1 LINEAR INTERPOLATION

The logic of Newton Divided Difference method is to take some points and generate equation from these points.Then, we put the searched point to this function and calculate the result. The number of points are depends of which kind of equation we want to generate.For instance, if we want to generate linear equation then we should take two point such that our searched point are between them. For cubic equation we need to take 4 points. We obtain more accurate results as we increase the order of the equation.

a. Our x values and corresponding y values are:

x:	0	1	2	4	6
y:	1	9	23	93	259

Let draw a table from this values:

x	b_0				
0	1	b_1			
		8	b_2		
1	9		3	b_3	
		14		1	b_4
2	23		7		0
		35		1	
4	93		12		
		83			
6	259				

In our table we calculate our values with this way:

For instance we calculate 8 from

$$\frac{f(1)-f(0)}{1-0}=\frac{9-1}{1-0}=8$$

Same way we calculate 14 with this equation:

$$\frac{f(2)-f(1)}{2-1}=\frac{23-9}{2-1}=14$$

Basically, we use previous values to complete our table.For instance we calculate 7 with this way :

$$\frac{35-14}{4-1}=7$$

With the same logic we calculate other values and build the table. After we build the table we can easily calculate coefficients of equation.Coefficients are the first row of our table.

Hence, $b_0 = 1$, $b_1 = 8$, $b_2 = 3$, $b_3 = 1$ and $b_4 = 0$.

Now, we have our coefficients and x values. So, we can write target equation. In order to write this equation general formula is:

$$\text{f(x)} = \text{f[x]} + \text{f}[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + ... + f[x_0, x_1, ...x_k](x - x_0)(x - x_1)...(x - x_{k-1})$$

Hence our equation is equal to :

$$\text{f(x)} = 1 + 8\text{(x-0)} + 3\text{(x-0)(x-1)} + 1\text{(x-0)(x-1)(x-2)}$$

$$\text{f(x)} = x^3 + 7\text{x} + 1$$

b.Comparison

Value at points 1.2, 10.5 and 17.3 for data1:

My algorithm values:

$$f(1.2) = 5.216, \quad f(10.5) = 3.625, \quad f(17.3) = 2.355$$

interp1d values:

$$f(1.2) = 5.226, \quad f(10.5) = 3.678, \quad f(17.3) = 2.375$$

Value at points 10.3, 20.2 and 29.7 for data2:

My algorithm values:

$$f(10.3) = 8.366, \quad f(20.2) = 30.2, \quad f(29.7) = 4.888$$

interp1d values:

$$f(10.3) = 13.066, \quad f(20.2) = 30.2, \quad f(29.7) = 4.857$$

Value at points -0.82, 0.40 and 0.91 for data3:

My algorithm values:

$$f(-0.82) = 0.056, \quad f(0.40) = 0.199, \quad f(0.91) = 0.0460$$

interp1d values:

$$f(-0.82) = 0.056, \quad f(0.40) = 0.199, \quad f(0.91) = 0.0460$$

Value at points -0.51, 0.72, 0.97 for data4:

My algorithm values:

