**COMSATS UNIVERSITY ISLAMABAD**

****

**Semester Project**

Submitted To: **Sir Umair Umar**

Submitted By: **Saifullah Waheed**

**(SP20-BSM-032)**

**Hafsa Ali**

**(SP20-BSM-013)**

Subject: **Numerical Computation**

Solving Non-linear Equations:

Bisection Method:

The bisection method is the simplest root-finding technique.

1. Take two points, aa and bb, on each side of the root such that f(a)f(a) and f(b)f(b) have opposite signs.
2. Calculate the midpoint c=a+b2c=a+b2
3. Evaluate f(c)f(c) and use cc to replace either aa or bb, keeping the signs of the endpoints opposite.

With this algorithm we successively half the length of the interval known to contain the root each time. We can repeat this process until the length of the interval is less than the tolerance to which we want to know the root. at each iteration (after the first iteration), one of f(a)f(a) or f(b)f(b) was computed during the previous iteration. Therefore, bisection method requires only one new function evaluation per iteration. Depending on how costly the function is to evaluate, this can be a significant cost savings.

Bisection method has linear convergence, with a constant of 1/2.

Drawbacks:

The bisection method requires us to know a little about our function. Specifically, f(x)f(x) must be continuous and we must have an interval [a,b][a,b] such that

sgn(f(a)) =−sgn(f(b)). sgn(f(a))=−sgn(f(b)).

Then, by the intermediate value theorem, we know that there must be a root in the interval [a,b][a,b]. This restriction means that the bisection method cannot solve for the root of x2x2, as it never crosses the x-axis and becomes negative.

**Example:**

From the graph above, we can see that f(x)f(x) has a root somewhere between 1 and 2. It is difficult to tell exactly what the root is, but we can use the bisection method to approximate it. Specifically, we can set a=1a=1 and b=2b=2.

**Iteration 1**

a=1b=2c=a+b2=32=1.5a=1b=2c=a+b2=32=1.5f(a)=f(1)=13−1−1=−1f(b)=f(2)=23−2−1=5f(c)=f(1.5)=1.53−1.5−1=0.875f(a)=f(1)=13−1−1=−1f(b)=f(2)=23−2−1=5f(c)=f(1.5)=1.53−1.5−1=0.875

Since f(b)f(b) and f(c)f(c) are both positive, we will replace bb with cc and further narrow our interval.

**Iteration 2**

a=1b=1.5c=a+b2=2.52=1.25a=1b=1.5c=a+b2=2.52=1.25f(a)=f(1)=−1f(b)=f(1.5)=0.875f(c)=f(1.25)=1.253−1.25−1=−0.296875f(a)=f(1)=−1f(b)=f(1.5)=0.875f(c)=f(1.25)=1.253−1.25−1=−0.296875

Since f(a)f(a) and f(c)f(c) are both negative, we will replace aa with cc and further narrow our interval.

Note that as described above, we didn’t need to recalculate f(a)f(a) or f(b)f(b) as we had already calculated them during the previous iteration. Reusing these values can be a significant cost savings.

**Iteration 3**

a=1.25b=1.5c=a+b2=1.25+1.52=1.375a=1.25b=1.5c=a+b2=1.25+1.52=1.375f(a)=f(1.25) =−0.296875f(b)=f(1.5)=0.875f(c)=f(1.375)=1.3753−1.375−1=0.224609375f(a)=f(1.25)=−0.296875f(b)=f(1.5)=0.875f(c)=f(1.375)=1.3753−1.375−1=0.224609375

Since f(b)f(b) and f(c)f(c) are both positive, we will replace bb with cc and further narrow our interval.

**Iteration nn**

When running the code for bisection method given below, the resulting approximate root determined is 1.324717957244502. With bisection, we can approximate the root to a desired tolerance (the value above is for the default tolerances).

Newton’s Method:

The Newton-Raphson Method (a.k.a. Newton’s Method) uses a Taylor series approximation of the function to find an approximate solution. Specifically, it takes the first 2 terms:

f(xk+h)≈f(xk)+f′(xk)hf(xk+h)≈f(xk)+f′(xk)h

Starting with the Taylor series above, we can find the root of this new function like so:

f(xk)+f′(xk)h=0f(xk)+f′(xk)h=0 h=−f(xk)f′(xk)h=−f(xk)f′(xk)

This value of hh can now be used to find a value of xx closer to the root of ff:

xk+1=xk+h=xk−f(xk)f′(xk)xk+1=xk+h=xk−f(xk)f′(xk)

Geometrically, (xk+1,0)(xk+1,0) is the intersection of the x-axis and the tangent of the graph at (xk,f(xk))(xk,f(xk)).

By repeatedly this procedure, we can get closer and closer to the actual root.

With Newton’s method, at each iteration we must evaluate both f(x)f(x) and f′(x)f′(x).

**Convergence**

Typically, Newton’s Method has quadratic convergence.

**Drawbacks**

Although Newton’s Method converges quickly, the additional cost of evaluating the derivative makes each iteration slower to compute. Many functions are not easily differentiable, so Newton’s Method is not always possible. Even in cases when it is possible to evaluate the derivative, it may be quite costly.

Convergence only works well if you are already close to the root. Specifically, if started too far from the root Newton’s method may not converge at all.

**Example**

We will need the following equations:

f(x)=x3−x−1f′(x)=3x2−1f(x)=x3−x−1f′(x)=3x2−1

**Iteration 1**

From the graph above, we can see that the root is somewhere near x=1x=1. We will use this as our starting position, x0x0.

x1=x0−f(x0)f′(x0)=1−f(1)f′(1)=1−13−1−13⋅12−1=1+12=1.5x1=x0−f(x0)f′(x0)=1−f(1)f′(1)=1−13−1−13⋅12−1=1+12=1.5

**Iteration 2**

x2=x1−f(x1)f′(x1)=1.5−f(1.5)f′(1.5)=1.5−1.53−1.5−13⋅1.52−1=1.5−0.8755.75=1.3478260869565217x2=x1−f(x1)f′(x1)=1.5−f(1.5)f′(1.5)=1.5−1.53−1.5−13⋅1.52−1=1.5−0.8755.75=1.3478260869565217

**Iteration 3:**

x3=x2−f(x2)f′(x2)=1.3478260869565217−f(1.3478260869565217)f′(1.3478260869565217)=1.3478260869565217−1.34782608695652173−1.3478260869565217−13⋅1.34782608695652172−1=1.3478260869565217−0.100682173091148244.449905482041588=1.325200398950907x3=x2−f(x2)f′(x2)=1.3478260869565217−f(1.3478260869565217)f′(1.3478260869565217)=1.3478260869565217−1.34782608695652173−1.3478260869565217−13⋅1.34782608695652172−1=1.3478260869565217−0.100682173091148244.449905482041588=1.325200398950907

As you can see, Newton’s Method is already converging significantly faster than the Bisection Method.

