ANT LAB Assignment 06 Least Square Fitting

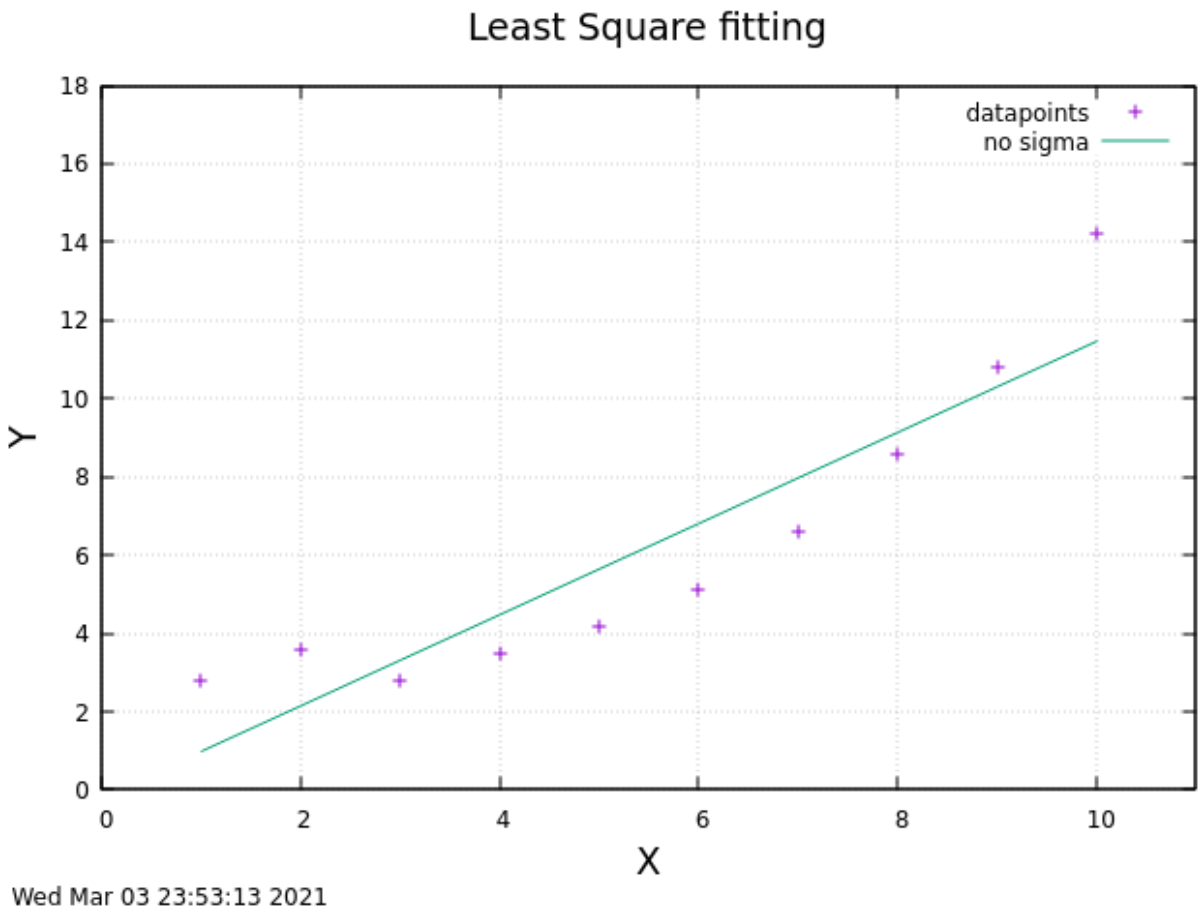
PROBLEM 1 : Case I (when no error is recorded or $\sigma_i = 0$)

