University of Dhaka

Department of Computer Science and Engineering

CSE-3212: Numerical Methods Lab

3rd Year 2nd Semester

Assignment: 02

Problems on Bisection, False Position, Newton-Raphson and Secant methods

Submitted by:
Syeed Abrar Zaoad, Roll: 23
Date of submission: 16th September, 2018

Submitted to:
Mr. Mubin Ul Haque,
Lecturer, Department of CSE, University of Dhaka

Problem 1:

Statement:

The velocity v of a falling parachutist is given by,

$$v = \frac{gm}{c}\left(1 - e^{-\left(\frac{c}{m}\right)t}\right)$$

where g = 9.8 m/s 2 . For a parachutist with a drag coefficient c = 15 kg/s, compute the mass m so that the velocity is v = 35 m/s at t = 9 s. By using (a) bisection and (b) false position.

Solution:

1a:

Source Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

double g = 9.8, c = 15, v = 35, t = 9;

double f(double mid)
{
    double ret = (g*mid) / c;
    double tmp1 = (c*t) / mid;
    tmp1 *= -1.0;
    double tmp = (1 - exp(tmp1));
    ret = ret * tmp;
    return ret - v;
}
void printfunction(int i)
{
    cout<<right<<fixed<<setprecision(6)<<i<<setw(14)<<f(i)<<endl;
    return;
}
void bisection(double a,double b)
{
    if(f(a)*f(b)>=0)
    {
        cout<<"you have to assumed right a and b"<<endl;
    }
    double c;
    int it=1;
    while((b-a)>=0.0001)
    {
        c=(a+b)/2;
        if(f(c)==0.0)
```

```
            break;
        else if(f(c)*f(a)<0)
            b=c;
        else
            a=c;
        cout<<"iteration "<<it<<" a "<<a<<" b "<<b<<endl;
        it++;
    }
    cout<<"the value of root is : "<<c<<endl;
}
int main()
{
    //freopen("out.csv", "w", stdout);
    double rE, lo, hi, Aerr, mid, prevMid = -1, xlo, xhi;
    cin>>xlo>>xhi>>Aerr;
    if((f(xlo)<0 && f(xhi)<0) || (f(xlo)>0 && f(xhi)>0))
    {
        cout<<"Root can't be found"<<endl;
        return 0;
    }

    int cnt = 65, caseno = 1;
    bool br = false;
    lo = xlo;
    hi = xhi;

    cout<<right<<"X"<<setw(14)<<"f(X)"<<endl;
    for(double i = lo; i <= hi; i += 0.1)
    {
        printfunction(i);
    }
    printfunction(hi);

    cout<<endl<<"Bisection"<<endl;

cout<<right<<"#"<<setw(14)<<"hi"<<setw(14)<<"lo"<<setw(14)<<"Xm"<<setw(14)<<
"f(Xm)"<<setw(14)<<"Error %"<<endl;
    while(true)
    {
        mid = (lo + hi) / 2.0;

        if(prevMid != -1)
        {
            rE =  fabs(mid - prevMid) * 100.0;
            rE /= mid;
            if(rE < Aerr)  br = true;
        }

        if(prevMid == -1) cout<<right<<fixed<<setprecision(6)<<caseno+
+<<setw(14)<<hi<<setw(14)<<lo<<setw(14)<<mid<<setw(14)<<f(mid)<<setw(14)<<"
```

```
N/A"<<endl;
        else cout<<right<<fixed<<setprecision(6)<<caseno+
+<<setw(14)<<hi<<setw(14)<<lo<<setw(14)<<mid<<setw(14)<<f(mid)<<setw(14)<<r
E<<endl;

        if(f(mid) > 0) hi = mid;
        else lo = mid;
        prevMid = mid;

        if(br) break;
    }

    cout<<fixed<<setprecision(6)<<"The root of bisection method is:
"<<mid<<endl<<endl;