**Iteration nn:**

When running the code for Newton’s method given below, the resulting approximate root determined is 1.324717957244746.

Secant Method:

Like Newton’s Method, secant method uses the Taylor Series to find the solution. However, you may not always be able to take the derivative of a function. Secant method gets around this by approximating the derivative as:

f′(xk)≈f(xk)−f(xk−1)xk−xk−1f′(xk)≈f(xk)−f(xk−1)xk−xk−1

**Convergence**

Secant method has superlinear convergence.

More specifically, the rate of convergence rr is:

r=1+√52≈1.618r=1+52≈1.618

This happens to be the golden ratio.

**Drawbacks**

This technique has many of the same drawbacks as Newton’s Method, but does not require a derivative. It does not converge as quickly as Newton’s Method. It also requires two starting guesses near the root.

**Example**

Let’s start with x0=1x0=1 and x−1=2x−1=2.

**Iteration 1**

First, find an approximate for the derivative (slope):

f′(x0)≈f(x0)−f(x−1)x0−x−1=f(1)−f(2)1−2=(13−1−1)−(23−2−1)1−2=(−1)−(5)1−2=6f′(x0)≈f(x0)−f(x−1)x0−x−1=f(1)−f(2)1−2=(13−1−1)−(23−2−1)1−2=(−1)−(5)1−2=6

Then, use this for Newton’s Method:

x1=x0−f(x0)f′(x0)=1−f(1)f′(1)=1−13−1−16=1+16=1.1666666666666667x1=x0−f(x0)f′(x0)=1−f(1)f′(1)=1−13−1−16=1+16=1.1666666666666667

**Iteration 2**

f′(x1)≈f(x1)−f(x0)x1−x0=f(1.1666666666666667)−f(1)1.1666666666666667−1=(1.16666666666666673−1.1666666666666667−1)−(13−1−1)1.1666666666666667−1=(−0.5787037037037035)−(−1)1.1666666666666667−1=2.5277777777777777f′(x1)≈f(x1)−f(x0)x1−x0=f(1.1666666666666667)−f(1)1.1666666666666667−1=(1.16666666666666673−1.1666666666666667−1)−(13−1−1)1.1666666666666667−1=(−0.5787037037037035)−(−1)1.1666666666666667−1=2.5277777777777777x2=x1−f(x1)f′(x1)=1.1666666666666667−f(1.1666666666666667)f′(1.1666666666666667)=1.1666666666666667−1.16666666666666673−1.1666666666666667−12.5277777777777777=1.1666666666666667−−0.57870370370370352.5277777777777777=1.3956043956043955x2=x1−f(x1)f′(x1)=1.1666666666666667−f(1.1666666666666667)f′(1.1666666666666667)=1.1666666666666667−1.16666666666666673−1.1666666666666667−12.5277777777777777=1.1666666666666667−−0.57870370370370352.5277777777777777=1.3956043956043955

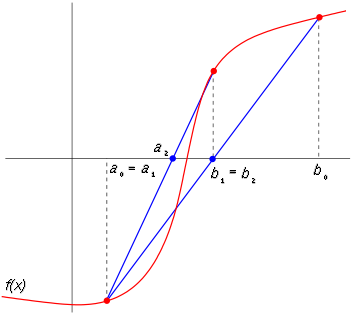
**Iteration 3**

f′(x2)≈f(x2)−f(x1)x2−x1=f(1.3956043956043955)−f(1.1666666666666667)1.3956043956043955−1.1666666666666667=(1.39560439560439553−1.3956043956043955−1)−(1.16666666666666673−1.1666666666666667−1)1.3956043956043955−1.1666666666666667=(0.3226305152401032)−(−0.5787037037037035)1.3956043956043955−1.1666666666666667=3.9370278683465503f′(x2)≈f(x2)−f(x1)x2−x1=f(1.3956043956043955)−f(1.1666666666666667)1.3956043956043955−1.1666666666666667=(1.39560439560439553−1.3956043956043955−1)−(1.16666666666666673−1.1666666666666667−1)1.3956043956043955−1.1666666666666667=(0.3226305152401032)−(−0.5787037037037035)1.3956043956043955−1.1666666666666667=3.9370278683465503x3=x2−f(x2)f′(x2)=1.3956043956043955−f(1.3956043956043955)f′(1.3956043956043955)=1.3956043956043955−1.39560439560439553−1.3956043956043955−13.9370278683465503=1.3956043956043955−0.32263051524010323.9370278683465503=1.3136566609098987x3=x2−f(x2)f′(x2)=1.3956043956043955−f(1.3956043956043955)f′(1.3956043956043955)=1.3956043956043955−1.39560439560439553−1.3956043956043955−13.9370278683465503=1.3956043956043955−0.32263051524010323.9370278683465503=1.3136566609098987

**Iteration nn:**

When running the code for secant method given below, the resulting approximate root determined is 1.324717957244753.

The regula falsi (false position) method:

[](https://en.wikipedia.org/wiki/File:False_position_method.svg)

The first two iterations of the false position method. The red curve shows the function f and the blue lines are the secants.

The convergence rate of the bisection method could possibly be improved by using a different solution estimate.

The regula falsi method calculates the new solution estimate as the [x-intercept](https://en.wikipedia.org/wiki/X-intercept) of the [line segment](https://en.wikipedia.org/wiki/Line_segment) joining the endpoints of the function on the current bracketing interval. Essentially, the root is being approximated by replacing the actual function by a line segment on the bracketing interval and then using the classical double false position formula on that line segment.[[9]](https://en.wikipedia.org/wiki/Regula_falsi#cite_note-9)

More precisely, suppose that in the k-th iteration the bracketing interval is (ak, bk). Construct the line through the points (ak, f (ak)) and (bk, f (bk)), as illustrated. This line is a [secant](https://en.wikipedia.org/wiki/Secant_method) or chord of the graph of the function f. In [point-slope form](https://en.wikipedia.org/wiki/Slope), its equation is given by

{\displaystyle y-f(b\_{k})={\frac {f(b\_{k})-f(a\_{k})}{b\_{k}-a\_{k}}}(x-b\_{k}).}

Now choose ck to be the x-intercept of this line, that is, the value of x for which y = 0, and substitute these values to obtain

{\displaystyle f(b\_{k})+{\frac {f(b\_{k})-f(a\_{k})}{b\_{k}-a\_{k}}}(c\_{k}-b\_{k})=0.}

Solving this equation for ck gives:

{\displaystyle c\_{k}=b\_{k}-f(b\_{k}){\frac {b\_{k}-a\_{k}}{f(b\_{k})-f(a\_{k})}}={\frac {a\_{k}f(b\_{k})-b\_{k}f(a\_{k})}{f(b\_{k})-f(a\_{k})}}.}

This last symmetrical form has a computational advantage:

As a solution is approached, ak and bk will be very close together, and nearly always of the same sign. Such a subtraction can lose significant digits. Because f (bk) and f (ak) are always of opposite sign the “subtraction” in the numerator of the improved formula is effectively an addition (as is the subtraction in the denominator too).

