

**COMSATS UNIVERSITY ISLAMABAD**

**Numerical Computations**

**By Sadia Nadeem & Bushra Bibi**

Table of Contents

About the members 2

Chapter 1: Method of non-linear equations 3

Chapter i: Bisection method 3

Chapter ii: Newton-Raphson method 6

Chapter iii: Secant method 8

Chapter iv: False position method 11

Chapter v: Fixed point method 14

Chapter vi: Comparisons with iterations 16

Chapter 2: Methods for system of linear equations 18

Chapter i: Gauss Jacobi method 21

Chapter ii: Gauss seidel method 26

Chapter iii: Successive Over Relaxation methods 29

Chapter 3: Interpolation 31

Chapter i: Lagrange Interpolation 31

Chapter ii: Newton interpolation 34

Chapter iii: Newton divided difference interpolation 44

Chapter iv: Spline interpolation 46

Chapter 4: Integration 55

Chapter i: Trapezoidal Rule 56

Chapter ii: Simpson’s ( 1/3 ) rule 58

Chapter iii: Simpson’s ( 3/8 ) rule 61

References 62

**About the members**

This project was done under the supervision of Dr Umair Umar who was the instructor of the course Numerical Computations at the Department of Mathematics, CUI, and Islamabad. The project comprises of all the course contents studied in the semester of spring-2022.

Miss Bushra Bibi alongside Sadia Nadeem worked upon the project. Miss Bushra has done her schooling from Pakistan international school of Cairo, Egypt and thereafter completed her high school studies at Punjab Group of colleges for women in Rawalpindi. For now, she is enrolled in the Undergraduate program of Mathematics at CUI Islamabad. While miss Sadia Nadeem completed her schooling from international school of Pakistan, Kuwait and is a high school graduate of international school of Pakistan, Kuwait. She is also enrolled in the Undergraduate program of Mathematics at CUI Islamabad.

**Chapter 1: Methods of non-linear equations**

**Chapter i: Bisection method**

Introduction: there is a lot of ambiguity in how the method actually came into being but it is thought that the method was not late developed after Bolzano in 1817 proved the Intermediate Value Theorem. One of the reasons why the method is traced back to this theorem is that this method actually uses the logic of Intermediate Value Theorem. it is also inferred that the method was a justification or application of the Intermediate Value Theorem.

**Procedure:**

The bisection method is used to find the roots of a continuous function. It uses the idea of Intermediate Value Theorem which is as follows:

If a function f(x) is continuous and there is a point **a** that is negative and a point **b** that is positive then there is a point **c** between **(a, b)** that equal zero.

So suppose for a given function f we have to find its numerical root. So firstly we check if the function is continuous or not. If the function is not continuous then this technique will fail and we will shift to some other method. But suppose the function is continuous then we will continue with the method. We will suppose two different points say a and b such that f(a) and f(b) have opposite sings or their product is less than zero. Hence now we are guaranteed that at least one of the roots lies between the interval [a,b].There can be multiple roots but for the sake of simplicity we shall consider the root is unique. Afterwards we will apply the formula (a+b)/2 and find a new root c. and if upon c we find the functional value turning to equal zero we will be done but if not then we will divide the interval into a new interval containing the root. This process will continue halving the interval until we reach a concluding criterion.

**Working rule:**

Let f be a continuous function. Consider two points a and b which when substituted in the function results as f(a) f(b)<0.

Formula: c=(a+b/2) , where c is the new root.

If f(c)=0 then answer has been found and the method concludes.

But if not then

Suppose f(a) < 0 (negative) and f(b) > 0 (positive)

Now there are two possibilities either

1) f(c)<0 or

2) f(c) >0

Consider case 1 when f(c)<0 then put a=c and b=b.

OR

Case 2 f(c) >0 then put b=c and a=a.

Similarly, if f(b) < 0 (negative) and f(a) > 0 (positive)

Now there are two possibilities either

1) f(c)<0 or

2) f(c) >0

Consider case 1 when f(c) < 0 then put b=c and a=a.

OR

Case 2 f(c) >0 then put a=c and b=b.

**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.

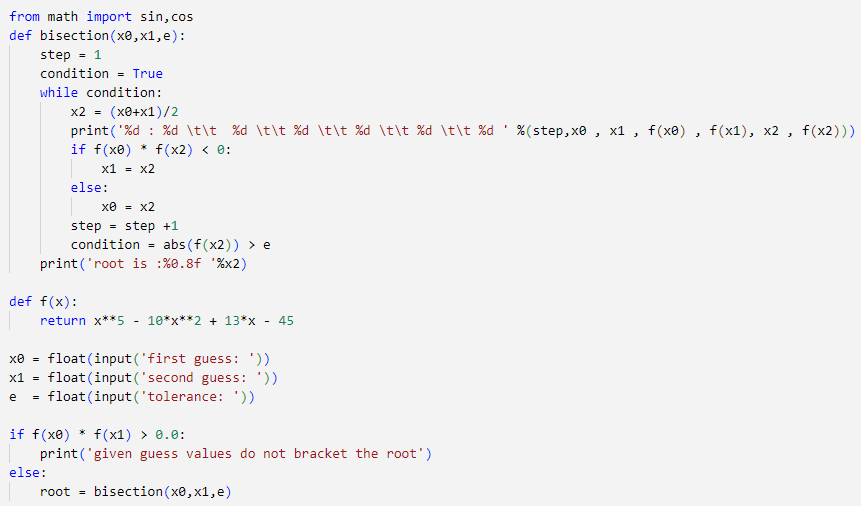
**Drawbacks:**

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.

**Algorithm :**

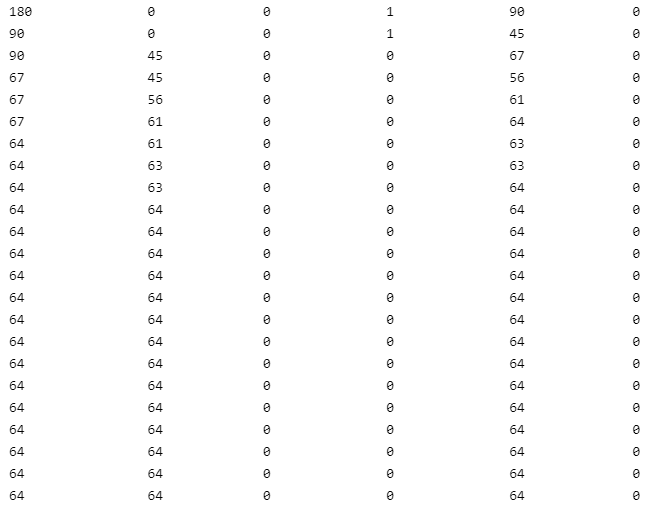
**Example:**

**Equation:**

F(x) = cos(x)

Guess1 = 180, Guess2 = 0, Tolerance = 0.000001, Root = 64.40264940

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | F(a) | F(b) | C | F(c) |



**Chapter ii: Newton-Raphson method**

**Introduction:**

Francois Vieta in 1600 designed a method that would solve second, third and fourth degree of polynomials. Newton further contributed to it by linearizing the polynomials. Raphson a student of newton developed a somewhat similar method to this one but his method was distinct due to the fact that it was derived differently. None of them thought it would be connected to calculus until 1740 when Thomas Simpson worked upon it to make it look very similar to the current method we use today.

**Procedure:**

Take an initial guess “c” from some interval [a,b] such that the function f(x) is continuous for the interval and initial guess belongs to [a,b]. Now must exist and if initial guess is substituted in then. The iterations shall continue until the tolerance criterion is reached.

