International Journal of Future Generation Communication and Networking
Vol. 13, No. 4, (2020), pp. 3887 -3896

Empirical Study on Various Computation Methods for Computer Applications

Dr. Darpan Anand, Dr. Surendra Singh, Saurabh, Narendra Kumar[2]

darpan.e8545@cumail.in
1 Associate Professor – Department of Computer Science and Engineering, Chandigarh University, Gharuan, Mohali, Punjab 140413,India
The ICFAI University, Jaipur [2]

Abstract

In this paper we present, computational investigation of some iterative techniques for unraveling non direct conditions. This manuscript is also do examination some important iterative methods which can be used for purpose for computation for various applications. Utilizing Bisection technique, Newton strategy and Secant technique comprehend non straight conditions and their outcomes are analyzed. The capacity and discover the foundation of capacity by these three techniques. By computational count , it was seen that Bisection strategy merges at the 24th cycles , Newton technique combines at fourth emphases and Secant strategy meets at fifth emphases. Newton technique required less number of emphases when contrasted with secant strategy. Be that as it may, when we think about execution, Newton technique needs two capacities and it requires some investment to figure. While Secant strategy needs just one capacity. By the computational tests, we saw that Secant Method is successful than different strategies.

Introduction

We know that solving the non-linear equations is most important and hard problems. Algebraic equations is of the form $ax^2+bx+c=0$, $y^3+y=1=0$. Any equation of the form which involves trigonometric functions, exponential functions or logarithmic functions is known as transcendental equations. For example:- $x+\cos x = 0$, $e^x+x=0$,$x^2-\log_e x =0$. The common methods for finding the root:- Bisection method,[1] [2] Newton-Raphson's method,[3] [4] secant method, [5] Regula falsi method [7] etc. There are different methods for finding the root and all these converge to the root at the different rates. The rate of convergence of some methods is faster than others. There is the study at comparing the rate of convergence of Newton Raphson method, Bisection method, Secant method.[8]

Newton Raphson method is very speedy and successful as compared to other methods. On the other hand, when we compare the performance, it is important to count both speed and cost of the convergence.[9] Newton Raphson method demanding only one iteration and calculating the derivative per iteration. Whereas Secant method does not require calculating the derivative. In most of the cases, calculating the derivative is difficult.[10], [11] Thus, we can determine that Newton Raphson method needs calculating the two functions per iteration. But Secant method needs only one function. Calculating the two functions in Newton Raphson method takes more time as compared to Secant method. Comparing the rate of convergence is in the following order:- Bisection method<Newton Raphson method<Secant method. There is a conclusion that Newton method is 7.678622465 times better than the Bisection method and Secant method is 1.389482397 times better than Newton method. [12]

METHODS

1. Bisection Method or Halving Method or Bolzano Method or Binary Chopping.

The bisection method is one of the simplest and most reliable of iterative methods for the solution of nonlinear equations. This method, also known as binary chopping or half-interval method, relies on the fact that if f(x) is real and continuous in the interval $a < x < b$, and f(a) and f(b) are of opposite signs, that is, [13], [14]

$$f(a)*f(b) < 0$$

ISSN: 2233 -7857 IJFGCN

Working method to find root by Bisection method:-

Step 1:- Take values 'a' and 'b'

f(a)*f(b)<0

Step 2:- Let x_1 = a+b/2

Step 3:- (a) If $f(a)f(x_1)<0$, at that time root lies between a and x_1. Proceed as in step 2 for finding second approximation to the root x_2.

(b) If $f(b)f(x_1)<0$, at that time root lies between b and x_1. Proceed as in step 2 for finding the second approximation to the root x_2.Repeat (a) or (b) in step 3 upto the required exactness of the root.

