University of Dhaka

Department of Computer Science and Engineering

CSE-3212: Numerical Methods Lab

Assignment-02

Submitted By

Name	Roll
Mohammad Habibullah Rakib	FH-25

Submitted To

1. Dr. Md. Haider Ali, Professor, CSE, DU
2. Mr. Mubin Ul Haque, Lecturer, CSE, DU

Problem 1

Statement: Water is flowing in a trapezoidal channel at a rate of Q = 20 m/s. The critical depth, y for such a channel must satisfy the equation

$$0 = 1 - \frac{Q^3}{gA_c^3}B$$

where, $g = 9.81\ m/s^2$, A_c = the cross-sectional area (m^2), and B = the width of the channel at the surface (m). For this case, the width and the cross-sectional area can be related to depth y by

$$B = 3 + y \quad \text{and} \quad A_c = 3y + \frac{y^2}{2}$$

Write a single program (source file name must be problem1 . extension) to solve for the critical depth using :

a) Bisection
b) False Position

Solution (C++ code): problem1.cpp

```
#include<bits/stdc++.h>

using namespace std;

const double g = 9.81;

const double Q = 20.00;

const double lo = 0.5, hi = 2.5;


void getAc_B (double y, double &Ac, double &B)

{

    Ac = 3*y + (y*y)/2;

    B = 3 + y;

}


double getFuncValue (double y)
```

```
{
    double Ac, B;
    getAc_B(y, Ac, B);
    return (1 -  (Q*Q*B)/(g*Ac*Ac*Ac));
}

double calcError(double neww, double old)
{
    return abs((neww - old)*100/neww);
}

void printValue ()
{
    double loCopy = lo, f_x;
    int cnt = 0;
    printf("\t    y \t\t\t   f(y) \n");
    while(loCopy <= hi + 0.001) {
        double f_x = getFuncValue(loCopy);
        printf("\t%lf \t|\t %lf\n",loCopy,f_x);
        loCopy += 0.1;
    }
}

void biSection (double ust, double lower, double upper) // ust = user
specified tolerance
{
    if(getFuncValue(upper) * getFuncValue(lower) > 0.0) {
        printf("\nThere is either no root or there is even number of roots in
the specified range");
        return;
    }
```

```
    printf("\nIteration No.\tUpper\t\tLower\t\tx_m\t\tf(x_m)\t\tRel. Approx. 
Error\n\n");

    int cnt = 0;

    double neww, old = 0.0, mid, hi, lo, midVal, lowVal;

    hi = upper;

    lo = lower;

    while(hi-lo >= ust) {

        mid = (lo + hi)/ 2.0;

        midVal = getFuncValue(mid);

        lowVal = getFuncValue(lo);

        if(midVal * lowVal <0.0)

            hi = mid;

        else

            lo = mid;

        cnt++;

        neww = mid;

        if(cnt == 1)

            printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t---(N/A)---\n", 
cnt,hi, lo, mid, midVal);

        else

            printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n", cnt,hi, 
lo, mid, midVal, calcError(neww, old));

        old = neww;

    }

    printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n", cnt + 1,hi, lo, 
mid, midVal, calcError(neww, old));

    printf("\nThe root is = %lf\n", mid);

}


double getFalseRoot(double x_l, double x_u)

{

    double fx_l = getFuncValue(x_l);

    double fx_u = getFuncValue(x_u);
```

```
    double x_r = (x_u * fx_l - x_l * fx_u)/(fx_l - fx_u);
    return x_r;
}

void falsePosition (double ust, double x_l, double x_u)
{
    double x_r, fx_l, fx_u, fx_r, x_r_o;
    x_r = getFalseRoot(x_l, x_u);
    fx_r = getFuncValue(x_r);
    fx_l = getFuncValue(x_l);
    fx_r = getFuncValue(x_r);
    if(fx_l * fx_u > 0.0) {
        printf("\nThere is either no root or there is even number of roots in
the specified range");
        return;
    }
    double error = 100.00 ;
    int cnt = 1;
    printf("\nIteration No.\tx_u\t\tx_l\t\tx_r\t\tf(x_r)\t\tRel. Approx.
Error\n\n");
    while (error > ust) {
        if(cnt == 1)
            printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t---(N/A)---\n",
cnt,x_u,x_l,x_r,fx_r);
        else
            printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n",
cnt,x_u,x_l,x_r,fx_r, error);
        if(fx_l * fx_r < 0)
            x_u = x_r;
        else
            x_l = x_r;
        x_r_o = x_r;
        x_r = getFalseRoot(x_l, x_u);
```

```
        fx_r = getFuncValue(x_r);

        fx_l  = getFuncValue(x_l);

        error = calcError(x_r, x_r_o);

        cnt++;

    }

    printf("\t%d\t%.6lf \t%.6lf \t%.6lf \t%.6lf \t%.6lf\n",
cnt,x_u,x_l,x_r,fx_r, error);

    printf("\nThe root is = %lf\n", x_r);

}


int main()

{

    double hi = 2.5, lo = 0.5, accuracy = 0.001, mid;

    int c;

    double loCopy = lo;

    printValue();

    printf("\n1.Bisection\t2.False Position\nEnter your choice: ");

    scanf("%d",&c);

    printf("\nEnter Low: ");

    scanf("%lf", &lo);

    printf("Enter High: ");

    scanf("%lf", &hi);

    printf("Enter Accuracy: ");

    scanf("%lf", &accuracy);

    if(c == 1)

        biSection(accuracy, lo, hi);

    if(c == 2)

        falsePosition(accuracy, lo, hi);

    return 0;

}
```