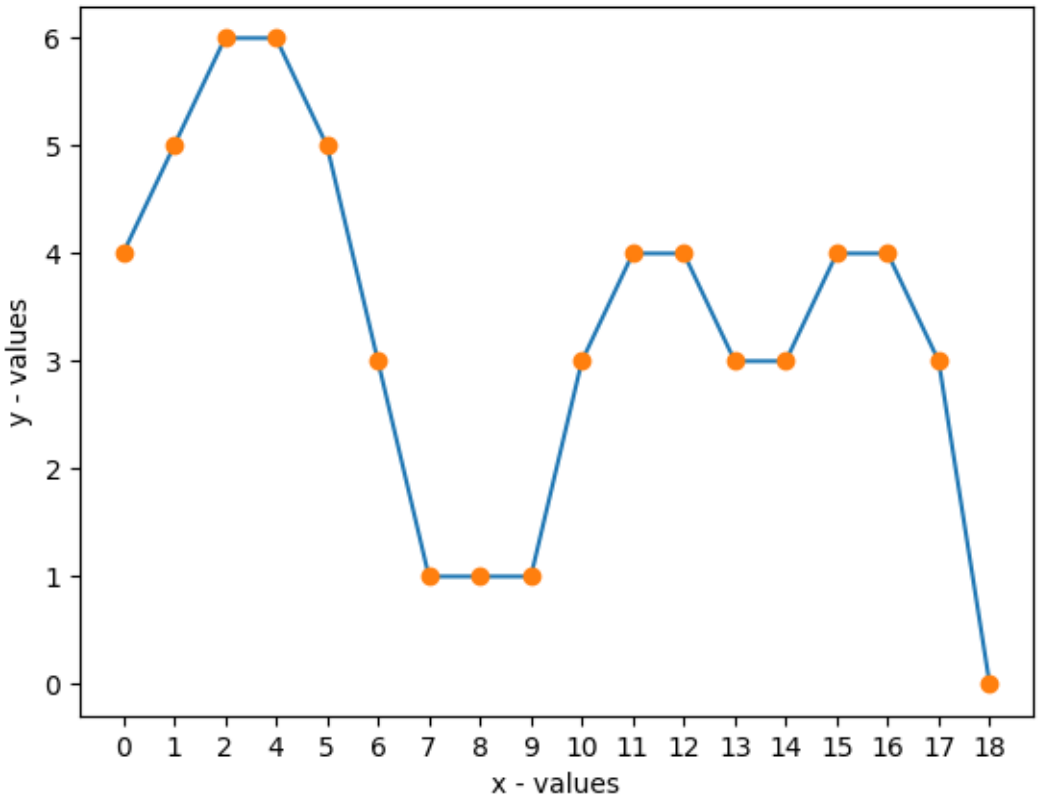
$$f(-0.51) = 0.133, \quad f(0.72) = 0.071, \quad f(0.97) = 0.040$$

interp1d values:

$$f(-0.51) = 0.133, \quad f(0.72) = 0.071, \quad f(0.97) = 0.040$$

As we can see from results, my algorithm and interp1d function calculate close results. Some of the results are exactly the same. On the other hand, some of them are a little different and I think that's because of logic of interp1d little bit different from Newton Divided Difference method.

Interpolation function graph for xData and yData values given in data1.txt



(Interpolation function graph)

Pseudo Codes

myInterpolationFunc(xData, yData, interpolationPoint):

```
1. n = np.shape(yData)[0]
2. create a square matrix coeffVector to hold our coefficent table
3. coeffVector[::,0] = yData
4. for k in range length of xData
5.     if interpolationPoint already calculated in data set return it
6. for j in range of 1 and n
7.     for i in range n- j
8.         if (xData[i+j] - xData[i]) == 0:     // if denominator value equal 0
9.             coeffVector[i][j] = 0
10.            continue
11.        coeffVector[i][j] = (coeffVector[i+1][j-1] - coeffVector[i][j-1]) / (xData[i+j] - xData[i])
12. coeffVector = coeffVector[0]
13. return result(coeffVector, interpolationPoint, xData)
```

result(coeffVector, point, xData):
// this function calculate and return result for interpolation point
1. result = coeffVector[0] + coeffVector[1]*(point - xData[0])
   + coeffVector[2]*(point - xData[0])*(point-xData[1])
   + coeffVector[3]*(point - xData[0])*(point-xData[1])*(point-xData[2])
2. **return** result