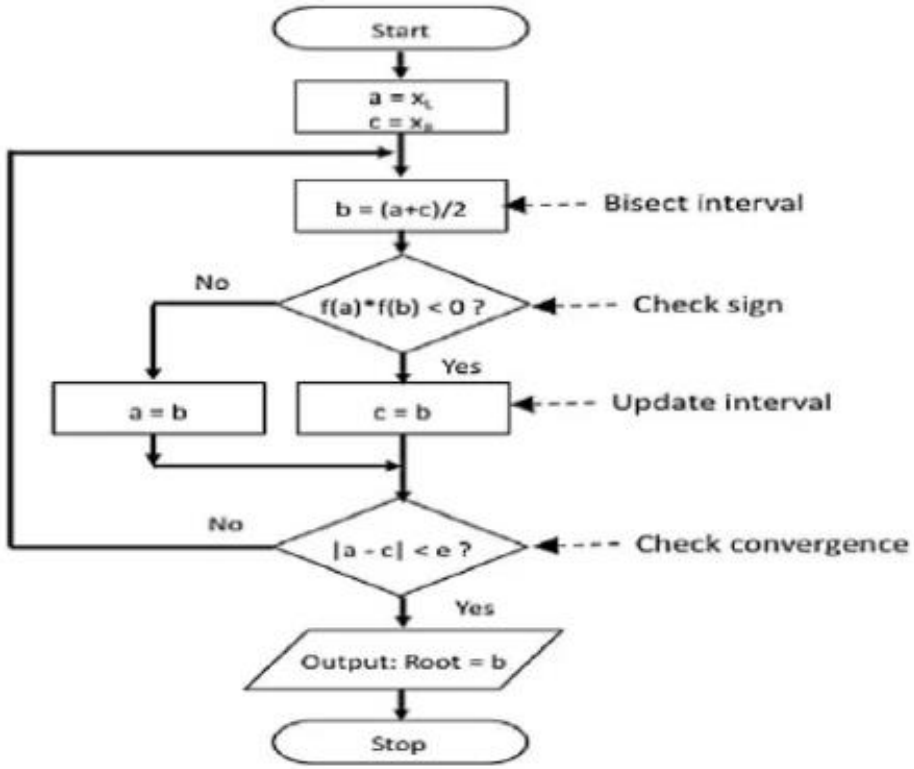

Figure 1: Flow of the computation methods for Bisection Method

Applications of Bisection Method for Computation:-

The bisection method is an iterative algorithm used to find roots of continuous functions. The main advantages to the method are the fact that it is guaranteed to converge if the initial interval is chosen appropriately, and that it is relatively simple to implement. The major disadvantage, however, is that convergence is slower than most other methods. You typically choose the method for tricky situations that cause difficulties for other methods. For example, if your choices are Bisection and Newton/Raphson, then Bisection will be useful if the function's derivative is equal to zero for some iteration, since that condition causes Newton's method to fail. It is not uncommon to develop hybrid algorithms that use Bisection for some iterations and faster methods for other iterations.[15]

2. Secant Method :- Working method to find root by secant method.

Step 1 :- Take values 'y' and 'z'.

Step 2 :- First calculation to the root is x_1 = y f(z) – z f(y)/f(z) – f(y).

Step 3 :- If $f(x_1) \neq 0$, at that point x_1 = y or z.

Choose that value for which $f(x_1)$ is close to zero.

ISSN: 2233 -7857 IJFGCN

Next, second approximation is $x_2 = y\ f(z) - z\ f(y)/f(z) - f(y)$.

Step 4 :- For calculating the following iterations, for valuing x_{n+1}, select x_{n-1} and x_n such as y and z.