}

/*

58 60 .00001

*/
```

<u>1b:</u>

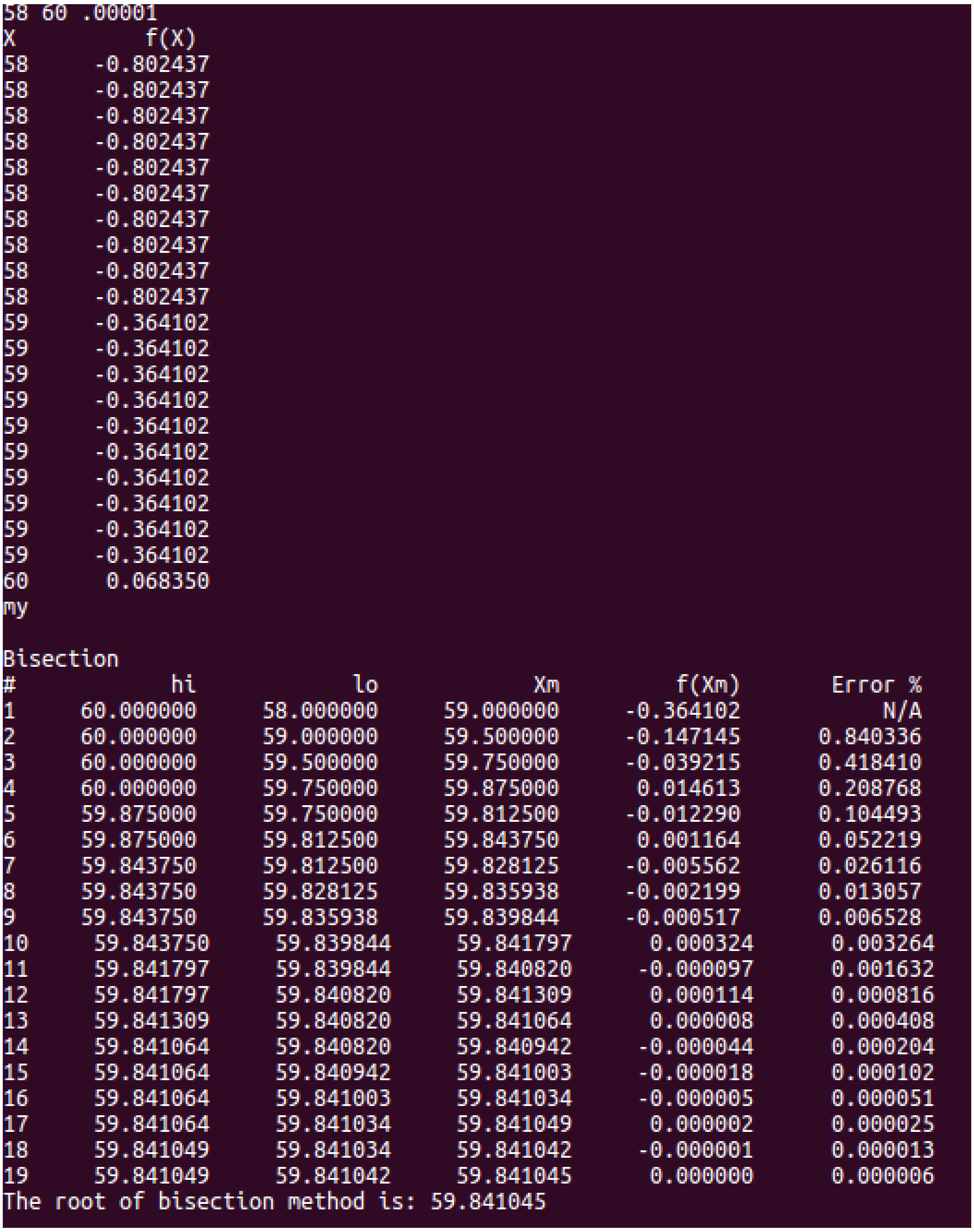
Source Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

double g = 9.8, c = 15, v = 35, t = 9;

double f(double mid)
{
    double ret = (g*mid) / c;
    double tmp1 = (c*t) / mid;
    tmp1 *= -1.0;
    double tmp = (1 - exp(tmp1));
    ret = ret * tmp;
    return ret - v;
}

double false_poistion(double lo, double hi)
{
    double tmp = (((hi * f(lo)) - (lo * f(hi))) / (f(lo) - f(hi)));
    return tmp;
```

