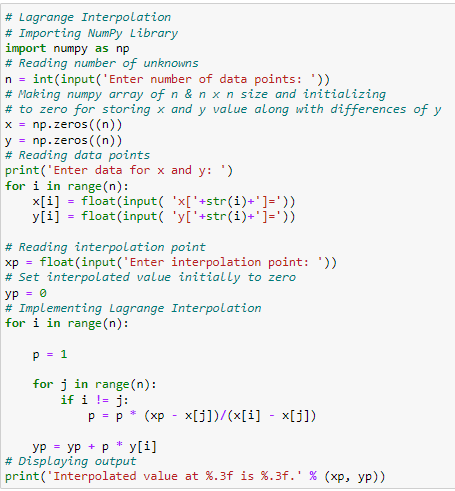
**SOR (Successive-Over Relaxation) Method:**



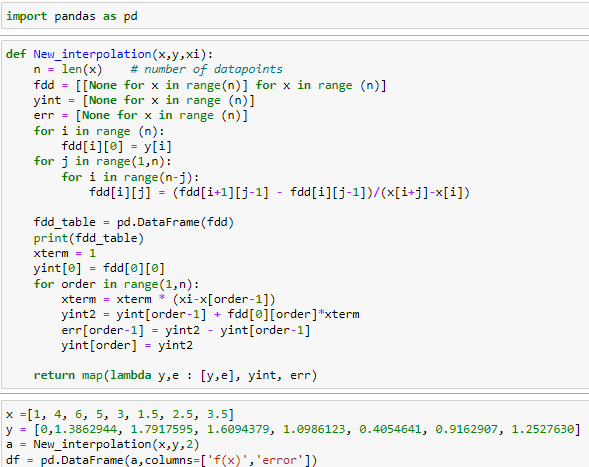
**Newton’s Interpolation:**



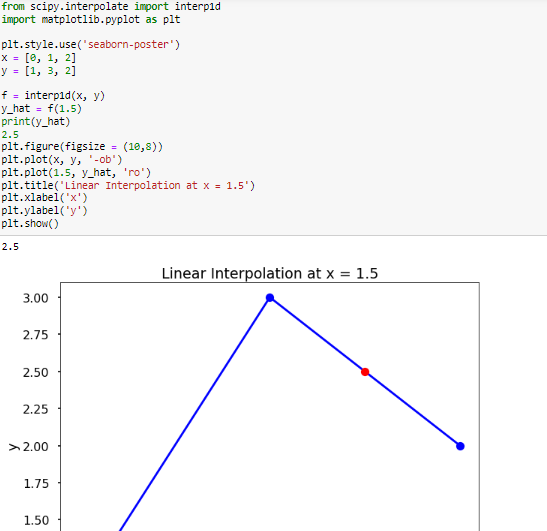
**Lagrange’s Interpolation:**

****

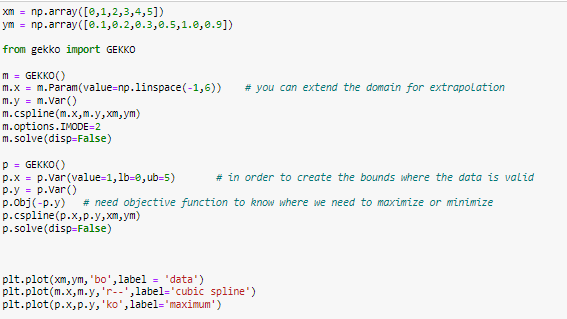
**Newton’s Divided Difference:**

****

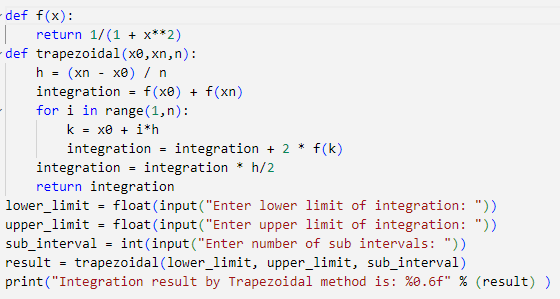
**Linear Spline:**



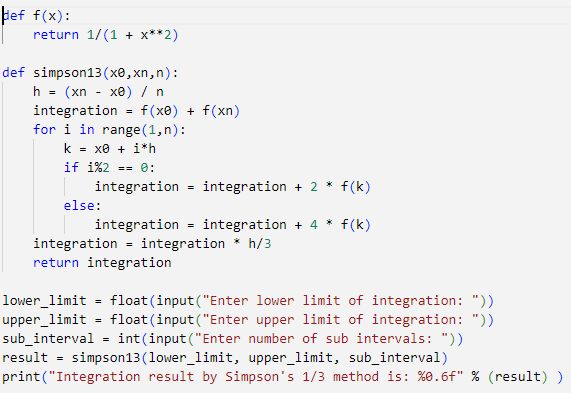
**Cubic Spline:**



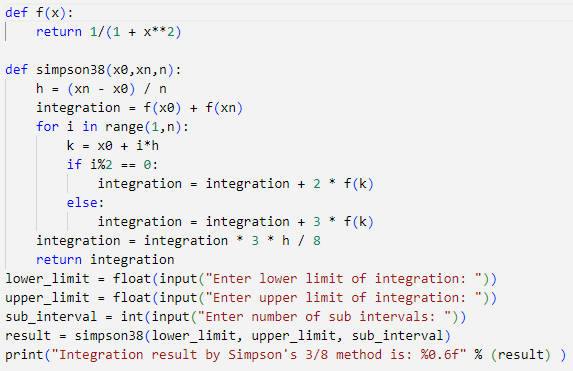
**Trapezoidal Rule:**

****

**Simpson’s ( 1/3 ) rule:**

****

**Simpson’s ( 3/8 ) rule:**



**Reference’ s:**

[**https://www3.nd.edu/~zxu2/acms40390F12/Lec-7.3.pdf**](https://www3.nd.edu/~zxu2/acms40390F12/Lec-7.3.pdf)

[**http://localhost:8888/notebooks/jacobi%20%2C%20siedel%20and%20or.ipynb**](http://localhost:8888/notebooks/jacobi%20%2C%20siedel%20and%20or.ipynb)

[**https://sites.google.com/site/knowyourrootsmaxima/introduction/bisectionmethod**](https://sites.google.com/site/knowyourrootsmaxima/introduction/bisectionmethod)

[**https://sites.google.com/site/knowyourrootsmaxima/introduction/newtonmethod**](https://sites.google.com/site/knowyourrootsmaxima/introduction/newtonmethod)[**https://www.youtube.com/watch?v=vfEq-WKyVbQ&t=30s**](https://www.youtube.com/watch?v=vfEq-WKyVbQ&t=30s)

[**https://www.youtube.com/watch?v=x7m0m5A5EiQ**](https://www.youtube.com/watch?v=x7m0m5A5EiQ)

[**https://www.youtube.com/watch?v=j\_jBK7zJ1vU&feature=youtu.be**](https://www.youtube.com/watch?v=j_jBK7zJ1vU&feature=youtu.be)

[**http://localhost:8888/notebooks/Downloads/interpolation%20and%20splines.ipynb**](http://localhost:8888/notebooks/Downloads/interpolation%20and%20splines.ipynb)

[**http://localhost:8888/notebooks/Downloads/cubic%20supline%20by%20GEKKO.ipynb**](http://localhost:8888/notebooks/Downloads/cubic%20supline%20by%20GEKKO.ipynb)