Sample Input/ Output:

```
Terminal
             y                         f(y)
        0.500000        |           -32.258215
        0.600000        |           -17.910278
        0.700000        |           -10.699416
        0.800000        |           -6.699595
        0.900000        |           -4.312155
        1.000000        |           -2.804055
        1.100000        |           -1.807448
        1.200000        |           -1.124169
        1.300000        |           -0.641159
        1.400000        |           -0.290786
        1.500000        |           -0.030946
        1.600000        |           0.165477
        1.700000        |           0.316466
        1.800000        |           0.434255
        1.900000        |           0.527355
        2.000000        |           0.601809
        2.100000        |           0.661983
        2.200000        |           0.711082
        2.300000        |           0.751493
        2.400000        |           0.785017
        2.500000        |           0.813032

1.Bisection     2.False Position
Enter your choice:
```

Fig: Value of y and f(y)

```
Terminal
1.Bisection     2.False Position
Enter your choice: 1

Enter Low: 0.5
Enter High: 2.5
Enter Accuracy: 0.0001

Iteration No.   Upper           Lower           x_m             f(x_m)          Rel. Approx. Error

        1       2.500000        1.500000        1.500000        -0.030946       ---(N/A)---
        2       2.000000        1.500000        2.000000        0.601809        25.000000
        3       1.750000        1.500000        1.750000        0.378909        14.285714
        4       1.625000        1.500000        1.625000        0.206927        7.692308
        5       1.562500        1.500000        1.562500        0.097956        4.000000
        6       1.531250        1.500000        1.531250        0.036261        2.040816
        7       1.515625        1.500000        1.515625        0.003383        1.030928
        8       1.515625        1.507812        1.507812        -0.013595       0.518135
        9       1.515625        1.511719        1.511719        -0.005060       0.258398
        10      1.515625        1.513672        1.513672        -0.000827       0.129032
        11      1.514648        1.513672        1.514648        0.001281        0.064475
        12      1.514160        1.513672        1.514160        0.000228        0.032248
        13      1.514160        1.513916        1.513916        -0.000300       0.016126
        14      1.514160        1.514038        1.514038        -0.000036       0.008063
        15      1.514099        1.514038        1.514099        0.000096        0.004031
        16      1.514099        1.514038        1.514099        0.000096        0.000000

The root is = 1.514099

Process returned 0 (0x0)   execution time : 21.479 s
Press ENTER to continue.
```

Fig: Problem 1.(a) console output

```
1.Bisection     2.False Position
Enter your choice: 2

Enter Low: 0.5
Enter High: 2.5
Enter Accuracy: 0.0001

Iteration No.   x_u             x_l             x_r             f(x_r)          Rel. Approx. Error

        1       2.500000        0.500000        2.450831        0.799873        ---(N/A)---
        2       2.450831        0.500000        2.403629        0.786123        1.963793
        3       2.403629        0.500000        2.358342        0.771792        1.920300
        4       2.358342        0.500000        2.314919        0.756894        1.875778
        5       2.314919        0.500000        2.273311        0.741447        1.830293
        6       2.273311        0.500000        2.233468        0.725474        1.783922
        7       2.233468        0.500000        2.195340        0.709003        1.736747
        8       2.195340        0.500000        2.158880        0.692065        1.688862
        9       2.158880        0.500000        2.124038        0.674695        1.640364
        10      2.124038        0.500000        2.090766        0.656933        1.591358
        11      2.090766        0.500000        2.059017        0.638822        1.541955
        12      2.059017        0.500000        2.028743        0.620408        1.492269
        13      2.028743        0.500000        1.999896        0.601740        1.442417
        14      1.999896        0.500000        1.972429        0.582868        1.392520
        15      1.972429        0.500000        1.946296        0.563846        1.342698
        16      1.946296        0.500000        1.921451        0.544727        1.293072
        17      1.921451        0.500000        1.897846        0.525565        1.243760
        18      1.897846        0.500000        1.875437        0.506414        1.194879
        19      1.875437        0.500000        1.854178        0.487327        1.146540
        20      1.854178        0.500000        1.834025        0.468358        1.098852
        21      1.834025        0.500000        1.814933        0.449556        1.051915
        22      1.814933        0.500000        1.796860        0.430969        1.005824
        23      1.796860        0.500000        1.779762        0.412643        0.960667
        24      1.779762        0.500000        1.763599        0.394621        0.916523
        25      1.763599        0.500000        1.748328        0.376941        0.873464
        26      1.748328        0.500000        1.733909        0.359639        0.831553
        27      1.733909        0.500000        1.720304        0.342748        0.790842
        28      1.720304        0.500000        1.707475        0.326295        0.751378
        29      1.707475        0.500000        1.695383        0.310304        0.713197
```

