ASSIGNMENT 03 SOLUTIONS (ALL IN ONE)

Creating a c file `interpolation.c` which contains the interpolation functions such as Lagrange method and Neville's methods.

In [ ]:

```
// it contains the interpolation functions
// (lagrange and neville methods)
// use #include"interpolation.c" in the program you wished to use this


//function to calculate Lagrange interpolated value
double lag(int n,double X[],double Y[],double x)
{
    // n=no of points (n-1=order of interpolation)
    double sum=0;
    int i,j;
    for(i=0;i<n;i++)
    {
        // initiating product part
        double prod=1;
        for(j=0;j<n;j++)
        {
            if(j!=i)
            prod=prod*(x-X[j])/(X[i]-X[j]);
        }
        sum=sum+prod*Y[i];
    }
    return sum;
}
//function to calculate neville interpolated value
double nev(int n,double X[],double Y[],double x)
{
    // n=no of points (n-1=order of interpolation)
    double Q[n][n];
    int i,j;
    // initialising null matrix
    for (i=0;i<n;i++)
    {
        for (j=0;j<n;j++)
        {
            Q[i][j] = 0.0;
        }
        Q[i][0] = Y[i]; //setting first column as Y(or P0) values
    }
    for (i=1;i<n;i++) //i=1 since first col is P0
    {
        for (j=1;j<=i;j++)
        {
            Q[i][j]=((x-X[i-j])*(Q[i][j-1])-(x-X[i])*(Q[i-1][j-1]))/(X[i]-X[i-j]
        }
    }
    return (Q[n-1][n-1]);
}
```

Now since we have written all the required functions in the `"interpolation.c"`, we'll use `#include"interpolation.c"` as a library.

How to use the functions?

- Lagrange Method: `lag(n,X,Y,x)`
- Neville's Method: `nev(n,X,Y,x)`

Where **X** and **Y** are the input arrays, **n** is the number of points to be used **(n-1=order of interpolation)** and **x** is the value where we want interpolated value.

[SHEET 1] PROBLEM 1:

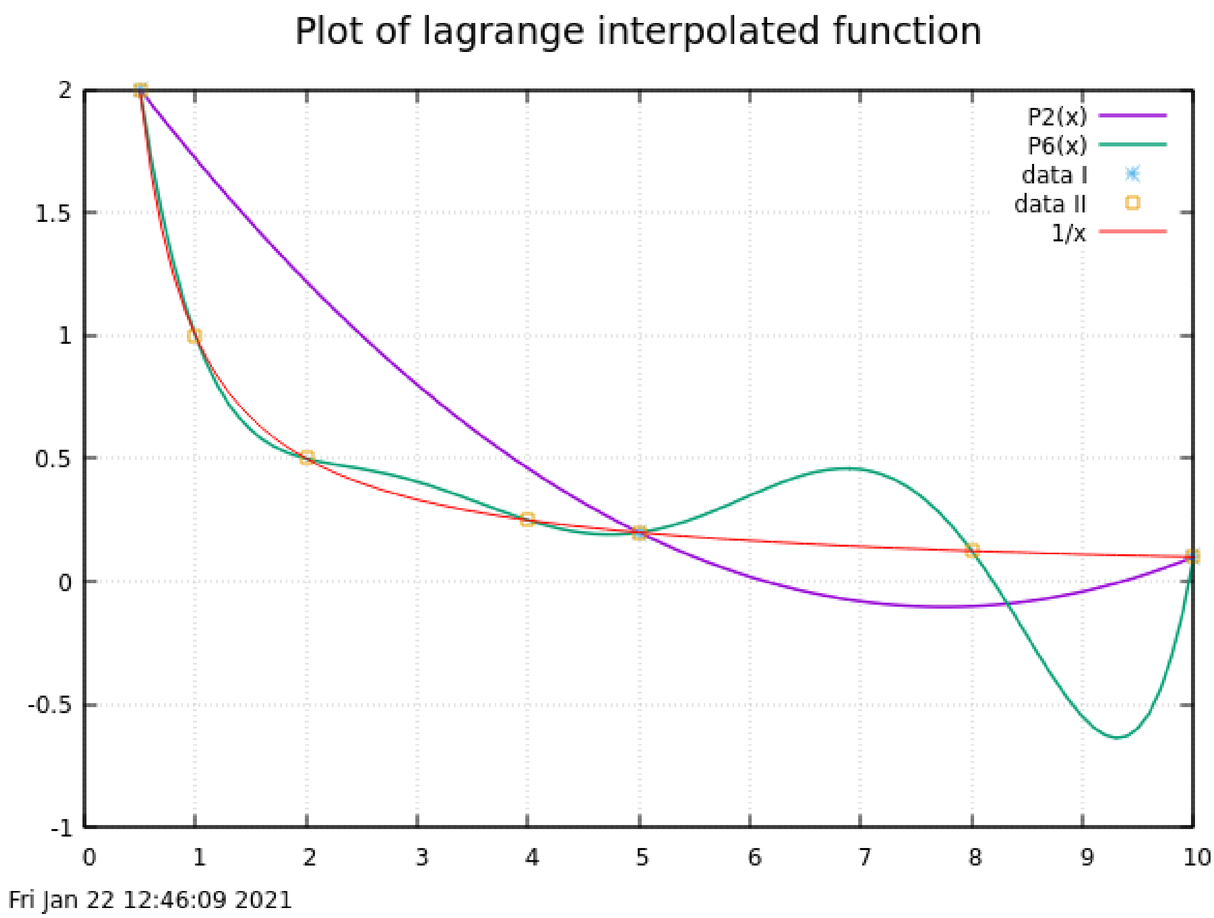
In [ ]:

```c
// sheet 1 problem 1
// make sure "interpolation.c" file is in the same directory
#include<stdio.h>
#include<math.h>
#include"interpolation.c"
int main()
{
    int i;
    FILE*fp=NULL;
    fp=fopen("sheet1.txt","w");
    // initialing array for table 1
        double X1[]={0.5,5,10};
        double Y1[]={2,0.2,0.1};
        // initialing array for table 2
        double X2[]={0.5,1,2,4,5,8,10};
        double Y2[]={2,1,0.5,0.25,0.2,0.125,0.1};
        double x=3;  //value of x for which interpolated value is required

    printf("For table I the interpolated value at x=3 is %lf",lag(3,X1,Y1,x));
    printf("\nFor table II the interpolated value at x=3 is %lf",lag(7,X2,Y2,x))

    for (x=0.5;x<=10;x+=0.1)
    {
        fprintf(fp,"%lf\t%lf\t%lf\t%lf\n",x,lag(3,X1,Y1,x),lag(7,X2,Y2,x),1/x);
    }
}
```