At iteration number k, the number ck is calculated as above and then, if f (ak) and f (ck) have the same sign, set ak + 1 = ck and bk + 1 = bk, otherwise set ak + 1 = ak and bk + 1 = ck. This process is repeated until the root is approximated sufficiently well.

The above formula is also used in the secant method, but the secant method always retains the last two computed points, and so, while it is slightly faster, it does not preserve bracketing and may not converge.

The fact that regula falsi always converges, and has versions that do well at avoiding slowdowns, makes it a good choice when speed is needed. However, its rate of convergence can drop below that of the bisection method.

Fixed point iteration:

Hefixed point iterationmethod 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.

Suppose we have an equation f(x) = 0, for which we must find the solution. The equation can be expressed as x = g(x). Choose g(x) such that |g’(x)| < 1 at x = xo where xo, is some initial guess called fixed point iterative scheme. Then the iterative method is applied by successive approximations given by xn = g (xn – 1), that is, x1 = g(xo), x2 = g(x1) and so on.

**Example 1:**

Find the first approximate root of the equation 2x3 – 2x – 5 = 0 up to 4 decimal places.

Given f(x) = 2x3 – 2x – 5 = 0

As per the algorithm, we find the value of xo, for which we must find a and b such that f(a) < 0 and f(b) > 0

F (0) = – 5

F (1) = – 5

F (2) = 7

Thus, a = 1 and b = 2

Therefore, xo = (1 + 2)/2 = 1.5

Now, we shall find g(x) such that |g’(x)| < 1 at x = xo

2x3 – 2x – 5 = 0

⇒ x = [(2x + 5)/2]1/3

g(x) = [(2x + 5)/2]1/3 which satisfies |g’(x)| < 1 at x = 1.5

Now, applying the iterative method xn, = g (xn – 1) for n = 1, 2, 3, 4, 5, …

For n = 1; x1 = g(xo) = [{2(1.5) + 5}/2]1/3 = 1.5874

For n = 2; x2 = g(x1) = [{2(1.5874) + 5}/2]1/3 = 1.5989

For n = 3; x3 = g(x2) = [{2(1.5989) + 5}/2]1/3 = 1.60037

For n = 4; x4 = g(x3) = [{2(1.60037) + 5}/2]1/3 = 1.60057

For n = 5; x5 = g(x4) = [{2(1.60057) + 5}/2]1/3 = 1.60059

For n = 6; x6 = g(x5) = [{2(1.60059) + 5}/2]1/3 = 1.600597 ≈ 1.6006

The approximate root of 2x3 – 2x – 5 = 0 by the fixed-point iteration method is 1.6006.

Method To Solve Linear Equation:

Successive over-relaxation(SOR):

The method of successive over-relaxation (SOR) is a variant of the [Gauss–Seidel method](https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method) for solving a [linear system of equations](https://en.wikipedia.org/wiki/Linear_system_of_equations), resulting in faster convergence. A similar method can be used for any slowly converging [iterative process](https://en.wikipedia.org/wiki/Iterative_method).

**Example:**

3x-y+z=-1, -x+3y-z=7, x-y+3z=-7  
3x-y+z=-1  
-x+3y-z=7  
x-y+3z=-7  
Now,  
xk+1=13(-1+yk-zk)  
yk+1=13(7+xk+1+zk)  
zk+1=13(-7-xk+1+yk+1)  
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)  
Initial gauss (x, y, z) = (0,0,0) and w=1.25

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

Jacobi Method:

The Jacobi method is a method of solving a [matrix equation](https://mathworld.wolfram.com/MatrixEquation.html) on a matrix that has no zeros along its main diagonal. Each diagonal element is solved for, and an approximate value plugged in. The process is then iterated until it converges.

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](https://byjus.com/maths/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.

**Example:**

**2*x*+5*y*=21, *x*+2*y*=8**

Solution:  
2x+5y=21  
x+2y=8  
From the above equations  
xk+1=12(21-5yk)

yk+1=12(8-xk)  
Initial gauss (x,y)=(0,0)

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 |

**Example:**

45x1 + 2x2 + 3x3 = 58

–3x1 + 22x2 + 2x3 = 47

5x1 + x2 + 20x3 = 67

Solution:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Iteration | 0 | 1 | 2 | 3 | 4 |
| x1 | 0.000 | 1.28889 | 0.99198 | 0.99958 | 1.0000 |
| x2 | 0.000 | 2.31212 | 2.00689 | 1.99979 | 1.99999 |
| x3 | 0.000 | 2.91217 | 3.00166 | 3.00012 | 3.00000 |

Gauss–Seidel method:

The Gauss–Seidel method is an [iterative technique](https://en.wikipedia.org/wiki/Iterative_method) for solving a square system of n linear equations with unknown x: Gauss–Seidel method is an improved form of Jacobi method, also known as the *successive displacement method*. Again, we assume that the starting values are *u*2 = *u*3 = *u*4 = 0. The difference between the Gauss–Seidel and Jacobi methods is that the Jacobi method uses the values obtained from the previous step while the Gauss–Seidel method always applies the latest updated values during the iterative procedures, as demonstrated in Table. The reason the Gauss–Seidel method is commonly known as the successive displacement method is because the second unknown is determined from the first unknown in the current iteration, the third unknown is determined from the first and second unknowns, etc.