Fig: Problem 1.(b) console output

Graph:

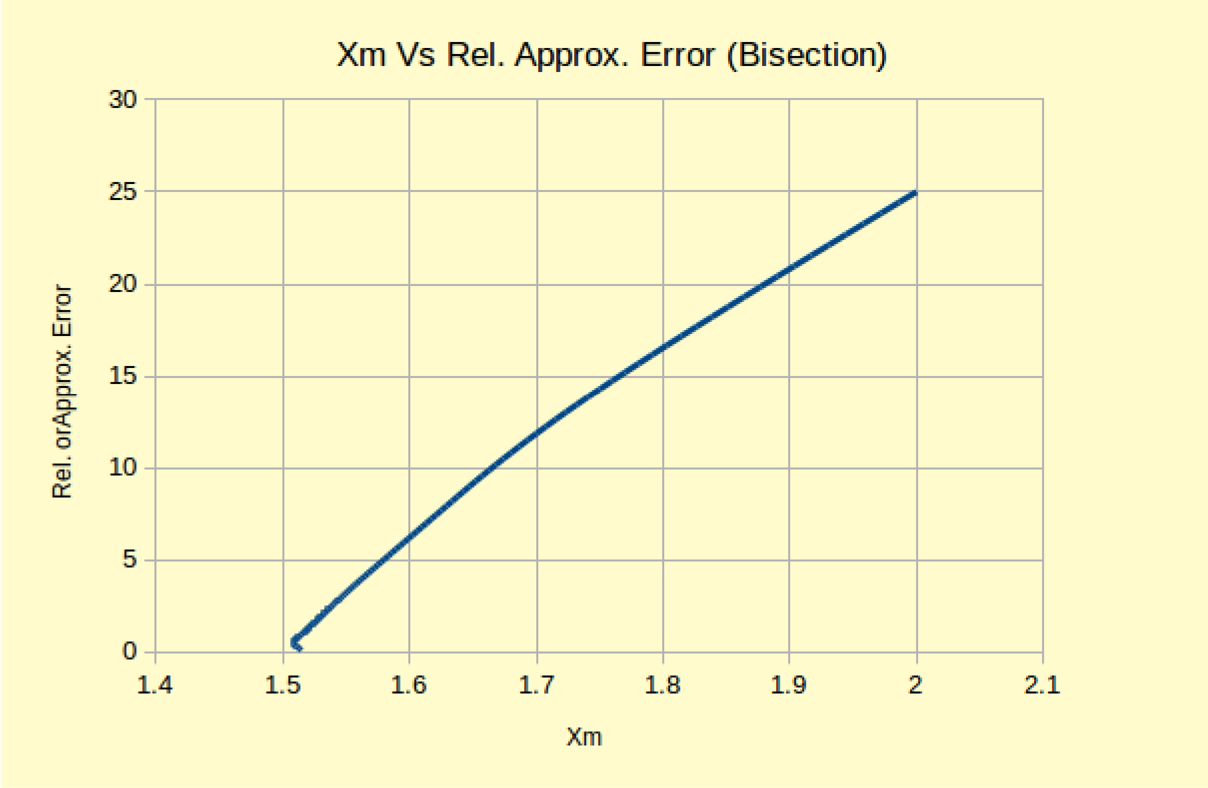

Fig: Graph 1

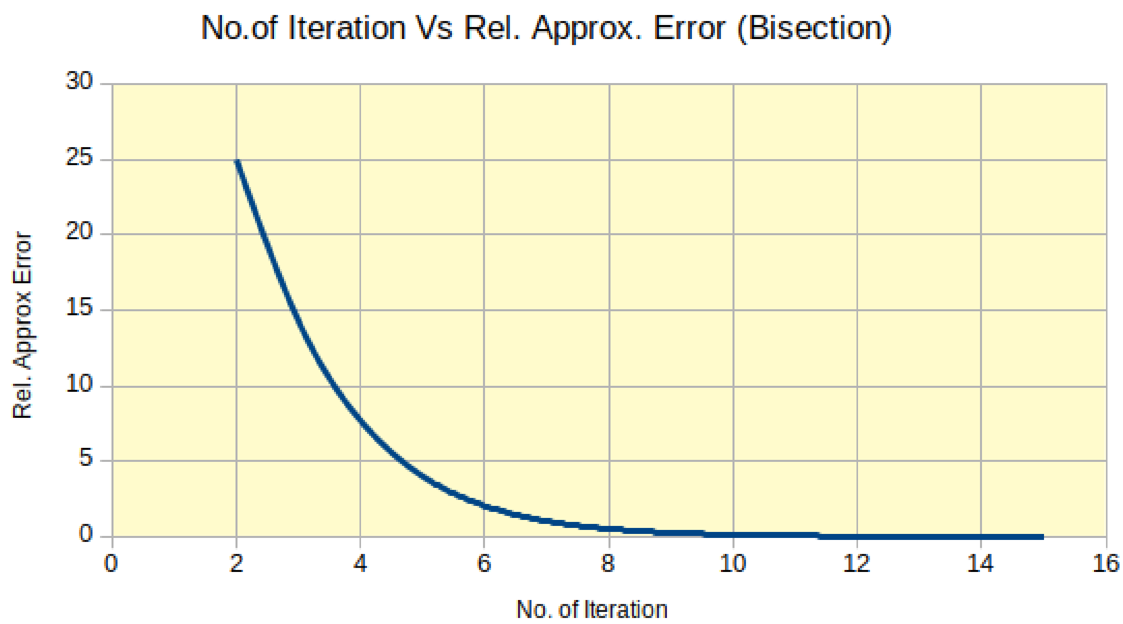

Fig: Graph 2

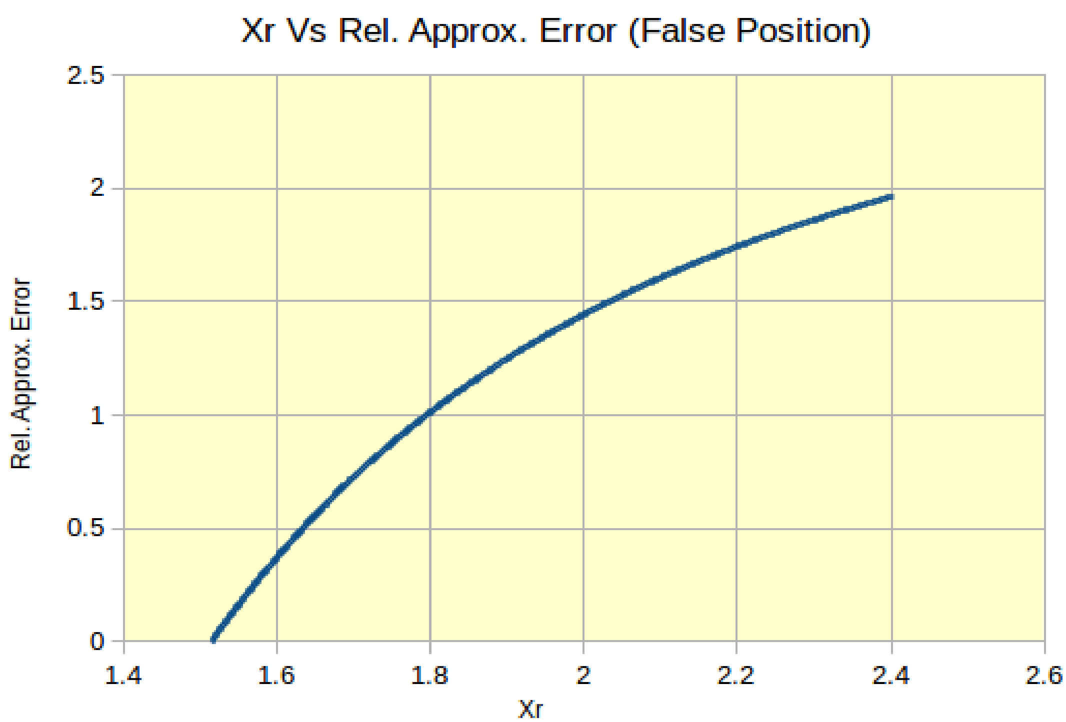

Fig: Graph 3

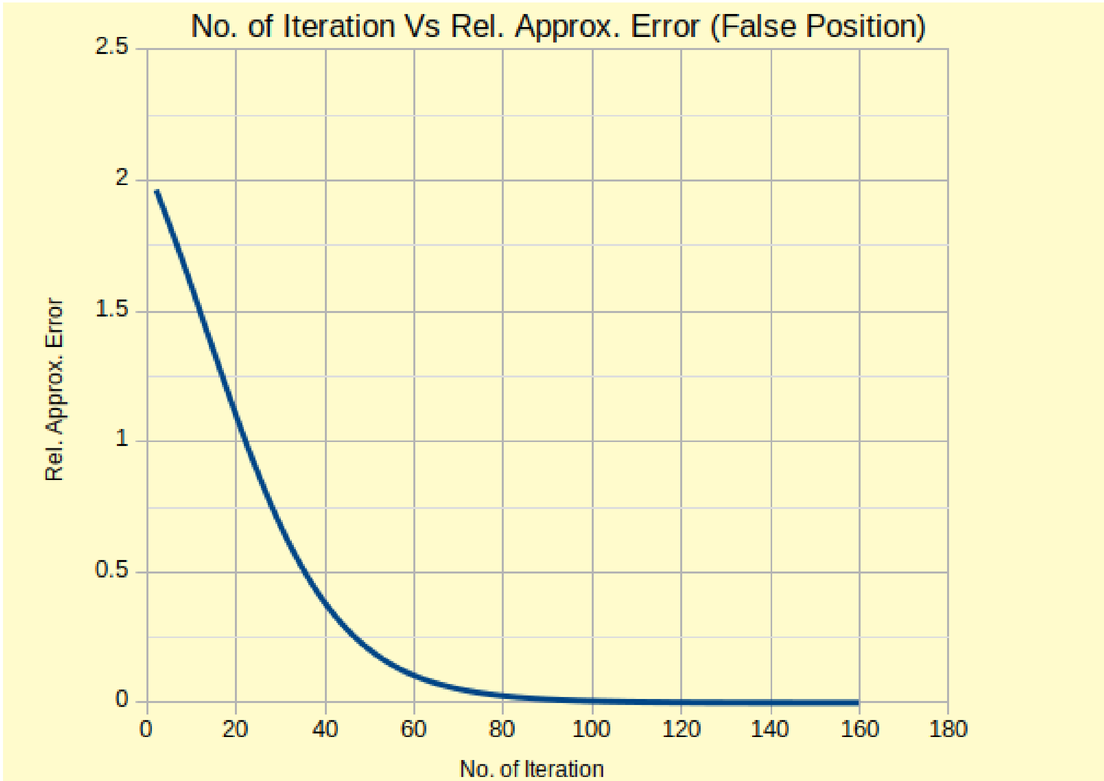

Fig: Graph 4

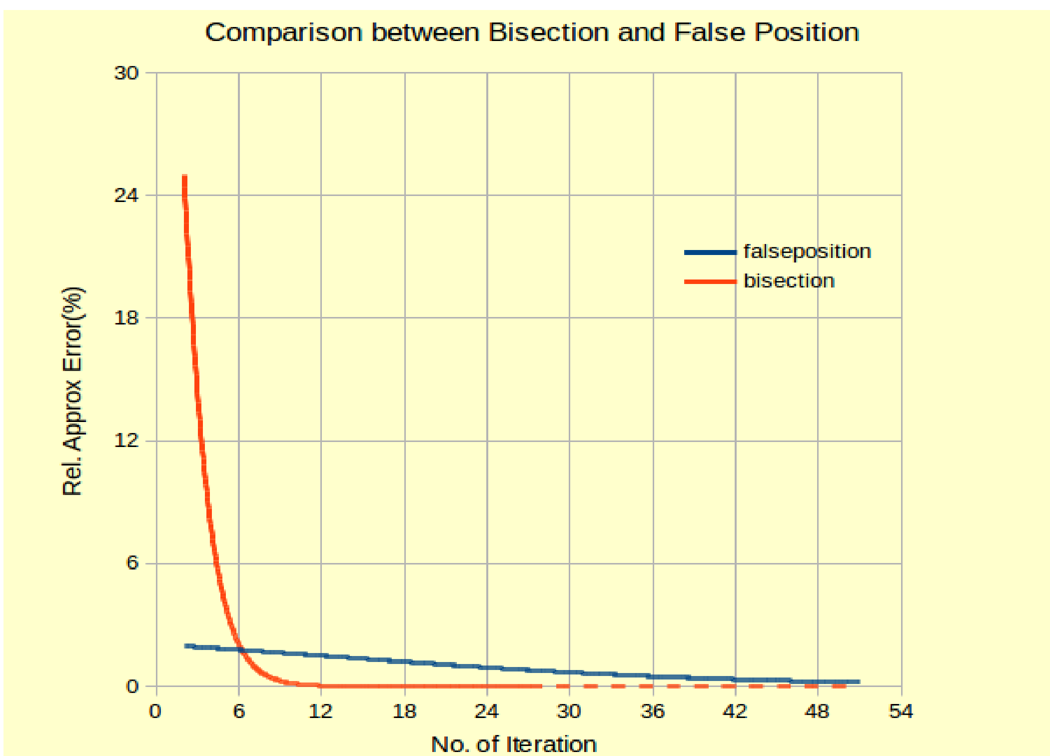

Fig: Graph 5

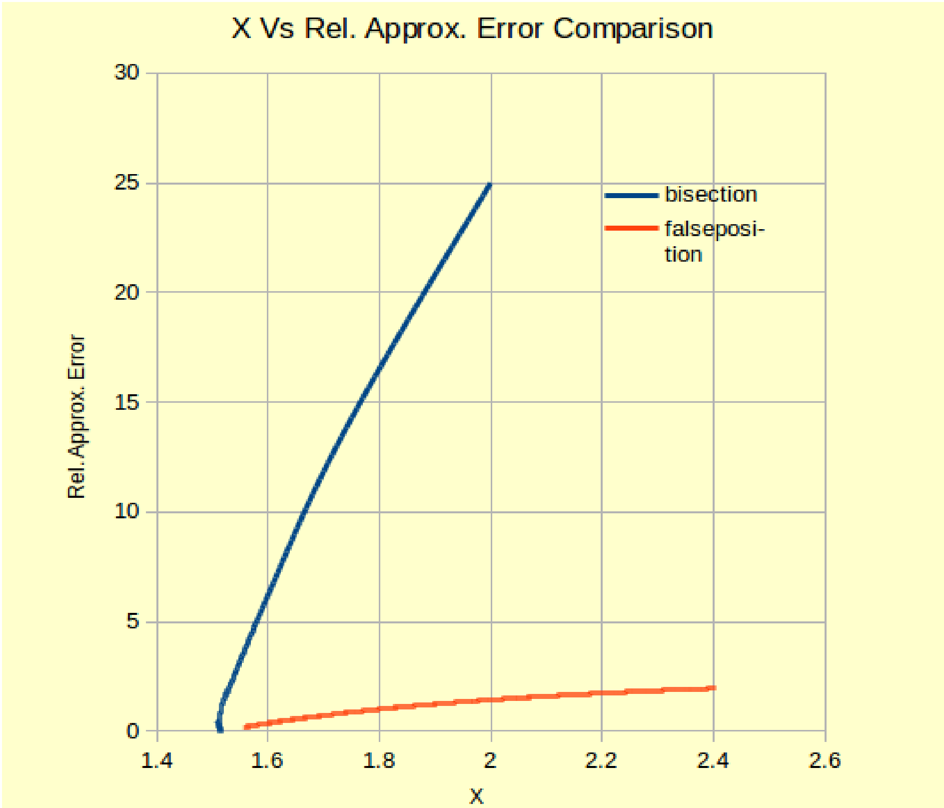

Fig: Graph 6

Problem 2

Statement: Write a single program to solve the following

a) A devotee of Newton-Raphson used the method to solve the equation $x^{100} = 0$, using the initial estimate $x_0 = 0.1$. Calculate the next five Newton Method estimates.