In [ ]:

```c
// case I : no error or all sigma[i] are zeros
#include<stdio.h>
#include<math.h>

int main()
{
    int i,n=10; // no of points
    double x[]={1,2,3,4,5,6,7,8,9,10};
    double y[]={2.8,3.6,2.8,3.5,4.2,5.1,6.6,8.6,10.8,14.2};

    FILE*fp=NULL;
    fp=fopen("1a.txt","w");
    // summing
    double X=0,Y=0,XY=0,X2=0;
    for(i=0;i<n;i++)
    {
        X  += x[i];
        X2 += x[i]*x[i];
        Y  += y[i];
        XY += x[i]*y[i];
    }
    // expected values
    X  /= n;
    X2 /= n;
    Y  /= n;
    XY /= n;

    double a0,a1;   // unknowns to find
    a0=(Y*X2-X*XY)/(X2-X*X);
    a1=(XY-X*Y)/(X2-X*X);
    printf("The coefficients a0=%lf\ta1=%lf\n",a0,a1);
    // calculating the fitted points
    for (i=0;i<n;i++){
        y[i]=a0+a1*x[i];
        fprintf(fp,"%lf\t%lf\n",x[i],y[i]);
    }
}
```

OUTPUT:

The coefficients a0=-0.186667 a1=1.164848



PROBLEM 1 : Case II (when error is found to be uniform or $\sigma_i = \text{constants}$)

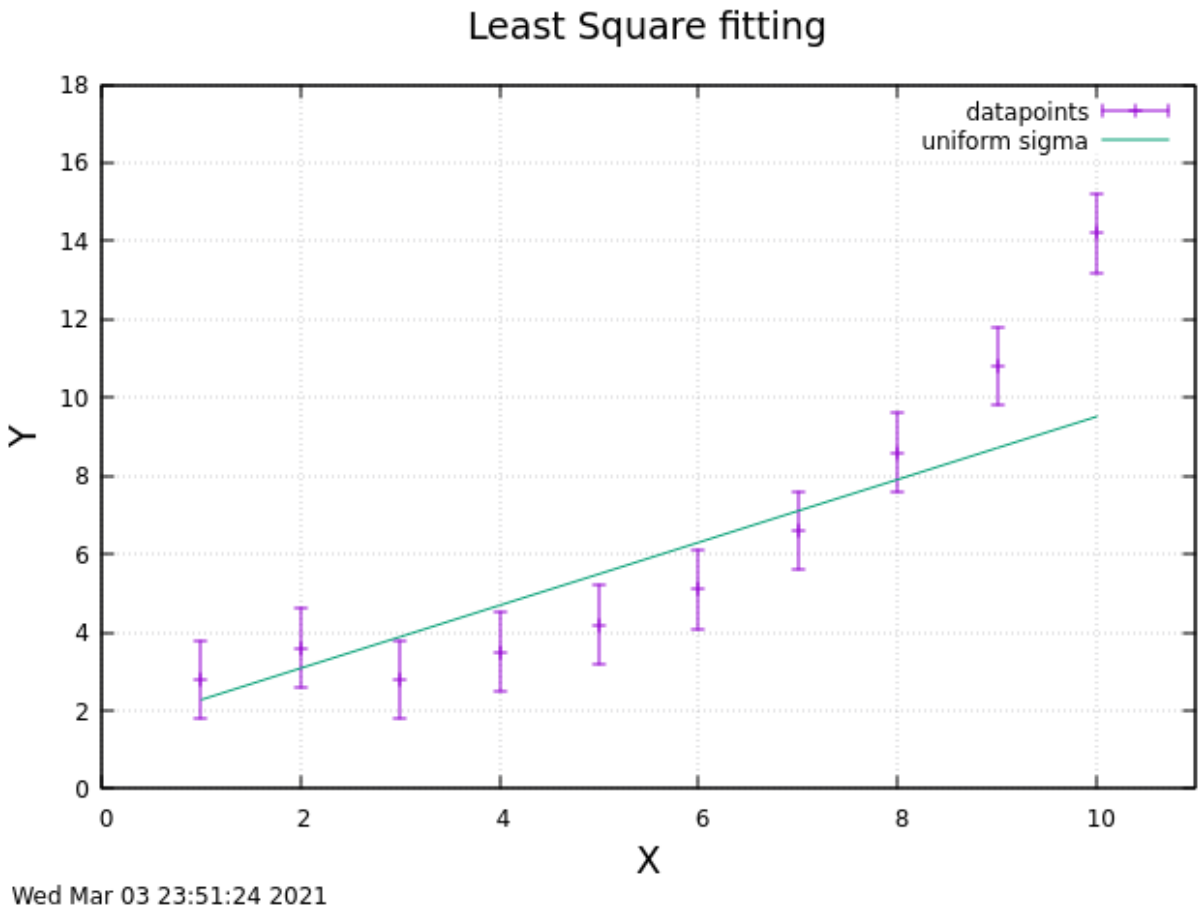
In [ ]:

```
// case II : uniform error or all sigma[i] are same
#include<stdio.h>
#include<math.h>

int main()
{
    int i,n=10; // no of points
    double x[]={1,2,3,4,5,6,7,8,9,10};
    double y[]={2.8,3.6,2.8,3.5,4.2,5.1,6.6,8.6,10.8,14.2};
    double sigma[]={0.3,0.5,0.55,0.6,0.65,0.7,0.75,0.9,1.1,1.3};

    FILE*fp=NULL;
    fp=fopen("1b.txt","w");

    // summing
    double weight,X=0,Y=0,XY=0,X2=0,w=0;
    for(i=0;i<n;i++)
    {
        weight=1/pow(sigma[i],2);
        X  += x[i]*weight;
        X2 += x[i]*x[i]*weight;
        Y  += y[i]*weight;
        XY += x[i]*y[i]*weight;
        w  += weight;
    }
    // expected values
```

```
    X  /= w;
    X2 /= w;
    Y  /= w;
    XY /= w;

    double a0,a1;   // unknowns to find
    a0=(Y*X2-X*XY)/(X2-X*X);
    a1=(XY-X*Y)/(X2-X*X);
    printf("The coefficients a0:%lf\ta1=%lf\n",a0,a1);
    // calculating the fitted points
    for (i=0;i<n;i++){
        y[i]=a0+a1*x[i];
        fprintf(fp,"%lf\t%lf\n",x[i],y[i]);
    }
}
```