The Gauss-Seidel method involves updating the sub-diagonal elements as the computation proceeds. The iteration process is {\displaystyle A\mathbf {x} =\mathbf {b} .}

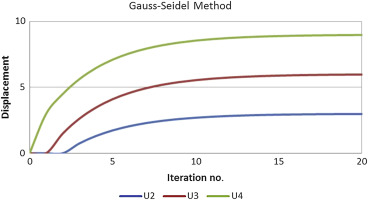
x1^(k+1)=1/a11(b1−a12x2^k−…−a1nxn^k),

x2^(k+1)=1/a22(b2−a21x1^(k+1)−…−a2nxn^k),

⋮

Xn^(k+1)=1/ann(bn−an1x1^(k+1)−…−an(n−1)^x n−1^(k+1)

| Iteration | U2 | U3 | U4 |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 3 |
| 2 | 0 | 1.5 | 4.5 |
| 3 | 0.75 | 2.625 | 5.625 |
| 4 | 1.3125 | 3.46875 | 6.46875 |
| 5 | 1.734375 | 4.101563 | 7.101563 |
| 6 | 2.050781 | 4.576172 | 7.576172 |
| 7 | 2.288086 | 4.932129 | 7.932129 |
| 8 | 2.466064 | 5.199097 | 8.199097 |
| 9 | 2.599548 | 5.399323 | 8.399323 |
| 10 | 2.699661 | 5.549492 | 8.549492 |
| 11 | 2.774746 | 5.662119 | 8.662119 |
| 12 | 2.831059 | 5.746589 | 8.746589 |
| 13 | 2.873295 | 5.809942 | 8.809942 |
| 14 | 2.904971 | 5.857456 | 8.857456 |
| 15 | 2.928728 | 5.893092 | 8.893092 |
| 16 | 2.946546 | 5.919819 | 8.919819 |
| 17 | 2.95991 | 5.939864 | 8.939864 |
| 18 | 2.969932 | 5.954898 | 8.954898 |
| 19 | 2.977449 | 5.966174 | 8.966174 |
| 20 | 2.983087 | 5.97463 | 8.97463 |



.

Forward Difference:

Consider a linear interpolation between the current data value (t0,I0) and the future data value (t1,I1). The slope of the secant line between these two points approximates the derivative by the forward (two-point) difference:

I'(t0) = (I1-I0) / (t1 - t0)

Forward differences are useful in solving ordinary differential equations by single-step predictor-corrector methods (such as Euler and Runge-Kutta methods). For instance, the forward difference above predicts the value of I1 from the derivative I'(t0) and from the value I0. If the data values are equally spaced with the step size h, the truncation error of the forward difference approximation has the order of O(h).

Formula

|  |
| --- |
| Newton's Forward Difference formula |
| p=x-x0h 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+... |

**Examples**1. Find Solution using Newton's Forward Difference formula

|  |  |
| --- | --- |
| 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 | Δ2y | Δ3y | Δ4y |
| 1891 | 46 |  |  |  |  |
|  |  | 20 |  |  |  |
| 1901 | 66 |  | -5 |  |  |
|  |  | 15 |  | 2 |  |
| 1911 | 81 |  | -3 |  | -3 |
|  |  | 12 |  | -1 |  |
| 1921 | 93 |  | -4 |  |  |
|  |  | 8 |  |  |  |
| 1931 | 101 |  |  |  |  |

The value of x at you want to find the f(x):x=1895  
h=x1-x0=1901-1891=10  
p=x-x0h=1895-189110=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

Backward difference

Consider a linear interpolation between the current data value (t0,I0) and the past data value (t-1,I-1). The slope of the secant line between these two points approximates the derivative by the backward (two-point) difference:

I'(t0) = (I0-I-1) / (t0 - t-1)

Backward differences are useful for approximating the derivatives if data in the future are not yet available. Moreover, data in the future may depend on the derivatives approximated from the data in the past (such as in control problems). If the data values are equally spaced with the step size h, the truncation error of the backward difference approximation has the order of O(h) (as bad as the forward difference approximation).  
  
Formula

|  |
| --- |
| Newton's Backward Difference formula |
| p=x-xnh 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 | f(x) |
| 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 | 93 |  | -4 |  |  |
|  |  | 8 |  |  |  |
| 1931 | 101 |  |  |  |  |

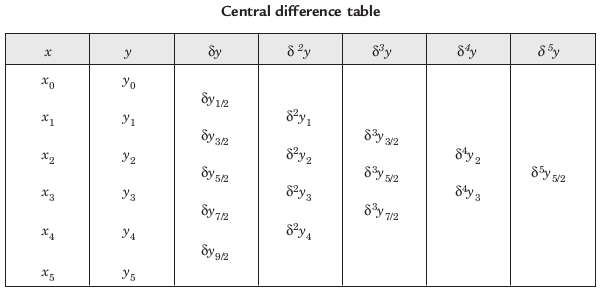
The value of x at you want to find the f(x):x=1925  
h=x1-x0=1901-1891=10  
p=x-xnh=1925-193110=-0.6  
Newton's backward difference interpolation formula is  
y(x)=yn+p∇yn+p(p+1)2!⋅∇2yn+p(p+1)(p+2)3!⋅∇3yn+p(p+1)(p+2)(p+3)4!⋅∇4yn  
y(1925)=101+(-0.6)×8+-0.6(-0.6+1)2×-4+-0.6(-0.6+1)(-0.6+2)6×-1+-0.6(-0.6+1)(-0.6+2)(-0.6+3)24×-3  
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

Central difference

Finally, consider a linear interpolation between the past data value (t-1,I-1) and the future data value (t1,I1). The slope of the secant line between these two points approximates the derivative by the central (three-point) difference:

I'(t0) = (I1-I-1) / (t1 - t-1)

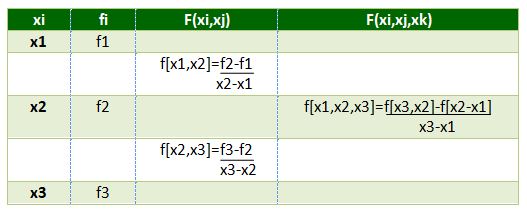
If the data values are equally spaced, the central difference is an average of the forward and backward differences. The truncation error of the central difference approximation is order of O(h2), where h is the step size. It is clear that the central difference gives a much more accurate approximation of the derivative compared to the forward and backward differences. Central differences are useful in solving partial differential equations. If the data values are available both in the past and in the future, the numerical derivative should be approximated by the central difference.



Newton divided difference interpolation:

Newton’s divided difference interpolation is a interpolation technique used when the interval difference is not same for all sequence of values.

Suppose f(x0), f(x1), f(x2)………f(xn) be the (n+1) values of the function y=f(x) corresponding to the arguments x=x0, x1, x2…xn, where interval differences are not same



Formula

|  |
| --- |
| Newton's Divided Difference Interpolation formula |
| y(x)=y0+(x-x0)f[x0,x1]+(x-x0)(x-x1)f[x0,x1,x2]+... |

**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 |
|  |  | 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

Lagrange's Interpolation:

Lagrange's interpolation is also an Nth degree polynomial approximation to f(x) and the Nth degree polynomial passing through (N+1) points is unique hence the Lagrange's and Newton's divided difference approximations are one and the same.