**Derivation:**

Equation of slope is

m=

=

=

=

=

The derivative is on the particular value

=

The tangent line is cutting the x-axis at y=0

=

So the first term becomes

And thus we can derive the general formula as

**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

||<

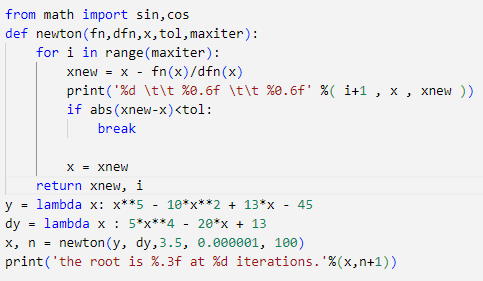
**Drawbacks:**

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.

**Algorithm:**

****

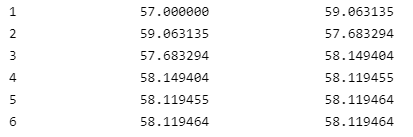
**Example:**

**Equation:**

F(x) = cos(x) , F’(x) = -sin(x)

Guess1 = 57, Tolerance = 0.000001, Root = 58.119

|  |  |  |
| --- | --- | --- |
| Iteration | X | xnew |



**Chapter iii: Secant method**

**History:**

Secant is a very old method that can be traced back to some 3000 years before newton was born. This method was essentially applied to liner equations. This method was developed owing to daily life problems such as predicting production, inheritance calculations, sales and distribution laws. A special case of this method known as double false position was used by Cardano in 1545 as an iterative procedure. Thus we got the secant method to solve non-linear equations.

**Procedure:**

Secant method is modified form of newton Raphson method. For this method select any two points as initial guesses but these guesses should be such that the functional value at these guesses doesn’t make the denominator in the formula equal to zero. Now using the formula calculate the first approximated root and check if the function f(x) =0. If condition is fulfilled then answer has been reached else continue with the iterations in such a manner that replace the value of first initial guess with the value of second initial guess and second initial guess with the newly found approximated root. And continue with the iterations until a tolerance criterion has been satisfied.

Say initial guesses are

First guess= a, second guess=b

Use the formula and find the new root c

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **C** | **f(c)** |
| **First Initial guess** | **Second initial guess** | **First New root** | **…..** |
| **Second initial guess** | **First New root** | **Second New root** | **…..** |
| **First New root** | **Second New root** | **Third New root** | **…..** |

This table shall continue until stopping criterion is satisfied.

**Derivation:**

Slope of AB=slope of AC

**Convergence:**

Using the mean-value theorem from calculus, we can show that

α - = (α - ) (α -) [] (1)

with †n and ¥n unknown points. The point †n is located between the minimum and maximum of,, and α; and ¥n is located between the minimum and maximum of and . Using (1), it can be shown that converges to α, and moreover,

== c

This assumes that and is chosen sufficiently close to α; and how close this is will vary with the function f. In addition, the above result assumes f (x) has two continuous derivatives for all in some interval about α. The above says that when we are close to α, that

≈c

Secant method is fast enough that we can show this

=1

and therefore ≈ is a good error estimator.

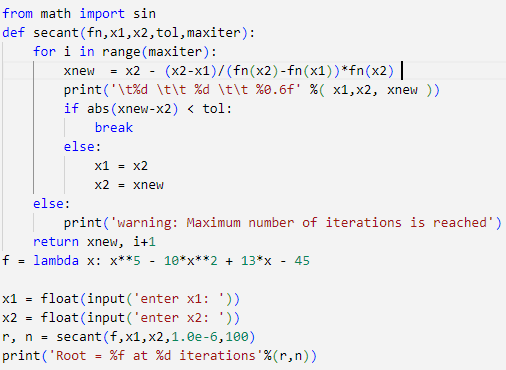
**Drawbacks:**

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.

**Algorithm:**

****

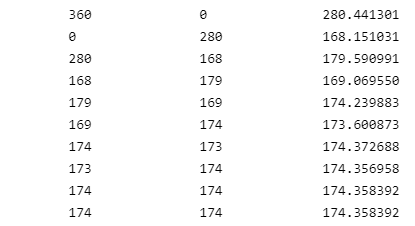
**Example:**

**Equation:**

F(x) = cos(x)

Guess1 = 360, Guess2 = 0, Tolerance = 0.000001, Root = 174.358392

|  |  |  |
| --- | --- | --- |
| x1 | x2 | xnew |



**Chapter iv: False position method**

**History:**

This method dates back to 200 to 100 CE of the ancient Chinese era. A book called the nine chapters on the mathematical art describes that the method was then used to solve linear equations which is today modified enough to solve non-linear equations.

**Introduction:**

This method is also known as Regula-Falsi method. This is one of the most ancient method of estimating numerical roots. For this method we need two initial guesses say x and y such that when these points are substituted in the given function, the joint product must be less than zero. This condition when satisfied guarantees the existence of root between the two points x and y.

**Procedure:**

Take two initial guesses and b. theses guesses should be such that the function f(x) at a and b gives f(a) f(b)<0. Or simply the function at both the points must be of opposite signs. This assures that the root lies between the interval [a,b]. Now apply the formula of the method and find the new approximated root say “c”. If at the new root function f(x) =0 then stop and answer has been found.. But if not then continue the iteration. Now either f(c)>0 or f(c) <0 where c is

If f(c)>0 then replace the value of a with c if f(a)< 0 else b if f(b)<0.

And if f(c)<0 then replace the value of a with c if f(a)> 0 else b if f(b)>0.

Continue the iterations until standard tolerance is reached.

**Derivation:**

Equation of the chord joining the points is

The chord meets the x-axis at the point

Put and in equation (1)

**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.

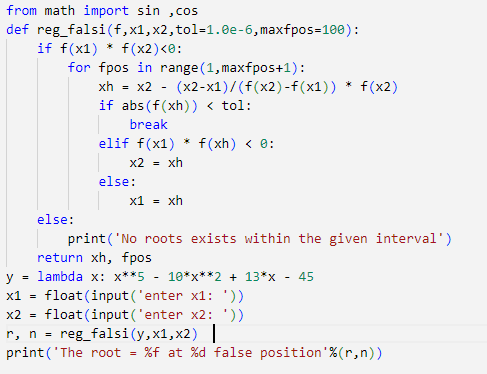
**Drawbacks:**

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.

**Algorithm:**

****

**Example:**

**Equation:**

F(x) = cos(x)

Guess1 = 180, Guess2 = 0, Tolerance = 0.000001, Root = 120.951317 at 10 false positions.

**Chapter v: Fixed point method:**

**Introduction**

The method has a long history but we shall restrict ourselves to the main idea behind the method. Since we are dealing with the idea of finding approximate roots of nonlinear equations of the sort

=0 (1)

In fixed point iteration method we rewrite equation (1) as

= (2)

So that any solution of equation (2), that is known as the fixed point of is a solution of equation (1).

**Procedure:**

Suppose we want to find the root of some function (1)

Write the equation (1) in the form

=

So that we may get x completely alone, being positive on one side.

Now we shall apply the formula of fixed point method

n=0, 1 …

**Convergence:**

**Existence and uniqueness theorem**

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

≤ k, for all x ∈ (a,b),

Note: ∈ C[a,b] – is continuous on [a,b]

∈ C[a,b], takes value in on [a,b].

**Fixed point theorem**

Let ∈ C[a,b] be such that ∈ [a,b] for all ∈ [a,b]

Suppose in addition, exists on (a,b) and a constant 0< k < 1 exists with

| ≤ k, for all x ∈ (a,b),

Then for any number [a,b], the formula defined by

=) converges to unique fixed point number p [a,b].

