***NUMERICAL COMPUTATION PROJECT***

**NAME:** MUHAMMAD UMAR

**Reg #** SP20-BSM-027

**NAME:** MUHAMMAD ASLAM

**Reg #** SP20-BSM-023

Methods to solve non linear equations:

# Bisection method

The bisection method is used to find the   roots of polynomial equation. . It works by narrowing the gap between the positive and negative intervals until it closes in on the correct answer.

 This method narrows the gap by taking the average of the positive and negative intervals. It is a simple method and it is relatively slow.

## Bisection Method Algorithm

Follow the below procedure to get the solution for the continuous function:

For any continuous function f(x),

* Find two points, say a and b such that a < b and f(a)\* f(b) < 0
* Find the midpoint of a and b, say “t”
* t is the root of the given function if f(t) = 0; else follow the next step
* Divide the interval [a, b] – If f(t)\*f(a) <0, there exist a root between t and a  
  – else if f(t) \*f (b) < 0, there exist a root between t and b
* Repeat above three steps until f(t) = 0.

**Bisection Method Example**

**Question:** Determine the root of the given equation x2-3 = 0 for x ∈ [1, 2]

**Solution:**

### Given: x2-3 = 0

### Let f(x) = x2-3

### Now, find the value of f(x) at a= 1 and b=2.

### f(x=1) = 12-3 = 1 – 3 = -2 < 0

### f(x=2) = 22-3 = 4 – 3 = 1 > 0

### The given function is continuous, and the root lies in the interval [1, 2].

### Let “t” be the midpoint of the interval.

### I.e., t = (1+2)/2

### t =3 / 2

### t = 1.5

### Therefore, the value of the function at “t” is

### f(t) = f(1.5) = (1.5)2-3 = 2.25 – 3 = -0.75 < 0

### If f(t)<0, assume a = t.

### and

### If f(t)>0, assume b = t.

### f(t) is negative, so a is replaced with t = 1.5 for the next iterations.

### The iterations for the given functions are:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Iterations | A | b | T | f(a) | f(b) | f(t) |
| 1 | 1 | 2 | 1.5 | -2 | 1 | -0.75 |
| 2 | 1.5 | 2 | 1.75 | -0.75 | 1 | 0.062 |
| 3 | 1.5 | 1.75 | 1.625 | -0.75 | 0.0625 | -0.359 |
| 4 | 1.625 | 1.75 | 1.6875 | -0.3594 | 0.0625 | -0.1523 |
| 5 | 1.6875 | 1.75 | 1.7188 | -01523 | 0.0625 | -0.0457 |
| 6 | 1.7188 | 1.75 | 1.7344 | -0.0457 | 0.0625 | 0.0081 |
| 7 | 1.7188 | 1.7344 | 1.7266 | -0.0457 | 0.0081 | -0.0189 |

So, at the seventh iteration, we get the final interval [1.7266, 1.7344]

Hence, 1.7344 is the approximated solution.

**Falsi Position Method**

Let the root of the equation f(x) =0, lie in the interval (xk-1,xk), that is, fk-1fk < 0, where f(xk-1) = fk-1, and f(xk) = fk. This method is also called linear interpolation method or chord method or false position method.

**Algoritm:**

False Position method (regula falsi method) Steps (Rule)

**Step-1:**

Find points x0 and x1 such that x0<x1 and f(x0)⋅f(x1)<0.

**Step-2:**

Take the interval [x0,x1] and  
find next value using  
Formula-1 : x2=x0-f(x0)⋅x1-x0f(x1)-f(x0)  
or Formula-2 : x2=x0⋅f(x1)-x1⋅f(x0)f(x1)-f(x0)  
or Formula-3 : x2=x1-f(x1)⋅x1-x0f(x1)-f(x0)  
(Using any of the formula, you will get same x2 value)

**Step-3:**

If f(x2)=0 then x2 is an exact root,  
else if f(x0)⋅f(x2)<0 then x1=x2,  
else if f(x2)⋅f(x1)<0 then x0=x2.

**Step-4:**

Repeat steps 2 & 3 until f(xi)=0 or |f(xi)|≤Accuracy

**Example-1**

Find a root of an equation f(x) = x3-x-1 using False Position method

solution:

Here x3-x-1=0

Let f(x) = x3-x-1

Here

x 0 1 2

f(x) -1 -1 5

1st iteration :  
  
Here f(1)=-1<0 and f(2)=5>0  
  
∴ Now, Root lies between x0=1 and x1=2  
  
x2=x0-f(x0)⋅x1-x0f(x1)-f(x0)  
  
x2=1-(-1)⋅2-15-(-1)  
  
x2=1.16667  
  
f(x2)=f(1.16667)=-0.5787<0  
  
  
2nd iteration :  
  
Here f(1.16667)=-0.5787<0 and f(2)=5>0  
  
∴ Now, Root lies between x0=1.16667 and x1=2  
  
x3=x0-f(x0)⋅x1-x0f(x1)-f(x0)  
  
x3=1.16667-(-0.5787)⋅2-1.166675-(-0.5787)  
  
x3=1.25311  
  
f(x3)=f(1.25311)=-0.28536<0  
  
  
3rd iteration :  
  
Here f(1.25311)=-0.28536<0 and f(2)=5>0  
  
∴ Now, Root lies between x0=1.25311 and x1=2  
  
x4=x0-f(x0)⋅x1-x0f(x1)-f(x0)  
  
x4=1.25311-(-0.28536)⋅2-1.253115-(-0.28536)  
  
x4=1.29344  
  
f(x4)=f(1.29344)=-0.12954<0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | x0 | f(x0) | X1 | F(x1) | X2 | F(x2) |
| 1 | 1 | -1 | 2 | 5 | 1.16667 | -0.5787 |
| 2 | 1.16667 | -0.5787 | 2 | 5 | 1.25311 | -0.28536 |
| 3 | 1.25311 | -0.28536 | 2 | 5 | 1.29344 | -0.12954 |
| 4 | 1.25311 | -0.12954 | 2 | 5 | 1.31128 | -0.05659 |
| 5 | 1.31128 | -0.05659 | 2 | 5 | 1.31899 | -0.0243 |
| 6 | 1.31899 | -0.0243 | 2 | 5 | 1.32228 | -0.01036 |
| 7 | 1.32228 | -0.01036 | 2 | 5 | 1.32368 | -0.0044 |

**Newton Raphson Method**

The Newton Raphson Method is referred to as one of the most commonly used techniques for finding the roots of given equations. It can be efficiently generalised to find solutions to a system of equations. Moreover, we can show that when we approach the root, the method is quadratically convergent. In this article, you will learn how to use the Newton Raphson method to find the roots or solutions of a given equation, and the geometric interpretation of this method.

**Newton Raphson Method Formula:**

Let x0 be the approximate root of f(x) = 0 and let x1 = x0 + h be the correct root. Then f(x1) = 0

⇒ f(x0 + h) = 0….(1)

By expanding the above equation using Taylor’s theorem, we get:

f(x0) + hf1(x0) + … = 0

⇒ h = -f(x0) /f’(x0)

Therefore, x1 = x0 – f(x0)/ f’(x0)

Now, x1 is the better approximation than x0.