OUTPUT:

The coefficients a0=1.470549 a1=0.804368



Wed Mar 03 23:51:24 2021

PROBLEM 1 : Case III (when error is found to be non-uniform or $\sigma_i = \text{varying}$)

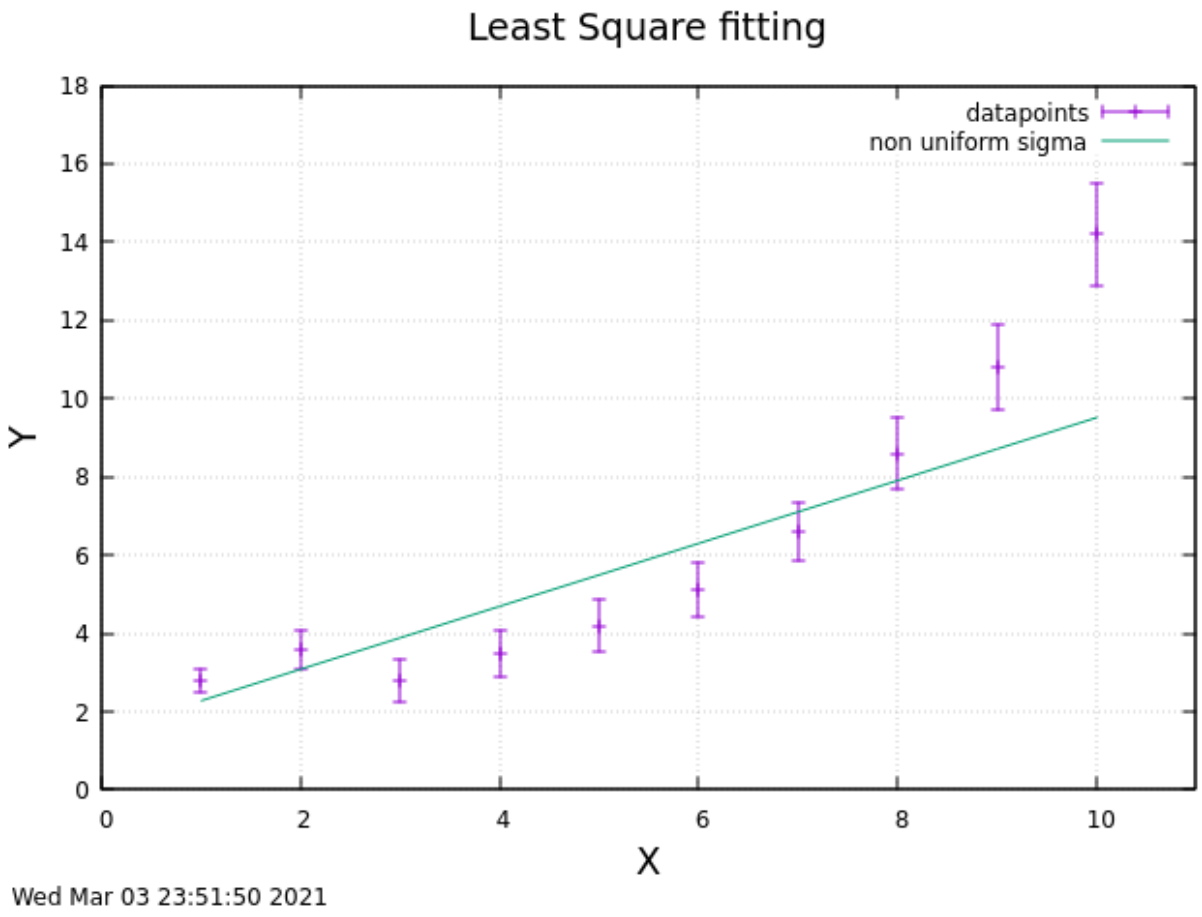
In [ ]:

```
// problem 4
// make sure "csplines.c" file is in the same directory
#include<stdio.h>
#include<math.h>
#include"csplines.c"