b) The devotee then tried to use the method to solve $3x^{\frac{1}{3}} = 0$, using $x_0 = 0.1$. Calculate the next ten Newton Method estimates.

Solution (C++ code):

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

double getFx(double x)
{
    double res = pow(x,100);
    return res;
}

double getFprimeX(double x)
{
    return 100*pow(x,99);
}

double getVal(double x_i)
{
    //double res = x_i - (getFx(x_i)/getFprimeX(x_i));
    //return res;
    return 0.99*x_i;
}

double getFxB(double x)
{
    double y = 0.33333333333333333333333333333333;
    if(x<0.0) {
        x *= -1.0;
        //printf("y = %lf ", pow(x,y));
        return (-3.0)*pow(x,y);
    }
    return 3*pow(x,y);
}

double getFprimeXB(double x)
{
    double y = getFxB(x);
    y /= 3.0;
    y *= y;
    return 1/y;
}

double getValB(double x_i)
{
    double res = x_i - (getFxB(x_i)/getFprimeXB(x_i));
    return res;
    //return -2.0*x_i;

}
```

```
double calcError(double neww, double old)
{
    return abs((neww - old)*100/neww);
}


void newton_raphson()
{
    printf("Problem (a):\n");
    printf("Intial guess X0 = 0.1, next five estimates are:\n");

    double a = 0.1, b;
    printf("\nIteration No.\tX_i\t\tF(X_i)\t\tF'(X_i)\t\tRel. Approx. 
Error\n\n");

    for(int i = 1; i<=5; i++) {
        b = getVal(a);

        printf("\t%d\t%.11lf \t%.11lf \t%.11lf \t%.11lf\n", 
i,b,getFx(b),getFprimeX(b),calcError(b,a));
        a = b;
    }
    puts("");

    printf("Problem (b):\n");
    printf("Intial guess X0 = 0.1, next ten estimates are:\n");
    a = 0.1;
    printf("\nIteration No.\tX_i\t\t\tF(X_i)\t\t\tF'(X_i)\t\t\tRel. Approx. 
Error\n\n");
    for(int i = 1; i<=10; i++) {
        b = getValB(a);
        printf("\t%d\t%.11lf\t\t%.11lf\t\t%.11lf\t\t%.11lf\n", 
i,b,getFxB(b),getFprimeXB(b),calcError(b,a));
        a = b;
 }