Similarly, the successive approximations x2, x3, …., xn+1 are given by

X(n+1) = Xn – F(Xn)/F’(Xn)

This is called Newton Raphson Method Formula.

**Newton’s Iterative Formula to Find bth Root of a Positive Real Number a**

The iterative formula is given by:

Xn+1 = 1/b[(b-1)Xn + a/Xn^b-1]

**Newton’s Iterative Formula to Find a Reciprocal of a Number N**

The iterative formula is given by:

xi+1 = xi(2 – xiN)

**Convergence of Newton Raphson Method:**

The order of convergence of Newton Raphson method is 2 or the convergence is quadratic. It converges if |f(x).f’’(x)| < |f’(x)|2. Also, this method fails if f’(x) = 0.

**Example 1:**

**Find the cube root of 12 using the Newton Raphson method assuming x0 = 2.5.**

Solution:

We know that, the iterative formula to find bth root of a is given by:

Xn+1 = 1/b[(b-1)Xn + a/Xn^b-1]

From the given, a = 12, b = 3

Let x0 be the approximate cube root of 12, i.e., x0 = 2.5.

So, x1 = (⅓) [2x0 + 12/x02]

= (⅓) [2(2.5) + 12/(2.5)2]

= (⅓) [5 + 12/6.25]

= (⅓)(5 + 1.92)

= 6.92/3

= 2.306

Now,

x2 = (⅓)[2x1 + 12/x12]

= (1/3) [2(2.306) + 12/(2.306)2]

= (⅓) [4.612 + 12/5.3176]

= (⅓) [4.612 + 2.256]

= 6.868/3

= 2.289

Therefore, the approximate cube root of 12 is 2.289.

**Example 2:**

Find a real root of the equation -4x + cos x + 2 = 0, by Newton Raphson method up to four decimal places, assuming x0 = 0.5.

Solution:

Given equation: -4x + cos x + 2 = 0

x0 = 0/5

Let f(x) = -4x + cos x + 2

f’(x) = -4 – sin x

Now,

f(0) = -4(0) + cos 0 + 2 = 1 + 2 = 3 > 0

f(1) = -4(1) + cos 1 + 2 = -4 + 0.5403 + 2 = -1.4597 < 0

Thus, a root lies between 0 and 1.

Let us find the first approximation.

x1 = x0 – f(x0)/f’(x0)

= 0.5 – [-4(0.5) + cos 0.5 + 2]/ [-4 – sin 0.5]

= 0.5 – [(-2 + 2 + cos 0.5)/ (-4 – sin 0.4)]

= 0.5 – [cos 0.5/ (-4 – sin 0.5)]

= 0.5 – [0.8775/ (-4 – 0.4794)]

= 0.5 – (0.8775/-4.4794)

= 0.5 + 0.1958

= 0. 6958.

**Secant Method**

The secant method is a root-finding procedure in numerical analysis that uses a series of roots of secant lines to better approximate a root of a function f.

**Algorithm:**

**Step-1:**

Find points x0 and x1 such that x0<x1 and f(x0)⋅f(x1)<0.

**Step-2:**

find next value using  
Formula-1 : x2=x0-f(x0)⋅x1-x0f(x1)-f(x0)  
or Formula-2 : x2=x0⋅f(x1)-x1⋅f(x0)f(x1)-f(x0)  
or Formula-3 : x2=x1-f(x1)⋅x1-x0f(x1)-f(x0)  
(Using any of the formula, you will get same x2 value)

**Step-3:**

If f(x2) =0 then x2 is an exact root,  
else x0=x1 and x1=x2

**Step-4:**

Repeat steps 2 & 3 until f(xi)=0 or |f(xi)|≤Accuracy

**Example-1  
Find a root of an equation f(x)=x^3-x-1 using Secant method**Solution:

Here x^3-x-1=0  
  
Let f(x)=x^3-x-1  
  
Here

|  |  |  |  |
| --- | --- | --- | --- |
| X | 0 | 1 | 2 |
| F(x) | -1 | -1 | 5 |

**1st iteration :**  
  
x0=1 and x1=2  
  
f(x0)=f(1)=-1 and f(x1)=f(2)=5  
  
∴x2=x0-f(x0)⋅x1-x0f(x1)-f(x0)  
  
x2=1-(-1)×2-15-(-1)  
  
x2=1.16667  
  
∴f(x2)=f(1.16667)=-0.5787  
  
  
**2nd iteration :**  
  
x1=2 and x2=1.16667  
  
f(x1)=f(2)=5 and f(x2)=f(1.16667)=-0.5787  
  
∴x3=x1-f(x1)⋅x2-x1f(x2)-f(x1)  
  
x3=2-5×1.16667-2-0.5787-5  
  
x3=1.25311  
  
∴f(x3)=f(1.25311)=-0.28536  
  
  
**3rd iteration :**  
x2=1.16667 and x3=1.25311  
  
f(x2)=f(1.16667)=-0.5787 and f(x3)=f(1.25311)=-0.28536  
  
∴x4=x2-f(x2)⋅x3-x2f(x3)-f(x2)  
  
x4=1.16667-(-0.5787)×1.25311-1.16667-0.28536-(-0.5787)  
  
x4=1.33721  
  
∴f(x4)=f(1.33721)=0.05388  
  
  
**4th iteration :**  
  
x3=1.25311 and x4=1.33721  
  
f(x3)=f(1.25311)=-0.28536 and f(x4)=f(1.33721)=0.05388  
  
∴x5=x3-f(x3)⋅x4-x3f(x4)-f(x3)  
  
x5=1.25311-(-0.28536)×1.33721-1.253110.05388-(-0.28536)  
  
x5=1.32385  
  
∴f(x5)=f(1.32385)=-0.0037  
  
  
**5th iteration :**  
  
x4=1.33721 and x5=1.32385  
  
f(x4)=f(1.33721)=0.05388 and f(x5)=f(1.32385)=-0.0037  
  
∴x6=x4-f(x4)⋅x5-x4f(x5)-f(x4)  
  
x6=1.33721-0.05388×1.32385-1.33721-0.0037-0.05388  
  
x6=1.32471  
  
∴f(x6)=f(1.32471)=-0.00004  
  
  
Approximate root of the equation x^3-x-1=0 using Secant method is 1.32471

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| n | x0 | f(x0) | x1 | f(x1) | x2 | f(x2) |
| 1 | 1 | -1 | 2 | 5 | 1.16667 | -0.5787 |
| 2 | 2 | 5 | 1.16667 | -0.5787 | 1.25311 | -0.28536 |
| 3 | 1.16667 | -0.5787 | 1.25311 | -0.28536 | 1.33721 | 0.05388 |
| 4 | 1.25311 | -0.28536 | 1.33721 | 0.05388 | 1.32385 | -0.0037 |
| 5 | 1.33721 | 0.05388 | 1.32385 | -0.0037 | 1.32471 | -0.00004 |

**Fixed point iteration method**

The**fixed point iteration**method in numerical analysis is used to find an approximate solution to algebraic and transcendental equations. Sometimes, it becomes very tedious to find solutions to cubic, bi-quadratic and transcendental equations; then, we can apply specific numerical methods to find the solution; one among those methods is the fixed point iteration method.