readFile(fileName, interpolationPoint):
// this function return xData and yData for given txt file and interpolationPoint
1. read x and y values from txt and store them in xTarget and yTarget respectively
2. create empty dict
3. **for** j **in** range of len(xTarget)
4. **if** xTarget[j] **in** dict: // if there is a duplicate value for x, take the first one
5. **continue**
6. **else**:
7. dict[xTarget[j]] = yTarget[j]
8. create smallXValues list for storing x values that's smaller than interpolationPoint
9. create bigXValues list for storing x values that's bigger than interpolationPoint
10. **for** j **in** range of len(xTarget)
11. **if** xTarget[i] <= interpolationPoint :
12. smallXValues.append(xTarget[i])
13. **else** :
14. bigXValues.append(xTarget[i])
15. convert smallXValues and bigXValues to numpay array
16. create xData array with length of 4 //since we will use 4 point to obtain our equation
17. **if** smallXValues has only one member:
18. add this point to xData as first point and for add other 3 points from bigXValues respectively
19. **elif** bigXValues has only one member:
20. add first 3 point from smallXValues to xData and add other point from bigXValues respectively
21. **else** bigXValues has only one member:
22. add first 2 point from smallXValues to xData and add other two point from bigXValues respectively
23. create yData array with length of 4
24. **for** i **in** range len(xData):
25. yData[i] = dict[xData[i]]
26. **return** xData,yData

Detailed Explanation of a Algorithm

At the beginning, my algorithm calculates xData and yData for a given data file and interpolation point with readFile() method.readFile method firstly,reads x and y values from txt files and separately store them in two lists. From these lists, I added x and corresponding y values to the dictionary as a key-value pair. If some of the x values are duplicated algorithm will take the first one. My algorithm method is cubic interpolation method. Thus,readFile() method returns "four point"such that interpolation point will be between them.Here, there is three possibilities.First possibility,it will return 3 smaller point and 1 bigger point.Second possibility is it will return 3 bigger point and 1 smaller point.The last possibility is it will return 2 bigger point and 2 smaller point. After readFile() method, I send xData, yData and interpolation point to myInterpolationFunc() method.In this method I create my divided difference table with the way we learn in our lesson.After, that as we know, my coefficients are in the first line of this table.Hence, I take only first row of my table. After, I found my coefficients, I send coefficients,xData and interpolation point to the result() method. In result() method, I generate my equation from coefficients and xData. Then, I put interpolation point and calculate the result.

2 NUMERICAL INTEGRATION

a. In calculus, an integral is a mathematical object that can be interpreted as an area or a generalization of area. One way to calculate approximate result of integral is the Trapezoidal Rule. Trapezoidal Rule divides the whole area into smaller trapezoids to calculate the area under the curves. This integration determines the area by approximating the region under the graph of a function as a trapezoid.In our question since n given as a 4, we will divide our area to 4 part and we will calculate their values and we will sum these values to find result.

Our integral is:

$$\int_1^2 \frac{e^{-x}+x}{x}dx$$

Now, our a = 1, b = 2 and n= 4. From this values we can calculate h value which is $\frac{a-b}{n}$. Hence, in our case h = $\frac{2-1}{4} = 0.25$

So, we can write our integral like that:

$$\int_1^{1.25} \frac{e^{-x}+x}{x}dx + \int_{1.25}^{1.50} \frac{e^{-x}+x}{x}dx + \int_{1.50}^{1.75} \frac{e^{-x}+x}{x}dx + \int_{1.75}^{2} \frac{e^{-x}+x}{x}dx$$

The solution using 4-segment Trapezoidal rule is

$$I = \frac{a-b}{2n}\left[f(a) + 2\left\{\sum_{i=1}^{n-1} f(a+ih)\right\} + f(b)\right]$$

Let apply this rule for our values and solve the equation

$$I = \frac{2-1}{8}\left[f(1) + 2\left\{\sum_{i=1}^{3} f(1+i*0.25)\right\} + f(2)\right]$$

We can extend our equation:

$$I = \frac{1}{8}\big[f(1) + 2\big\{f(1.25) + f(1.50) + f(1.75)\big\} + f(2)\big]$$

Let calculate f(1), f(1.25), f(1.50), f(1.75) and f(2):

$$f(1) = \frac{e^{-1}+1}{1} = 1.3678$$

$$f(1.25) = \frac{e^{-1.25}+1.25}{1.25} = 1.2292$$

$$f(1.50) = \frac{e^{-1.50}+1.50}{1.50} = 1.1487$$

$$f(1.75) = \frac{e^{-1.75}+1.75}{1.75} = 1.0992$$

$$f(2) = \frac{e^{-2}+2}{2} = 1.0676$$

So our result is:

$$I = \frac{1}{8}\big[1.3678 + 2\big\{1.2292 + 1.1487 + 1.0992\big\} + 1.0676\big]$$

$$I = \frac{1}{8}\big[9.3896\big] = 1.1737$$

b. While Trapezoidal Rule was based on approximating the integrand by a first order polynomial,Simpson's 1/3rd rule based on approximating the integrand by a second order polynomial.Hence, Simpson's 1/3rd rule more accurate than Trapezoidal Rule.