The flow of computation can be illustrated from the Figure 2 as:-

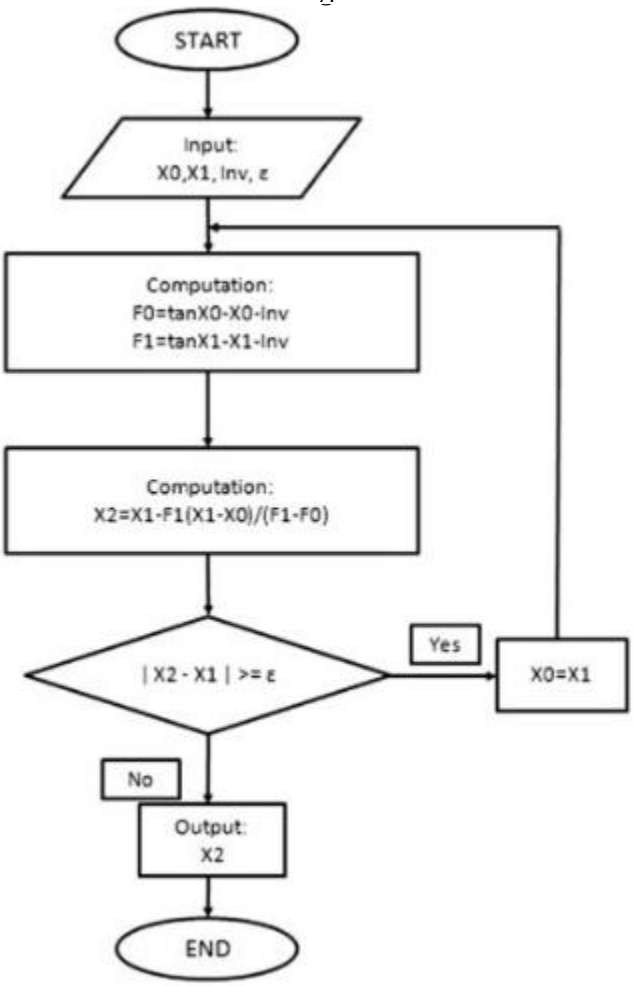

Figure Flow of the Secant Method`

Applications of Secant Method

Clearly utilization of the secant strategy to complex multi-story multi-level of-opportunity frameworks would be incredibly inconvenient if significant definite data about the conduct of the framework during a tremor is to be created. Examination of huge structures generally involves displaying hundreds, if not a great many components, and thousands, if not countless degrees of opportunity. In these cases, significantly under ideal conditions, leading the broad nonlinear investigation needed so as to gauge framework debasement is overwhelming. Further, emphasizing to assembly utilizing wasteful, aberrant emphasis calculations is likewise an impressive assignment when a great many boundaries are liable to change in every cycle. More awful yet, structures in many cases show vertical solidness or mass non-consistencies that render the utilization of 'same' first-mode approximations wrong, successfully disposing of an essential instrument from the specialist's scientific tool compartment. [16] Similarly as regularly, auxiliary frameworks frequently show torsional methods of conduct that should not to be disregarded in view of their capability to impact both burden circulation and disappointment succession, which additionally eliminates two-dimensional approximations from the logical tool compartment.

ISSN: 2233 -7857 IJFGCN

Subsequently, a requirement for a dependable and precise methods for consolidating the factors in an intricate structure to a sensible number is available. Further, a requirement for a productive emphasis plot that can deal with a generous number of factors – adequate to save key trademark practices, for example, torsional and higher mode reaction – and that can rapidly meet is clear. As of late, the creators have effectively utilized the Secant Method to contemplate the exhibition of a few extremely huge structures in late quakes, and have utilized productive cycle plans while thoroughly adjusting to the investigative necessities of the Secant Method as depicted in the references gave. One of the structures contemplated and the procedures used to consider it are portrayed underneath. [17]

3. Newton-Raphson's Method:- Working method to find to by Newton-Raphson's method.

Step 1 :- Take x_0 be the initial approximation, which is near to the exact root.
Step 2 :- Using Newton-Raphson's formula - $x_{n+1} = x_n - f(x_n)/f'(x_n)$. where n= 0,1,2.....
Step 3 :- Calculate $f(x_n)$ and $f'(x_n)$.
Step 4 :- If $f(x_{n+1}) = 0$, at that point x_{n+1} is itself a root of $f(x) = 0$.
If $f(x_{n+1}) \neq 0$, at that point continue as in step 2.

The flow of computation of this method can be illustrated as Figure 3.