The fixed point iteration method uses the concept of a fixed point in a repeated manner to compute the solution of the given equation. A fixed point is a point in the domain of a function g such that g(x) = x. In the fixed point iteration method, the given function is algebraically converted in the form of g(x) = x.

**Important Facts**

Some interesting facts about the fixed point iteration method are

The form of x = g(x) can be chosen in many ways. But we choose g(x) for which |g’(x)|<1 at x = xo.

By the fixed-point iteration method, we get a sequence of xn, which converges to the root of the given equation.

Lower the value of g’(x), and fewer iterations are required to get the approximate solution.

The rate of convergence is more if the value of g’(x) is smaller.

The method is useful for finding the real roots of the equation, which is the form of an infinite series.

The type of convergence seen is linear.

Example-1  
Find a root of an equation f(x)=x3-x-1 using Fixed Point Iteration method  
  
Solution:  
Method-1  
Let f(x)=x3-x-1  
  
x3-x-1=0  
  
∴x3=x+1  
  
∴x=3√x+1  
  
∴ϕ(x)=3√x+1  
  
Here

|  |  |  |  |
| --- | --- | --- | --- |
| x | O | 1 | 2 |
| f(x) | -1 | -1 | 5 |

Here f(1)=-1<0 and f(2)=5>0  
  
∴ The root lies between 1 and 2  
  
x0=1+2/2=1.5  
  
x1=ϕ(x0)=ϕ(1.5)=1.35721  
  
x2=ϕ(x1)=ϕ(1.35721)=1.33086  
  
x3=ϕ(x2)=ϕ(1.33086)=1.32588  
  
x4=ϕ(x3)=ϕ(1.32588)=1.32494

The approximate root of the equation x3-x-1=0 using Iteration method is 1.32494

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | x0 | x1=ϕ(x0) | Update | Difference  x1-x0 |
| 2 | 1.5 | 1.35721 | x0=x1 | 0.14279 |
| 3 | 1.35721 | 1.33086 | x0=x1 | 0.02635 |
| 4 | 1.33086 | 1.32588 | x0=x1 | 0.00498 |
| 5 | 1.32588 | 1.32494 | x0=x1 | 0.00094 |

**Methods to solve systems of linear equations:**

**Gauss Seidel method**

In numerical linear algebra, the Gauss-Seidel method, also known as the Liebmann method or the method of successive displacement, is an iterative method used to solve a system of linear equations.

#### Example 1

Let's apply the Gauss-Seidel Method to the system from Example 1:

                                           4X1 – X2 – X3 = 3

                                          -2X1 + 6X2 + X3 =9

                                            -X1 + X2 + 7X3 -6

At each step ,given the current values X1^(k),X2^(k),X3^(k), we solve for X1^(k+1),X2^(k+1),X3^(k+1) in

4X1^(k+1)  -  X2^(k)  - X3^(k)  = 3

-2X1^(k+1)  + 6X2^(k+1)  +  = 9

-X1^(k+1)  +  X2^(k+1)  +  7X3^(k+1) = -6

To compare our results from the two methods, we again choose X^(0) = (0,0,0). We find X^(1) = (X1^(1),X 2^(1),X3^(1)) by solving

                                             4X1^(1) – 0 -0 = 3

                                     -2X1^(1)  +  6X2^(1) + 0 = 9

                                     -X1^(1)  +  X2^(1)  +7X3^(1)  =  -6

 .

Let us be clear about how we solve this system. We first solve for *X1*^(1) in the first equation and find that

X1^(1) = 3/4 = 0.750.

We then solve for X2^(1) in the second equation, using the new value of X1^(1) = 0.750, and find that

X2^(1) = [9 + 2(0.750)] / 6 = 1.750.

Finally, we solve for X3^(1) in the third equation, using the new values of X1^(1) = 0.750 and X2^(1) = 1.750, and find that

X3^(1) = [-6 + 0.750 − 1.750] / 7 = − 1.000.

The result of this first iteration of the Gauss-Seidel Method is

X1^(1)= (X1^(1*)*, X2^(1), X3^(1)) = (0.750, 1.750, − 1.000).

**Jacobi Method**

The jacobian method or **Jacobi method**is one the iterative methods for approximating the solution of a system of n linear equations in n variables. The Jacobi iterative method is considered as an iterative algorithm which is used for determining the solutions for the system of linear equations in numerical linear algebra, which is diagonally dominant. In this method, an approximate value is filled in for each diagonal element. Until it converges, the process is iterated. This algorithm was first called the Jacobi transformation process of matrix diagonalization. Jacobi Method is also known as the simultaneous displacement method.

The first iterative technique is called the Jacobi method, named after Carl Gustav Jacob Jacobi (1804–1851) to solve the system of linear equations. This method makes two assumptions:

**Assumption 1:** The given system of equations has a unique solution.

### Example1: 2*x*+5*y*=21, *x*+2*y*=8

**Solving Equations 2x+5y=21,x+2y=8 using Gauss Jacobi method**  
  
**Solution:**  
Total Equations are 2  
  
2*x*+5*y*=21  
  
*x*+2*y*=8  
  
  
From the above equations  
*xk*+1=12(21-5*yk*)  
  
*yk*+1=12(8-*xk*)  
  
Initial gauss (*x*,*y*)=(0,0)  
  
Solution steps are  
**1*st* Approximation**  
  
*x*1=12[21-5(0)]=12[21]=10.5  
  
*y*1=12[8-(0)]=12[8]=4  
 **2*nd* Approximation**  
  
*x*2=12[21-5(4)]=12[1]=0.5  
  
*y*2=12[8-(10.5)]=12[-2.5]=-1.25  
  
**3*rd* Approximation**  
  
*x*3=12[21-5(-1.25)]=12[27.25]=13.625  
  
*y*3=12[8-(0.5)]=12[7.5]=3.75  
  
**4*th* Approximation**  
  
*x*4=12[21-5(3.75)]=12[2.25]=1.125  
  
*y*4=12[8-(13.625)]=12[-5.625]=-2.8125  
  
**5*th* Approximation**  
  
*x*5=12[21-5(-2.8125)]=12[35.0625]=17.5312  
  
*y*5=12[8-(1.125)]=12[6.875]=3.4375  
  
**6*th* Approximation**  
  
*x*6=12[21-5(3.4375)]=12[3.8125]=1.9062  
  
*y*6=12[8-(17.5312)]=12[-9.5312]=-4.7656  
  
**7*th* Approximation**  
  
*x*7=12[21-5(-4.7656)]=12[44.8281]=22.4141  
  
*y*7=12[8-(1.9062)]=12[6.0938]=3.0469  
  
  
Equations are Divergent...  
Iterations are tabulated as below.

|  |  |  |
| --- | --- | --- |
| Iteration | X | y |
| 1 | 10.5 | 4 |
| 2 | 0.5 | -1.25 |
| 3 | 13.625 | 3.75 |
| 4 | 1.125 | -2.8125 |
| 5 | 17.5312 | 3.4375 |
| 6 | 1.9062 | -4.7656 |
| 7 | 22.4141 | 3.0469 |

**Successive Over relaxation (SOR) Method**

A third iterative method, called the Successive Over relaxation (SOR) Method, is a generalization of and improvement on the Gauss-Seidel Method.