Our integral is:

$$\int_1^2 \frac{e^{-x} + x}{x} dx$$

Now, our a = 1, b = 2 and n= 2. From this values we can calculate h value which is $\frac{a-b}{n}$. Hence,

$$h = \frac{2-1}{4} = 0.5$$

So, we can write our integral like that:

$$\int_1^{1.5} \frac{e^{-x} + x}{x} dx + \int_{1.5}^2 \frac{e^{-x} + x}{x} dx$$

Simpson 1/3rd rule:

$$\int_a^b f_2(x)dx = \frac{b-a}{6}\left[f(a) + 4f\left\{f\left(\frac{a+b}{2}\right)\right\} + f(b)\right]$$

Let apply Simpson 1/3rd rule to the this integrals

$$\int_1^{1.5} \frac{e^{-x} + x}{x} dx = \frac{1.5-1}{6}\left[f(1) + 4f\left\{f\left(\frac{1+1.5}{2}\right)\right\} + f(1.5)\right]$$

Let calculate f(1), f(1.25) and f(2):

$$f(1) = \frac{e^{-1} + 1}{1} = 1.3678$$

$$f(1.25) = \frac{e^{-1.25} + 1.25}{1.25} = 1.2292$$

$$f(1.50) = \frac{e^{-1.50} + 1.50}{1.50} = 1.1487$$

So, our result for the first integral is

$$\int_1^{1.5} \frac{e^{-x} + x}{x} dx = \frac{0.5}{6}\left[1.3678 + 4 * (1.2292) + 1.1487\right]$$

$$= \frac{1}{12}\big[7,4333\big] = 0.6194$$

We calculated our first integral, let calculate the other one

$$\int_{1.5}^{2} \frac{e^{-x}+x}{x}dx = \frac{2-1.5}{6}\left[f(1.5) + 4f\left\{f\left(\frac{2+1.5}{2}\right)\right\} + f(2)\right]$$

Let calculate f(1.5), f(1.75) and f(2):

$$f(1.50) = \frac{e^{-1.50}+1.50}{1.50} = 1.1487$$

$$f(1.75) = \frac{e^{-1.75}+1.75}{1.75} = 1.0992$$

$$f(2) = \frac{e^{-2}+2}{2} = 1.0676$$

So, our result for the second integral is

$$\int_{1.5}^{2} \frac{e^{-x}+x}{x}dx = \frac{0.5}{6}\left[1.1487 + 4 * (1.0992) + 1.0676\right]$$

$$= \frac{1}{12}\left[6,6131\right] = 0.5510$$

Finally, we should sum the result of this two integrals

$$result = 0.5510 + 0.6194 = 1,1704$$

So, our result is 1,1704

Result
As we said before, the Trapezoidal Rule used a first order polynomial to approximate the integrand. On the other hand, Simpson's 1/3rd Rule used a second order polynomial to approximate the integrand. Since, Simpson's 1/3rd Rule is second order polynomial it gives more accurate results. For our question real result of integral is 1.1705. We calculated value of integral with Simpson's 1/3rd Rule as 1.1704 while Trapezoidal Rule's result was 1.1737. As a result, Simpson's 1/3rd Rule is more accurate than Trapezoidal Rule.