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'S FORWARD DIFFERENCE INTERPOLATION:

Newton's forward difference formulae :

Let the function  f  is known at  n+1  equally spaced data points  a = x0 < x1 < ... <  =  xn = b  in the interval [a,b] as f0, f1, . . . fn. Then the n  the degree polynomial approximation of  f(x)  can be given as

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | n |  |  |  |  |
| f(x)      Pn(x)   = |  | ( | r i | ) | i f0 |
|  | i=0 |  |  |  |  |

where r = (x-x0 ) / n      x  = x0  +  r h   0 <r <n

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| and | ( | r | ) | are the binomial coefficients defined as | ( | r | ) = 1, | ( | r | ) | = | r(r - 1) . . . (r - i + 1) | for any integer i > 0 |
| i | 0 | i | i! |

Forward difference table : Consider the function value (xi, fi)  i = 0,1,2,--5 then the

forward difference table is

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **xi** | **fi** | **fi** | **2fi** | **3fi** | **4fi** | **5fi** |
|  |  |  |  |  |  |  |
| x0 | f0 |  |  |  |  |  |
|  |  | f0 = f1- f0 |  |  |  |  |
| x1 | f1 |  | 2f0 = f1- f0 |  |  |  |
|  |  | f1 = f2 - f1 |  | 3f0 = 2f1- 2f0 |  |  |
| x2 | f2 |  | 2f1 = f2 - f1 |  | 4f0 = 3f1- 3f0 |  |
|  |  | f2 = f3 - f2 |  | 3f1 = 2f2 - 2f1 |  | 5f0 = 4f1- 4f0 |
| x3 | f3 |  | 2f2 = f3 - f2 |  | 4f1 = 3f2 - 3f1 |  |
|  |  | f3 = f4 - f3 |  | 3f2 = 2f3 - 2f2 |  |  |
| x4 | f4 |  | 2f3 = f4 - f3 |  |  |  |
|  |  | f4 = f5 - f4 |  |  |  |  |
| x5 | f5 |  |  |  |  |  |

**Example :**

If  f(x) is known at the following data points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| xi | 0 | 1 | 2 | 3 | 4 |
| fi | 1 | 7 | 23 | 55 | 109 |

then find f(0.5) and f(1.5) using Newton's forward difference formula.

Solution :

Forward difference table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| xi | fi | fi | 2fi | 3fi | 4fi |
|  |  |  |  |  |  |
| 0 | 1 |  |  |  |  |
|  |  | 6 |  |  |  |
| 1 | 7 |  | 10 |  |  |
|  |  | 16 |  | 6 |  |
| 2 | 23 |  | 16 |  | 0 |
|  |  | 32 |  | 6 |  |
| 3 | 55 |  | 22 |  |  |
|  |  | 54 |  |  |  |
| 4 | 109 |  |  |  |  |

(Note : The given data satifies f(x) = x3 + 2x2 + 3x +1, i.e the function is a third degree polynomial and hence third forward differences are constant by the result).

By Newton's forward difference formula

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| f(x)  = f0 + rf0 + | r(r-1) | 2f0 | + | r(r-1)(r-2) | 3f0 |
| 2! | 3! |

at x = 0.5,  r = (x - x0) / h = (0.5 - 0) / 1 = 0.5

|  |  |  |  |
| --- | --- | --- | --- |
| f(0.5) = 1 + 0.5 x 6 + | 0.5(0.5 - 1) x 10 5 | + | 0.5(0.5 - 1)(0.5 - 2) x 6 |
| 2 | 6 |

         = 1 + 3 + 2.5 x (-0.5) + (-0.25)(-1.5)

         = 3.125

Exact value is  
f(0.5) = (0.5)3 + 2(0.5)2 + 3(0.5) + 1  
          = 0.125 + 0.5 + 1.5 + 1  
          = 3.125

**Error in the Interpolation :**

En(x) = (x - x0)(x - x1) . . .(x - xn)  f(n+1)() / (n+1)!                 x0 <  < xn

So for the Newton's method where the nodel points  xi,  i = 0, 1, . . . n  are equally spaced, the error is   En(x) = (x - x0)(x - x0 - h) . . .(x - x0 - nh)  f(n+1)() / (n+1)!

|  |  |  |
| --- | --- | --- |
| = | r(r-1). . .(r-n) | h(n+1)f(n+1) |
| (n+1)! |

|  |  |  |
| --- | --- | --- |
| = ( | r | ) h(n+1)f(n+1) |
| n+1 |
|  |  |  |

BACKWARD DIFFERNECE INTERPOLATION:

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

As a particular case, lets again consider the linear approximation to **f(x)**

|  |  |  |  |
| --- | --- | --- | --- |
|  | f1 - f0 |  | f0x1 - f1x0 |
| f(x)P1(x) = |  | x + |  |
|  | x1 - x0 |  | x1 - x0 |

|  |  |  |  |
| --- | --- | --- | --- |
| x1 - x |  | x - x0 |  |
|  | f0 + |  | f1 |
| x1 - x0 |  | x1 - x0 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | x - x1 |  |  | ( | x - x0 |  | ) |
| = | f1 - |  | f0 | + |  | - 1 | f1 |
|  |  | x0 - x1 |  |  | x - x0 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | x - x1 |  |
| = | f1 - |  | (f1- f0) |
|  |  | x1 - x0 |  |

= f1 + sf1

where**s = (x - x1) / (x1 - x0)** and**f1**is the backward difference of **f** at **x1**.

The same can be obtained from the difference operators as follows.

|  |  |  |  |
| --- | --- | --- | --- |
| f(x)  =  f (xn + | x - xn | h) |  |
|  |  |
| h |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | =  f(xn + sh)   =   Es f(xn),      s  =  ( x - xn )/ h |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
|  | **=  ( 1 - )s f( xn)** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| = fn + sfn + | s(s + 1) | 2fn | + . . . + | s(s + 1) . . . (s + n -1) | nfn + . . . |
| 2! | n! |

Neglecting the (n+1) difference as onwards we get

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| f(x)   Pn(x)  = fn + sfn + | s(s + 1) | 2fn | + . . . + | s(s + 1) . . . (s + n -1) | nfn |
| 2! | n! |

This polynomial is called the Newton - Gregory backward difference formula.