The formula of Successive over relaxation (SOR) Method is.

https://www.maa.org/sites/default/files/images/cms_upload/AlgorithmSOR45121.gif

**1. Example 3*x*-*y*+*z*=-1,-*x*+3*y*-*z*=7,*x*-*y*+3*z*=-7**

**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  
1. Total Equations are 3  
  
3*x*-*y*+*z*=-1  
  
-*x*+3*y*-*z*=7  
  
*x*-*y*+3*z*=-7  
  
  
2. From the above equations, First write down the equations for Gauss Seidel method  
*xk*+1=13(-1+*yk*-*zk*)  
  
*yk*+1=13(7+*xk*+1+*zk*)  
  
*zk*+1=13(-7-*xk*+1+*yk*+1)  
  
3. 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*⋅13(-1+*yk*-*zk*)  
  
*yk*+1=(1-*w*)⋅*yk*+*w*⋅13(7+*xk*+1+*zk*)  
  
*zk*+1=(1-*w*)⋅*zk*+*w*⋅13(-7-*xk*+1+*yk*+1)  
  
4. Initial gauss (*x*,*y*,*z*)=(0,0,0) and *w*=1.25  
  
Solution steps are  
1*st* Approximation  
  
*x*1=(1-1.25)⋅0+1.25⋅13[-1+(0)-(0)]=(-0.25)⋅0+1.25⋅13[-1]=0±0.41667=-0.41667  
  
*y*1=(1-1.25)⋅0+1.25⋅13[7+(-0.41667)+(0)]=(-0.25)⋅0+1.25⋅13[6.58333]=0+2.74306=2.74306  
  
*z*1=(1-1.25)⋅0+1.25⋅13[-7-(-0.41667)+(2.74306)]=(-0.25)⋅0+1.25⋅13[-3.84028]=0±1.60012=-1.60012  
  
**2*nd* Approximation**  
  
*x*2=(1-1.25)⋅-0.41667+1.25⋅13[-1+(2.74306)-(-1.60012)]=(-0.25)⋅-0.41667+1.25⋅13[3.34317]=0.10417+1.39299=1.49715  
  
*y*2=(1-1.25)⋅2.74306+1.25⋅13[7+(1.49715)+(-1.60012)]=

(-0.25)⋅2.74306+1.25⋅13[6.89704]=-0.68576+2.87377=2.188  
  
*z*2=(1-1.25)⋅-1.60012+1.25⋅13[-7-(1.49715)+(2.188)]=(-0.25)⋅-1.60012+1.25⋅13[-6.30915]=0.40003±2.62881=-2.22878  
  
**3*rd* Approximation**  
*x*3=(1-1.25)⋅1.49715+1.25⋅13[-1+(2.188)-(-2.22878)]=(-0.25)⋅1.49715+1.25⋅13[3.41679]=-0.37429+1.42366=1.04937  
  
*y*3=(1-1.25)⋅2.188+1.25⋅13[7+(1.04937)+(-2.22878)]=(-0.25)⋅2.188+1.25⋅13[5.82059]=-0.547+2.42524=1.87824  
  
*z*3=(1-1.25)⋅-2.22878+1.25⋅13[-7-(1.04937)+(1.87824)]=(-0.25)⋅-2.22878+1.25⋅13[-6.17113]=0.5572±2.5713=-2.01411  
  
**4*th* Approximation**  
  
*x*4=(1-1.25)⋅1.04937+1.25⋅13[-1+(1.87824)-(-2.01411)]=

(-0.25)⋅1.04937+1.25⋅13[2.89235]=-0.26234+1.20515=0.9428  
  
*y*4=(1-1.25)⋅1.87824+1.25⋅13[7+(0.9428)+(-2.01411)]=

(-0.25)⋅1.87824+1.25⋅13[5.9287]=-0.46956+2.47029=2.00073  
  
*z*4=(1-1.25)⋅-2.01411+1.25⋅13[-7-(0.9428)+(2.00073)]=(-0.25)⋅-2.01411+1.25⋅13[-5.94207]=0.50353±2.47586=-1.97234  
  
**5*th* Approximation**  
  
*x*5=(1-1.25)⋅0.9428+1.25⋅13[-1+(2.00073)-(-1.97234)]=

(-0.25)⋅0.9428+1.25⋅13[2.97307]=-0.2357+1.23878=1.00308  
  
*y*5=(1-1.25)⋅2.00073+1.25⋅13[7+(1.00308)+(-1.97234)]=

(-0.25)⋅2.00073+1.25⋅13[6.03074]=-0.50018+2.51281=2.01263  
  
*z*5=(1-1.25)⋅-1.97234+1.25⋅13[-7-(1.00308)+(2.01263)]=(-0.25)⋅-1.97234+1.25⋅13[-5.99045]=0.49308±2.49602=-2.00294  
  
**6*th* Approximation**  
  
*x*6=(1-1.25)⋅1.00308+1.25⋅13[-1+(2.01263)-(-2.00294)]=(-0.25)⋅1.00308+1.25⋅13[3.01556]=-0.25077+1.25648=1.00572  
  
*y*6=(1-1.25)⋅2.01263+1.25⋅13[7+(1.00572)+(-2.00294)]=(-0.25)⋅2.01263+1.25⋅13[6.00278]=-0.50316+2.50116=1.998  
  
*z*6=(1-1.25)⋅-2.00294+1.25⋅13[-7-(1.00572)+(1.998)]=(-0.25)⋅-2.00294+1.25⋅13[-6.00771]=0.50073±2.50321=-2.00248  
 **7*th* Approximation**  
  
*x*7=(1-1.25)⋅1.00572+1.25⋅13[-1+(1.998)-(-2.00248)]=

(0.25)⋅1.00572+1.25⋅13[3.00048]=-0.25143+1.2502=0.99877  
  
*y*7=(1-1.25)⋅1.998+1.25⋅13[7+(0.99877)+(-2.00248)]=

(-0.25)⋅1.998+1.25⋅13[5.99629]=-0.4995+2.49845=1.99895  
  
*z*7=(1-1.25)⋅-2.00248+1.25⋅13[-7-(0.99877)+(1.99895)]=(-0.25)⋅-2.00248+1.25⋅13[-5.99982]=0.50062±2.49992=-1.9993  
  
**8*th* Approximation**  
  
*x*8=(1-1.25)⋅0.99877+1.25⋅13[-1+(1.99895)-(-1.9993)]=(-0.25)⋅0.99877+1.25⋅13[2.99826]=-0.24969+1.24927=0.99958  
  
