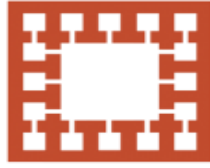


VARENDRA UNIVERSITY



বরেন্দ্র
বিশ্ববিদ্যালয়

VARENDRA UNIVERSITY

Department of Computer Science & Engineering

Lab Report

Course title: Numerical methods Lab

Course code: CSE 312

Date of Submitted: 17th April, 2021

Submitted By: Name: Md. Fatin Ilham ID: 191311102 Semester: 7 th Section: A Batch: 20 th Department of Computer Science and Engineering, Varendra University	Submitted To: Mst. Jannatul Ferdous Lecturer, Department of Computer Science and Engineering, Varendra University
--	--

Index

Lab	Problem	Date	Signature	Remarks
1	Error Calculation	14-01-2021		
2	Implementation of Bisection method	21-01-2021		
3	Implementation of False position method	28-01-2021		
4	Implementation of Newton Raphson method	04-02-2021		
5	Implementation of Iteration method	11-02-2021		
6	Newton's Forward Interpolation	11-03-2021		
7	Curve Fitting on Straight line	25-03-2021		
8	Non-linear Curve fitting	01-04-2021		
9	Lagrange's Interpolation	08-04-2021		

1. Suppose the measured value of the temperature is $T_m = 146.2$, but the true temperature is $T = 145.9$ (Same unit). What is the absolute error, and the relative error?
2. The width of a rectangular piece of land is measured to be 48.0474 ft. If the relative error is 0.0042 then find the true width of that land. Also find the absolute and percentage error.
3. Find the relative error of 8.6 where both of digits are correct.

Ans:

Hand-note:

1.

$$\text{absolute error} = 145.9 - 146.2$$

$$= -0.3$$

$$\text{relative error} = \frac{-0.3}{145.9} = -2.056 \times 10^{-3}$$

2.

$$E_R = \frac{x - x_1}{x}$$

$$0.0042 = \frac{x - 48.0474}{x}$$

$$x - 0.0042x = 48.0474$$

$$0.9958x = 48.0474$$

$$x = \frac{48.0474}{0.9958} = 48.2500$$

$$\text{absolute error} = (48.2500 - 48.0474) = 0.2026 \text{ (Ans.)}$$

$$\therefore \text{Percentage error} = (0.0042 \times 100) = 0.42 \text{ (Ans.)}$$

3.

$$\Delta x = \frac{1}{2} (10^{-N})$$

$x = 8.6$ is correct to 1 decimal places.

$$\text{then, } \Delta x = \frac{1}{2} (10^{-1}) = 0.05 = E_A$$

$$\text{Hence, } E_R = \frac{0.05}{8.6} = 0.0058 \text{ (Ans.)}$$

Code

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    double Tm=146.2,T=145.9;

    ///question 1
    cout<<"Answer 1:"<<endl;
    double absolute_error=T-Tm;
    cout<<absolute_error<<endl;
    double relative_error=absolute_error/(T*1.0);
    cout<<relative_error<<endl;
    cout<<endl;

    ///question 2
    cout<<"Answer 2:"<<endl;
    double measured_value=48.0474;
    double r_error=0.0042;
    double x= measured_value/(1-r_error);
    double abs_error=(x-measured_value);
    cout<<abs_error<<endl;
    double percentage_error=(r_error)*100;
    cout<<percentage_error<<endl;
    cout<<endl;

    ///question 3
    cout<<"Answer 3:"<<endl;
    x=8.6;
    double del_x;
    del_x=0.5*(pow(10,-1));
    double Er=del_x/x;
    printf("%.4lf",Er);
}
```

Output:

```
Answer 1:  
-0.3  
-0.0020562  
  
Answer 2:  
0.20265  
0.42  
  
Answer 3:  
0.0058
```

Conclusion: In this lab, we implement error calculation. We learned how to code of these kind of problems. And we hope all the outputs are correct.

1. Calculate the roots of the following equation using Bisection Method considering error up to 0.001: $x^3 - x = 4$

2. Calculate the first 10 roots of the following equation using Bisection Method: $f(x) = x^3 - 3x - 5 = 0$

Ans:

Hand-note:

1. ~~$f(x) = x^3 - x - 4$~~ $x^3 - x = 4$

Here, $x^3 - x - 4 = 0$

Let $f(x) = x^3 - x - 4$

$x = 1$, $f(x) = 1^3 - 1 - 4 = -4$

$x = 2$, $f(x) = 2^3 - 2 - 4 = 2$

1st iteration,

Here, $f(1) = -4 < 0$ and $f(2) = 2 > 0$

Now root lies between 1 and 2.

$$c = \frac{1+2}{2} = 1.5$$

$$f(c) = f(1.5) = (1.5)^3 - 1.5 - 4 = -2.125 < 0$$

2nd iteration,

Here, $f(1.5) = -2.125 < 0$ and $f(2) = 2 > 0$

Root lies between 1.5 and 2

$$c_1 = \frac{1.5+2}{2} = 1.75$$

$$f(c) = (1.75)^3 - 1.75 - 4 = -0.3906 < 0$$

iteration	a	b	c	f(c)	Error
1	1	2	1.5	-2.125	—
2	1.5	2	1.75	-0.3906	0.25
3	1.75	2	1.875	0.7168	0.125
4	1.75	1.875	1.8125	0.1418	0.062
5	1.75	1.812	1.7812	-0.1296	0.031
6	1.7812	1.8125	1.7969	0.0048	0.016
7	1.7812	1.7969	1.7891	-0.0627	0.008
8	1.7891	1.7969	1.793	-0.029	0.004
9	1.793	1.7969	1.7949	-0.0121	0.002
10	1.7949	1.7969	1.7959	-0.0037	0.001

So, the root is = 1.7959

Code-1:

```
#include<bits/stdc++.h>
using namespace std;
double f(double x){
    return pow(x,3)-x-4;
}
int main(){
    long long int i=1;
    double a,b,c;
level:
    cout<<"Enter two guessing number: ";
    cin>>a>>b;
    if(f(a)*f(b)>0){
        goto level;
    }
    double x1=0,x2,Er;
    cout<<"iteration\t a\t b\t c\t f(c)\t Error\n";
    while(1){
        c=(a+b)/(2*1.0);
        cout<<" "<<i<<"\t\t";
        x2=c;
        printf("%.4lf\t%.4lf\t%.4lf",a,b,c);
        Er=abs(x1-x2)/(x2*1.0);
        printf("\t%.4lf\t%.3lf\n",f(c),Er);
        if(Er<=0.001)
            break;
        if(f(a)*f(c)<0){
            b=c;
        }
        else{
            a=c;
        }
        x1=x2;
        i++;
    }
    cout<<"root is : "<<c<<endl;
}
```


Output:

```
Enter two guessing number: 1 2
iteration    a        b        c        f(c)    Error
1           1.0000    2.0000    1.5000   -2.1250    1.000
2           1.5000    2.0000    1.7500   -0.3906    0.143
3           1.7500    2.0000    1.8750    0.7168    0.067
4           1.7500    1.8750    1.8125    0.1418    0.034
5           1.7500    1.8125    1.7813   -0.1296    0.018
6           1.7813    1.8125    1.7969    0.0048    0.009
7           1.7813    1.7969    1.7891   -0.0627    0.004
8           1.7891    1.7969    1.7930   -0.0290    0.002
9           1.7930    1.7969    1.7949   -0.0121    0.001
10          1.7949    1.7969    1.7959   -0.0037    0.001
root is : 1.7959
```

Code-2:

```
#include<bits/stdc++.h>
using namespace std;
double f(double v)
{
    return pow(v,3)-3*v-5;
}
int main()
{
    long long int i;
level:
    double a,b,c;
    cout<<"Enter two guessing number:";
    cin>>a>>b;
    if(f(a)*f(b)>0)
    {
        goto level;
    }
    double x1,x2,Er;
    cout<<"iteration\t a\t b\t c\t f(c)\n";
    for(i=1; i<=10; i++)
    {
        c=(a+b)/(2*1.0);
        cout<<"<i<<"\t\t";
        printf("%.4lf\t%.4lf\t%.4lf",a,b,c);
        x2=c;
```

```

printf("\t%.4lf\n",f(c));
if(f(a)*f(c)<0)
{
    b=c;
}
else
{
    a=c;
}
}
cout<<"root is:"<<c<<endl;
}

```

Output:

```

Enter two guessing number:2 3
iteration      a          b          c          f(c)
1             2.0000    3.0000    2.5000    3.1250
2             2.0000    2.5000    2.2500   -0.3594
3             2.2500    2.5000    2.3750    1.2715
4             2.2500    2.3750    2.3125    0.4290
5             2.2500    2.3125    2.2813    0.0281
6             2.2500    2.2813    2.2656   -0.1673
7             2.2656    2.2813    2.2734   -0.0700
8             2.2734    2.2813    2.2773   -0.0211
9             2.2773    2.2813    2.2793    0.0035
10            2.2773    2.2793    2.2783   -0.0088
root is:2.27832

```

Conclusion: In this lab, we implement Bisection method which is a root finding method. It is easier to implement. For the first question we also calculate the absolute error which is considered until 0.001. And then finally we get the root of the equation. In the second question we calculate the first 10 roots of the equation. We hope all the values are correct.

1. Calculate the roots of the following equation using False Position Method that correct to three decimal places with the interval of $[0,0.5]$: $f(m)=4*e^{(-m)}*sin(m)-1=0$

2. Calculate the interval of the above equation.

Ans:

Code-1:

```
#include<bits/stdc++.h>
using namespace std;
double f(double v)
{
    return 4*exp(-v)*sin(v)-1;
}
int main()
{
    long long int i=1;
    double a,b,c;
level:
    cout<<"Enter two guessing number: ";
    cin>>a>>b;
    if(f(a)*f(b)>0)
    {
        goto level;
    }
    double x1=0,x2,Er;
    cout<<"iteration\ta\tb\tc\tf(c)\tError\n";
    while(1)
    {
        c=(a*f(b)-b*f(a))/((f(b)-f(a))*(1.0));
        cout<<" "<<i<<"\t\t";
        x2=c;
        Er=abs(x1-x2);
        printf("%.4lf\t%.4lf\t%.4lf",a,b,c);

        if(Er==abs(c))
        {
            printf("\t%.4lf  NULL\n",f(c));
        }
        else
        {
            printf("\t%.4lf  %.3lf\n",f(c),Er);
        }
    }
```

```

    if(Er<=0.001)
        break;

    if(f(a)*f(c)<0)
    {
        b=c;
    }
    else
    {
        a=c;
    }
    x1=x2;
    i++;
}
cout<<"root is : "<<c<<endl;
}

```

Output:

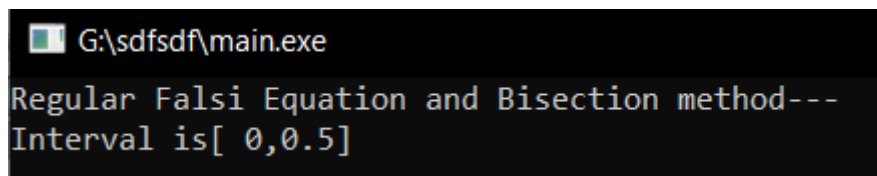
```