OUTPUT:

For table **I** the interpolated value at x=3 is 0.800000

For table **II** the interpolated value at x=3 is 0.406250



[SHEET 2] PROBLEM 1:

In [ ]:

```
// sheet 2 problem 1
// make sure "interpolation.c" file is in the same directory
#include<stdio.h>
#include<math.h>
#include"interpolation.c"
#define pi 3.141592

int main()
{
        int i,n=3;
        double x[]={1,1.25,1.6};
        double y[n],X=1.4;

        printf("[Part a] f(x)=sin(pi*x)\n");
        for(i=0;i<n;i++)
        {
                y[i]=sin(pi*x[i]);
        }
        printf("\nP2(x)=%f\n",lag(n,x,y,X));
        printf("with error:%f\n",fabs(lag(n,x,y,X)-sin(pi*X)));

        printf("-------------------------------------------\n");
        printf("\n[Part b] f(x)=pow(x-1,0.3333)\n");
        for(i=0;i<n;i++)
        {
                y[i]=pow(x[i]-1,0.3333);
        }
```

```
        printf("\nP2(x)=%f\n",lag(n,x,y,X));
        printf("with error:%f\n",fabs(lag(n,x,y,X)-pow(X-1,0.3333)));

        printf("-----------------------------------------\n");
        printf("\n[Part c] f(x)=log10(3*x-1)\n");
        for(i=0;i<n;i++)
        {
                y[i]=log10(3*x[i]-1);
        }
        printf("\nP2(x)=%f\n",lag(n,x,y,X));
        printf("with error:%f\n",fabs(lag(n,x,y,X)-log10(3*X-1)));

        printf("-----------------------------------------\n");
        printf("[Part d] f(x)=exp(2*x)-x\n");
        for(i=0;i<n;i++)
        {
        y[i]=exp(2*x[i])-x[i];
        }

        printf("\nP2(x)=%f\n",lag(n,x,y,X));
        printf("with error:%f\n",fabs(lag(n,x,y,X)-(exp(2*X)-X)));

}
```

OUTPUT:

[Part a] f(x)=sin(pi*x)

P2(x)=-0.918228
with error:0.032828

[Part b] f(x)=pow(x-1,0.3333)

P2(x)=0.816975
with error:0.080147

[Part c] f(x)=log10(3*x-1)

P2(x)=0.507122
with error:0.001972

[Part d] f(x)=exp(2*x)-x

P2(x)=15.269763
with error:0.225117

Same can be done to get the polynomial of order 1 by changing the n in the function.