The error in this interpolation

|  |  |  |  |
| --- | --- | --- | --- |
| En(x)  = | s(s + 1) . . . (s + n) | hn+1f(n+1)() | ( x0 < () < xn ) |
| (n + 1)! |

**Example :**

Find f(0.15) using Newton backward difference table from the data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | f(x) | f | 2f | 3f | 4f |
| 0.1 | 0.09983 |  |  |  |  |
|  |  | 0.09884 |  |  |  |
| 0.2 | 0.19867 |  | -0.00199 |  |  |
|  |  | 0.09685 |  | -0.00156 |  |
| 0.3 | 0.29552 |  | -0.00355 |  | 0.00121 |
|  |  | 0.0939 |  | -0.00035 |  |
| 0.4 | 0.38942 |  | -0.0039 |  |  |
|  |  | 0.09 |  |  |  |
| 0.5 | 0.47943 |  |  |  |  |

s  =  ( x - xn ) / h  =  (0.15 - 0.5) / 0.1  =  -3.5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| f(0.15)  = fn + sfn + | s(s + 1) | 2fn + | s(s + 1)(s + 2) | 3fn + | s(s + 1)(s + 2)(s + 3) | 4fn |
| 2! | 3! |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| = .97943 + (-3.5)\*.09 + | (-3.5)(-3.5 + 1) | (-.0039)+ | (-3.5)(-3.5 + 1)(-3.5 + 2) | (-.00035) + |
| 2! | 3! |

|  |  |
| --- | --- |
| (-3.5)(-3.5 + 1)(-3.5 + 2)(-3.5 + 3) | (.00121) |
| 4! |

=  0.97943 - 0.315 - 0.01706 + 0.000765625 + 0.00033086

= 0.14847

Error in the approximation

|  |  |  |  |
| --- | --- | --- | --- |
| E5(x)  = | -3.5(-3.5 +1)(-3.5 + 2)(-3.5 + 3)(-3.5 + 4) | h5f5() | ( 0.1 <  < 0.5 ) |
| 5! |

= 2.734375 x 10-7 f5()

CENTRAL DIFFERENCE INTERPOLATION:

Consider a function f(x) tabulated for equally spaced points x0, x1, x2, . . ., xn with step length h. In many problems one may be interested to know the behaviour of f(x) in the neighbourhood of xr (x0 + rh). If we take the transformation X = (x - (x0 + rh)) / h,  the data points for X and f(X) can be written as

|  |  |  |
| --- | --- | --- |
| x | X | f(X) |
|  |  |  |
| x0 + (r - 2)h | -2 | f-2 |
| x0 + (r -1)h | -1 | f-1 |
| x0 + rh | 0 | f0 |
| x0 + (r + 1)h | 1 | f1 |
| x0 + (r + 2)h | 2 | f2 |

now the central difference table can be generated using the definition of central differences:

f(X) =  f(X + h/2) - f(X - h/2)

fi =  (E1/2 - E-1/2)fi  = ( fi +1/2 - fi -1/2)

2fi =  (E1/2 - E-1/2) ( fi +1/2 - fi -1/2)

=  f1 - f0 - f0 + f-1   =  f1 - 2f0 + f-1

Now the central difference table is

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Xi | fi | fi | 2fi | 3fi | 4fi |
|  |  |  |  |  |  |
| -2 | f-2 |  |  |  |  |
|  |  | f-3/2 ( = f-1 - f-2) |  |  |  |
| -1 | f-1 |  | 2f-1 ( = f-1/2-f-3/2) |  |  |
|  |  | f-1/2 ( = f0 - f-1) |  | 3f-1/2 ( = 2f0 - 2f-1) |  |
| 0 | f0 |  | 2f0 ( = f1/2-f-1/2) |  | 4f0 ( = 3f1/2 -3f-1/2) |
|  |  | f1/2 ( = f1 - f0) |  | 3f1/2 ( = 2f1 - 2f0) |  |
| 1 | f1 |  | 2f1 ( = f3/2-f1/2) |  |  |
|  |  | f3/2 ( = f2 - f1) |  |  |  |
| 2 | f2 |  |  |  |  |
|  |  |  |  |  |  |

SPLINE INTERPOLATION :

Interpolation:

spline interpolation is a form of [interpolation](https://en.wikipedia.org/wiki/Interpolation) where the interpolant is a special type of [piecewise](https://en.wikipedia.org/wiki/Piecewise) [polynomial](https://en.wikipedia.org/wiki/Polynomial) called a [spline](https://en.wikipedia.org/wiki/Spline_(mathematics)). That is, instead of fitting a single, high-degree polynomial to all of the values at once, spline interpolation fits low-degree polynomials to small subsets of the values, for example, fitting nine cubic polynomials between each of the pairs of ten points, instead of fitting a single degree-ten polynomial to all of them. Spline interpolation is often preferred over [polynomial interpolation](https://en.wikipedia.org/wiki/Polynomial_interpolation) because the [interpolation error](https://en.wikipedia.org/wiki/Interpolation_error) can be made small even when using low-degree polynomials for the spline. Spline interpolation also avoids the problem of [Runge's phenomenon](https://en.wikipedia.org/wiki/Runge%27s_phenomenon), in which oscillation can occur between points when interpolating using high-degree polynomials.

Linear Interpolation:

Its simplest formula is given below:

y=y1+(x−x1)(y2−y1)x2−x1

This formula is using coordinates of two given values to find the best fit curve as a straight line. Then this will give any required value of y at a known value of x.

In this formula, we are having terms as:

* x1 and y1 are the first coordinates
* x2 and y2 are the second coordinates
* x is the point to perform the interpolation
* y is the interpolated value.

Q.Find the value of y at x = 4 given some set of values (2, 4), (6, 7).

Solution: Given the known values are,

x=4 x1=2  x2=6 y1=4 ; y2=7

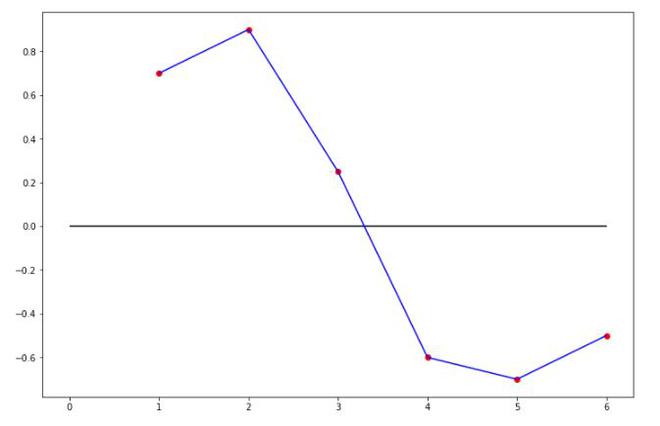
The interpolation formula is,

y=y1+(x−x1)(y2−y1)x2−x1

i.e. y=4+(4−2)×(7−4)(6−2)

y = 112

Linear Interpolation is a way of curve fitting the points by using linear polynomial such as the equation of the line. This is just similar to joining points by drawing a line b/w the two points in the dataset.