*y*8=(1-1.25)⋅1.99895+1.25⋅13[7+(0.99958)+(-1.9993)]=

(-0.25)⋅1.99895+1.25⋅13[6.00028]=-0.49974+2.50012=2.00038  
  
*z*8=(1-1.25)⋅-1.9993+1.25⋅13[-7-(0.99958)+(2.00038)]=(-0.25)⋅-1.9993+1.25⋅13[-5.9992]=0.49983±2.49967=-1.99984  
 **9*th* Approximation**  
  
*x*9=(1-1.25)⋅0.99958+1.25⋅13[-1+(2.00038)-(-1.99984)]=(-0.25)⋅0.99958+1.25⋅13[3.00022]=-0.2499+1.25009=1.0002  
  
*y*9=(1-1.25)⋅2.00038+1.25⋅13[7+(1.0002)+(-1.99984)]=(-0.25)⋅2.00038+1.25⋅13[6.00035]=-0.50009+2.50015=2.00005  
  
*z*9=(1-1.25)⋅-1.99984+1.25⋅13[-7-(1.0002)+(2.00005)]=(-0.25)⋅-1.99984+1.25⋅13[-6.00014]=0.49996±2.50006=-2.0001  
  
  
Solution By SOR (successive over-relaxation) method.  
*x*=1.0002≅1  
  
*y*=2.00005≅2  
  
*z*=-2.0001≅-2

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

**Finite Differences:**

**Newton Differences**

**Newton forward difference:**

Newton's forward difference formula is a [finite difference](https://mathworld.wolfram.com/FiniteDifference.html) identity giving an interpolated value between tabulated points  in terms of the first value  and the [powers](https://mathworld.wolfram.com/Power.html) of the [forward difference](https://mathworld.wolfram.com/ForwardDifference.html) . For , the formula states

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Xi | Fi | fi | 2fi | 3fi | 4fi | 5fi |
|  |  |  |  |  |  |  |
| x0 | f0 |  |  |  |  |  |
|  |  | f0 = f1- f0 |  |  |  |  |
| x1 | f1 |  | 2f0 = Df1- Df0 |  |  |  |
|  |  | f1 = f2 - f1 |  | 3f0 = D2f1- D2f0 |  |  |
| x2 | f2 |  | 2f1 = Df2 - Df1 |  | 4f0 = D3f1- D3f0 |  |
|  |  | f2 = f3 - f2 |  | 3f1 = D2f2 - D2f1 |  | 5f0 = D4f1- D4f0 |
| x3 | f3 |  | 2f2 = Df3 - Df2 |  | 4f1 = D3f2 - D3f1 |  |
|  |  | f3 = f4 - f3 |  | 3f2 = D2f3 - D2f2 |  |  |
| x4 | f4 |  | 2f3 = Df4 - Df3 |  |  |  |
|  |  | f4 = f5 - f4 |  |  |  |  |
| x5 | f5 |  |  |  |  |  |

Newton Forward difference formula is:

p=x-x0/h

y(x)=y0 + pΔy0 + p(p-1)/2!⋅Δ2y0 + p(p-1)(p-2)/3!⋅Δ3y0 + p(p-1)(p-2)(p-3)/4!⋅Δ4y0+...

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

X=1895

Solution:  
The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1891 | 1901 | 1911 | 1921 | 1931 |
| Y | 46 | 66 | 81 | 93 | 101 |

Newton's forward difference interpolation method to find solution  
  
Newton's forward difference table is

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

The value of x at you want to find the f(x):x=1895  
  
h=x1-x0=1901-1891=10  
  
p=x-x0/h=1895-1891/10=0.4  
  
**Newton's forward difference interpolation formula is**

y(x)=y0 + pΔy0 + p(p-1)/2!⋅Δ2y0 + p(p-1)(p-2)/3!⋅Δ3y0 + p(p-1)(p-2)(p-3)4!⋅Δ4y0  
  
y(1895)=46 + 0.4×20 + 0.4(0.4-1)2×-5 + 0.4(0.4-1)(0.4-2)6×2 + 0.4(0.4-1)(0.4-2)(0.4-3)24×-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

**Error: =** h^(n+1)÷(n+1**)** !.p(p-1)(p-2)…(p-n) Δ^(n+1)Fo

**Newton Backward Difference:**

This is another way of approximating a function with an **nth** degree polynomial passing through **(n+1)** equally spaced points.

**Newton backward difference formla:**

p=x-xn/h

y(x)=yn + p∇yn + p(p+1)/2!⋅∇2yn + p(p+1)(p+2)/3!⋅∇3yn + p(p+1)(p+2)(p+3)/4!⋅∇4yn + ...

**Examples:**

**1. Find Solution using Newton's Backward Difference formula**

|  |  |
| --- | --- |
| X | Y |
| 1891 | 46 |
| 1901 | 66 |
| 1911 | 81 |
| 1921 | 93 |
| 1931 | 101 |

**x = 1925**  
**Solution:**  
The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1891 | 1901 | 1911 | 1921 | 1931 |
| Y | 46 | 66 | 81 | 93 | 101 |

Newton's backward difference interpolation method to find solution  
  
Newton's backward difference table is.

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

The value of x at you want to find the f(x):x=1925  
  
h=x1-x0=1901-1891=10  
  
p=x-xn/h=1925-1931/10=-0.6  
  
**Newton's backward difference interpolation formula is**

y(x)=yn + p∇yn + p(p+1)/2!⋅∇2yn + p(p+1)(p+2)/3!⋅∇3yn + p(p+1)(p+2)(p+3)/4!⋅∇4yn  
  
y(1925)=101 + (-0.6)×8 + -0.6(-0.6+1)2×-4 + -0.6(-0.6+1)(-0.6+2)6×-1 + -0.6(-0.6+1)(-0.6+2)(-0.6+3)24×-3  
  
y(1925)=101-4.8+0.48+0.056+0.1008  
  
y(1925)=96.8368  
  
  
Solution of newton's backward interpolation method y(1925)=96.8368

Error: h^(n+1)÷(n+1**)** !.p(p+1)(p+2)…(p+n) Δ^(n+1)Fn

**Newton central difference:**

The Central Difference calculates multiple integrals by using polynomial interpolation. Central differences are useful in solving partial differential equations. the numerical derivative should be approximated by the central difference.

**Newton Divided difference:**

|  |  |  |
| --- | --- | --- |
| Let us assume that the function **f(x)** is linear then we have | f(xi) - f(xj) |  |
|  |  |
| (xi - xj) |  |

where xi and xj are any two tabular points, is independent of xi and xj. This ratio is called the first divided difference of f(x) relative to xi , xj and is denoted byf [xi, xj].

**Newton Divided Difference Formula:**

y(x)=y0 + (x-x0)f[x0,x1] + (x-x0)(x-x1)f[x0,x1,x2] + (x-xo)(x-x1)(x-x2)F[Xo,X1,X2,X3]...