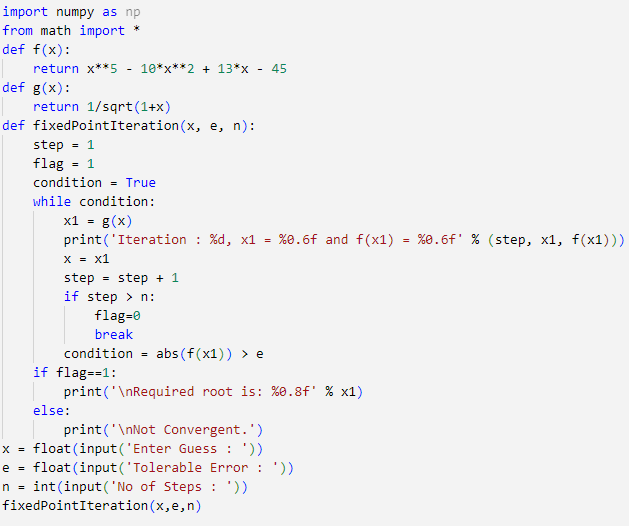
**Drawbacks:**

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.

**Algorithm:**

****

**Chapter vi: Comparison with iterations is given below.**

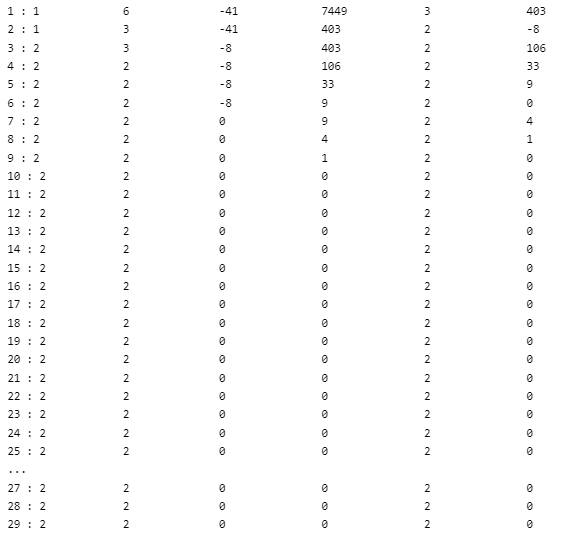
**Equation = x\*\*5 - 10\*x\*\*2 + 13\*x – 45**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Bisection method** | **Newton raphson method** | **Regula falsi method** | **Secant method** |
| **First guess** | **1** | **3** | **1** | **2** |
| **Second guess** | **6** | **5** | **6** | **4** |
| **Tolerance** | **0.00001** | **0.00001** | **0.00001** | **0.00001** |
| **Root** | **2.33286606** | **2.33** | **2.315693** | **2.332866** |

The iterations of every method are mentioned below.

1. **Bisection method**

F(x) = x\*\*5 - 10\*x\*\*2 + 13\*x - 45

Guess1 = 180, Guess2 = 0, Tolerance = 0.000001, Root = 2.33286606

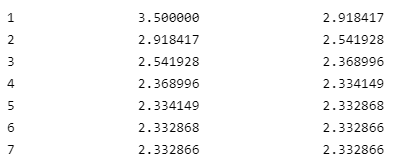
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | F(a) | F(b) | C | F(c) |

1. **Newton Raphson method**

F(x) = x\*\*5 - 10\*x\*\*2 + 13\*x – 45 , F’(x) = 5\*x\*\*4 - 20\*x + 13

Guess1 = 3.5, Tolerance = 0.000001, Root = 2.333

|  |  |  |
| --- | --- | --- |
| Iteration | X | xnew |



1. **Regula falsi method**

F(x) = x\*\*5 - 10\*x\*\*2 + 13\*x - 45

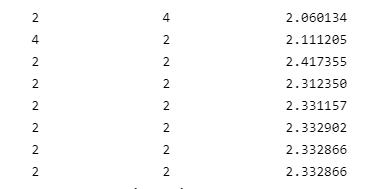
Guess1 = 180, Guess2 = 0, Tolerance = 0.000001, Root = 2.315693 at 100 false positions.

1. **Secant method**

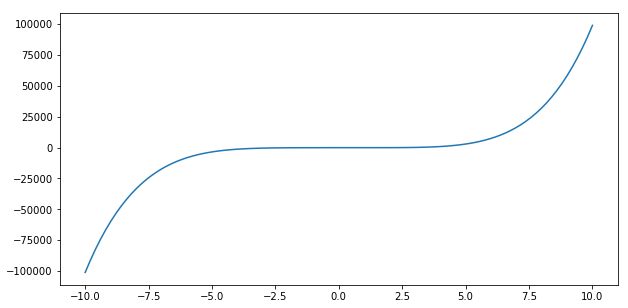
F(x) = x\*\*5 - 10\*x\*\*2 + 13\*x - 45

Guess1 = 2, Guess2 = 4, Tolerance = 0.000001, Root = 2.332866

|  |  |  |
| --- | --- | --- |
| x1 | x2 | xnew |



The graph for the above equation is given below.

****

**Chapter 2: Methods for system of linear equations**

**Vector norm:**

For any vector **x** of length one, a function ║**.** ║ ℝn →ℝ is called a **vector norm** if it satisfies the following properties:

1. ║**x**║≥ 0 for all **x**ϵ ℝn
2. ║**x**║= 0 if and only if **x** = 0
3. For any scalar α ϵ ℝ and vector **x** ϵ ℝ, ║α**x**║ = ║α ║║**x**║
4. For vectors **x, y ϵ** ℝ, ║**x + y**║≤║**x**║ + ║**y**║

The last property is known as the triangular inequality.

Some of the vector norms can be

║**x**║n =

1. For n=1

║**x**║1 =

1. For n = 2

║**x**║2 =

This is known as the Euclidean norm and it is the distance formula between **x**, y

1. For n = , max norm

**Triangular inequality:**

**Theorem: (Cauchy-shrwaz)**

Triangular inequality for :

=

This shows that the norm *I*2 satisfies the triangular inequality.

Now we discuss the distance and the converge of the vector, let **x, y ϵ ℝn**

**Distance:**

For vectors **x** and y, the distance is the function║.║ from ℝn → ℝ and for *I*2 and

**Convergence:**

A sequence of vectors in ℝn is said to be convergent to the norm if for any ϵ >0 there e**x**ist an integer N(ϵ) such that:

This shows that the distance between **x**k and **x** must approach to zero.

The theorem states that a sequence in ℝn with respect to the ma**x** norm if and only if

i=1,2,3,….

If two norms are equivalent, then a sequence of vectors that converges to a limit with respect to one norm will converge to the same limit in the other. It can be shown that all norms are *I*p norms are equivalent. For instance, if **x** ϵ ℝn then,

**Matrix norm**

A matrix norm is defined as the real valued function ║.║ for n x n matrices that maps from ℝn xn → ℝ. Following are the properties that matrices **A** and **B** say or order n x n satisfies for the matrix norm:

1. ║**A**║≥ 0
2. ║**A**║= 0 if and only if **A** has all the zero entries that is it is a null matrix.
3. For any scalar α ϵ ℝ, ║α**A**║ = ║α ║║**A**║
4. ║**A + B**║≤║**A**║ + ║**B**║ (Triangular inequality)
5. ║**AB**║≤║**A**║║**B**║

**Induced matrix norm:**

For a given vector norm ║**.**║ a matrix norm is defined as

**Eigen values and Eigen vectors:**

A n x n matrix **A** has an eigenvector **x** if there exist a scalar λ such that

**Ax = λx**

λ is the eigen value of the matrix **A.** After rearranging we get

**(A-λ*I)* x = 0**

As **x** must be the nonzero matrix so we can find the eigen values by putting (A-λI) = 0

The characteristic polynomial p of n x n matrix **A** is

**P(A) = det(A-λ*I)***

λ is the real value and if λ >1 then **x** is stretched with the factor of λ

and if λ is real and 0<λ <1 then x is shrink by the factor of its corresponding eigen value λ

and if the eigen value is negative λ<0 then the direction of **Ax** is reversed.

