**Informe métodos de interpolación Lagrange y Newton**

1.- Implementación en c++ método de lagrange

```
///Lagrange
/*Funcion que calcula los polinomios coeficientes de lagrange*/
float Lk(float* X,float x,int k,int n)
{
    float num=1,den=1;
    for(int i=0;i<n;i++)
        if(i!=k)
        {
            num*=x-X[i];
            den*=X[k]-X[i];
        }
    return num/den;
}
```

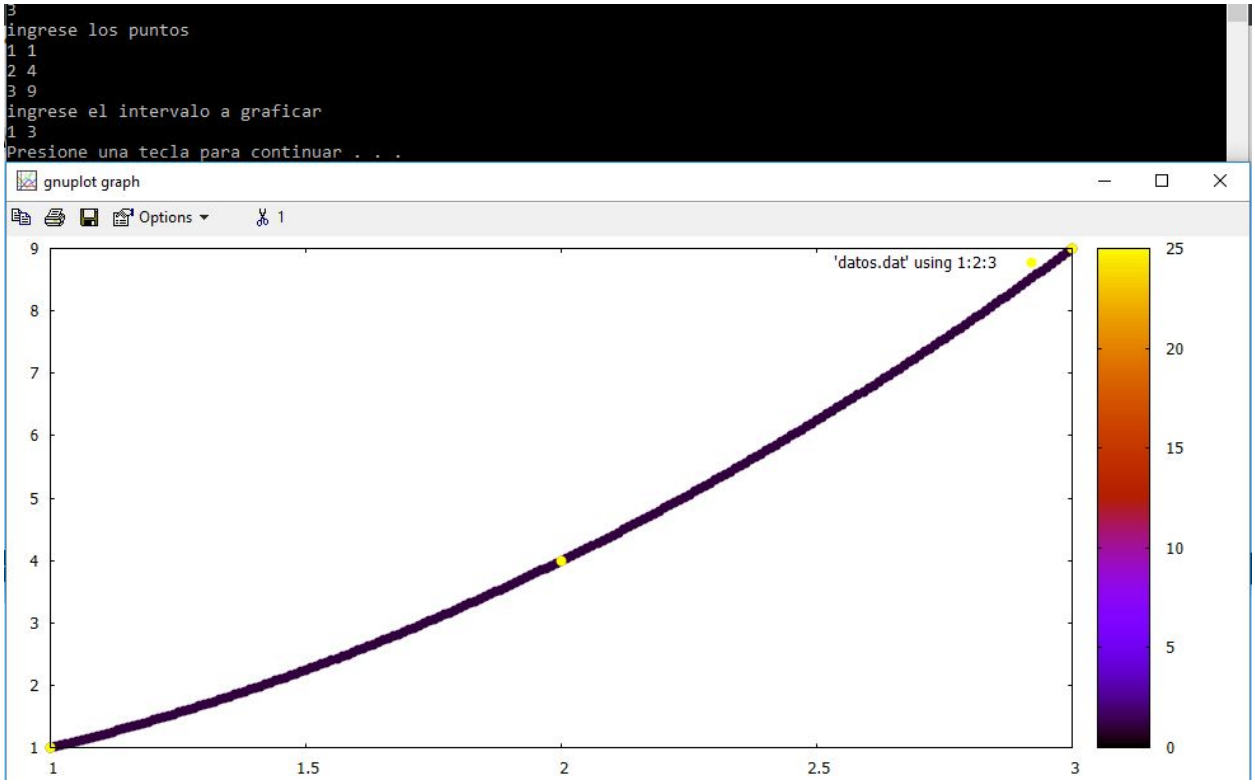
Función calcula los polinomios de lagrange y los evalúa dado un punto

Función que calcula la sumatoria de los polinomios de lagrange generados por la función definida anteriormente.

```
float lagrange(float* X,float* Y,int n,float x)
{
    float Yk=0;
    for(int i=0;i<n;i++)
        Yk+=Y[i]*Lk(X,x,i,n);
        //cout<<Yk<<" ";
    return Yk;
}
```

Función que grafica en un rango dado una función generada por lagrange y marca los puntos ingresados por el usuario

Ejemplo de ejecución:




2.- método Newton

Porción de la función que inicializa la tabla de diferencias divididas y crea la tabla.

```
FILE * Arch;
Arch=fopen("datos.dat","w+");
//tabla diferencias divididas
float ** dif_div=new float *[n]();
for(int i=0;i<n;i++)
    dif_div[i]=new float[n+1]();

for(int j=0;j<n;j++)
{
    dif_div[j][0]=x[j];
    dif_div[j][1]=y[j];
}
```

Porción de la función que llena la tabla de diferencias repetidas

```
int inc=1;
for(int c=2;c<n+1;c++)
{
    for(int f=c-1;f<n;f++)
    {
      dif_div[f][c]=(dif_div[f][c-1]-dif_div[f-1][c-1])/(dif_div[f][0]-dif_div[f-inc][0]);
      cout<<dif_div[f][c]<<" ";
    }
    inc++;
 }
 print_m(dif_div,n);
```

Porción de la función que genera los coeficientes del polinomio P(n)

```
 float *r=new float [n];
 for(int s=0;s<n;s++)
 {
     if(dif_div[s][s+1]!=0)
         r[s]=dif_div[s][s+1];
 }
```

Evaluando y generando un archivo con dichos puntos en un rango

```
float res;
float temp=1;
for(float num=a;num<=b;num+=0.01)
{
    res=r[0];
    for(unsigned int i=1;i<sizeof(r);i++)
    {
        for(unsigned int j=0;j<i;j++)
            temp*=(num-x[j]);
        temp=temp*r[i];
        res+=temp;
        temp=1;
    }
    fprintf(Arch,"%f %f %d\n",num,res,10);
    res=0;
    //fprintf(Arch,"%f %f %d \n",num,pow(num,3)-2*pow(num,2)-2,40);
}
```