3 ROOT FINDING

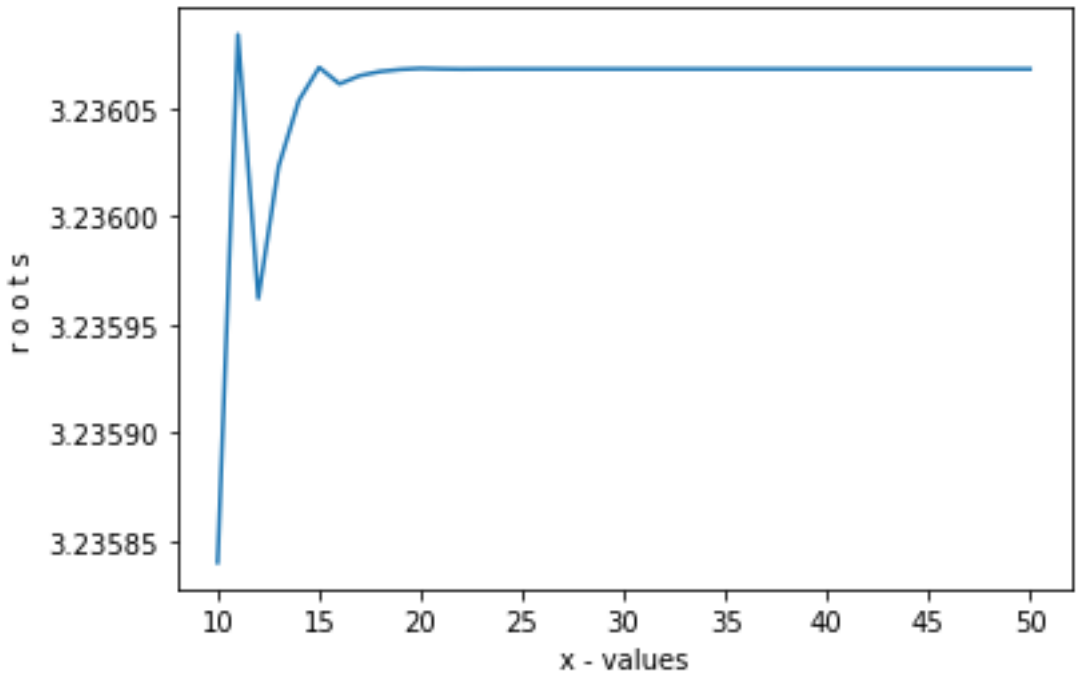
Pseudo Codes

newtonRaphson(f,x_0,numIter):
1. calculate Df function which is differentation of function f
2. **for** n **in** range numIter+1:
3. calculate f(x_0)
4. calculate Df(x_0)
5. **if** Df(x_0) equal 0:
6. **return** None
7. x_0 = x_0 - (f(x_0)/D(x_0))

bisection(f,a,b,iterNum):

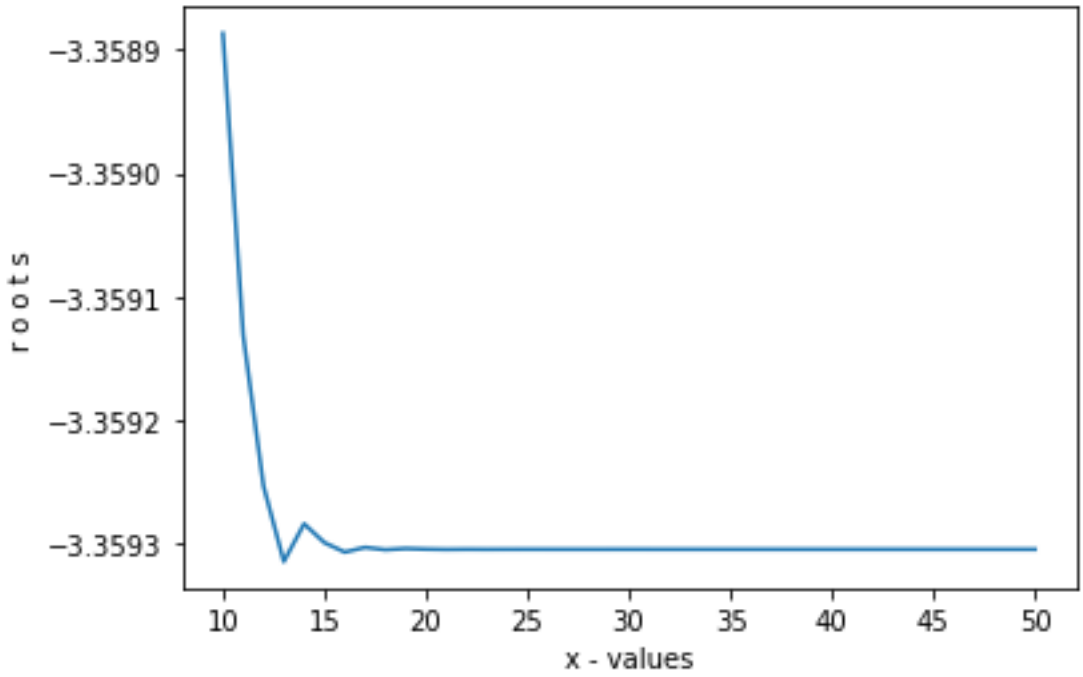
```
1. if f(a) * f(b) < 0:
2.     return None
3. for i in range iterNum+1
4.     midPoint = (a + b ) / 2.0
5.     print(midPoint)
6.     if f(a) * f(midPoint) < 0:
7.         b = midPoint
8.     elif f(b) * f(midPoint) < 0:
9.         a = midPoint
10.    elif midPoint == 0:
11.        return midPoint
12.    else:
13.        return None
14. return (a+b) / 2
```