**Examples:**

**1. 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**  
  
**Solution:**  
The value of table for x and y

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 300 | 304 | 305 | 307 |
| Y | 2.4771 | 2.4829 | 2.4843 | 2.4871 |

Numerical divided differences method to find solution  
  
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 |
| 307 | 2.4871 | 0.0014 |  |

The value of x at you want to find the f(x):x=301  
  
Newton's divided difference interpolation formula is

f(x)=y0 + (x-x0)f[x0,x1] + (x-x0)(x-x1)f[x0,x1,x2]  
  
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)=2.47858

**Lagrange interpolation:**

The Newton’s forward and backward interpolation formulae can be used only when the values of *x* are at equidistant. If the values of *x* are at equidistant or not at equidistant, we use Lagrange’s interpolation formula.

Lagrange's Interpolation formula

y(x)=(x-x1)(x-x2)...(x-xn) ÷ (x0-x1)(x0-x2)...(x0-xn)×y0 + (x-x0)(x-x2)...(x-xn) ÷ (x1-x0)(x1-x2)...(x1-xn)×y1  + (x-x0)(x-x1)(x-x3)...(x-xn)÷(x2-x0)(x2-x1)(x2-x3)...(x2-xn)×y2 +...+ (x-x0)(x-x1)...(x-xn-1)(xn-x0) ÷ (xn-x1)...(xn-xn-1)×yn

**Examples  
1. Find Solution using Lagrange's Interpolation formula**

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

**x = 301**  
**Solution:**  
The value of table for x and y.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 300 | 304 | 305 | 307 |
| Y | 2.4771 | 2.4829 | 2.4843 | 2.4871 |

Lagrange's Interpolating Polynomial  
The value of x at you want to find Pn(x):x=301  
  
**Lagrange's formula is**,

f(x)=(x-x1)(x-x2)(x-x3) ÷ (x0-x1)(x0-x2)(x0-x3)×y0 + (x-x0)(x-x2)(x-x3) ÷ (x1-x0)(x1-x2)(x1-x3)×y1 + (x-x0)(x-x1)(x-x3) ÷ (x2-x0)(x2-x1)(x2-x3)×y2 + (x-x0)(x-x1)(x-x2) ÷ (x3-x0)(x3-x1)(x3-x2)×y3  
  
y(301)=(301-304)(301-305)(301-307) ÷ (300-304)(300-305)(300-307)×2.4771 + (301-300)(301-305)(301-307) ÷ (304-300)(304-305)(304-307)×2.4829 + (301-300)(301-304)(301-307) ÷ (305-300)(305-304)(305-307)×2.4843 + (301-300)(301-304)(301-305) ÷ (307-300)(307-304)(307-305)×2.4871  
  
y(301)=(-3)(-4)(-6) ÷ (-4)(-5)(-7)×2.4771 + (1)(-4)(-6) ÷ (4)(-1)(-3)×2.4829 + (1)(-3)(-6)(5)(1)(-2)×2.4843 + (1)(-3)(-4) ÷ (7)(3)(2)×2.4871  
  
y(301)=-72-140×2.4771 + 2412×2.4829 + 18-10×2.4843 + 1242×2.4871  
  
y(301)=2.4786  
  
  
**Solution of the polynomial at point 301 is y(301)=2.4786**

**Newton difference interpolation:**

Newton's Divided Difference Interpolation formula,

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

**Example:  
1. 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**  
  
**Solution:**The value of table for x and y,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 300 | 304 | 305 | 307 |
| Y | 2.4771 | 2.4829 | 2.4843 | 2.4871 |

Numerical divided differences method to find solution  
  
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 you want to find the f(x):x=301  
  
Newton's divided difference interpolation formula is

f(x)=y0 + (x-x0)f[x0,x1] + (x-x0)(x-x1)f[x0,x1,x2]  
  
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)=2.47858

**Linear spline interpolation:**

Spline interpolation is a form of interpolation where the interpolate is a special type of piecewise polynomial called a spline.

The linear spline represents a set of line segments between the two adjacent data points (Vk,Ik) and (Vk+1,Ik+1). The equations for each line segment can be immediately found in a simple form.

Here is example:

(Xo,Yo),(X1,Y1),…(Xn,Yn)

As we know that about the form:

(Xo,Yo),(X1,Y1),(X2,Y2),(X3,Y3)

(0,0),(1,5),(2,9),(3,7)

Using formula:

F(x)= F(X1) + F(X2)-F(X1)÷X2-X1(X-X1) 0≤x≤1

F(X) = 5 + (9-5)÷2-1(X-1) 1≤x≤2

F(x) = 5+4(x-1)=4x+1 1≤x≤2

**Quadratic spline interpolation:**

**find the quadratic interpolating function given the following 4 data points (-1, 0.038) (-0.8, 0.058), (-0.60, 0.10), (-0.4, 0.20)**

**SOLUTION**

The four data points (k=4) with three intervals (k-1=3), therefore, there are 3(k-1)=9 unknowns. The following are the nine equations according to scheme 1 to be solved to find the unknowns:

a1+ b1(-1) + c1(-1)^2=0.038

a1 + b1(-0.8) + c1(-0.8)^2=0.058

a2 + b2(-0.8) + c2(-0.8)^2=0.058

a2 + b2(-0.6) + c2(-0.6)^2=0.10

a3 + b3(-0.6) + c3(-0.6)^2=0.10

a3 + b3(-0.4) + c3(-0.4)^2=0.20

b1 + 2c1(-0.8) - b2 - 2c2(-0.8)=0

b2 + 2c2(-0.6) - b3 - 2c3(-0.6)=0

2c1=0

Solving the above equations produces the following interpolation function:

s1(x)=0.138 + 0.1x, -1≤x<-0.8

s2(x)=0.49 + 0.98x + 0.55x^2, -0.8 ≤x<-0.6

s3(x)=0.616 + 1.4 x + 0.9 x^2, -0.6≤x≤-0.4

**cubic spline interpolation:**

Cubic spline interpolation is a way of finding a curve that connects data points with a degree of three or less. Splines are polynomial that are smooth and continuous across a given plot and also continuous first and second derivatives where they join.

**Cubic spline formula is**

 f(x)=(xi-x)^3 ÷6h (Mi-1) + (x-xi-1)^3 ÷6h (Mi) + (xi-x) ÷h (yi-1 –h ^2 ÷6 (Mi-1)) + (x-xi-1) ÷h (yi-h ^2 ÷6 (Mi))→(1)

Mi-1 + 4Mi+Mi+1 = 6 ÷h ^2(yi-1 - 2yi + yi+1)→(2)

**Example  
1. Calculate Cubic Splines:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **1** | **2** | **3** | **4** |
| **Y** | **1** | **5** | **11** | **8** |

**y(1.5), y'(2)  
  
Solution:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 1 | 2 | 3 | 4 |
| Y | 1 | 5 | 11 | 8 |

**Cubic spline formula is**

f(x)=(xi-x)^3 ÷ 6h (Mi-1) + (x-xi-1)^3 ÷6h (Mi) + (xi-x) ÷h (yi-1 - h^2 ÷6 (Mi-1)) + (x-xi-1) ÷h (yi- h^2 ÷6 (Mi))→(1)  
  