}

int main()
{
    newton_raphson();
    return 0;
}
```

Sample Input/ Output:

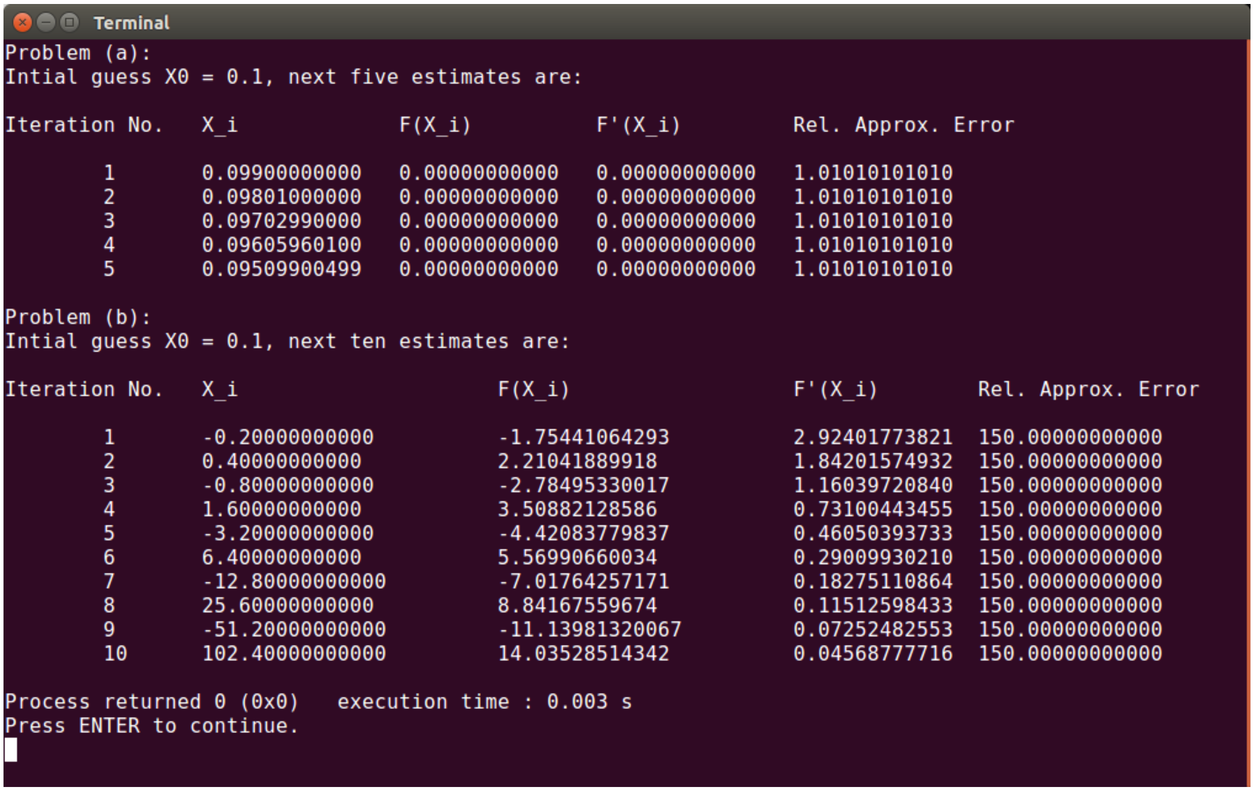

Fig: Problem 2 console output

Problem 3

Statement: Write a single program to solve the following

a) $e^{0.5x} = 5 - 5x$
Use the secant method, when initial guesses of $x_{i-1} = 0$ and $x_i = 2$ with user specified tolerance.

b) Locate the first positive root of, $f(x) = \sin x + \cos(1 + x^2) - 1$ where x is in radians. Use four iterations of the secant method with initial guesses of

a) $x_{i-1} = 1.0$ and $x_i = 3.0$;
b) $x_{i-1} = 1.5$ and $x_i = 2.5$, and
c) $x_{i-1} = 1.5$ and $x_i = 2.25$

to locate the root.

Solution (C++ code):

```
#include<bits/stdc++.h>

using namespace std;

double getFuncValue(double xn)

{

    double temp = exp(0.5 * xn) - 5 + 5*xn;

    return temp;

}

double getFuncValue2( double x)

{

    double res = sin(x) + cos (1 + x*x) - 1;

    return res;

}

double sec(double xn, double x1)

{

    double temp = ((xn - x1) * getFuncValue(xn))/ (getFuncValue(xn) -
getFuncValue(x1));

    return xn - temp;

}

double sec2(double xn, double x1)

{

    double temp = ((xn - x1) * getFuncValue2(xn))/ (getFuncValue2(xn) -
getFuncValue2(x1));

    return xn - temp;

}


double calcError(double neww, double old)