Bisection Graphs



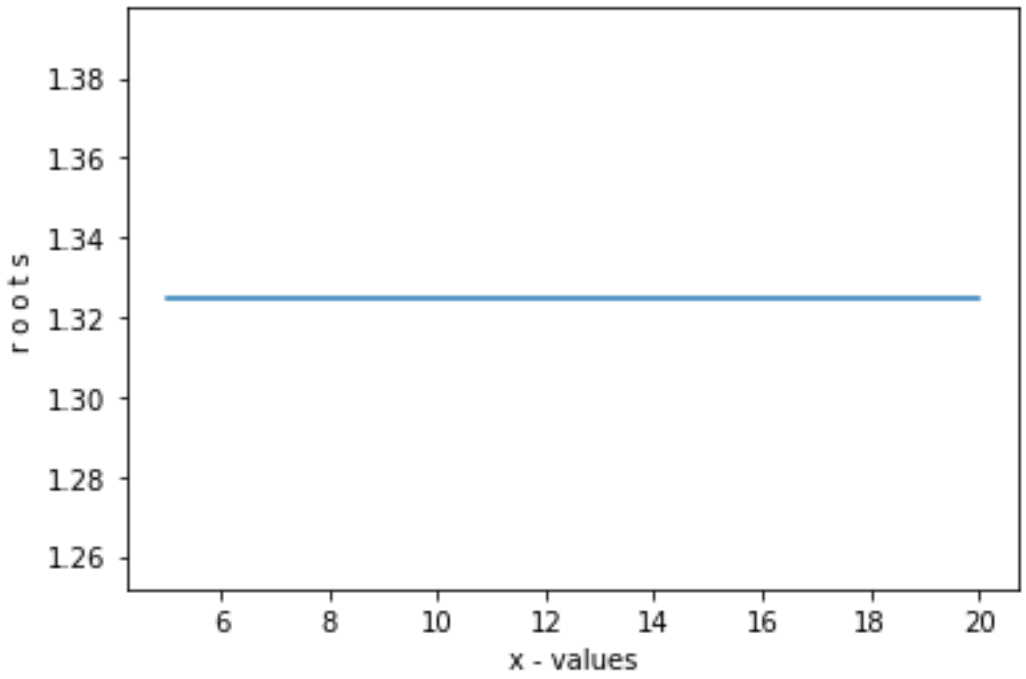
(First function graph)

Note: For the second function, I define interval between -4 and -3.