We have, Mi-1 + 4Mi + Mi+1=6 ÷h ^2 (yi-1 - 2yi + yi+1)→(2)  
  
Here h=1,n=3  
  
Mo=0,M3=0  
  
Substitute i=1 in equation (2)  
  
M0 + 4M1 + M2=6 ÷ h ^2 (y0 - 2y1 +y2)  
  
⇒0 + 4M1 + M2=6 ÷1⋅(1-2 ⋅ 5 + 11)  
  
⇒4M1 + M2=12  
  
Substitute i=2 in equation (2)  
  
M1 + 4M2+ M3=6 ÷h^2 (y1 - 2y2 + y3)  
  
⇒M1 + 4M2 + 0=61⋅(5-2⋅11+8)  
  
⇒M1+4M2=-54  
  
**Solving these 2 equations using elimination method**

Substitute i=1 in equation (1), we get cubic spline in 1st interval [x0,x1]=[1,2]  
  
f1(x)=(x1-x) ^3 ÷6h (M0) + (x-x0)^3 ÷6h (M1) + (x1-x) ÷h(yo – h^2 ÷6 (Mo)) + (x-x0) ÷h (y1-h^2 ÷6 (M1)  
  
f1(x)=(2-x) ^3 ÷6 ⋅0 + (x-1) ^3 ÷6 ⋅6.8 + (2-x) ÷1(1-1 ÷6⋅0)+(x-1) ÷1(5-1 ÷6 ⋅6.8)  
  
f1(x)=1.1333x^3 - 3.4x^2 + 6.2667x-3, for 1≤x≤2

Substitute i=2 in equation (1), we get cubic spline in 2nd interval [x1,x2]=[2,3]  
  
f2(x)=(x2-x)^3 ÷6h(M1) + (x-x1)^3 ÷6h (M2) + (x2-x) ÷h (y1-h ^2 ÷6 (M1)) + (x-x1) ÷h (y2-h^2 ÷6 (M2))  
  
f2(x)=(3-x)^3 ÷ 6 ⋅(6.8) + (x-2)^3 ÷ 6 ⋅ (-15.2) + (3-x) ÷1 (5-1 ÷6 ⋅6.8) + (x-2) ÷1(11-1 ÷ 6 ⋅ -15.2)  
  
f2(x)=-3.6667x ^3 + 25.4x ^2 - 51.3333 x + 35.4, for 2≤x≤3

Substitute i=3 in equation (1), we get cubic spline in 3rd interval [x2,x3]=[3,4]  
  
f3(x)=(x3-x) ^3 ÷6h (M2) + (x-x2) ^3÷ 6h (M3) + (x3-x) ÷ h (y2 – h^2 ÷6 (M2)) + (x-x2) ÷h(y3 – h^2 ÷6 (M3))  
  
f3(x)=(4-x) ^3 ÷6 ⋅ -15.2 + (x-3)^3 ÷6 ⋅ 0 + (4-x)^1(11-1 ÷6 ⋅ -15.2)+(x-3) ÷1(8-1 ÷6 ⋅0)  
  
f3(x)=2.5333x ^3 - 30.4x^2 + 116.0667x-132, for 3≤x≤4

For y(1.5), 1.5∈[1,2], so substitute x=1.5 in f1(x), we get  
  
f1(1.5)=2.575  
  
For y′(2), 2∈[1,2], so find f′1(x)  
  
*f*′1(*x*)=3.4*x^*2 - 6.8*x* + 6.2667  
  
Now substitute x=2 in f′1(x), we get  
  
*f*′1(2)=6.2667

**Integration:**

**Trapizedual rule:**

In Calculus, “Trapezoidal Rule” is one of the important integration rules. The name trapezoidal is because when the area under the curve is evaluated, then the total area is divided into small trapezoids instead of rectangles. This rule is used for approximating the definite integrals where it uses the linear approximations of the functions.

The trapezoidal rule is mostly used in the numerical analysis process. To evaluate the definite integrals, we can also use Riemann Sums, where we use small rectangles to evaluate the area under the curve.

The formula of trapizedual rule is:

∫ydx = h÷2 (y0 + 2( y1 + y2 + y3+...+yn-1) +yn)

**Example  
1. Find Solution using Trapezoidal rule**

|  |  |
| --- | --- |
| **x** | **F(x)** |
| 1.4 | 4.0552 |
| 1.6 | 4.9530 |
| 1.8 | 6.0436 |
| 2.0 | 7.3891 |
| 2.2 | 9.0250 |

**Solution:**  
The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1.4 | 1.6 | 1.8 | 2.0 | 2.2 |
| Y | 4.0552 | 4.9530 | 6.0436 | 7.3891 | 9.0250 |

Using Trapezoidal Rule  
∫y dx = h ÷2[yo+y4+2(y1+y2+y3)]  
  
∫y dx = 0.2 ÷2[4.0552 + 9.025 + 2×(4.953 + 6.0436 + 7.3891)]  
  
∫ydx=0.2 ÷2[4.0552 + 9.025 +2×(18.3857)]  
  
∫y dx=4.9852  
  
Solution by Trapezoidal Rule is 4.9852

**Simpson’s 1/3 rule:**

Simpson’s rule is one of the numerical methods which is used to evaluate the definite integral. Usually, to find the definite integral, we use the fundamental theorem of calculus, where we have to apply the antiderivative techniques of integration. However, sometimes, it isn’t easy to find the antiderivative of an integral, like in Scientific Experiments, where the function has to be determined from the observed readings. Therefore, numerical methods are used to approximate the integral in such conditions. Other numerical methods used are[trapezoidal rule](https://byjus.com/maths/trapezoidal-rule/), midpoint rule, left or right approximation using Riemann sums. Here, we will discuss Simpson’s rule formula, 1/3 rule

∫y dx = h ÷3 (yo + 4(y1+y3+y5+...+yn-1)+2(y2+y4+y6+...+yn-2)+yn)

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

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

Solution:  
The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1.4 | 1.6 | 1.8 | 2 | 2.2 |
| Y | 4.0552 | 4.9530 | 6.0436 | 7.3891 | 9.0250 |

**Using Simpsons 13 Rule**  
  
∫y dx = h÷3 [(yo+y4)+4(y1+y3)+2(y2)]  
  
∫y dx = 0.23[(4.0552+9.025)+4×(4.953+7.3891)+2×(6.0436)]  
  
∫ydx=0.23[(4.0552+9.025)+4×(12.3421)+2×(6.0436)]  
  
∫ydx=4.9691  
  
Solution by Simpson's 13 Rule is 4.9691.

**Simpson’s 3/8 Rule:**

Simpson’s rule is one of the numerical methods which is used to evaluate the definite integral. Usually, to find the definite integral, we use the fundamental theorem of calculus, where we have to apply the antiderivative techniques of integration. However, sometimes, it isn’t easy to find the antiderivative of an integral, like in Scientific Experiments, where the function has to be determined from the observed readings. Therefore, numerical methods are used to approximate the integral in such conditions. Other numerical methods used are[trapezoidal rule](https://byjus.com/maths/trapezoidal-rule/), midpoint rule, left or right approximation using Riemann sums. Here, we will discuss Simpson’s rule formula, 1/3 rule, 3/8 rule.