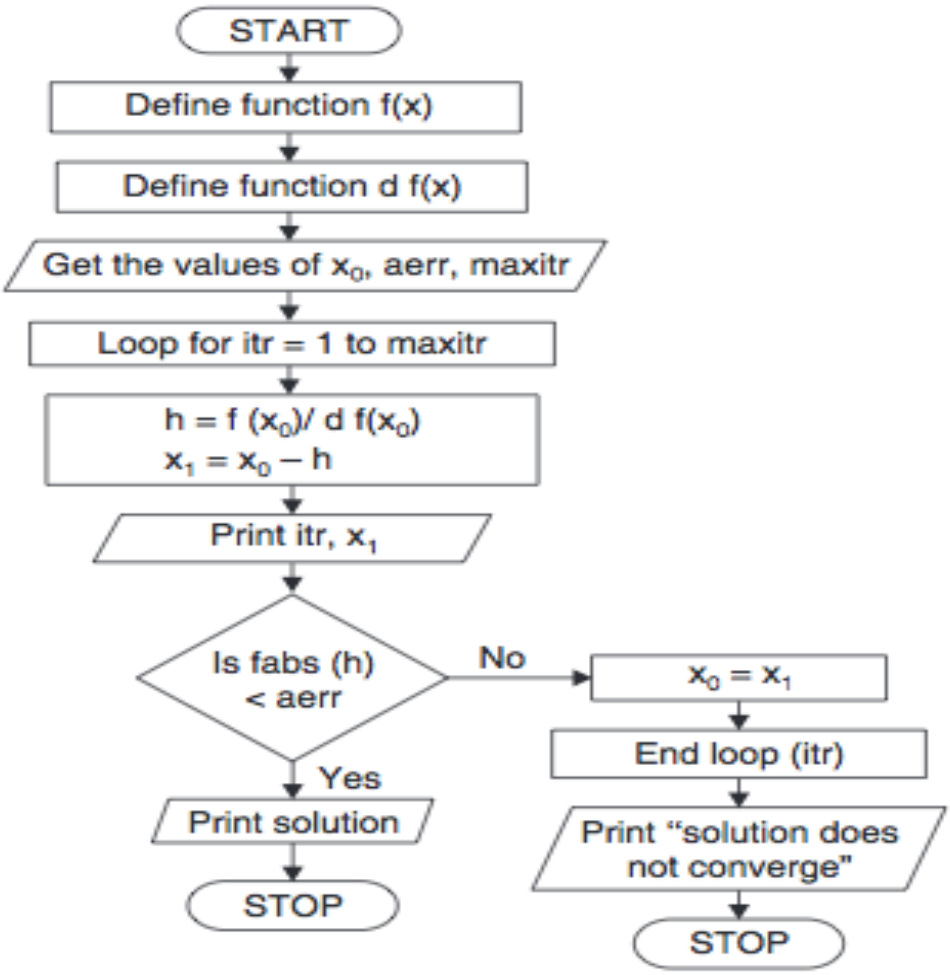

Figure 3 Flow of computation of Newton-Raphson's methods

ISSN: 2233 -7857 IJFGCN

Numerical Experiments and Comparative discussion

Code of bisection method-

```c
#include<stdio.h>
#include<math.h>

double F( double x) //Function definition
{
    //This return the value of the Function
    return (pow(x,3) + 3*x -5);
}

int main()
{
    printf("\n\n\t\tStudytonight - Best place to learn\n\n\n");

    printf("This program illustrates the bisection method in C:\n\n");
    printf(" x^3 + 3*x - 5 = 0\n\n");
    double x0,x1;

    printf("\nEnter the first approximation to the root :  ");
    scanf("%lf",&x0);

    printf("\n\nEnter the second approximation to the root :   ");
    scanf("%lf",&x1);

    int iter;
    printf("\n\nEnter the number of iteration you want to perform :   ");
    scanf("%d",&iter);

    int ctr=1;
    double l1=x0;
    double l2=x1;
    double r,f1,f2,f3;
```

```c
if(F(l1)==0)/
    r=l1;
else if(F(l2)==0)
    r=l2;
else
{
    while(ctr<=iter)
    {

        f1=F(l1);
        r=(l1+l2)/2.0;
        f2=F(r);
        f3=F(l2);

        if(f2==0)
        {
            r=f2;
            break;
        }
        printf("The root after %d iteration is %lf \n\n",ctr,r);

        if(f1*f2<0)
            l2=r;
        else if(f2*f3<0)
            l1=r;
        ctr++;
    }
}

printf("\n\n\nThe approximation to the root is %lf\n",r);
```

Finding root by bisection method : - we have, f(x) = 10x – cos (x) -5

ISSN: 2233 -7857 IJFGCN

Table 1.