// main program to do our job
int main()
{
```

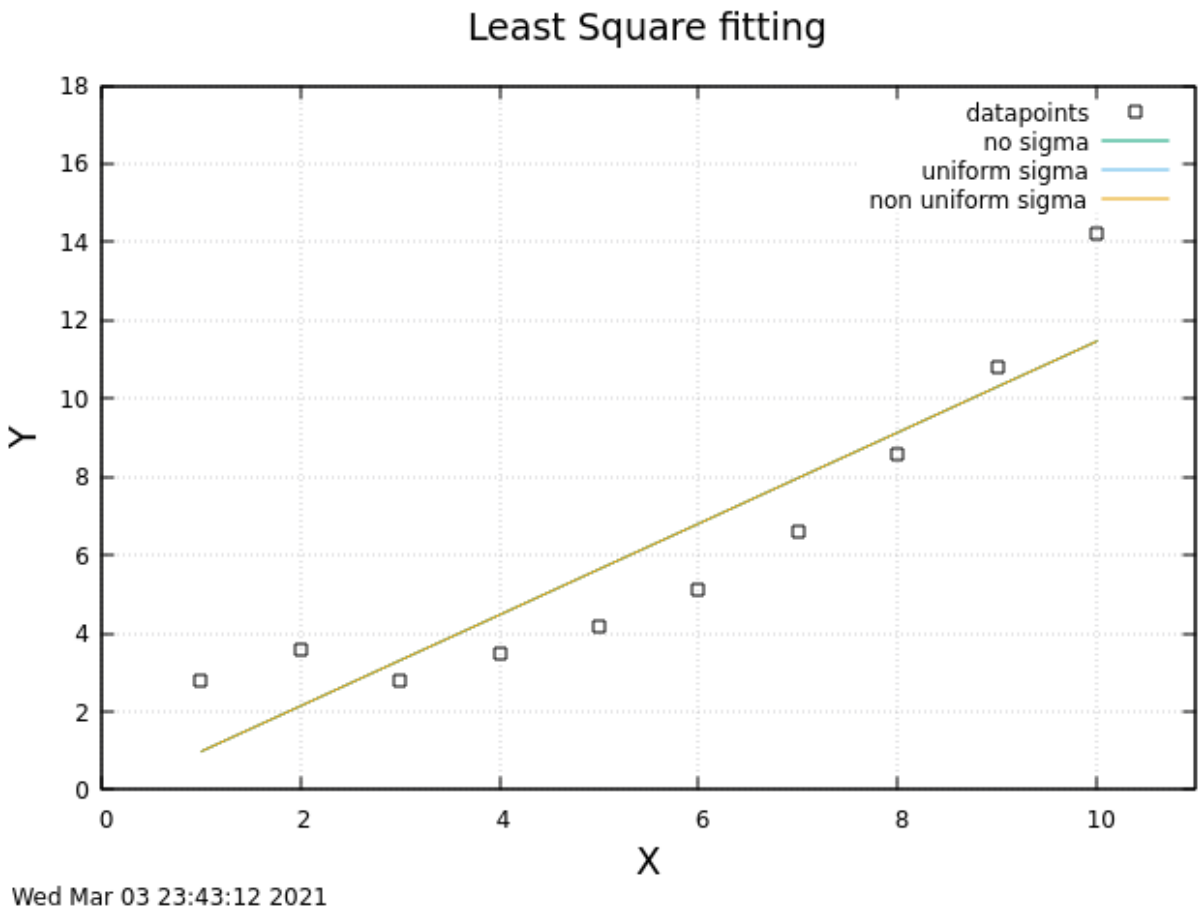
```
    double X,F0=1,F1=exp(3); // clamped conditions
    double x[]={0,1,2,3};
    double y[]={exp(0),exp(1),exp(2),exp(3)};
    FILE*fp=NULL;
    fp=fopen("clamp.txt","w");
    for (X=0;X<3;X+=0.01)
    {
        fprintf(fp,"%lf\t%lf\t%lf\n",X,clamped(4,x,y,F0,F1,X),exp(X));
    }
}
```