**Formula of simpson’s 3/8 Rule:**

∫ydx=3h ÷8 (yo+2(y3+y6+...+yn-3) + 3(y1+y2+y4+y5+...+yn-2+yn-1)+yn)

**Example**  
1. Find Solution using Simpson's 3/8 rule

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

**Solution:**  
The value of table for x and y

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | 1.4 | 1.6 | 1.8 | 2 | 2.2 |
| Y | 4.0552 | 4.9530 | 6.0436 | 7.3891 | 9.0250 |

Using Simpson's 38 Rule  
  
∫ydx=3h ÷8[(y0+y4)+2(y3)+3(y1+y2)]  
  
∫ydx=3×0.28[(4.0552 + 9.025) + 2×(7.3891) + 3×(4.953 + 6.0436)]  
  
∫ydx=3×0.28[(4.0552 + 9.025) + 2×(7.3891) + 3×(10.9966)]  
  
∫ydx=4.5636  
  
Solution by Simpson's 38 Rule is 4.5636.

**Differential equations:**

In Mathematics, a differential equation is an equation that contains one or more functions with its derivatives. The derivatives of the function define the rate of change of a function at a point. The primary purpose of the differential equation is the study of solutions that satisfy the equations and the properties of the solutions.

dy/dx = f(x)

**Eulers Method:**

Euler's method is based on approximating the graph of a solution y(x) with a sequence of tangent line approximations computed sequentially, in “steps”.

**Formula of euler’s method:**

y=y(Xi) + f(Xi,y(Xi)) (x−xi).

if an initial value problem

y′=f(x,y),y(xo)=yo

xi=Xo +ih, i=0,1,…,n,

where

h=b−xo ÷n

is

y=y(xi)+f(xi,y(xi))(x−xi)

Setting x=Xi+1=Xi+hx

yi+1=y(xi)+h f(xi,y(xi))

y1=Yo+hf(Xo,Yo)

 i=1

y2=y(X1) + h f(x1,y(x1)),

y3=y2 + hf(X2,Y2).y3

In general, Euler’s method starts with the known value y(x0)=y0 and computes y1, y2, …, yn successively by with the formula

yi+1=Yi + h f(Xi,Yi), 0≤i≤n−1.

**Example:**

**Use Euler’s method with h=0.1 to find approximate values for the solution of the initial value problem**

**y′+2y=X^3 e^(−2x)**

**,y(0)=1**

**at x=0.1,0.2,0.3**

**Solution**

y′=−2y+X^3 e^(−2x) ,

y(0)=1

f(x,y)=−2y+X^3 e^(−2x) ,Xo=0,andYo=1.

Euler’s method yields

y1=y0+hf(x0,y0)=1+(0.1)f(0,1)=1+(0.1)(−2)=0.8,

Y2=y1+hf(x1,y1)=0.8+(0.1)f(0.1,0.8)=0.8+(0.1)(−2(0.8)+(0.1) ^3 e^(−0.2))=0.640081873,

Y3=y2+hf(x2,y2)=0.640081873+(0.1)(−2(0.640081873)+(0.2)^3 e^(−0.4))=0.512601754

**Taylor Series:**

Taylor series is the polynomial or a function of an infinite sum of terms. Each successive term will have a larger exponent or higher degree than the preceding term.

**Formula:**

F(a) + F’(a)÷1 (x-a) + F’’(a)÷2.1 (x-a)^2 + F’’’(a)÷3.2.1 (x-a)^3 +…

Proof:

We know that the power series can be defined as

When x = 0,

f(x)= a0

So, differentiate the given function, it becomes,

f’(x) = a1+ 2a2x + 3a3x2 + 4a4x3 +….

Again, when you substitute x = 0, we get

f’(0) =a1

So, differentiate it again, we get

f”(x) = 2a2 + 6a3x +12a4x2 + …

Now, substitute x=0 in second-order differentiation, we get

f”(0) = 2a2

Therefore, [f”(0)/2!] = a2

By generalising the equation, we get

f n (0) / n! = an

Now substitute the values in the power series we get,

Generalise f in more general form, it becomes

f(x) = b + b1 (x-a) + b2( x-a)2 + b3 (x-a)3+ ….

Now, x = a, we get

bn = fn(a) / n!

now ,substitute Bn in a generalized form

**Programmes/code:**

import numpy as np

import matplotlib.pyplot as plt

f = lambda x:0.4\*x\*5-5\*x\*3+0.2\*x-0.3

h = 0.01

# will create an array of element between -1 and 1 having 50 equal

x = np.linspace(-2,2,50)

#central difference

dff1 = (f(x+h) - f(x-h))/(2\*h)

dff2 = (f(x+h) - 2\*f(x-h))/h\*\*2

#plot

plt.plot(x,f(x),'-k',x,dff1,'--b',x,dff2,'-.r')

plt.xlabel('x')

plt.ylabel('y')

plt.legend(["f(x)" "f'(x)", "f''(x)"])

plt.grid()

import numpy as np

f = lambda x: 0.1\*x\*5 - 0.2\*x\*3 + 0.1\*x -0.2

x = 0.1

h = 0.1

df1 = 0.09405

df2 = -0.118

print("\t f'(x)\t\t err\t\t f''(x)\t\t err")

# forward difference

dff1 = (f(x+h) - f(x))/h

dff2 = (f(x+2\*h) - 2\*f(x+h) + f(x))/h\*\*2

print("FFD\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))

# backward difference

dff1 = (f(x) - f(x-h))/h

dff2 = (f(x) - 2\*f(x-h) + f(x-2\*h) )/h\*\*2

print("BFD\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))

# central differences

dff1 = (f(x+h) - f(x-h))/(2\*h)

dff2 = (f(x+h) - 2\*f(x) + f(x-h))/h\*\*2

print("CFD\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))

import numpy as np

f = lambda x: 0.1\*x\*5 - 0.2\*x\*3 + 0.1\*x -0.2

x = 0.1

h = 0.1

df1 = 0.09405

df2 = -0.118

print("\t f'(x)\t\t err\t\t f''(x)\t\t err")

# forward difference

dff1 = (f(x+h) - f(x))/h

dff2 = (f(x+2\*h) - 2\*f(x+h) + f(x))/h\*\*2

print("FFD\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))

# backward difference

dff1 = (f(x) - f(x-h))/h

dff2 = (f(x) - 2\*f(x-h) + f(x-2\*h) )/h\*\*2

print("BFD\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))

# central differences

dff1 = (f(x+h) - f(x-h))/(2\*h)

dff2 = (f(x+h) - 2\*f(x) + f(x-h))/h\*\*2

print("CFD\t% f\t% f\t% f\t% f"%(dff1,dff1-df1,dff2,dff2-df2))

**References:**

**Book from numerical analysis by Richard L. Burden.**

**Book from Introduction to numerical analysis by Nadeem Akhtar.**

**By lecture Notes.**

**THE END**