Enter two guessing number: 0 0.5
iteration      a      b      c      f(c)      Error
1      0.0000  0.5000  0.429869  0.0845     NULL
2      0.0000  0.4299  0.396359  0.0389     0.034
3      0.0000  0.3964  0.381512  0.0169     0.015
4      0.0000  0.3815  0.375159  0.0072     0.006
5      0.0000  0.3752  0.372482  0.0030     0.003
6      0.0000  0.3725  0.371361  0.0013     0.001
7      0.0000  0.3714  0.370893  0.0005     0.000
root is : 0.370893

```

Code-2:

```
#include<bits/stdc++.h>
using namespace std;
double f(double x)
{
    return 4* exp(-x) * sin(x)-1;
}
int main()
{
    double a=-1.5, b=-1;
    cout<<"Regular Falsi Equation and Bisection method---"<<endl;
    while(1)
    {
        if (f(a)*f(b)<0)
        {
            cout<<"Interval is[ "<<a<<","<<b<<"]"<<endl;
            break;
        }
        a=a+0.5, b=b+0.5;
    }
}
```

Output:

```
G:\sdfsdf\main.exe
Regular Falsi Equation and Bisection method---
Interval is[ 0,0.5]
```

Conclusion: In this lab, we find the root of the given equation by using false position method. And In the second question we find the interval of the given equation. We hope all the Outputs are correct.

1. Calculate the roots of the following equation using Newton-Raphson Method that correct to three decimal places with the starting root $a_0=2$. $f(a)=a^3-2a-5=0$

2. Calculate the first three roots of the following equation using Newton-Raphson Method with the starting value π . $f(x)=x\sin(x)+\cos(x)=0$

Ans:

Code-1:

```
#include<bits/stdc++.h>
using namespace std;
double f(double x)
{
    return pow(x,3)-2*x-5;
}
double derivative_f( double x)
{
    return 3*pow(x,2)-2;
}
int main()
{
    double x0,x,error;
    cout<<"Choose a value of x0= ";
    cin>>x0;
    int i=0;
    cout<<" n\t Xn\t f(Xn)\t Xn+1\tError"<<endl;

    while(1){

        x=x0-(f(x0)/(derivative_f(x0)*1.0));
        error = abs(x-x0);
        printf(" %d\t %.4lf %.9lf %.4lf\t%.4lf\n",i,x0,f(x0),x,error);
        if(error<=0.001) break;
        x0=x;
        i++;
    }

    printf("\n\n");
    printf("The root is : %.4lf\n",x);
}
```

Output:

```
Choose a value of x0= 2
n      Xn      f(Xn)      Xn+1      Error
0      2.0000  -1.000000000  2.1000  0.1000
1      2.1000  0.061000000  2.0946  0.0054
2      2.0946  0.000185723  2.0946  0.0000

The root is : 2.0946
```

Code-2:

```
#include<bits/stdc++.h>
using namespace std;
double f(double x)
{
    return x*sin(x)+cos(x);
}
double derivative_f( double x)
{
    return x*cos(x)+sin(x)-sin(x);
}
int main()
{
    double x0,x;
    cout<<"Choose a value of x0= ";
    cin>>x0;
    int i;
    cout<<" n \tXn\tf(Xn)\t Xn+1"<<endl;

    for(i=0;i<3;i++){

        x=x0-(f(x0)/(derivative_f(x0)*1.0));
        printf(" %d\t%.4lf %.9lf %.4lf\n",i,x0,f(x0),x);
        x0=x;

    }

    printf("\n\n");
    printf("The root is : %.4lf",x);
}
```

Output:

```
Choose a value of x0= 3.1416
n      Xn      f(Xn)      Xn+1
0      3.1416  -1.000023079  2.8233
1      2.8233  -0.066188065  2.7986
2      2.7986  -0.000563605  2.7984