OUTPUT:

The coefficients a0=1.470549 a1=0.804368



Superimposed plot of the fitted data



PROBLEM 2 (i) : values of the coefficients and corresponding minimum chi-square

In [ ]:

```
#include<stdio.h>
#include<math.h>
// function to solve AX=b for X
void gausspivot(int n,double A[n][n+1],double x[]){
    int i,j,k;
    for(i=0;i<n-1;i++){
        //Partial Pivoting
        for(k=i+1;k<n;k++){
            //If the diagonal element is less than the terms below it
            if(fabs(A[i][i])<fabs(A[k][i])){
                //Swap the rows in the matrix
                for(j=0;j<=n;j++){
                    double temp;
                    temp=A[i][j];
                    A[i][j]=A[k][j];
                    A[k][j]=temp;
                }
            }
        }
        //Begin the Gauss Elimination
        for(k=i+1;k<n;k++){
            double term;
            term=A[k][i]/A[i][i];
            for(j=0;j<=n;j++){
                A[k][j]=A[k][j]-term*A[i][j];
            }
```

```
        }
    }
    //Start with the back-substitution
    for(i=n-1;i>=0;i--){
        x[i]=A[i][n];
        for(j=i+1;j<n;j++){
            x[i]=x[i]-A[i][j]*x[j];
        }
        x[i]=x[i]/A[i][i];
    }
    // printing the x array
    for(i=0;i<n;i++) {
        printf(" a[%d]= %.3lf\n",i,x[i]);
    }
}
int main()
{
    int n,N=19;   // no of datapoints
    // n is order of matrix or n-1 is the order of polynomial
    printf("enter the order of augmented matrix:");
    scanf("%d",&n);
    int i,j,k;
    double x[]={0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,
                0.1,0.11,0.12,0.13,0.14,0.15,0.16,0.17,0.18};
    double y[]={0.2,0.231895,0.264668,0.289191,0.332345,0.368007,
                0.403062,0.43739,0.472877,0.508413,0.543893,
                0.579221,0.614274,0.648984,0.683257,0.717008,
                0.717008,0.782653,0.814414};
    double A[n][n+1];
    // part A of augmented matrix [A:b]
    for(i=0;i<n;i++) {
        for(j=0;j<n;j++) {
            double sum=0;
            for(k=0;k<N;k++)
                sum+=pow(x[k],i+j);
            A[i][j]=sum/N;
        }
    }
    // part b of augmented matrix [A:b]
    for(i=0;i<n;i++) {
        double sum=0;
        for(k=0;k<N;k++)
            sum=sum+(pow(x[k],i)*y[k]);
        A[i][n]=sum/N;
    }
    double a[n],yf[N]; //yf[N] is fitted values
    // finding the coefficients
    printf("The coefficients for order [%d] are:\n",n-1);
    gausspivot(n,A,a);

    FILE*fp=NULL;
    fp=fopen("2.txt","w");
    // defining the polynomial
    k=0;
    while(k<N) {
        double sum=0;
        for (i=0;i<n;++i)
            sum += a[i]*pow(x[k],i);
        yf[k]=sum;
        fprintf(fp,"%.3lf\t%lf\n",x[k],yf[k]);
        k++;
```

```
    }
    // calculating the chi square
    double chi=0;
    for (i=0;i<N;++i)
        chi += (yf[i]-y[i])*(yf[i]-y[i]);
    printf("Chi square for order [%d]:%lf\n",n-1,chi);
}
```