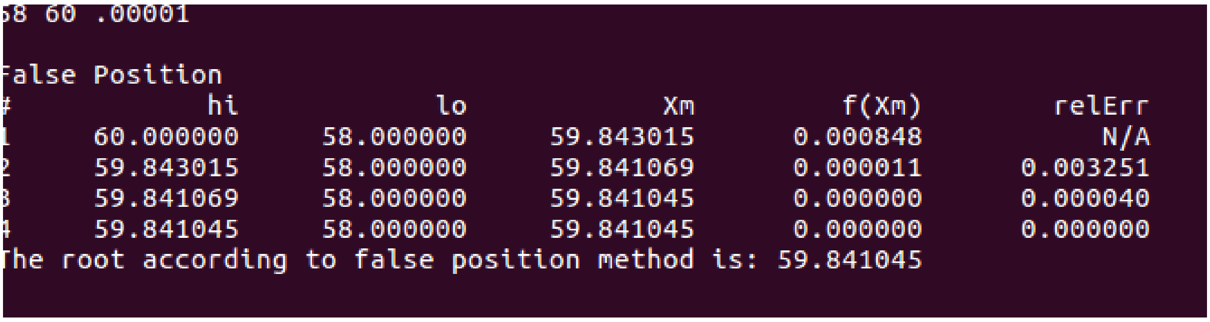
```
}

double isPossible(double lo, double hi)
{
   if((f(lo)<0 && f(hi)<0) || (f(lo)>0 && f(hi)>0)) return false;
   return true;
}

int main()
{
   double relErr, lo, hi, accpErr, mid, prevMid = -1, xlo, xhi;
   cin>>xlo>>xhi>>accpErr;
   cout<<endl<<"False Position"<<endl;

cout<<right<<"#"<<setw(14)<<"hi"<<setw(14)<<"lo"<<setw(14)<<"Xm"<<setw(14)<<
"f(Xm)"<<setw(14)<<"relErr"<<endl;
   lo = xlo;
   hi = xhi;
   prevMid = -1;
   int caseno = 1;
   bool br = false;

   while(true)
   {
      mid = false_poistion(lo, hi);

      if(prevMid != -1)
      {
         relErr =  fabs(mid - prevMid) * 100.0;
         relErr /= mid;
         if(relErr < accpErr)  br = true;
      }

      if(prevMid == -1) cout<<right<<fixed<<setprecision(6)<<caseno+
+<<setw(14)<<hi<<setw(14)<<lo<<setw(14)<<mid<<setw(14)<<f(mid)<<setw(14)<<"
N/A"<<endl;
      else cout<<right<<fixed<<setprecision(6)<<caseno+
+<<setw(14)<<hi<<setw(14)<<lo<<setw(14)<<mid<<setw(14)<<f(mid)<<setw(14)<<r
elErr<<endl;

      if(f(lo)*f(mid) < 0) hi = mid;
      else lo = mid;
      prevMid = mid;

      if(br) break;
   }

   cout<<fixed<<setprecision(6)<<"The root according to false position method is:
"<<mid<<endl<<endl;
```

```
}

/*

58 60 .00001

*/
```

Sample Input/Output:



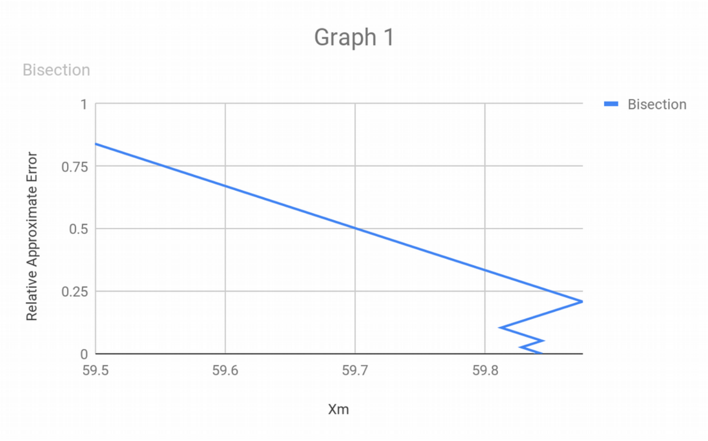
```
58 60 .00001
X          f(X)
58     -0.802437
58     -0.802437
58     -0.802437
58     -0.802437
58     -0.802437
58     -0.802437
58     -0.802437
58     -0.802437
58     -0.802437
58     -0.802437
59     -0.364102
59     -0.364102
59     -0.364102
59     -0.364102
59     -0.364102
59     -0.364102
59     -0.364102
59     -0.364102
59     -0.364102
59     -0.364102
60      0.068350
my

Bisection
#          hi            lo            Xm          f(Xm)        Error %
1     60.000000     58.000000     59.000000     -0.364102         N/A
2     60.000000     59.000000     59.500000     -0.147145       0.840336
3     60.000000     59.500000     59.750000     -0.039215       0.418410
4     60.000000     59.750000     59.875000      0.014613       0.208768
5     59.875000     59.750000     59.812500     -0.012290       0.104493
6     59.875000     59.812500     59.843750      0.001164       0.052219
7     59.843750     59.812500     59.828125     -0.005562       0.026116
8     59.843750     59.828125     59.835938     -0.002199       0.013057
9     59.843750     59.835938     59.839844     -0.000517       0.006528
10     59.843750     59.839844     59.841797      0.000324       0.003264
11     59.841797     59.839844     59.840820     -0.000097       0.001632
12     59.841797     59.840820     59.841309      0.000114       0.000816
13     59.841309     59.840820     59.841064      0.000008       0.000408
14     59.841064     59.840820     59.840942     -0.000044       0.000204
15     59.841064     59.840942     59.841003     -0.000018       0.000102
16     59.841064     59.841003     59.841034     -0.000005       0.000051
17     59.841064     59.841034     59.841049      0.000002       0.000025
18     59.841049     59.841034     59.841042     -0.000001       0.000013
19     59.841049     59.841042     59.841045      0.000000       0.000006
The root of bisection method is: 59.841045
```