Cubic interpolation:

Cubic spline interpolation is a mathematical method commonly used to construct new points within the boundaries of a set of known points. These new points are function values of an interpolation function (referred to as *spline*), which itself consists of multiple cubic piecewise polynomials. This article explains how the computation works mathematically. Linear Interpolation

Cubic spline interpolation is the process of constructing a spline f:[x1,xn+1]→R which consists of n polynomials of degree three, referred to as f1 to fn. A spline is a function defined by piecewise polynomials. Opposed to regression, the interpolation function traverses all n+1 pre-defined points of a data set D. The resulting function has the following structure:

f(x)={a1x3+b1x2+c1x+d1if x∈[x1,x2]a2x3+b2x2+c2x+d2if x∈(x2,x3]…anx3+bnx2+cnx+dnif x∈(xn,xn+1].

Note that all polynomials are just valid within an interval; they compose the interpolation function. While extrapolation predicts a development outside the range of the data, interpolation works just within the data boundaries [x1,xn+1].

With properly chosen coefficients ai, bi, ci, and di for the polynomials, the resulting function traverses the points smoothly. For determining the coefficients, several equations are formulated which all together compose a uniquely solvable system of equations.

Examples  
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)36hMi-1+(x-xi-1)36hMi+(xi-x)h(yi-1-h26Mi-1)+(x-xi-1)h(yi-h26Mi)→(1)  
  
We have, Mi-1+4Mi+Mi+1=6h2(yi-1-2yi+yi+1)→(2)  
  
Here h=1,n=3  
  
M0=0,M3=0  
  
Substitute i=1 in equation (2)  
  
M0+4M1+M2=6h2(y0-2y1+y2)  
  
⇒0+4M1+M2=61⋅(1-2⋅5+11)  
  
⇒4M1+M2=12  
  
Substitute i=2 in equation (2)  
  
M1+4M2+M3=6h2(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)36hM0+(x-x0)36hM1+(x1-x)h(y0-h26M0)+(x-x0)h(y1-h26M1)  
  
f1(x)=(2-x)36⋅0+(x-1)36⋅6.8+(2-x)1(1-16⋅0)+(x-1)1(5-16⋅6.8)  
  
f1(x)=1.1333x3-3.4x2+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)36hM1+(x-x1)36hM2+(x2-x)h(y1-h26M1)+(x-x1)h(y2-h26M2)  
  
f2(x)=(3-x)36⋅6.8+(x-2)36⋅-15.2+(3-x)1(5-16⋅6.8)+(x-2)1(11-16⋅-15.2)  
  
f2(x)=-3.6667x3+25.4x2-51.3333x+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)36hM2+(x-x2)36hM3+(x3-x)h(y2-h26M2)+(x-x2)h(y3-h26M3)  
  
f3(x)=(4-x)36⋅-15.2+(x-3)36⋅0+(4-x)1(11-16⋅-15.2)+(x-3)1(8-16⋅0)  
  
f3(x)=2.5333x3-30.4x2+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)=3.4x2-6.8x+6.2667  
  
Now substitute x=2 in f′1(x), we get  
  
f′1(2)=6.2667

Quadratic Spline Interpolation:

A Quadratic Spline is the creation of a set of polynomial functions that are quadratic, or, easier to understand, follow the format f(x)=ax²+bx+c, where a, b and c are the values obtained while doing the Splines to create the desired functions.

To create the splines, it is necessary for the user to provide 2 or more points, as with 1 point it is impossible to calculate because the splines, as mentioned before, return a set of functions that contain n-1 functions. Given 2 points, the set contains only 1 function, as with 1, it would contain 0.

As Example, To create splines the User would Input the following set of points:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | 0 | 1 | 3 | 4 |
| Y | -2.5 | 1 | 3.6 | 2.3 |

To create a Linear Spline we have 1 Possible methods to find the desired equation Values:

A Matrix, which is Built using the values. This Matrix has the size of (n-1)\*3.

With the Matrix Method we have:

The first (n-1)\*2 equations correspond to the given values equations following the format F(xi) = ai(x)²+bi(x)+ci.

For this example we would have:

f(x1): a1(0)²+b1(0)+c1 = -2.5

f(x2): a1(1)²+b1(1)+c1 = 1

f(x3): a2(1)²+b2(1)+c2 = 1

f(x4): a2(3)²+b2(3)+c2 = 3.6

f(x5): a3(3)²+b3(3)+c3 = 3.6

f(x6): a3(4)²+b3(4)+c3 = 2.3

Here we assume each set of 2 equations are the same evaluated in the different points.

The next set of equations (in this case, the next 2), are formed by following the principle of continuity, in which a the derivative of a function in a point has to be the same as the derivative of the other function on the same point. For this we take the functions that pass through the same points.

f(x7): 2a1(1)+b1 = 2a2(1) +b2 -> f(x5): 2a1(1)+b1-2a2(1)-b2 = 0

f(x8): 2a2(3)+b2 = 2a3(3)+b3  -> f(x6): 2a2(3)+b2-2a3(3)-b3 = 0

In this case, we're missing 1 equation to have the same number of equations as there is of unknown values, therefore, we assume that a1 = 0

f(x9): a1 = 0

Having the desired number of equations, we create a matrix using the multipliers of ai and bi as follows:

[0,0,1,0, 0,0,0,0,0]        First equations Values

[1,1,1,0,0,0,0,0,0]     Second equation Values

[0,0,0,1,1,1,0,0,0]        Third equation Values

[0,0,0,9,3,1,0,0,0]        Fourth equation Values

[0,0,0,0,0,0,9,3,1]        Fifth equation Values

[0,0,0,0,0,0,16,4,1]        Sixth equation Values

[2,1,0,-2,-1,0,0,0,0]        Seventh equation Values

[0,0,0,6,1,0,-6,-1,0]        Eighth equation Values

[1,0,0,0,0,0,0,0,0]        Ninth equation Values

Next, we have to create the b Vector to be able to solve a system of linear equations, we do that with the Y values of the 6 equations. we have:

[-2.5]        Results of equation f(x1)

[     1]        Results of equation f(x2)

[     1]        Results of equation f(x3)

[  3.6]        Results of equation f(x4)

[  3.6]        Results of equation f(x5)

[  2.3]        Results of equation f(x6)

[     0]        Results of equation f(x7)

[     0]        Results of equation f(x8)

[     0]        Results of equation f(x9)

Now we have a System of Ax = b, where we need to find the x. For this, any method on the Systems of Linear Equations can be used to solve the system.