If **A** is a symmetric matrix, then its eigenvalues are real.

**Spectral radius**

The spectral radius **ρ(A)** of a matrix A is defined as the

**ρ (A) = max**

**λ** is the scalar and is the eigenvalues. It is used to check whether the system is convergent or not. A matrix is said to be convergent if ρ(**A**) < 1.

**Theorems of spectral radius:**

**A is a matrix of order n x n**

1. for any natural norm.

**Chapter i: Gauss Jacobi method:**

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.

**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**

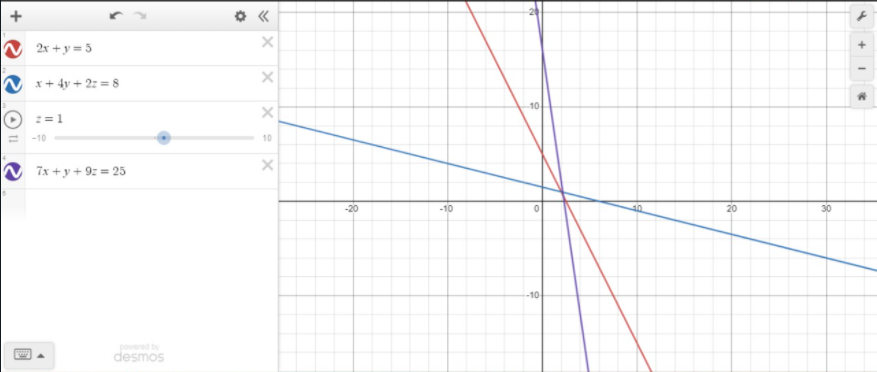
Where D, L and U are the diagonal, lower and upper matrix.

**Algorithm for Jacobi method:**

**Example:**

**solve the following system by gauss Jacobi method**

**2x1 + x2 = 5**

****

To check the convergence, we use spectral radius;

For Jacobi method,

**ρ (A) = <1**

**A =**

**D = , , L= , U=**

**L+U =**

**=**

**ρ (T) = max │λ│**

**To find the values of lambda, we solve**

**= -2λ**

**= -2λ (**

**=**

**λ = 0.341, imaginary value, imaginary value**

**As λ should be real so we consider**

**λ = 0.341**

**ρ (T) = max │λ│**

**ρ (T) < 1**

**0.341 < 1**

**⸫ The given system is convergent.**

**Now we solve the systems of equation by Jacobi method.**

**The given system is:**

**2x1 + x2 = 5**

**→ 1**

**We take the initial guess**

**Putting initial guess in 1,2,3**

**1st iteration:**

**=**

**Table for 5 iterations:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No. iterations** | **X1** | **X2** | **X3** |
| **1** | **2.5** | **2** | **2.778** |
| **2** | **1.5** | **-0.014** | **0.6111** |
| **3** | **2.5** | **1.319** | **1.612** |
| **4** | **1.8405** | **0.5690** | **0.6867** |
| **5** | **2.215** | **1.1965** | **1.28305** |

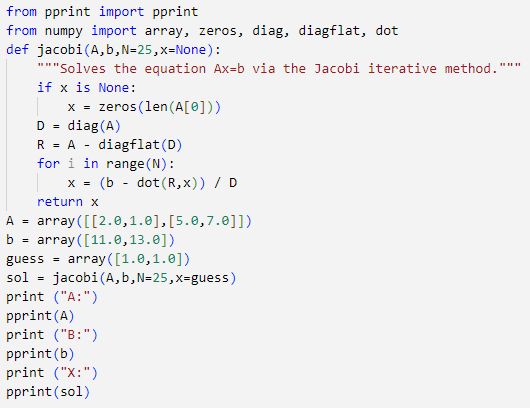
**Error:**

|  |
| --- |
| **Error** |
| **144.4%** |
| **53.32%** |
| **50.2%** |
| **28.3%** |
|  |

**Error = = 144.4%**

**Also, for next iterations:**

**Algorithm:**

****

**Chapter ii: 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**

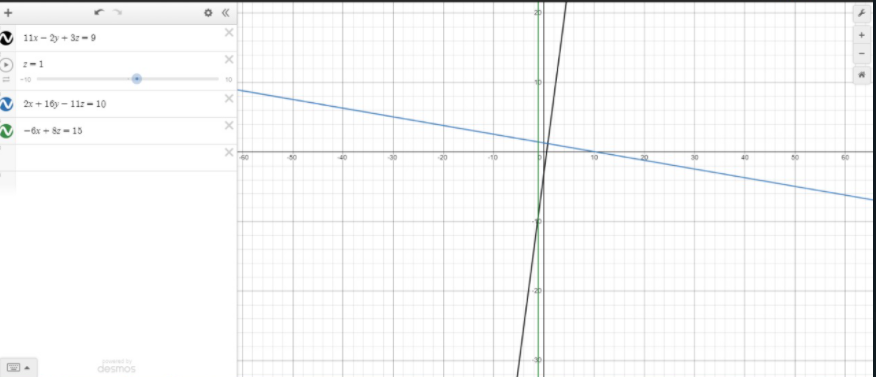
**For convergence, we can also check the spectral radius**

**ρ (Tg) < 1**

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

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

**Example:**

****

Checking the convergence:

The system is diagonally dominant; we can apply gauss seidal method.

**→ 4**

**Taking initial guess**

**1st iteration: putting the values in eq. 4,5,6**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **K** | **0** | **1** | **2** | **3** | **4** | **5** |
| **X1** | **0** | **0.818** | **0.044** | **-0.125** | **-0.017** | **-0.026** |
| **X2** | **0** | **0.522** | **2.329** | **1.925** | **1.850** | **1.9083** |
| **X3** | **0** | **2.488** | **1.908** | **1.780** | **1.862** | **1.855** |

**Error:**

**Error = =**

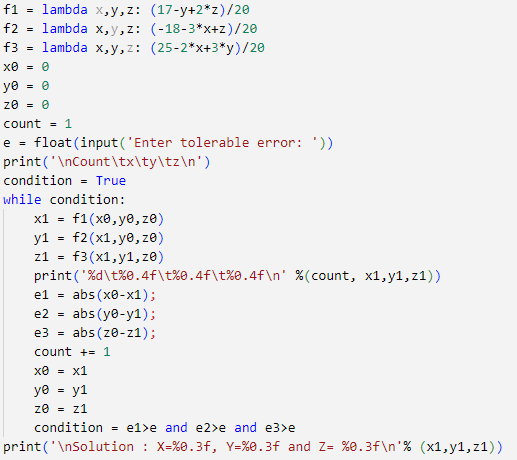
**Also, for next iterations:**

|  |
| --- |
| **Error** |
| **77%** |
| **20%** |
| **5.8%** |
| **3%** |

Gauss Seidel method has low convergence rate. It requires large number of iterations to converge. It is not productive for the large systems. A system may or may not converge.

Gauss Seidel method converges faster than Gauss Jacobi method but for some systems Gauss Seidel is not suitable as it takes more time to converge for some systems, there exist some systems for which gauss Jacobi converges quickly than gauss Seidel.