Loop no	a	b	x_n	$f(x_n)$
1.	0.5	1	0.75	1.7683112
2.	0.5	0.75	0.625	0.43903689
3.	0.5	0.625	0.5625	-0.22092449
4.	0.5625	0.625	0.59375	0.10865152
5.	0.5625	0.59375	0.578125	-0.0562387
6.	0.578125	0.59375	0.5859375	-0.0521536
7.	0.5859375	0.59375	0.58984375	0.0674099
8.	0.5859375	0.58984375	0.58789062	0.04679385
9.	0.5859375	0.587890625	0.586914062	0.03648702
10.	0.5859375	0.586914062	0.586425781	0.0313339
11.	0.5859375	0.586425781	0.58618164	0.028757397
12.	0.5859375	0.58618164	0.58605957	0.027469176
13.	0.5859375	0.58605957	0.585998535	-0.05180915
14.	0.585998535	0.58605957	0.58602905	0.027147097
15.	0.5859375	0.58602905	0.58598325	0.0266637
16.	0.5859375	0.58598325	0.585960375	0.026422367
17.	0.5859367	0.585960375	0.585948937	0.026301662
18.	0.5859375	0.585948937	0.585943218	0.02624131
19.	0.5859375	0.585943218	0.585940359	0.013105569
20.	0.5859375	0.585940359	0.585938929	0.026196048
21.	0.5859375	0.585938929	0.585938214	0.026188502
22.	0.5859375	0.585938214	0.585937857	0.026184735
23.	0.5859375	0.585937857	0.585937678	0.026182846
24.	0.5859375	0.585937678	0.585937589	0.026182907

Table 1 shows that function converges to the root at 0.585937589 at 24th iterations.

ISSN: 2233 -7857 IJFGCN

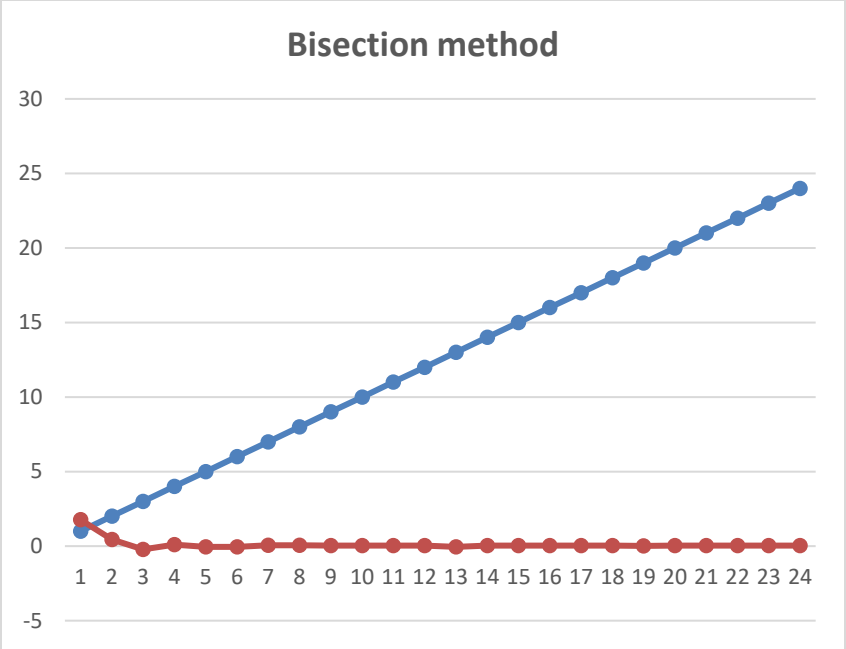

Figure 4 Relationship between Loop's and F(x)

Finding root by Secant method :We have, f(x) = 10x – cos (x) – 5

Code:-

```
#include<stdio.h>
float f(float x)
{
    return(x*x*x-4);
}
float main()
{
    float a,b,c,d,e;
    int count=1,n;
    printf("\n\nEnter the values of a and b:\n");
    scanf("%f%f",&a,&b);
    printf("Enter the values of allowed error and maximun number of iterations:\n");
    scanf("%f %d",&e,&n);
    do
    {
        if(f(a)==f(b))
        {
            printf("\nSolution cannot be found as the values of a and b are same.\n");
        return;
        }
        c=(a*f(b)-b*f(a))/(f(b)-f(a));
        a=b;
        b=c;
        printf("Iteration No-%d    x=%f\n",count,c);
        count++;
        if(count==n)
        {
        break;
        }
    } while(fabs(f(c))>e);
    printf("\n The required solution is %f\n",c);

}
```


ISSN: 2233 -7857 IJFGCN

Loop no.	y	Z	f(y)	f(z)	X_{n+1}	$f(x_{n+1})$
1.	0.5	1	-0.8775825	4.4596977	0.5822125	-0.0131231
2.	0.5822125	1	-0.0131231	4.4596977	0.58343827	-0.0001907
3.	0.58343827	1	-0.0001907	4.4596977	0.58345607	-0.0000029
4.	0.58345607	1	-0.0000029	4.4596977	0.583456319	0.28365146
5.	0.58345607	0.583456319	-0.0000029	0.28365146	0.583456194	0.28365034

Table 2 Shows that function converges to the root 0.583456194 at 5th iteration.