OUTPUT:

The coefficients for order [1] are:

a[0]= 0.197

a[1]= 3.427

Chi square [1]:0.001149

The coefficients for order [2] are:

a[0]= 0.194

a[1]= 3.551

a[2]= -0.687

Chi square [2]:0.001085

The coefficients for order [3] are:

a[0]= 0.198

a[1]= 3.254

a[2]= 3.550

a[3]= -15.692

Chi square [3]:0.001009

The coefficients for order [4] are:

a[0]= 0.202

a[1]= 2.567

a[2]= 21.812

a[3]= -176.206

a[4]= 445.874

Chi square [4]:0.000876

The coefficients for order [5] are:

a[0]= 0.199

a[1]= 3.355

a[2]= -12.080

a[3]= 344.168

a[4]= -2844.548

a[5]= 7312.049

Chi square [5]:0.000799

The coefficients for order [6] are:

a[0]= 0.201

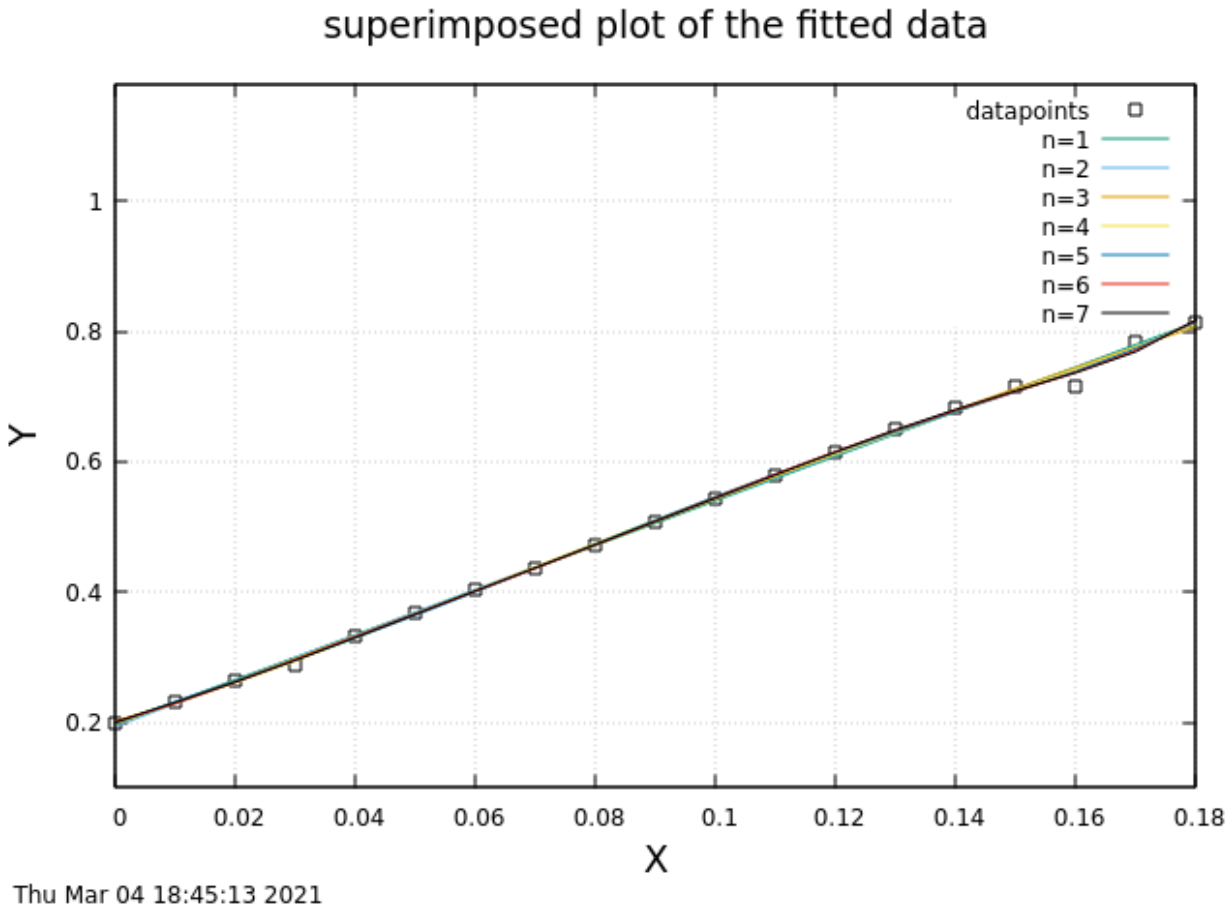
a[1]= 2.321

a[2]= 54.210

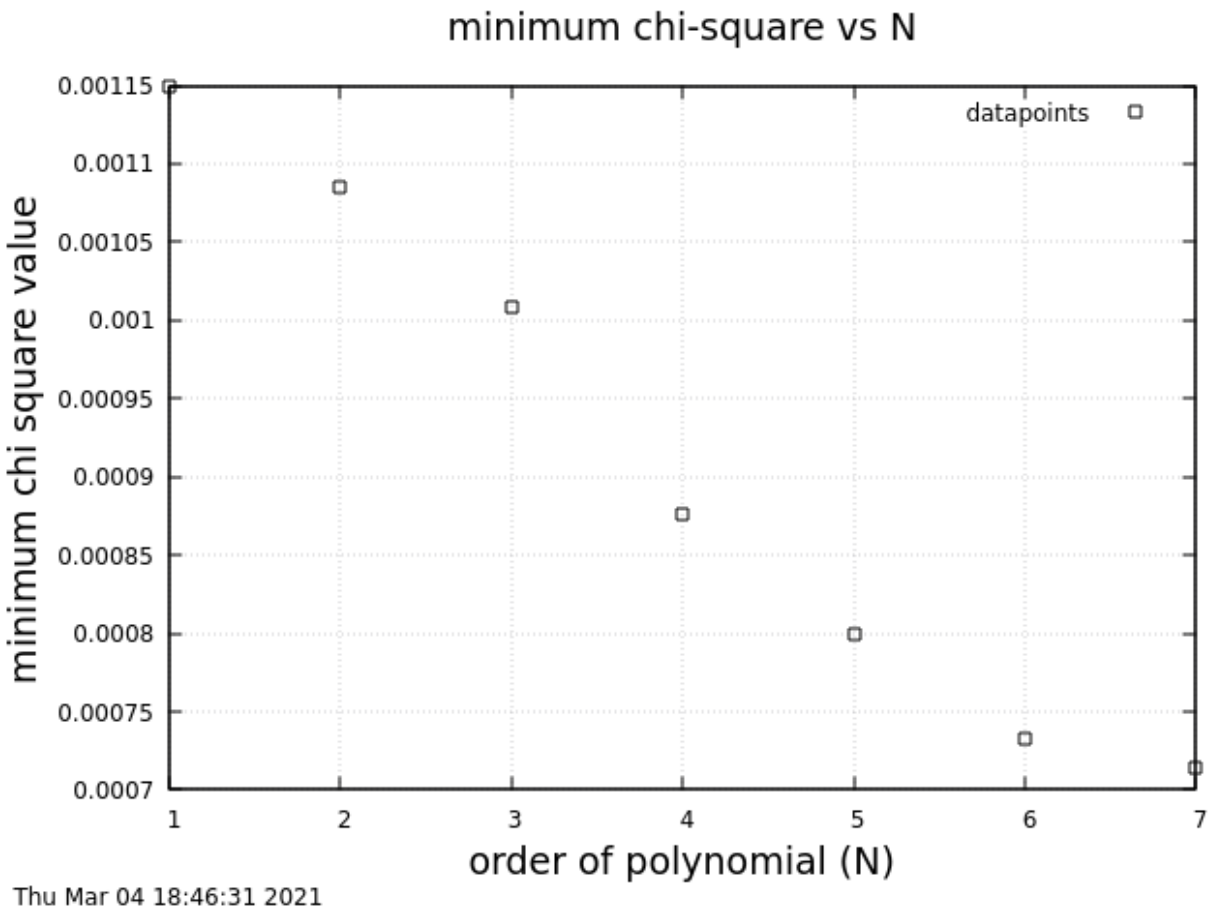
a[3]= -1211.315
a[4]= 13736.810
a[5]= -74425.102
a[6]= 151365.093
Chi square [6]:0.000733