**Algorithm:**

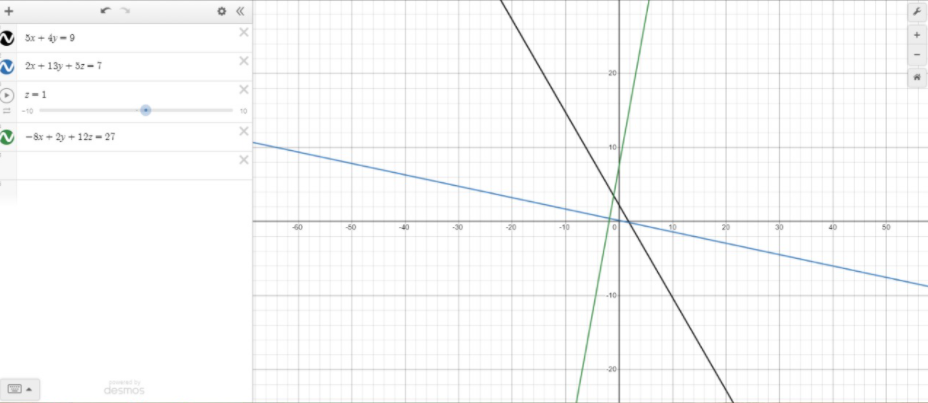
****

**Chapter iii: Successive Over Relaxation methods:**

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:**

**Solve the following system by SOR method for ω = 1.8:**

****

**ω = 1.8**

The system is strictly dominant.

**Taking initial guess**

**Putting initial guess in eq. 7,8 and 9**

**1st iteration:**

**= 3.24**

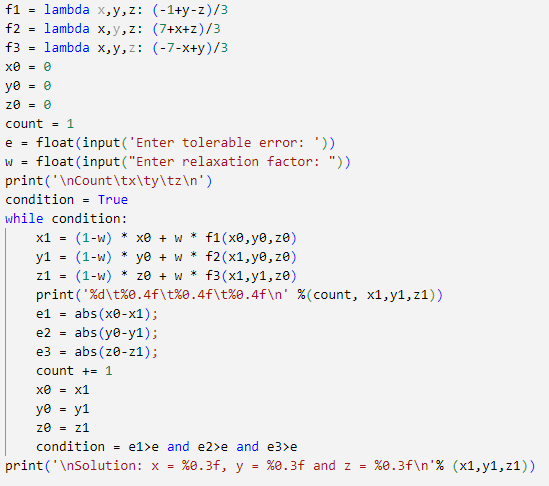
**Error:**

**Error = =**

**Tale for next iterations and errors:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **k** | **X1** | **X2** | **X3** | **Error** |
| **0** | **0** | **0** | **0** | **100%** |
| **1** | **3.24** | **0.072** | **7.9164** | **149%** |
| **2** | **5.443** | **-4.719** | **-0.213** | **102%** |
| **3** | **5.680** | **3.318** | **10.041** | **209%** |
| **4** | **-6.081** | **-6.952** | **-9.195** | **129%** |
| **5** | **18.115** | **7.8799** | **30.780** | **174%** |
| **6** | **-22.599** | **-20.385** | **-41.579** |  |

**Algorithm:**

****

**Chapter 3: Interpolation**

**What is Interpolation?**

This is a method that helps out in finding a polynomial which for a given set of points can accurately answer the behaviour of some unknown point between two known values provided in the problem.

**Choice of suitable interpolation formula:**

* Interpolation formula depends on the spacing between data points.

Some interpolations are for equally spaced data points and some are for unequally spaced points.

* Whether the interpolation is desired towards the begininig, center end of the difference table

**Interpolation for equally spaced data points**

1. Newton forward difference interpolation formula
2. Newton backward difference interpolation formula
3. Central difference interpolation formula

**Interpolation for unequally spaced data points**

1. Lagrange formula
2. Newton divided difference interpolation formula

Now let us discuss some of the interpolations mentioned above,

**Chapter i: Lagrange Interpolation**

**History:**

In 1779 LaGrange’s interpolation formula was published first by Waring. But this interpolation is also known as Lagrange’s interpolating polynomial. Euler in 1783 rediscovered the interpolation. It was used to interpolate a smooth function for n number of values. Lagrange’s interpolation is a tedious task in practice that is why it can be avoided and finite difference formulas can be used instead.

**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

**Advantages**

1. No need to compute coefficients thus we can avoid lengthy calculations.
2. We can directly plug in the values and start constructing the polynomial.
3. Convergence towards exact solution for higher order polynomials is quicker.

**Disadvantages**

1. Calculations are long as we have to take product of many terms

**Chapter ii: Newton interpolation**

**Newton forward difference interpolation formula**

* 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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** |  |  |  |  |

The standard format of displaying finite differences is called difference table.

* 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:**

**(a)** **compute the difference table for the following set of data point**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **2.5** | **3.5** | **4.5** | **5.5** |
| **F(x)** | **7** | **9** | **15** | **23** |

**(b) use newton forward difference formula to find 3rd degree polynomial**

**(c) use the above formula to interpolate for f(1.25)**

1. The forward differences is computed as follow

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Xi** | **fi** | **∆f** | **∆2f** | **∆3f** |
| **2.5** | **7** |  |  |  |
|  |  | **2** |  |  |
| **3.5** | **9** |  | **4** |  |
|  |  | **6** |  | **-4** |
| **4.5** | **15** |  | **2** |  |
|  |  | **8** |  |  |
| **5.5** | **23** |  |  |  |

1. h = X1- X0 = 3.5-2.5 = 1

P= = = -1.25

Since the value of p is within the range 0 to 1 its makes the forward difference formula applicable.

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

= 7+ p (2) + +

**=** -0.3 P3 +3 P2 -0.6P +7

1. Inserting p=-1.25 in above polynomial, we get

Fp= -0.3 (-1.25)3 +3 (-1.25)2 –(-1.25)6+7

=19.77

**Remarks:**

This formula is used when the value of p is between 0<p<1

The first two terms of this formula give the linear interpolation while the first three terms give a parabolic interpolation and so on…

**Example:**

**construct the difference table for the function f(x)=x2 for x = -2 to x=2 , at interval of 1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Xi** | **fi= xi2** | **1st order** | **2nd order** | **3rd order** |
| **-2** | **4** |  |  |  |
|  |  | **-3** |  |  |
| **-1** | **1** |  | **2** |  |
|  |  | **-1** |  | **0** |
| **0** | **0** |  | **2** |  |
|  |  | **1** |  | **0** |
| **1** | **1** |  | **2** |  |
|  |  | **3** |  |  |
| **2** | **4** |  |  |  |

**Newton backward difference interpolation formula**

* 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 bionmial 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:**

**(a)** **compute the difference table for the following set of data point**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **2.5** | **3.5** | **4.5** | **5.5** |
| **F(x)** | **7** | **9** | **15** | **23** |

**(b) use newton backward difference formula to find 3rd degree polynomial**

**(c) use the above formula to interpolate for f(3.42)**

1. The forward differences is computed as follow

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Xi** | **fi** | **∇f** | **∇2f** | **∇3f** |
| **2.5** | **7** |  |  |  |
|  |  | **2** |  |  |
| **3.5** | **9** |  | **4** |  |
|  |  | **6** |  | **-4** |
| **4.5** | **15** |  | **2** |  |
|  |  | **8** |  |  |
| **5.5** | **23** |  |  |  |

1. h = X1- X0 = 3.5-2.5 = 1

P= = = 0.92

Since the value of p is within the range 0 to 1 so we can use x0 =5.5 as origin so its makes the backward difference formula applicable

Fp= f0+ p∇ f0 +p(p+1) ∇ 2 f0+p(p+1)(p+2) ∇ 3 f0

= 23 + p85) + +

**=**-0.66P3 -0.98P2 +7.68P+23

1. Inserting p=0.92 in above polynomial, we get

Fp=- 0.66(0.92)3 -0.98(0.92)2 +7.6(0.92) +23

=28.64

**Central difference 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. **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