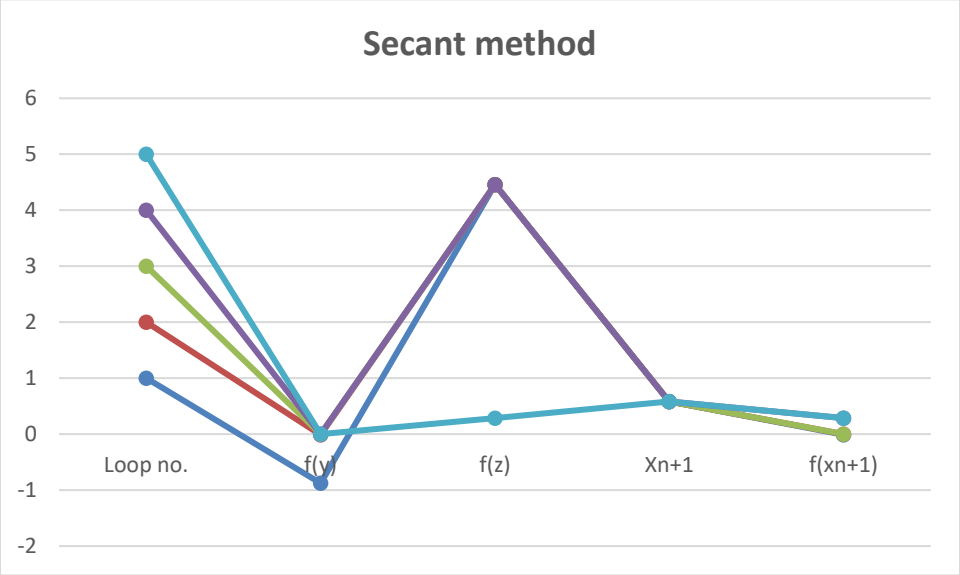

Figure 5: Relationship between Loop's and f(x), f(y), f(z), X_{n+1}, $f(x_{n+1})$

Finding root by Newton-Raphson's method : - we have , f(x) = 10x – cos (x) -5

Loop no.	x_n	$f(x_n)$	$f'(x_n)$	X_{n+1}
1.	0.5	-0.8775825	10.4794255	0.5837433
2.	0.5837433	0.0030277	10.5511512	0.5834563
3.	0.5834563	-0.0000005	10.5509117	0.583456347
4.	**0.583456347**	**-0.000000029**	**10.55091176**	**0.583456349**

Table 3 shows that function converges to the root at 0.5834563 at 4th iteration.

ISSN: 2233 -7857 IJFGCN

Code for Newton-Raphson's method

```c
#include<stdio.h>
#include<math.h>
float f(float x)
{
    return x*log10(x) - 1.2;
}
float df (float x)
{
    return log10(x) + 0.43429;
}
void main()
{
    int itr, maxmitr;
    float h, x0, x1, allerr;
    printf("\nEnter x0, allowed error and maximum iterations\n");
    scanf("%f %f %d", &x0, &allerr, &maxmitr);
    for (itr=1; itr<=maxmitr; itr++)
    {
        h=f(x0)/df(x0);
        x1=x0-h;
        printf(" At Iteration no. %3d, x = %9.6f\n", itr, x1);
        if (fabs(h) < allerr)
        {
            printf("After %3d iterations, root = %8.6f\n", itr, x1);
            return 0;
        }
        x0=x1;
    }
    printf(" The required solution does not converge or iterations are insufficient\n");
    return 1;
}
```

Sr.no	Bisection method	Newton-Raphson's method	Secant method
1.	Number of iterations=24	Number of iterations= 4	Number of iterations= 5
2.	Converges too slow	Converges fast	Convergence rate close to Newton-Raphson's method
3.	Number of iterations are more as compared to other methods. Thus it takes more time.	Number of iterations are less but when we consider performance, there is need to calculate two functions and thus it takes more time.	This method takes less time because there is need to calculate only one function.