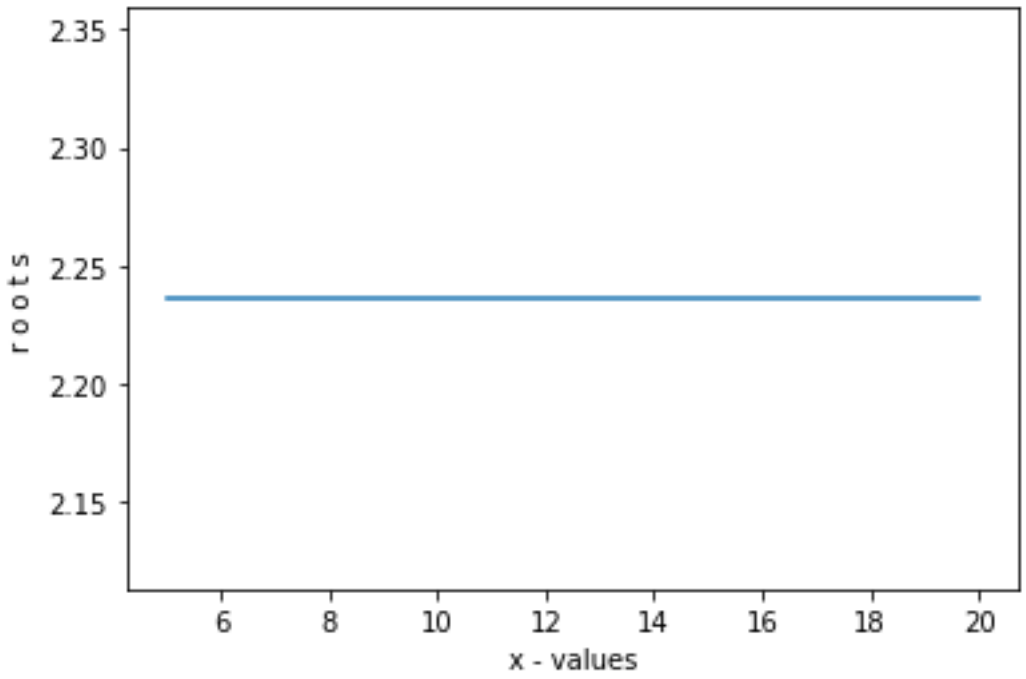
(Second function graph)

Newton-Raphson Graphs



(First function graph)

Note:I define my equation as $x^2 + \sqrt{5}x = 0$ and initial $x_0 = 2$. As we can see from graph, our function calculate root of this equation as 2.236 which is equal to $\sqrt{5}$.



(Second function graph)

Result
Bisection method is used for solving non linear equations that cannot be solved analytically. For a given intervals function should be continuous and function should return opposite sign result for these intervals. Because when the function is continuous and it returns positive and negative result for some values, we know that certainly at some point function has a root.The logic is simple, until it closes to the correct answer it narrowing the gap between negative and positive interval. In each steps it change one of the intervals such that the result of new intervals will return opposite sign from function.On the other, there is a another root finding which method is called as Newton-Raphson method. The problem of Bisection method is slowness.Newton-Raphson is based on finding another point close to the root by using the tangent at the selected point.Thus it's faster. Also, there is a another difference between them: while Bisection method is a closed interval method, Newton-Raphson method is a open interval method.Newton-Raphson method take function and initial point and result for this point. Then it calculate tangent of this point, with this way it get new selected point. Then, it repeats same operation until it closes enough to the correct answer. As a result, when we compare Bisection method and Newton-Raphson method, Newton-Raphson method is more effective and fast.

4 SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

a.Our given equation and initial conditions are:

$$\frac{dy}{dx} + x = y, \quad y(0) = 0, \quad h = 0.1$$

First , let write a Euler formula:

$$y(x_i + h) = y(x_i) + y'(x_i) * h$$

Second, let define x_i and h values:

$$\frac{dy}{dx} = y' = y - x$$

$$y(0) = 0 \quad \Rightarrow \quad x_i = 0 \quad , \quad y_i = 0 \quad , \quad h = 0.1$$

Our aim is to find value of y(0.1) which is equal to y(0+0.1). So, basically we need to calculate y(0+0.1).