[SHEET 2] PROBLEM 2:

In [ ]:
```
// sheet 2 problem 2
// make sure "interpolation.c" file is in the same directory
#include <stdio.h>
#include <math.h>
#include"interpolation.c"

int main()
```

```
{
        int i,n=4;
    double X=1.5,err1,err2;
        double x1[]={-2,-1,0,1,2};
        double x2[]={0,1,2,4,5};
    double a[n],b[n];
    for(i=0;i<n;i++)
    {
        a[i]=pow(3,x1[i]);
    }
    for(i=0;i<n;i++)
    {
        b[i]=pow(x2[i],0.5);
    }
        printf("Using Neville the approx. value for part (a): %lf and for part (
        // calculating the abs error
    err1=fabs(nev(n,x1,a,X)-pow(3,X));
    err2=fabs(nev(n,x2,b,X)-pow(X,0.5));
        printf("and the accuracy in part (a): %lf and in part (b): %lf\n",err1,e
}
```

OUTPUT:

Using Neville the approx. value for part (a) is :4.777778 and for part (b) is:1.256663

and the accuracy in part (a) is 0.418375 and in part (b) is0.031918

[SHEET 2] PROBLEM 3:

Problem-3 [marked as 22]

$x_j = j$ for $j = 0, 1, 2, 3$

$P_{0,1}(x) = x+1$

$P_{1,2}(x) = 3x-1$

$P_{1,2,3}(1.5) = 4$

find $P_{0123}(1.5) = ?$

0	P_0			
1	P_1	$P_{01}(1.5) = 2.5$		
2	P_2	$P_{12}(1.5) = 3.5$	$P_{012}(1.5) = 3.25$	
3	P_3	$P_{23}(1.5) = 5.5$	$P_{123}(1.5) = 4$	$P_{0123}(1.5) =$

$$P_{012}(1.5) = \frac{(1.5-2)P_{01}(1.5) - (1.5-0)P_{12}(1.5)}{0-2}$$

$$= 3.25$$

$$P_{0123}(1.5) = \frac{(1.5-3)P_{012}(1.5) - (1.5-0)P_{123}(1.5)}{6-3}$$

$$= 5.4375$$

[SHEET 2] PROBLEM 4:

Problem - 4 [marked as 26]

suppose $f \in C^1[a,b]$, $f'(x) \neq 0$ has one zero p

let $x_0, \ldots\ldots, x_n$ be $n+1$ distinct numbers in $[a,b]$

with $f(x_k) = y_k$

to approximate the root of function p construct the interpolating polynomial of degree n on the nodes $y_0, \ldots y_n$ for the function f^{-1}

since $y_k = f(x_k)$ and $0 = f(p)$

$\Rightarrow f^{-1}(y_k) = x_k$ and $f^{-1}(0) = p$

$$f(x) = x - e^{-x} = 0$$

x	0.3	0.4	0.5	0.6
e^{-x}	0.740818	0.670320	0.606531	0.548812

Using the code

the approx root is 0.567111

```
In [ ]: // sheet 2 problem 4
        // make sure "interpolation.c" file is in the same directory
        #include<stdio.h>
        #include<math.h>
        #include"interpolation.c"

        int main()
        {
            int i,n=3;
            double X=0,root[10];
            // initialing array for given table
```

```c
    double x[]={0.3,0.4,0.5,0.6};
    double ex[]={0.740818,0.670320,0.606531,0.548812};
    for(i=0;i<=n;i++)
    {
        root[i]=x[i]-ex[i];
    }
    printf("The approx. solution is: %f\n",nev(n,root,x,X));
}
```

OUTPUT:

The approx. solution is: 0.567111

[SHEET 2] PROBLEM 5:

Problem - 5 [marked as 27]

input numbers $x_0, x_1, x_2, \dots, x_n$

corresponding val $y_0, y_1, y_2 \dots, y_n$

as first coloumn $P_{00}, P_{10}, \dots$ of P

output the table P with P_{nn} approximating $f^{-1}(0)$

step: for $i = 1, 2, \dots n$

for $j = 1, 2, \dots i$

$$P_{ij} = \frac{x_j P_{i-1, j-1} - x_{j-1} P_{i, j-1}}{x_j - x_{i-j}}$$

[SHEET 3] PROBLEM 1 & 2:

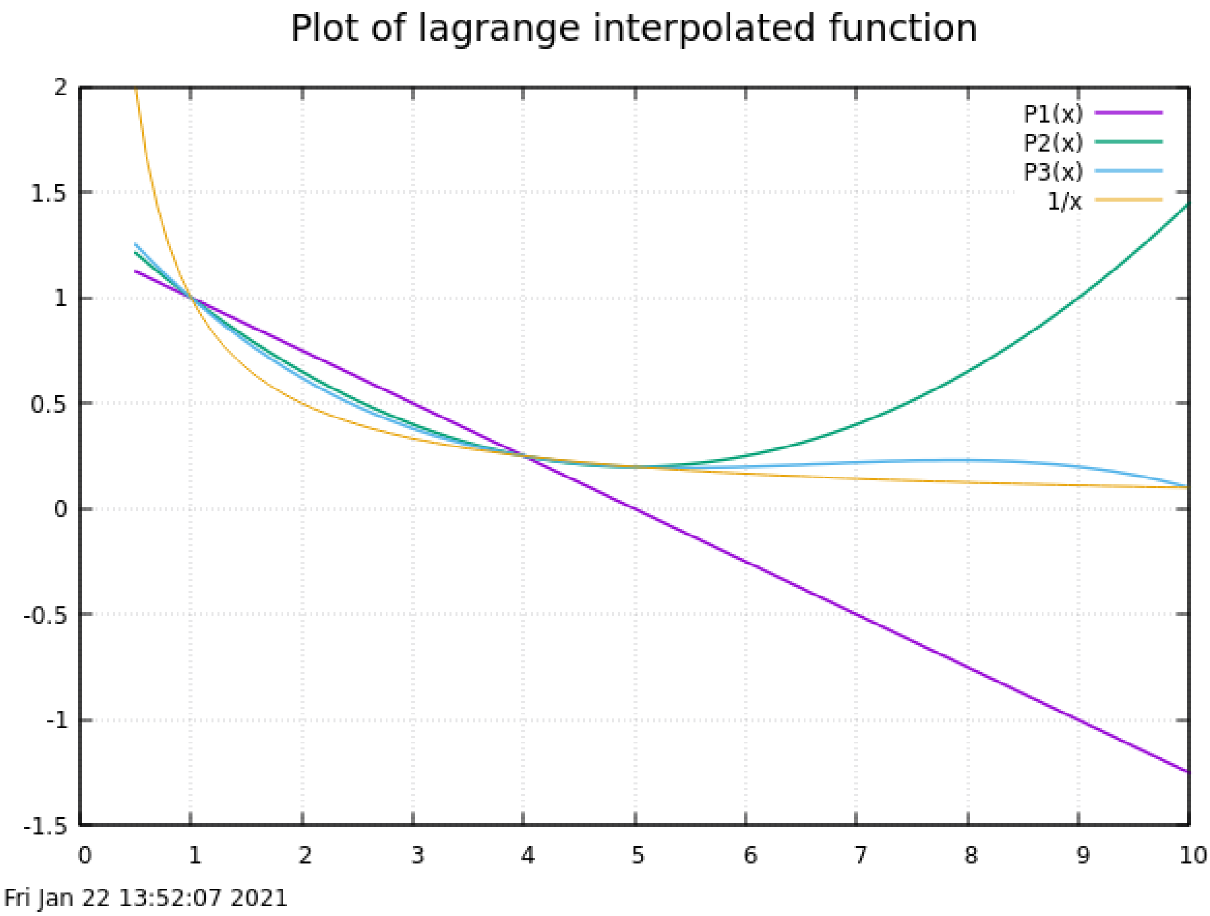
In [ ]:

```c
// sheet 3 problem 1 & 2
// make sure "interpolation.c" file is in the same directory
#include<stdio.h>
#include<math.h>
#include"interpolation.c"

int main()
{
    int i;
    double x;
    FILE*fp=NULL;
    FILE*fp1=NULL;
    fp=fopen("sheet3lag.txt","w");   // using lagrange
    fp1=fopen("sheet3nev.txt","w");  // using neville
    // initialing array for given table
    double X[]={1,4,5,10};
    double Y[]={1,0.25,0.2,0.1};
    // getting data to plot the function
    for (x=0.5;x<=10;x+=0.1)
```