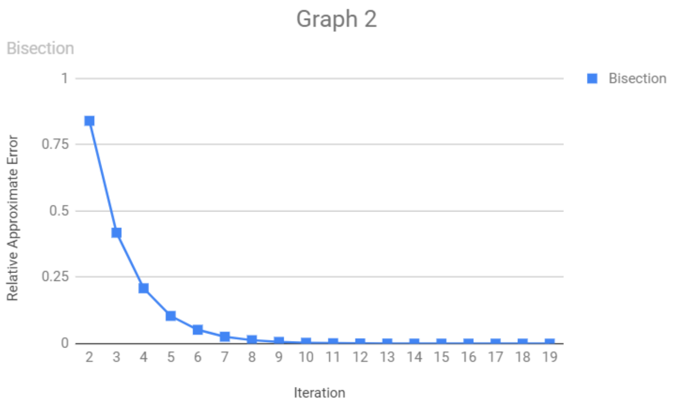
```
58 60 .00001

alse Position
#            hi              lo              Xm            f(Xm)           relErr
1      60.000000       58.000000       59.843015       0.000848              N/A
2      59.843015       58.000000       59.841069       0.000011         0.003251
3      59.841069       58.000000       59.841045       0.000000         0.000040
4      59.841045       58.000000       59.841045       0.000000         0.000000
he root according to false position method is: 59.841045
```

Graphs:

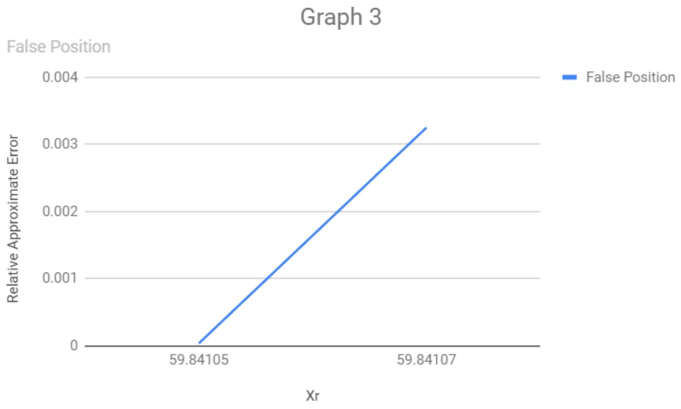
Graph 1: Graph of Xm and relative approximation error (bisection)



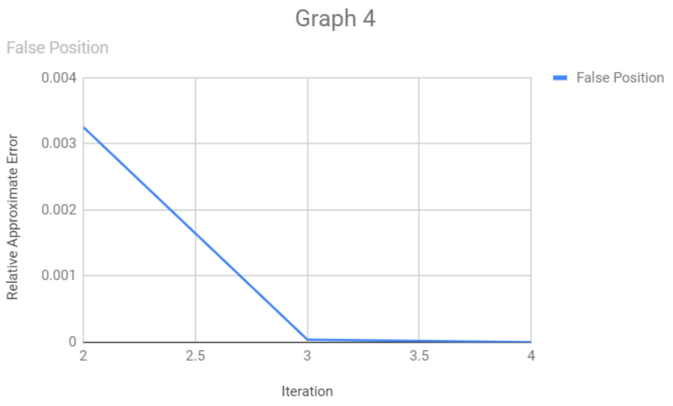
Graph 2: Graph of no of iteration and relative approximation error (bisection)



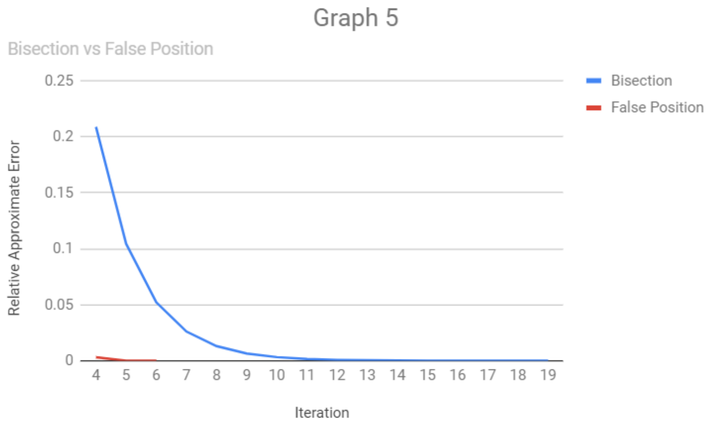
Graph 3: Graph of Xr and relative approximation error (false position)