Results and Discussions:

From the above outcomes, we see that Newton-Raphson technique requires two capacities at each progression, these capacities are f(x) and f'(x). In Secant technique there is have to compute just one capacity. Consequently, we see that Newton-Raphson technique takes additional time when contrasted with Secant strategy. Then again, based on mathematical calculation we thought about that combination pace of cut technique is excessively moderate yet sure. In light of our outcomes, we reason that Secant strategy is powerful when contrasted with Newton-Raphson's technique.

Conclusion

We conclude that Secant method is very valuable as compared to Newton-Raphson's method, because Newton method requires two functions while secant method need only one function. Also observe that Bisection method converges too slow but sure. There are various techniques but, further we need to evaluate the computation complexity of all the methods to compare in terms of the performance as space and time complexity while, it is also important to get the comparison of the above said methods. Further we need to compare the impact of each method for the single application to evaluate the performance.

ISSN: 2233 -7857 IJFGCN

References:

[1] Boyd, S., Balakrishnan, V., & Kabamba, P. (1989). A bisection method for computing the H∞ norm of a transfer matrix and related problems. *Mathematics of Control, Signals and Systems*, *2*(3), 207-219.

[2] Bachrathy, D., & Stépán, G. (2012). Bisection method in higher dimensions and the efficiency number. *Periodica Polytechnica Mechanical Engineering*, *56*(2), 81-86.

[3] Ypma, T. J. (1995). Historical development of the Newton–Raphson method. *SIAM review*, *37*(4), 531-551.

[4] Hartmann, S. (2005). A remark on the application of the Newton-Raphson method in non-linear finite element analysis. *Computational Mechanics*, *36*(2), 100-116.

[5] Wolfe, P. (1959). The secant method for simultaneous nonlinear equations. *Communications of the ACM*, *2*(12), 12-13.

[6] Gragg, W. B., & Stewart, G. W. (1976). A stable variant of the secant method for solving nonlinear equations. *SIAM Journal on Numerical Analysis*, *13*(6), 889-903.

[7] FOX, L., & Sankar, R. (1973). The regula-falsi method for free-boundary problems. *IMA Journal of Applied Mathematics*, *12*(1), 49-54.

[8] Danandeh, M. A. (2018). Comparative and comprehensive review of maximum power point tracking methods for PV cells. *Renewable and Sustainable Energy Reviews*, *82*, 2743-2767.

[9] Gupta, R. K. (2019). *Numerical Methods: Fundamentals and Applications*. Cambridge University Press.

[10] Kim, J. H., & Rhee, J. G. (2006, March). Simulation and analysis of electromagnetic environment for the mobile receiver. In *2006 17th International Zurich Symposium on Electromagnetic Compatibility* (pp. 77-80). IEEE.

[11] Ansari, K. A., & Dichone, B. (2019). *An Introduction to Numerical Methods Using MATLAB*. SDC Publications.

[12] Phuengrattana, W., & Suantai, S. (2012). Comparison of the rate of convergence of various iterative methods for the class of weak contractions in Banach spaces. *Thai Journal of Mathematics*, *11*(1), 217-226.

[13] Kim, J. H., & Rhee, J. G. (2006, March). Simulation and analysis of electromagnetic environment for the mobile receiver. In *2006 17th International Zurich Symposium on Electromagnetic Compatibility* (pp. 77-80). IEEE.

[14] Kim, J. H., & Rhee, J. G. (2006, March). Simulation and analysis of electromagnetic environment for the mobile receiver. In *2006 17th International Zurich Symposium on Electromagnetic Compatibility* (pp. 77-80). IEEE.

[15] Jia, Y. B. (2004). Computation on parametric curves with an application in grasping. *The International Journal of Robotics Research*, *23*(7-8), 827-857.

[16] Zheng, Q., Zhao, P., Zhang, L., & Ma, W. (2010). Variants of Steffensen-secant method and applications. *Applied mathematics and computation*, *216*(12), 3486-3496.

[17] Tomota, Y. (1997). Application of the secant method to prediction of flow curves in multi-microstructure steels. *Acta materialia*, *45*(5), 1923-1929.

ISSN: 2233 -7857 IJFGCN