We can calculate y(0+0.1) with Euler formula:

$$y(0+0.1) = y(0) + y'(0).(0.1)$$

We already know y(0) which is equal to 0, we just need to find $y'(0)$.We can calculate $y'(0)$ with this way:

Since, we know y' = y- x we can calculate $y'(0)$ easily,

$$y' = y - x \quad \Rightarrow \quad y'(0) = y(0) - 0 = 0$$

If we put this value to the our equation we will find y(0+0.1)

$$y(0+0.1) = y(0) + y'(0).(0.1) = 0 + 0 * (0.1) = 0$$

As a result, y(0.1) is equal to 0.

b.Our given equation and initial conditions are:

$$\frac{dy}{dx} + x = y, \quad y(0) = 0, \quad h = 0.1$$

For this question we will use 4th order Runge-Kutta method , let write a Runge-Kutta method:

$$y(x_i + h) = y(x_i) + \frac{1}{6}\big[k_1 + 2 * k_2 + 2 * k_3 + k_4\big]$$

We calculate k values in this formula with this way:

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{k_1 * h}{2}\right)$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{k_2 * h}{2}\right)$$

$$k_4 = f(x_i + h, y_i + k_3 * h)$$

Before applying Runge-Kutta method, we should define x_i,y_i and h values:

$$\frac{dy}{dx} = y' = y - x$$

$$y(0) = 0 \quad \Rightarrow \quad x_i = 0 \quad , \quad y_i = 0 \quad , \quad h = 0.1$$

Let find our k values:

$$k_1 = f(x_i, y_i) = f(0,0), \quad since \quad y' = f(x,y) = y - x$$
$$k_1 = f(0,0) = y(0) - 0 = 0$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{k_1 * h}{2}\right) = f\left(0 + \frac{0.1}{2}, 0 + \frac{0 * 0.1}{2}\right) = f(0.05, 0)$$

$$k_2 = f(0.05, 0) = 0 - 0.05 = -0.05$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{k_2 * h}{2}\right) = f\left(0 + \frac{0.1}{2}, 0 + \frac{-0.05 * 0.1}{2}\right) = f(0.05, -0.0525)$$

$$k_3 = f(0.05, -0.0525) = -0.0525 - 0.05 = -0.0525$$

$$k_4 = f(x_i + h, y_i + k_3 * h) = f(0 + 0.1, 0 + (-0.0525) * (0.1)) = f(0.1, -0.00525)$$

$$k_4 = f(0.1, -0.00525) = -0.00525 - 0.1 = -0.10525$$

Now, let put this values to the formula and calculate y(0+0.1) value.

$$y(0 + 0.1) = y(0) + \frac{1}{6}\big(0 + 2 * (-0.05) + 2 * (-0.0525) + (-0.10525)\big) * (0.1)$$

$$y(0 + 0.1) = -0.005170$$

As a result, we found y(0.1) value as a -0.005170 using 4th order Runge-Kutta method.

c.Pseudo Codes

myEuler(x0, y, x, h):
1. **while** x_0 < x:
2. y = y + h * dydx(x0,y) // dydx is function a which is equal to y'
3. x0 = x0 +h
4. **print** x_0

myRungeKutta(x_0**, **y_0**, x, h)**:
1. n = (int)((x-x_0)/h) // n is number of steps
2. **for** i **in**range of 1 and n+1
3. k_1 = dydx(x_0, y_0) quad // dydx is function a which is equal to y'
4. k_2 = dydx(x_0 + h*0.5, y_0 + (k_1*h)*0.5)
5. k_3 = dydx(x_0 + h*0.5, y_0 + (k_2*h)*0.5)
6. k_4 = dydx(x_0 + h, y_0 + (k_3*h))
7. y_0 = y_0 + (1.0 / 6.0)*(k_1 + 2 * k_2 + 2 * k_3 + k_4)*h //update y_0
8. x_0 = x_0 + h
9. x_0 = x_0 + h //update x_0
10.**return** y_0

Result

Euler's method is first order method. It is a basic method that predicts the next point based on the rate of change at the current point. While Euler method just uses the first term of the Taylor series, Runge-Kutta methods are using higher order terms of the Taylor series. Hence, Runge-Kutta method are more preferable than Euler's method .