**Example:**

**(a)** **compute the difference table upto 3rd order only for the following set of data point**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **2.5** | **3.5** | **4.5** | **5.5** | **6.5** |
| **F(x)** | **7** | **9** | **15** | **23** | **30** |

**(b) interpolate f(2.9) using the following formulas centred at x= 4.5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **F(x)** | **𝛿f** | **𝛿2f** | **𝛿 3f** | **δ4f** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2.5** | **7** |  |  |  |  |
|  |  | **2** |  |  |  |
| **3.5** | **9** |  | **4** |  |  |
|  |  | **6** |  | **-4** |  |
| **X =4.5** | **15** |  | **2** |  | **1** |
|  |  | **8** |  | **-3** |  |
| **5.5** | **23** |  | **-1** |  |  |
|  |  | **7** |  |  |  |
| **6.5** | **30** |  |  |  |  |

P= = = 0.4

fp= f0 +  p( ) +  P2𝛿2f0 + ( +

fp= 15 + (0.4) ( ) +  (0.4)2(18) + (

= 26.8

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

**Example:**

**(a)** **compute the difference table upto 3rd order only for the following set of data point**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **2.5** | **3.5** | **4.5** | **5.5** | **6.5** |
| **F(x)** | **7** | **9** | **15** | **23** | **30** |

**(b) interpolate f() using the following formulas centered at x= 4.5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Y** | **𝛿f** | **𝛿2f** | **𝛿 3f** | **δ 4 f** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2.5** | **7** |  |  |  |  |
|  |  | **11.5** |  |  |  |
| **3.5** | **9** |  | **11.75** |  |  |
|  |  | **12** |  | **13.2** |  |
| **X = 4.5** | **15** |  | **15.5** |  | **16** |
|  |  | **19** |  | **19** |  |
| **5.5** | **23** |  | **22.5** |  |  |
|  |  | **26.5** |  |  |  |
| **6.5** | **30** |  |  |  |  |

fp= f0 + p +( + +

fp= 15 + (0.4)(12) + (11.5+15.5) +

= 18.23

1. **Everett’s interpolation formula**

Everett’s formula follows the path through the difference table :

X0 f0  …

X1 f1  …

Everett’s formula can be expressed as follows

fp= qf0 + 𝛿2f0 + 𝛿4f0 +…

+ pf1 + 𝛿2f1 + 𝛿4f1 +…

Everett’s formulas is simple and fast and generally most useful.

**Example:**

**(a)** **compute the difference table upto 3rd order only for the following set of data point**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **2.5** | **3.5** | **4.5** | **5.5** | **6.5** |
| **F(x)** | **7** | **9** | **15** | **23** | **30** |

**(b) interpolate f(6.3) using the following formulas centered at x=4.5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Y** | **𝛿f** | **𝛿2f** | **𝛿 3f** | **δ 4 f** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2.5** | **7** |  |  |  |  |
|  |  | **11.5** |  |  |  |
| **3.5** | **9** |  | **11.75** |  |  |
|  |  | **12** |  | **13.2** |  |
| **X = 4.5** | **15** |  | **15.5** |  | **16** |
|  |  | **19** |  | **19** |  |
| **5.5** | **23** |  | **22.5** |  |  |
|  |  | **26.5** |  |  |  |
| **6.5** | **30** |  |  |  |  |

q=1-p=1-0.4 =0.6

fp= qf0 + 𝛿2f0 + pf1 + 𝛿2f1

= (0.6)(7)+ (11.5)+(0.4)(9)+ (11.75)

=6.406

1. **gaussian forward and backward**

**Example:**

**(a)** **compute the difference table upto 3rd order only for the following set of data point**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **2.5** | **3.5** | **4.5** | **5.5** | **6.5** |
| **F(x)** | **7** | **9** | **15** | **23** | **30** |

**(b) interpolate f(2.7) using the following formulas centered at x= 4.5**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Y** | **𝛿f** | **𝛿2f** | **𝛿 3f** | **δ 4 f** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **2.5** | **7** |  |  |  |  |
|  |  | **11.5** |  |  |  |
| **3.5** | **9** |  | **11.75** |  |  |
|  |  | **12** |  | **13.2** |  |
| **X = 4.5** | **15** |  | **15.5** |  | **16** |
|  |  | **19** |  | **19** |  |
| **5.5** | **23** |  | **22.5** |  |  |
|  |  | **26.5** |  |  |  |
| **6.5** | **30** |  |  |  |  |

**Forward:** fp= f0 + p + +

= 7 + (0.4)11.5 + (11.75) + (13.2)

= 9.45

**Backward:**  fp= f0 + p + +

=30 + (0.4)(26.5) + (22.5)+ (19)

= 45.83

**Difference operators:**

To refer to specific entries in a difference table we use some operators, called difference operators.

The following operators are commonly used:

**∆ forward difference operator**

**∇ backward difference operator**

**µ average operator**

**𝛿 central difference operator**

**E shift operator**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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** |  |  |  |  |

1. **Forward Difference Operator:**

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** |  |  |  |  |
|  |  | **∇ 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. **Backward Difference Operator:**

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** | **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. **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

**Chapter iii: Newton divided difference interpolation**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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** |  |  |  |
| **.**  **.**  **.** | **.**  **.**  **.** | **.**  **.**  **.** |  |  |

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

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.

**Example:**

**(a)** **compute the newton divided difference table for the following set of data point (1,2),(3,5),(4,9),(8,11)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **1** | **3** | **4** | **8** |
| **F(x)** | **2** | **5** | **9** | **11** |

**(b) find polynomial of degree 3 by newton divided difference method**

**(c) interpolate f(0.018)**

**(a)construction a table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **F(x)** | **1st order** | **2nd order** | **3rd order** |
| **1** | **2** | **1.5** | **0.833** | **-0.219** |
| **3** | **5** |
| **4** | **9** | **4** | **-0.6** |
| **8** | **11** | **0.5** |

**(b) polynomial by newton divided difference method**

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

Substituting values from the table,we get

F(x)=2+(x-1)(1.5) +(x-1)(x-3)0.8333 +(x-1)(x-3)(x-4)(-0.219)

=-0.219x3+2.585x2-5.161x+5.62

**(c) f(0.115)**

F(0.018)= 0.219(0.115)3+2.585(0.115)2-5.161(0.115)+5.62

=5.527

.