The root is : 2.7984
```

Conclusion: In this lab, we find the root of the given equation by using Newton Raphson method. In the first question we calculate the root until we get the error to three decimal places. For error calculation we use absolute error. And For the second question we calculate the first three roots of the equation. And this method is so much faster than others method. We hope the outputs are correct.

1. Calculate the roots of the following equation using Iteration/Fixed Position Iteration Method that correct to three decimal places with $b_0=1$: $f(b)=\sin(b)-10b+10=0$

Ans:

Code

```
#include<bits/stdc++.h>
using namespace std;
double g(double x)
{
    return (sin(x)+10)/(10.0);
}
int main()
{
    int i=1;
    double n,x,error;
    cout<<"Choose the values of x0: ";
    cin>>n;

    cout<<"\n i\tXn\tXn+1\tError"<<endl;
    while(1)
    {
        x=g(n);
        error=abs(x-n);
        printf(" %d\t%.4f\t%.4f %%.4f\n",i,n,x,error);

        if(error<=0.001){
            break;
        }

        i++;
        n=x;
    }

    printf("\nThe root is : %.4f\n",x);
}
```

Output:

```
Choose the values of x0: 1

i      Xn      Xn+1    Error
1      1.0000  1.0841  0.0841
2      1.0841  1.0884  0.0042
3      1.0884  1.0886  0.0002

The root is : 1.0886
```

Conclusion: In this lab, we find the root of the given equation by using Iteration method. Here we first from the given equation $f(x)$ into $g(x)$ form. And here a point is given. Then we calculate the root from this given point with the accuracy up to three decimal places. We hope all the outputs are correct.

Consider the following table:

$x = 1 \ 2 \ 3 \ 4 \ 5$

$y = 3 \ 7 \ 13 \ 21 \ 31$

Q-1. Show the difference table.

Q-2. Find $y(7)$ using Newtons forward interpolation formulae.

Ans:

Code 1 and 2(dynamically):

```
#include<bits/stdc++.h>
using namespace std;
int factorial(int l)
{
    int val=1;
    for(int i=1; i<=l; i++){
        val*=i;
    }
    return val;
}
int main()
{
    long long int n;
    cout<<"Enter number of rows: ";
    cin>>n;

    double x[n],y[n][n];

    cout<<"x[] = ";
    for(int i=0; i<n; i++)
    {
        cin>>x[i];
    }
    cout<<"y[] = ";
    for(int i=0; i<n; i++)
    {
        cin>>y[i][0];
    }

    cout<<"\n\nX\tY\t\t\t";
    int del=2;
    for(int i=1; i<n; i++)
    {
        if(i>=2)
        {
```

```

        cout<<"del^"<<del<<"\t";
        del++;
    }
    for(int j=0; j<n-i; j++)
    {
        y[j][i]=y[j+1][i-1]-y[j][i-1];
    }
}

```

```

cout<<endl;

```

```

for(int i=0; i<n; i++)
{
    cout<<x[i]<<"\t";
    for(int j=0; j<n-i; j++)
    {
        cout<<y[i][j]<<"\t";
    }
    cout<<endl;
}
cout<<endl;

```

```

double x1,p,h;
cout<<"Enter the value of x: ";
cin>>x1;
p=(x1-x[0])/(x[1]-x[0]);
h=(x[1]-x[0]);
cout<<"x0="<<x[0]<<endl;
cout<<"x="<<x1<<endl;
cout<<"h="<<h<<endl;
cout<<"p="<<p<<endl;

```

```

double sum=y[0][0],l;
for(int i=0; i<n-1; i++)
{
    l=p;

    for(int j=1; j<=i; j++)
    {
        l*=(p-j);
    }
    l=l/(factorial(i+1)*1.0);
    l*=y[0][i+1];
    sum+=l;
}

```

```

    cout<<"\n\nValue at "<<x1<<" is : "<<sum<<endl;
}

```

Output-1:

```

Enter number of rows: 5
x[]= 1 2 3 4 5
y[]= 3 7 13 21 31

X      Y      del      del^2      del^3      del^4
1      3      4      2      0      0
2      7      6      2      0
3      13     8      2
4      21     10
5      31

```

Output-2:

```

Enter the value of x: 7
x0=1
x=7
h=1
p=6

Value at 7 is :57

```

Conclusion: In this lab, we implement the newton forward interpolation method. For the first question we calculate the difference table for the given points. Then we print the difference table. And For the second question we take as input a value x to find the unknown value. We hope all the outputs are correct.

**Q-1. Find The best values of a_0 and a_1 if straight line $Y = a_0 + a_1 * X$ is fitted to the data (x_i, y_i) :
 (1,0.6), (2,2.4), (3,3.5), (4,4.8), (5,5.7)**

Ans:

Code

```
#include<bits/stdc++.h>
using namespace std;
int main()
{

    double x[5]={1,2,3,4,5};
    double y[5]={0.6,2.4,3.5,4.8,5.7};
    double sum_x=0,sum_y=0;
    int m=sizeof(x)/sizeof(x[0]);
    double x_square[m],sum_x_square=0.0;
    double xy[m],sum_xy=0;
    cout<<"\n X(i)\tY(i)\tX(i)^2\tX(i)*Y(i)"<<endl;
    for(int i=0;i<m;i++)
    {
        sum_x+=x[i];
        sum_y+=y[i];
        x_square[i]=pow(x[i],2);
        sum_x_square+=x_square[i];
        xy[i]=x[i]*y[i];
        sum_xy+=xy[i];
    }

    for(int i=0;i<m;i++)
    {
        cout<<" "<<x[i]<<"\t"<<y[i]<<"\t"<<x_square[i]<<"\t"<<xy[i]<<endl;
    }
    cout<<"-----"<<endl;
    cout<<" "<<sum_x<<"\t"<<sum_y<<"\t"<<sum_x_square<<"\t"<<sum_xy<<endl;
    double average_x=sum_x/(m*1.0);
    double average_y=sum_y/(m*1.0);

    double a1=((m*sum_xy)-(sum_x*sum_y))/(((m*sum_x_square)-pow(sum_x,2))*1.0);
    double a0=average_y-(a1*average_x);

    cout<<"\n\n\n a0 = "<<a0<<"\n a1 = "<<a1<<endl;

    double new_y[m];

    for(int i=0;i<m;i++)
    {
```

```

    new_y[i]=a0+(a1*x[i]);
}

cout<<"\n New points are given below----\n";
for(int i=0;i<m;i++)
{
    cout<<" (x"<<i+1<<","<<"y"<<i+1<<") = ("<<x[i]<<","<<new_y[i]<<)"<<endl;
}
}

```

Output:

X(i)	Y(i)	X(i)^2	X(i)*Y(i)
1	0.6	1	0.6
2	2.4	4	4.8
3	3.5	9	10.5
4	4.8	16	19.2
5	5.7	25	28.5

15	17	55	63.6

$a_0 = -0.38$
 $a_1 = 1.26$

New points are given below----
 $(x_1, y_1) = (1, 0.88)$
 $(x_2, y_2) = (2, 2.14)$
 $(x_3, y_3) = (3, 3.4)$
 $(x_4, y_4) = (4, 4.66)$
 $(x_5, y_5) = (5, 5.92)$

Conclusion: In this lab, we implement least square method for straight line. Here some point was given. Here we find the value of a_0 and a_1 . And we also find the new points. We hope the outputs are correct.

Q-1. Using the method of least squares, fit a curve of the form $y = \frac{x}{a+bx}$ to the following data.
(3, 7.148), (5, 10.231), (8, 13.509), (12, 16.434)

Ans:

Code