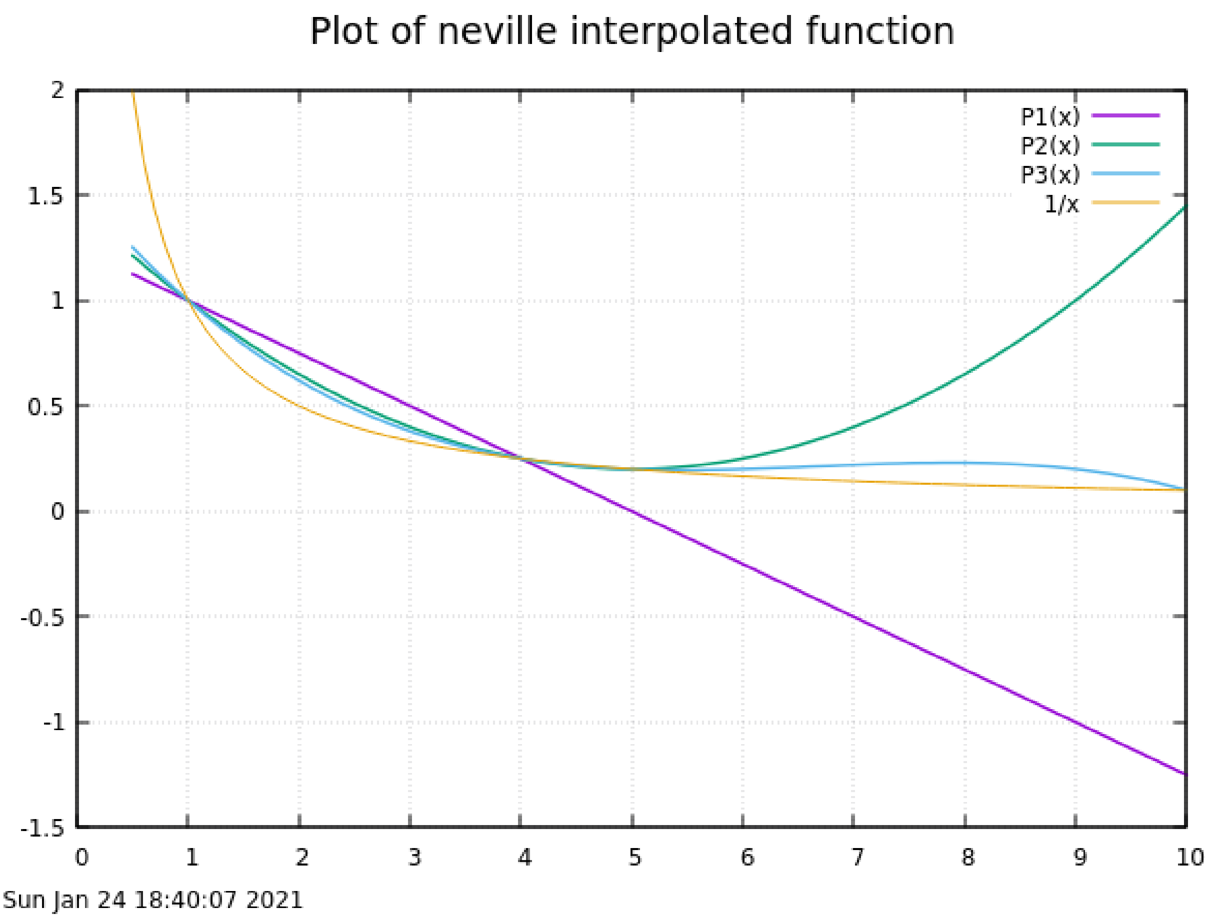
```
    {
        fprintf(fp,"%lf\t%lf\t%lf\t%lf\t%lf\n",x,lag(2,X,Y,x),lag(3,X,Y,x),lag(4
        fprintf(fp1,"%lf\t%lf\t%lf\t%lf\t%lf\n",x,nev(2,X,Y,x),nev(3,X,Y,x),nev(
    }
}
```

OUTPUT:

Plotting the datafile `sheet3lag.txt` which contains lagrange interpolated data.



Plotting the datafile `sheet3nev.txt` which contains neville interpolated data.



Since both the datafiles obtained using lagrange and neville methods are same which is visually represented in the above plots we can say Navilles polynomials are same as Lagrange interpolating polynomials which is obvious as we are using the lagrange polynomials to calculate neville polynomials.