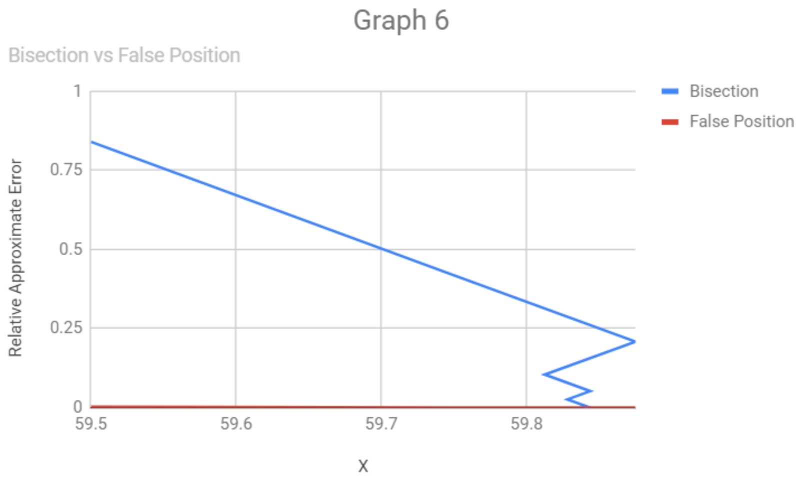
Graph 4: Graph of no of iteration and relative approximation error (false position)



Graph 5: Compare the relative approximate error with respect to number of iteration between the bisection method and false position method



Graph 6: Compare the relative approximate error with respect to x between the bisection method and false position method



Problem 2:

Statement:

Write a single program to solve the following,

(a) Use the Newton-Raphson method to determine a root of f (x) = −x2 + 1.8x + 2.5 using x0 = 5.

(b) Use the Newton-Raphson method to find the root of

f (x) = $e^{(-0.5x)}$ (4 − x) − 2

Employ initial guesses of (i) 2, (ii) 6, and (iii) 8

Solution:

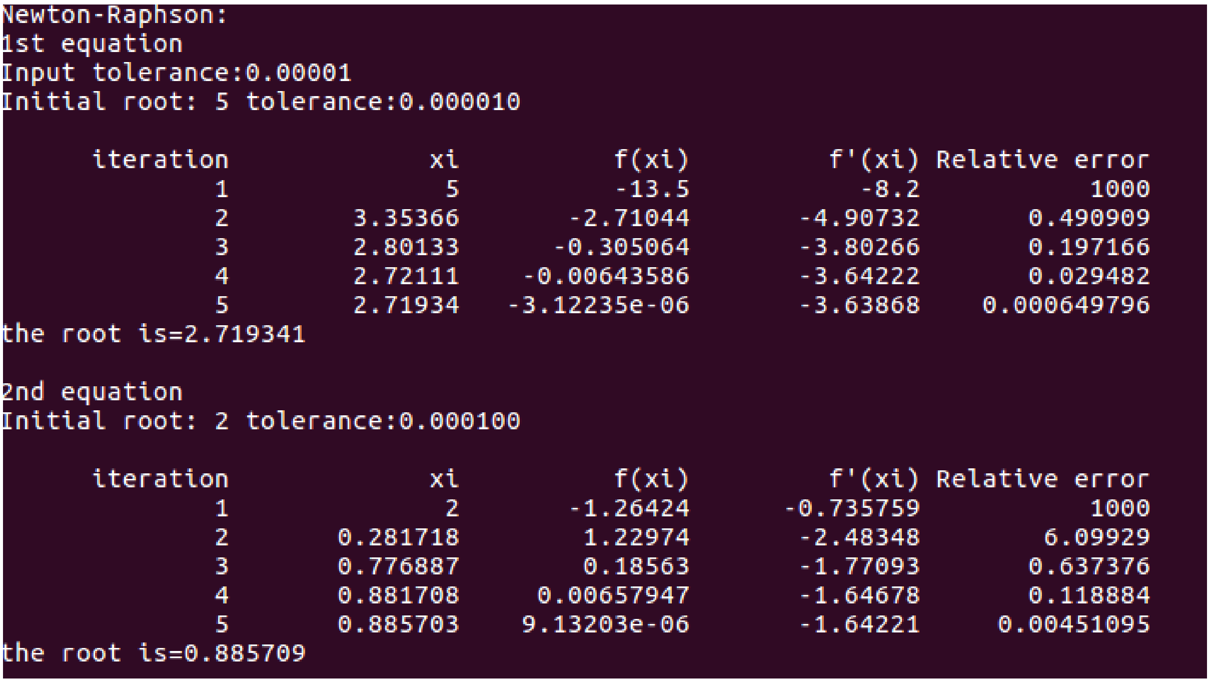
Source Code:

```
#include<bits/stdc++.h>
using namespace std;

double funca(double x)
{
   double r= -x*x + 1.8*x +2.5;
   return r;
}

double func1b(double x)
{
   double r= -2*x +1.8;
   return r;
}
void print(double v1, double v2, double v3, double v4, double v5)
{
   cout<< setw(15) << v1 << setw(15) << v2 << setw(15) << v3 <<  setw(15) << v4
<<  setw(15) << v5 <<endl;
}

void print(string v1, string v2, string v3, string v4, string v5)
{
   cout<<  setw(15) << v1 <<  setw(15) << v2 <<  setw(15) << v3 <<  setw(15) << v4
<<  setw(15) << v5 << endl;
}
```

```
double func2a(double x)
{
    double r= exp(-0.5*x) * (4-x) -2;
    return r;
}

double func2b(double x)
{
    double r= -exp(-0.5*x) - 0.5 * exp(-0.5*x) * (4-x);
    return r;
}

void Newton_Raphson(double initGuess, double input_tolerance, int cs)
{

    double x0, tolerance;;
    x0=initGuess;
    tolerance=input_tolerance;
    double x1=x0,rError=1000;


    print("iteration", "xi", "f(xi)", "f'(xi)", "Relative error");
    int cnt=0;
    while(rError>=tolerance)
    {
        x0=x1;

        double r0,r1;
        if(cs==1)
        {
            r0=funca(x0);
            r1=func1b(x0);
        }
        else
        {
            r0=func2a(x0);
            r1=func2b(x0);
        }

        //printf("iteration=%d xi=%.6f  f(xi)=%.6f f'(xi)=%.6f rError=%.6f\n",+
+cnt,x0,f0(x0), f1(x0),rError);
        print(++cnt,x0,r0, r1,rError);
        if(r1==0)
        {
            printf("Causing division by zero hence terminating\n");
            return ;
        }
        x1= x0 - r0/r1;
        rError=fabs((x1-x0)/x1);
    }
    printf("the root is=%.6f\n",x1);


}

int main()
{

    cout<<"Newton-Raphson:"<<endl;
    cout<<"1st equation"<<endl;
    printf("Input tolerance:");
    double tol;
    cin>>tol;
    printf("Initial root: 5 tolerance:%.6f\n\n",tol);
    Newton_Raphson (5,tol,1);
    cout<<endl;

    cout<<"2nd equation"<<endl;
    tol=0.0001;
    printf("Initial root: 2 tolerance:%.6f\n\n",tol);
    Newton_Raphson (2,tol,2);
    cout<<endl;

    printf("Initial root: 6 tolerance:%.6f\n\n",tol);
    Newton_Raphson (6,tol,2);
    cout<<endl;

    printf("Initial root: 8 tolerance:%.6f\n\n",tol);
    Newton_Raphson (8,tol,2);
   cout<<endl;


}
```

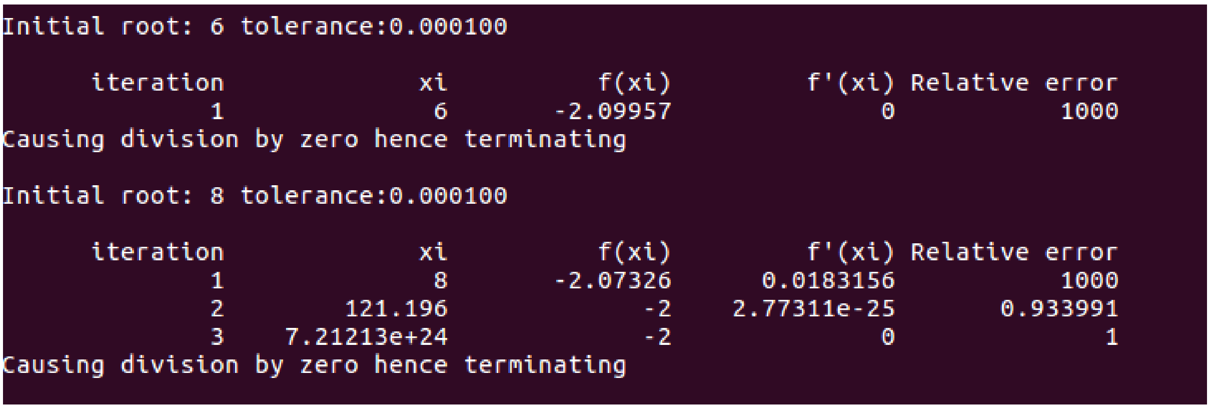
Sample Input/Output:

1.



```
Newton-Raphson:
1st equation
Input tolerance:0.00001
Initial root: 5 tolerance:0.000010

      iteration              xi          f(xi)         f'(xi) Relative error
              1               5          -13.5           -8.2           1000
              2         3.35366       -2.71044       -4.90732       0.490909
              3         2.80133      -0.305064       -3.80266       0.197166
              4         2.72111    -0.00643586       -3.64222       0.029482
              5         2.71934   -3.12235e-06       -3.63868    0.000649796
the root is=2.719341

2nd equation
Initial root: 2 tolerance:0.000100

      iteration              xi          f(xi)         f'(xi) Relative error
              1               2       -1.26424      -0.735759           1000
              2        0.281718        1.22974       -2.48348        6.09929
              3        0.776887        0.18563       -1.77093       0.637376
              4        0.881708     0.00657947       -1.64678       0.118884
              5        0.885703    9.13203e-06       -1.64221     0.00451095
the root is=0.885709
```

2.



```
Initial root: 6 tolerance:0.000100

      iteration              xi          f(xi)         f'(xi) Relative error
              1               6       -2.09957              0           1000
Causing division by zero hence terminating

Initial root: 8 tolerance:0.000100

      iteration              xi          f(xi)         f'(xi) Relative error
              1               8       -2.07326      0.0183156           1000
              2         121.196             -2    2.77311e-25       0.933991
              3     7.21213e+24             -2              0              1
Causing division by zero hence terminating
```

Problem 2(b) Discussion:

For this problem we were asked to find the root for three initial guess, x_0 = 2, 6 and 8. Newton Raphson could successfully determine the value for an initial guess of 2, but for guess greater than 6, The value of $f'_{\square}(x)$ becomes 0 and as such $\frac{f(x)}{f'(x)}$ approaches infinity. Due to this Newton Raphson can’t calculate the roots for 6 and 8.

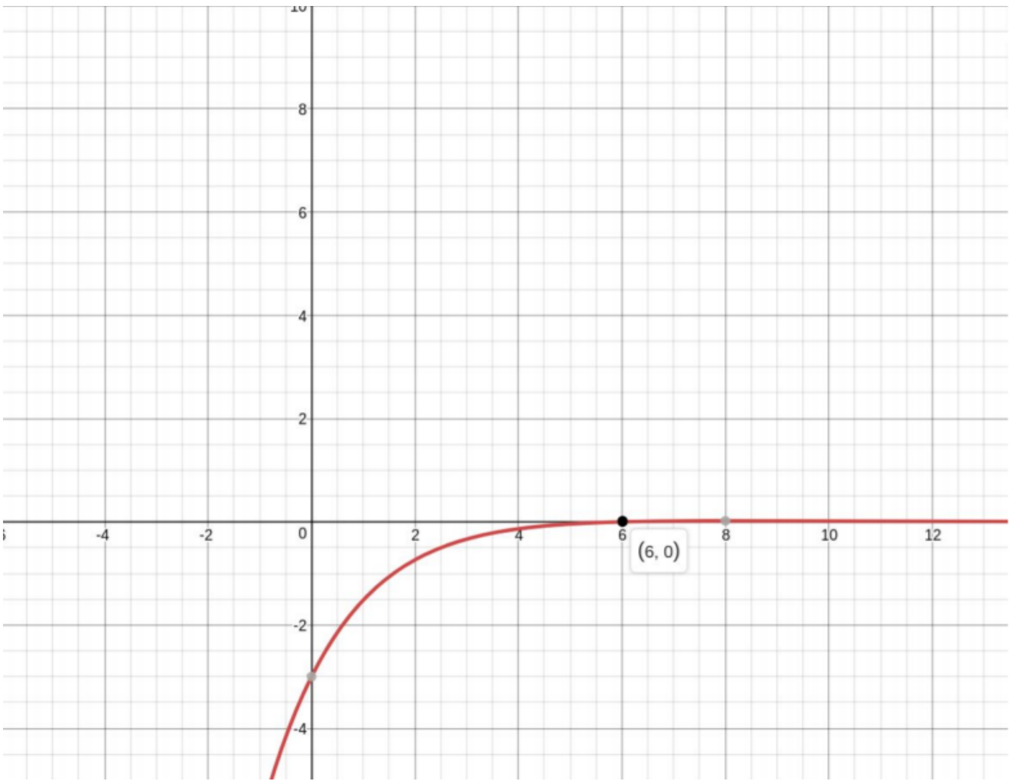


Fig: Graph of the derivative of the given function

Problem 3:

Statement:

(a) Consider following easily differentiable function,

f (x) = 8 sin(x)e–x − 1:

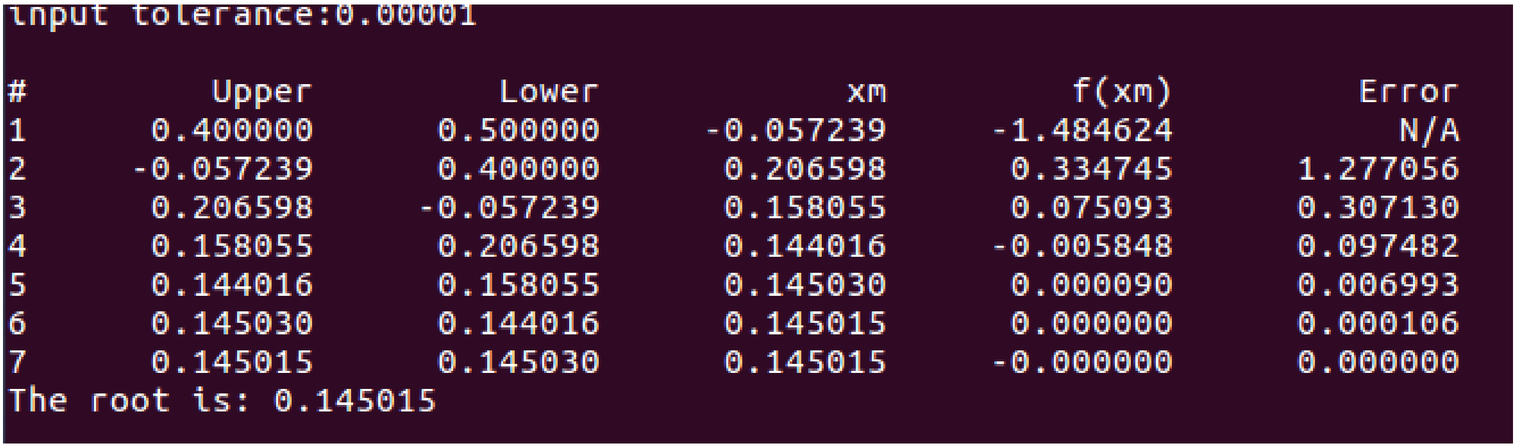
Use the secant method, when initial guesses of xi–1 = 0.5 and xi = 0.4

Solution:

Source Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

double f(double x) {
    return ((8.0 * sin(x) * exp(-1.0 * x)) - 1.0);
}

double xm(double xi, double xii) {
    return (xi - ((f(xi) * (xi - xii)) / (f(xi) - f(xii))));
}

void secant(double xi, double xii, double accpErr) {
    double relErr = INT_MAX, x_;
    double prev = -1;

cout<<endl<<"#"<<setw(14)<<"Upper"<<setw(14)<<"Lower"<<setw(14)<<"xm"<<set
w(14)<<"f(xm)"<<setw(14)<<"Error"<<endl;
    int caseno = 1;

    while(1) {
        x_ = xm(xi, xii);

        if(prev != -1) {
            relErr = fabs(prev - x_) / fabs(x_);
        }

        if(prev == -1) cout<<right<<fixed<<setprecision(6)<<caseno+
+<<setw(14)<<xi<<setw(14)<<xii<<setw(14)<<x_<<setw(14)<<f(x_)<<setw(14)<<"N/
A"<<endl;
        else cout<<right<<fixed<<setprecision(6)<<caseno+
+<<setw(14)<<xi<<setw(14)<<xii<<setw(14)<<x_<<setw(14)<<f(x_)<<setw(14)<<rel
Err<<endl;

        prev = x_;
        xii = xi;
        xi = x_;
        if(relErr < accpErr) break;
    }

    cout<<fixed<<setprecision(6)<<"The root is: "<<x_<<endl;
}

int main() {
    double xi = 0.4, xii = 0.5, accpErr;
    cin>>accpErr;
    secant(xi, xii, accpErr);
    return 0;
}
```

Sample Input/Output:



```
input tolerance:0.00001

#         Upper          Lower             xm          f(xm)          Error
1      0.400000       0.500000      -0.057239      -1.484624            N/A
2     -0.057239       0.400000       0.206598       0.334745       1.277056
3      0.206598      -0.057239       0.158055       0.075093       0.307130
4      0.158055       0.206598       0.144016      -0.005848       0.097482
5      0.144016       0.158055       0.145030       0.000090       0.006993
6      0.145030       0.144016       0.145015       0.000000       0.000106
7      0.145015       0.145030       0.145015      -0.000000       0.000000
The root is: 0.145015
```