```
#include<bits/stdc++.h>
using namespace std;
double X[100],Y[100];

double transform_xy(double x[],double y[],int m){
    for(int i=0;i<m;i++){
        X[i]=1/(x[i]*1.0);
        Y[i]=1/(y[i]*1.0);
    }
}

int main()
{
    double x[4]={3,5,8,12};
    double y[4]={7.148,10.231,13.509,16.434};

    int m=sizeof(x)/sizeof(x[0]);

    double xy[m],sum_xy=0,x_square[m],sum_x_square=0.0,sum_x=0,sum_y=0;

    transform_xy(x,y,m);

    cout<<"\n X(i)\tY(i)\tX(i)^2\tX(i)*Y(i)"<<endl;
    for(int i=0;i<m;i++)
    {
        sum_x+=X[i];
        sum_y+=Y[i];

        x_square[i]=pow(X[i],2);
        sum_x_square+=x_square[i];

        xy[i]=X[i]*Y[i];
        sum_xy+=xy[i];
        cout<<"
"<<setprecision(3)<<X[i]<<"\t"<<setprecision(3)<<Y[i]<<"\t"<<x_square[i]<<"\t"<<xy[i]<<endl;
    }
    cout<<"-----"<<endl;
    cout<<" "<<sum_x<<"\t"<<sum_y<<"\t"<<sum_x_square<<"\t"<<sum_xy<<endl;

    double average_x=sum_x/(m*1.0);
    double average_y=sum_y/(m*1.0);
```



```

double a1=((m*sum_xy)-(sum_x*sum_y))/(((m*sum_x_square)-pow(sum_x,2))*1.0);
double a0=average_y-(a1*average_x);

double a=a1;
double b=a0;
cout<<"\n\n\n a = "<<a<<"\n b = "<<b<<endl;
cout<<" y = x / ("<<a<<" + "<<b<<"x)"<<endl;
}

```

Output:

```

X(i)    Y(i)    X(i)^2    X(i)*Y(i)
0.333    0.14    0.111    0.0466
0.2      0.0977    0.04     0.0195
0.125    0.074    0.0156    0.00925
0.0833    0.0608    0.00694    0.00507
-----
0.742    0.373    0.174    0.0805

a = 0.316
b = 0.0345
y = x / (0.316 + 0.0345x)

```

Conclusion: In this lab, we implement the least square method for non-linear equation. Here, we have to convert the non-linear equation into linear. So, we transform those terms into linear form. Then we calculate the same as linear equation and get the values of A0 and A1 then we re-transform the term to get the value of a and b.

Q-1. Certain corresponding values of x and $\log_{10} x$ are (300, 2.4771), (304, 2.4829), (305, 2.4843), (307, 2.4871). Find $\log_{10} 301$.

Ans:

Code

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n,i,j;
    double x_arr[n+1],y_arr[n+1],X;
    char c;
    cout<<"Enter number of terms: ";
    cin>>n;
    for(i=1;i<=n;i++){
        cout<<"x["<<i<<"] , y["<<i<<"] = ";
        cin>>x_arr[i]>>c>>y_arr[i];
    }

    cout<<"\n\nEnter the value of X: ";
    cin>>X;

    double upper=1,lower=1,sum=0,term=1;
    for(i=1;i<=n;i++){
        upper=1,lower=1;
        for(j=1;j<=n;j++){
            if(j!=i){
                upper*=(X-x_arr[j]);
                lower*=(x_arr[i]-x_arr[j]);
            }
        }
        term=(upper*(y_arr[i])/(lower*1.0));
        sum+=term;
    }
    cout<<"\nValue at X is "<<sum<<endl;
}
```

Output:

```
Enter number of terms: 4
x[1] , y[1] = 300 , 2.4771
x[2] , y[2] = 304 , 2.4829
x[3] , y[3] = 305 , 2.4843
x[4] , y[4] = 307 , 2.4871

Enter the value of X: 301

Value at X is 2.4786
```

Conclusion: In this lab, we implement the Lagrange's interpolation method to get the unknown value. Here the interval of x is uneven. But this method works for both even and uneven interval. We hope the outputs are correct.