**Chapter iv: 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
4. quartic interpolation

let us discuss one by one

1. **linear interpolation:**

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

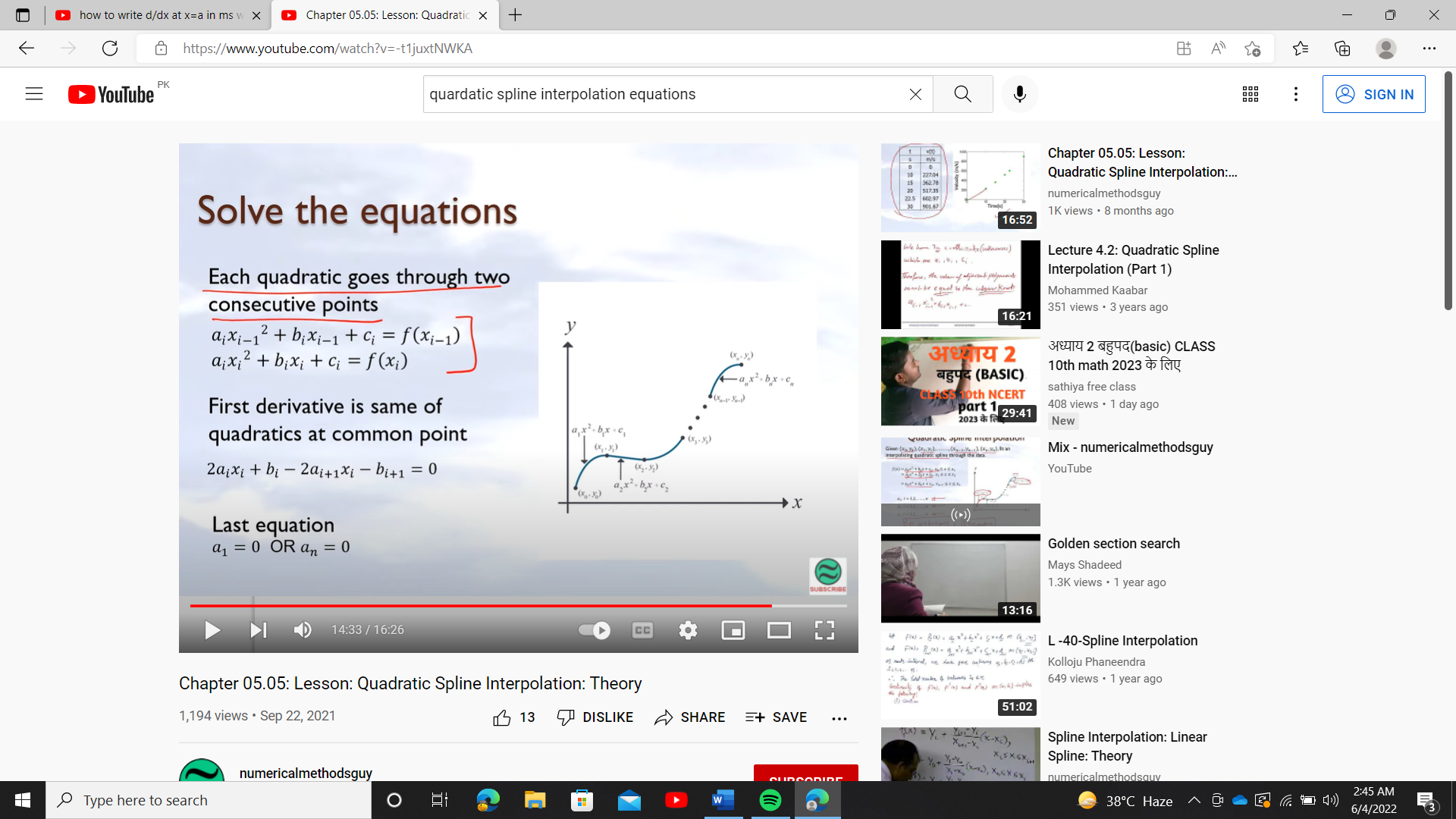
**Example: from the given set of a data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **1** | **4** | **7** | **10** |
| **F(x)** | **2** | **5** | **10** | **15** |

**(b) find piece wise polynomial by linear spline interpolation**

**(c)find f(3)**

*F(x) = f(x0) + (x-x0)*  1 ≤ x≤ 4

 =2+*(x-1)*

= x+2 = x + 2

*F(x) = f(x1) + (x-x1)*  4 ≤ x≤ 7

=5+*(x-4)*

= 5+ 5/3 (x-4) = 5 + ( 5x-20/3 ) = 5x-5 / 3

*F(x) = f(x2) + (x-x2)*  7 ≤ x≤ 10

=10+*(x-7)*

=10+ x-7) = = 5x-5 / 3

For f(3) we will use *F(x) =* 5x-5 / 3 4 ≤ x≤ 7

*F(3) =*10 / 3

1. **Quadratic polynomial:**

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 (as required)

**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** |
| **5** | **100** |
| **10** | **200** |
| **15** | **300** |
| **20** | **400** |
| **25** | **500** |

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

**(b)Using the quadratic splines as velocity functions, find the acceleration of the rocket at t = 16s.**

**Solution a)**

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

V(t) = a1t2+b1t+c1 0≤t≤5

a2t2+b2t+c2 5≤t≤10

a3t2+b3t+c3 10≤t≤15

a4t2+b4t+c4 15≤t≤20

a5t2+b5t+c5 20≤t≤25

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

**a1t2+b1t+c1 passes through t = 0 and t = 5 .**

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

a1(5)2+b1t(5)+c1 = 113(2)

**a2t2+b2t+c2 passes through t = 5 and t = 10**

a2(5)2+b2(5)+c2 = 113 (3)

a2(10)2+b2(10)+c2 = 226(4)

**a3t2+b3t+c3 passes through t = 10 and t = 15**

a3(10)2+b3(10)+c3 = 226 (5)

a3(15)2+b3(15)+c3 = 362 (6)

**a4t2+b4t+c4 passes through t = 15 and t = 20**

a4(15)2+b4(15)+c4 = 362 (7)

a4(20)2+b4(20)+c4 = 245 (8)

**a5t2+b5t+c5 passes through t = 20 and t = 25**

a5(20)2+b5(20)+c5 = 245 (9)

a5(25)2+b5(25)+c5 = 468 (10)

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

**At t = 5**

a1(5)+b1-a2(5)-b2=0 (11)

**At t = 10**

a2(10)+b2-a3(10)-b3=0 (12)

**At t = 15**

a3(15)+b3-a4(15)-b4=0 (13)

**At t = 25**

a4(25)+b4-a5(25)-b5=0 (14)

3. for last equation

|t1-t0|≤|t5-t4|

|5-25|≤|25-20|

20≤ 5 not true

So we will choose a5=0

a5t2+b5t+c5

b5t+c5 = 362 (15)

now we will write matrix from those 15 equations

A = =

|  |  |  |  |
| --- | --- | --- | --- |
| **I** | **ai** | **bi** | **ci** |
| **1** | **0** | **147.2** | **0** |
| **2** | **0.458** | **56.8** | **44.5** |
| **3** | **0.145** | **63.66** | **465** |
| **4** | **365** | **77.6** | **54.3** |
| **5** | **0.29** | **65.2** | **–135** |

After solving the matrix we have,

Therefore, splines are given by

V(t) = 22.704t0≤t≤5

= 0.458t2+56.8t+44.55≤t≤10

=0.145t2+63.66t-46510≤t≤15

=365t2+77.6t+54.3 15≤t≤20

=0.29t2+65.2t-13520≤t≤25

At t =25 s

V(16)= 0.458 (25)2+63.66(25)-456

=597.35m/s

**Solution b)**

**What is acceleration at t=25**

a(16)=

a(16)=0.458 (25)2+63.66(25)-456)

= (-0.654t+456

= -0.1572(16)+456

=24.107m/s2

1. **cubic spline interpolation:**

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

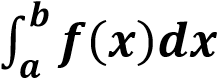
So we will get n-1 equations by equating the second derivative at n-1 interior points.

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

**Chapter 4: 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.

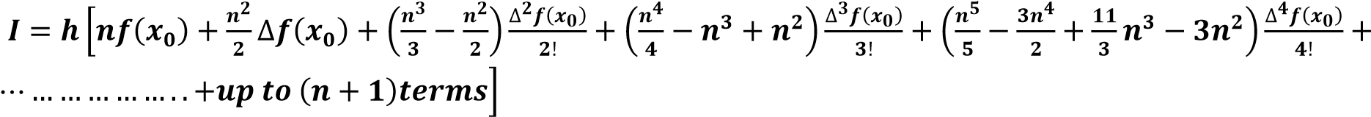
**A general formula for solving numerical integration**

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.