{

    return abs((neww - old)*100/neww);

}
```

```
void secantA(double a, double b, double ust)  // a = xn, b = xn+1

{

    double c, error;

    c = sec(a, b);

    error = calcError(c,b);

    int iter = 1;

    printf("\nIteration No.\tUpper\t\tLower\t\tx_m\t\tf(x_m)\t\tRel. Approx. 
Error\n\n");

    while (error > ust) {

        printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, 
c,a,b,getFuncValue(b),error);

        a = b;

        b = c;

        c = sec(a, b);

        error = calcError(c,b);

        iter++;

    }

    printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, 
c,a,b,getFuncValue(b),error);

    printf("\nThe root is %.8f %d\n", b);

}


void secantB(double a, double b)  // a = xn, b = xn+1

{

    double c, error;

    c = sec2(a, b);

    error = calcError(c,b);

    int iter = 1;

    printf("\nIteration No.\tUpper\t\tLower\t\tx_m\t\tf(x_m)\t\tRel. Approx. 
Error\n\n");

    while (iter <= 3) {

        printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, 
c,a,b,getFuncValue2(b),error);
```

```
        a = b;
        b = c;
        c = sec2(a, b);
        error = calcError(c,b);
        iter++;
    }
    printf("\t%d\t%.8lf \t%.8lf \t%.8lf \t%.8lf \t%.8lf\n", iter, 
c,a,b,getFuncValue2(b),error);
    printf("\nThe root is %.8f\n", b);
}
int main()
{
    double hi = 2.0, lo = 0.0, accuracy = 0.0001;
    printf("Problem (3A): \n");
    printf("\nEnter Low: ");
    scanf("%lf", &lo);
    printf("Enter High: ");
    scanf("%lf", &hi);
    printf("Enter Accuracy: ");
    scanf("%lf", &accuracy);
    secantA(lo, hi, accuracy);
    printf("Problem (3B): \n");
    printf("Intial Guesses: (a)\nLow = 1.0\tHigh = 3.0\n");
    secantB(1.0, 3.0);
    puts("");
    printf("Intial Guesses:(b)\nLow = 1.5\tHigh = 2.5\n");
    secantB(1.5,2.5);
    puts("");
    printf("Intial Guesses:(c)\nLow = 1.0\tHigh = 3.0\n");
    secantB(1.5, 2.25);
    return 0;
}
```

Sample Input/Output:

```
Terminal
Problem (3A):

Enter Low: 0
Enter High: 2
Enter Accuracy: 0.00001

Iteration No.   Upper           Lower           x_m             f(x_m)          Rel. Approx. Error

        1       0.68269394      0.00000000      2.00000000      7.71828183      65.86530296
        2       0.71266435      2.00000000      0.68269394      -0.17968901     4.39002129
        3       0.71417019      0.68269394      0.71266435      -0.00859641     0.21129703
        4       0.71416871      0.71266435      0.71417019      0.00000842      0.00020635
        5       0.71416872      0.71417019      0.71416871      -0.00000000     0.00000001

The root is 0.71416871 1

Process returned 0 (0x0)   execution time : 7.104 s
Press ENTER to continue.
```

Fig: Problem 3.(a) console output

```
Terminal
Problem (3B):
Intial Guesses: (a)
Low = 1.0       High = 3.0

Iteration No.   Upper           Lower           x_m             f(x_m)          Rel. Approx. Error

        1       -0.02321428     1.00000000      3.00000000      -1.69795152     13023.08094796
        2       -1.22634748     3.00000000      -0.02321428     -0.48336344     98.10703908
        3       0.23395122      -0.02321428     -1.22634748     -2.74475001     624.18939941
        4       0.39636577      -1.22634748     0.23395122      -0.27471727     40.97592885

The root is 0.23395122

Intial Guesses:(b)
Low = 1.5       High = 2.5

Iteration No.   Upper           Lower           x_m             f(x_m)          Rel. Approx. Error

        1       2.35692873      1.50000000      2.50000000      0.16639632      6.07024145
        2       2.54728716      2.50000000      2.35692873      0.66984231      7.47298649
        3       2.52633909      2.35692873      2.54728716      -0.08282791     0.82918687
        4       2.53210693      2.54728716      2.52633909      0.03147109      0.22778830

The root is 2.52633909

Intial Guesses:(c)
Low = 1.0       High = 3.0

Iteration No.   Upper           Lower           x_m             f(x_m)          Rel. Approx. Error

        1       1.92701799      1.50000000      2.25000000      0.75382086      16.76071567
        2       1.95147933      2.25000000      1.92701799      -0.06176948     1.25347672
        3       1.94460446      1.92701799      1.95147933      0.02414683      0.35353588
        4       1.94460843      1.95147933      1.94460446      -0.00001394     0.00020403

The root is 1.94460446
```

Fig: Problem 3.(b) console output