INTEGRATION:

TRAPEZOIDAL RULE:

We apply the trapezoidal rule formula to solve a definite integral by calculating the area under a curve by dividing the total area into little [trapezoids](https://www.cuemath.com/geometry/trapezoid/) rather than [rectangles](https://www.cuemath.com/geometry/rectangle/). This rule is used for approximating the definite integrals where it uses the linear approximations of the functions. The trapezoidal rule takes the average of the left and the right sum.

Let y = f(x) be [continuous](https://www.cuemath.com/calculus/continuous-function/) on [a, b]. We divide the interval [a, b] into n equal subintervals, each of width, h = (b - a)/n,

such that a = x0 < x1 < x2 < ⋯ < xn = b

Area = (h/2) [y0 + 2 (y1 + y2 + y3 + ..... + yn-1) + yn]

**Example 1 :**

Approximate the area under the curve y = f(x) between x =0 and x=8 using Trapezoidal Rule with n = 4 subintervals. A function f(x) is given in the table of values.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | 0 | 2 | 4 | 6 | 8 |
| f(x) | 3 | 7 | 11 | 9 | 3 |

**Solution:**

The Trapezoidal Rule formula for n= 4 subintervals is given as:

T4 =(Δx/2)[f(x0)+ 2f(x1)+ 2f(x2)+2f(x3) + f(x4)]

Here the subinterval width Δx = 2.

Now, substitute the values from the table, to find the approximate value of the area under the curve.

A≈ T4 =(2/2)[3+ 2(7)+ 2(11)+2(9) + 3]

A≈ T4 = 3 + 14 + 22+ 18+3 = 60

Therefore, the approximate value of area under the curve using Trapezoidal Rule is 60.

**Example 2:**

Approximate the area under the curve y = f(x) between x =-4 and x= 2 using Trapezoidal Rule with n = 6 subintervals. A function f(x) is given in the table of values.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | -4 | -3 | -2 | -1 | 0 | 1 | 2 |
| f(x) | 0 | 4 | 5 | 3 | 10 | 11 | 2 |

**Solution:**

The Trapezoidal Rule formula for n= 6 subintervals is given as:

T6 =(Δx/2)[f(x0)+ 2f(x1)+ 2f(x2)+2f(x3) + 2f(x4)+2f(x5)+ f(x6)]

Here the subinterval width Δx = 1.

Now, substitute the values from the table, to find the approximate value of the area under the curve.

A≈ T6 =(1/2)[0+ 2(4)+ 2(5)+2(3) + 2(10)+2(11) +2]

A≈ T6 =(½) [ 8 + 10 + 6+ 20 +22 +2 ] = 68/2 = 34

Therefore, the approximate value of area under the curve using Trapezoidal Rule is 34.

SIMPSOMS 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, 3/8 rule and examples.

Simpson’s 1/3rd rule is an extension of the trapezoidal rule in which the integrand is approximated by a second-order polynomial. Simpson rule can be derived from the various way using Newton’s divided difference polynomial,  Lagrange polynomial and the method of coefficients. Simpson’s 1/3 rule is defined by:

|  |
| --- |
| ∫ab f(x) dx = h/3 [(y0 + yn) + 4(y1 + y3 + y5 + …. + yn-1) + 2(y2 + y4 + y6 + ….. + yn-2)] |

This rule is known as Simpson’s One-third rule.

Simpson’s ⅓ Rule for Integration

We can get a quick approximation for definite integrals when we divide a small interval [a, b] into two parts. Therefore, after dividing the interval, we get;

x0= a, x1= a + b, x2 = b

Hence, we can write the approximation as;

∫ab f(x) dx ≈ S2 = h/3[f(x0) + 4f(x1) + f(x2)]

S2 = h/3 [f(a) + 4 f((a+b)/2) + f(b)]

Where h = (b – a)/2

This is the Simpson’s ⅓ rule for integration.

**Example**:

 Evaluate ∫01exdx, by Simpson’s ⅓ rule.

Solution:

Let us divide the range [0, 1] into six equal parts by taking h = 1/6.

If x0 = 0 then y0 = e0 = 1.

If x1 = x0 + h = ⅙, then y1 = e1/6 = 1.1813

If x2 = x0 + 2h = 2/6 = 1/3 then, y2 = e1/3 = 1.3956

If x3 = x0 + 3h = 3/6 = ½ then y3 = e1/2= 1.6487

If x4 = x0 + 4h = 4/6 ⅔ then y4 = e2/3 = 1.9477

If x5 = x0 + 5h = ⅚ then y5 = e5/6 = 2.3009

If x6 = x0 + 6h = 6/6 = 1 then y6 = e1 = 2.7182

We know by Simpson’s ⅓ rule;

∫ab f(x) dx = h/3 [(y0 + yn) + 4(y1 + y3 + y5 + …. + yn-1) + 2(y2 + y4 + y6 + ….. + yn-2)]

Therefore,

∫01exdx = (1/18) [(1 + 2.7182) + 4(1.1813 + 1.6487 + 2.3009) + 2(1.39561 + 1.9477)]

=  (1/18)[3.7182 + 20.5236 + 6.68662]

= 1.7182 (approx.)

Simpson’s 3/8 Rule

Another method of numerical integration is called “Simpson’s 3/8 rule”. It is completely based on the cubic interpolation rather than the quadratic interpolation. Simpson’s 3/8 or three-eight rule is given by:

|  |
| --- |
| ∫ab f(x) dx = 3h/8 [(y0 + yn) + 3(y1 + y2 + y4 + y5 + …. + yn-1) + 2(y3 + y6 + y9 + ….. + yn-3)] |

This rule is more accurate than the standard method, as it uses one more functional value. For 3/8 rule, the composite Simpson’s 3/8 rule also exists which is similar to the generalized form. The 3/8 rule is known as Simpson’s second rule of integration.

**Examples:  
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 | 4.953 | 6.0436 | 7.3891 | 9.025 |

Using Simpson's 38 Rule  
  
∫ydx=3h8[(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

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

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

Solution:  
The value of table for x and y

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

Using Simpson's 38 Rule  
  
∫ydx=3h8[(y0+y4)+2(y3)+3(y1+y2)]  
  
∫ydx=3×0.18[(1+0.8604)+2×(0.9776)+3×(0.9975+0.99)]  
  
∫ydx=3×0.18[(1+0.8604)+2×(0.9776)+3×(1.9875)]  
  
∫ydx=0.36668  
  
Solution by Simpson's 38 Rule is 0.36668

**THANK YOU**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_