# **Chapter i: 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 –

**Derivation (1st method)**

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

A picture containing diagram

Description automatically generated O a= x0 x1 b= x2



f (x

0

)



f (x

1

)



Y



O



X



a=x

0



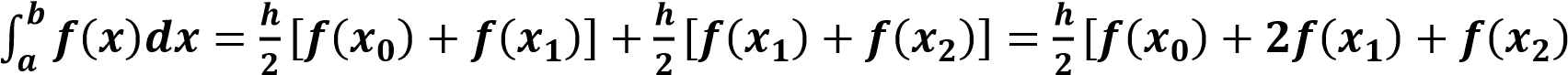
B=x

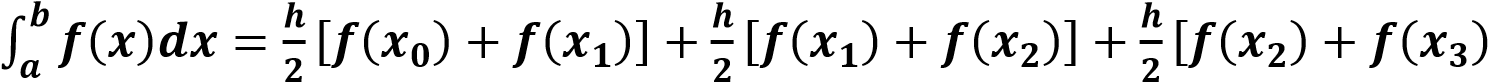
1

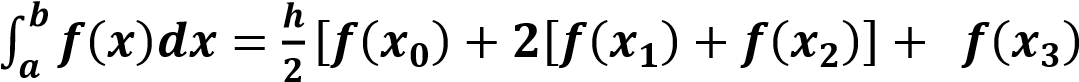
= area of trapezium =



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.

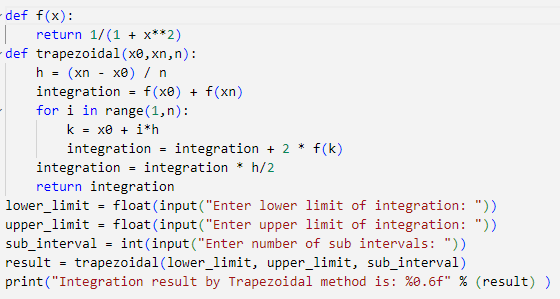
**Error :**

**Example:**

Evaluate f(x)= using Trapezoidal Rule when h=0.98

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **0** | **5/2** | **3/4** | **7/6** | **.98** |
| **F(x)** | **1** | **0.3698** | **0.5874** | **0.1472** | **0.6841** |

**Algorithm:**

****

**Chapter ii: 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.

## 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 

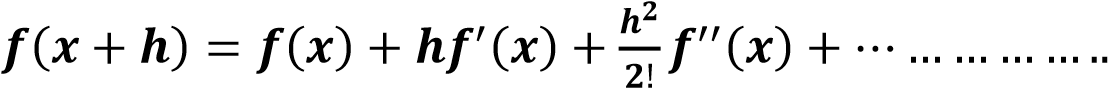
Shape

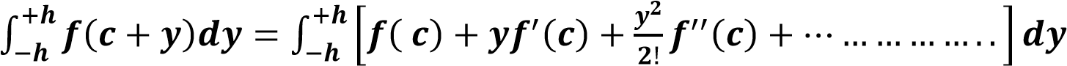
Description automatically generated with medium confidence 



Now Shape

Description automatically generated with medium confidence where y is small change

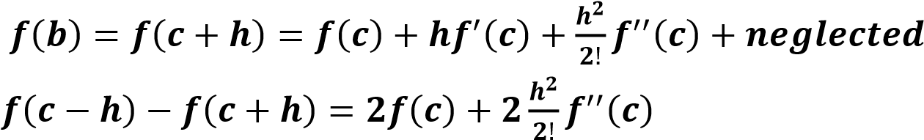
Using Taylor Series Formula 



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



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



-

-

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

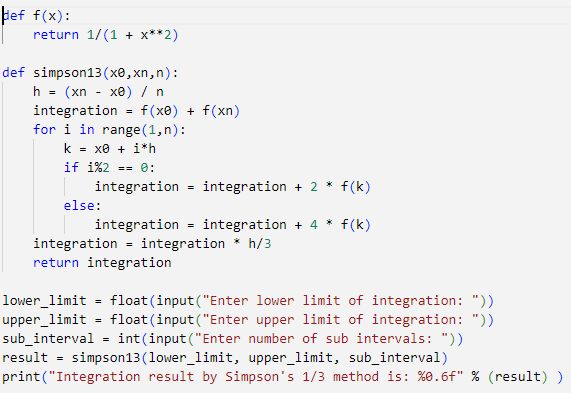
## Example:

Compute using Simpson’s (1/3) Rule when

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| X | 0 | 0.059 | 0.064 | 0.067 | 0.076 | 0.0357 | 0.041 | 0.065 | 0.987 |
| F(x) | 0.235 | 0.147 | 0.567 | 0.841 | 0.852 | 0.961 | 0.753 | 0.963 | 0.657 |

Since by Simpson’s Rule

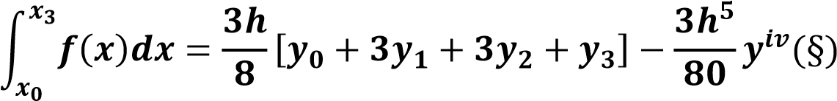
**Algorithm:**

****

**Chapter iii: Simpson’s ( 3/8 ) rule**

Rule is based on fitting four points by a cubic.

Simpson’s Rule is defined as for simple case



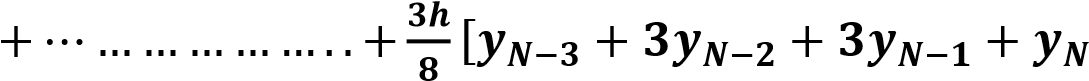
While in composite form (“n” must be divisible by 3) it is defined as



## Derivation:

Shape

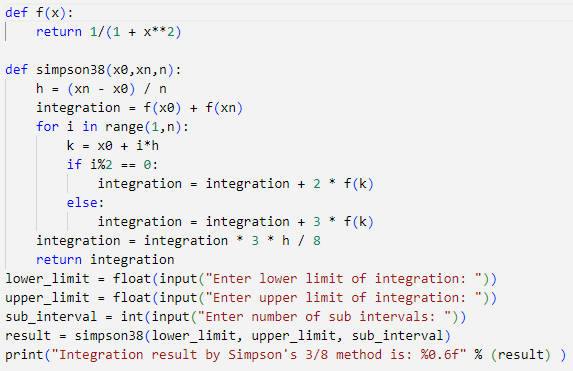
Description automatically generated with medium confidence

-



**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.

**Algorithm:**



**References:**

<https://sites.google.com/site/knowyourrootsmaxima/introduction/bisectionmethod>

<https://sites.google.com/site/knowyourrootsmaxima/introduction/newtonmethod> <https://www.youtube.com/watch?v=vfEq-WKyVbQ&t=30s>

<https://www.youtube.com/watch?v=x7m0m5A5EiQ> <http://article.sapub.org/10.5923.j.ajsp.20170702.01.html> <http://compmath-journal.org/dnload/Robin-Kumar-and-Vipan-/CMJV06I06P0290.pdf>

<https://www.youtube.com/watch?v=8F-IY4oihR4>

<http://compmath-journal.org/dnload/Robin-Kumar-and-Vipan-/CMJV06I06P0290.pdf>

<https://www.math.usm.edu/lambers/mat772/fall10/lecture17.pdf>

<https://dmpeli.math.mcmaster.ca/Matlab/Math4Q3/NumMethods/Lecture2-3.html>

<https://ece.uwaterloo.ca/~dwharder/NumericalAnalysis/10RootFinding/bisection/examples.html>

<http://compmath-journal.org/dnload/Robin-Kumar-and-Vipan-/CMJV06I06P0290.pdf>

<https://files.eric.ed.gov/fulltext/EJ1231189.pdf>