The coefficients for order [7] are:
a[0]= 0.201
a[1]= 3.054
a[2]= -11.742
a[3]= 972.247
a[4]= -20732.018
a[5]= 205594.892
a[6]= -978760.443
a[7]= 1793850.059
Chi square [7]:0.000714

PROBLEM 2 (ii) : Superimposed plot of the data and the best-fit polynomials



PROBLEM 2 (iii) : Chi-square vs order of polynomial



Chi Square Minimization

This is an interesting problem. In the least square method what we have been doing so far is that we have a suffiencient amount of experimental dataset and we knew the theoritical nature of the function with some controlling parameters. Then using the standard matrix method we calculate the unknown parameters.

Here instead of using the typical matrix approach our approach will be minimizing the chi square value and finding the value of unkonwn corresponding to χ^2_{min} in some range.

But first using the typical matrix approach so that we can appreciate the Chi square approach.

In [ ]:
```c
#include<stdio.h>
#include<math.h>

int main()
{
    int i,n=7; // no of points
    double x[]={1,2,3,4,5,6,7};
    double y[]={4,5,8,16,30,38,70};
    double sigma[]={2,2,3,3,4,5,5};
    // for expontial fitting
    for(i=0;i<n;i++) {
        y[i]=log(y[i]);
    }
    // FILE*fp=NULL;
    // fp=fopen("3.txt","w");
```

```
    // summing
    double weight,X=0,Y=0,XY=0,X2=0,w=0;
    for(i=0;i<n;i++)
    {
        weight=1/pow(sigma[i],2);
        X  += x[i]*weight;
        X2 += x[i]*x[i]*weight;
        Y  += y[i]*weight;
        XY += x[i]*y[i]*weight;
        w  += weight;
    }
    // expected values
    X  /= w;
    X2 /= w;
    Y  /= w;
    XY /= w;

    double a0,a1;   // unknowns to find
    a0=(Y*X2-X*XY)/(X2-X*X);
    a1=(XY-X*Y)/(X2-X*X);
    a0=exp(a0);
    printf("The coefficients a0=%lf\ta1=%lf\n",a0,a1);
    // calculating the fitted points
    // for (i=0;i<n;i++){
    //     y[i]=a0+a1*x[i];
    //     fprintf(fp,"%lf\t%lf\n",x[i],y[i]);
    // }
}
```

OUTPUT :

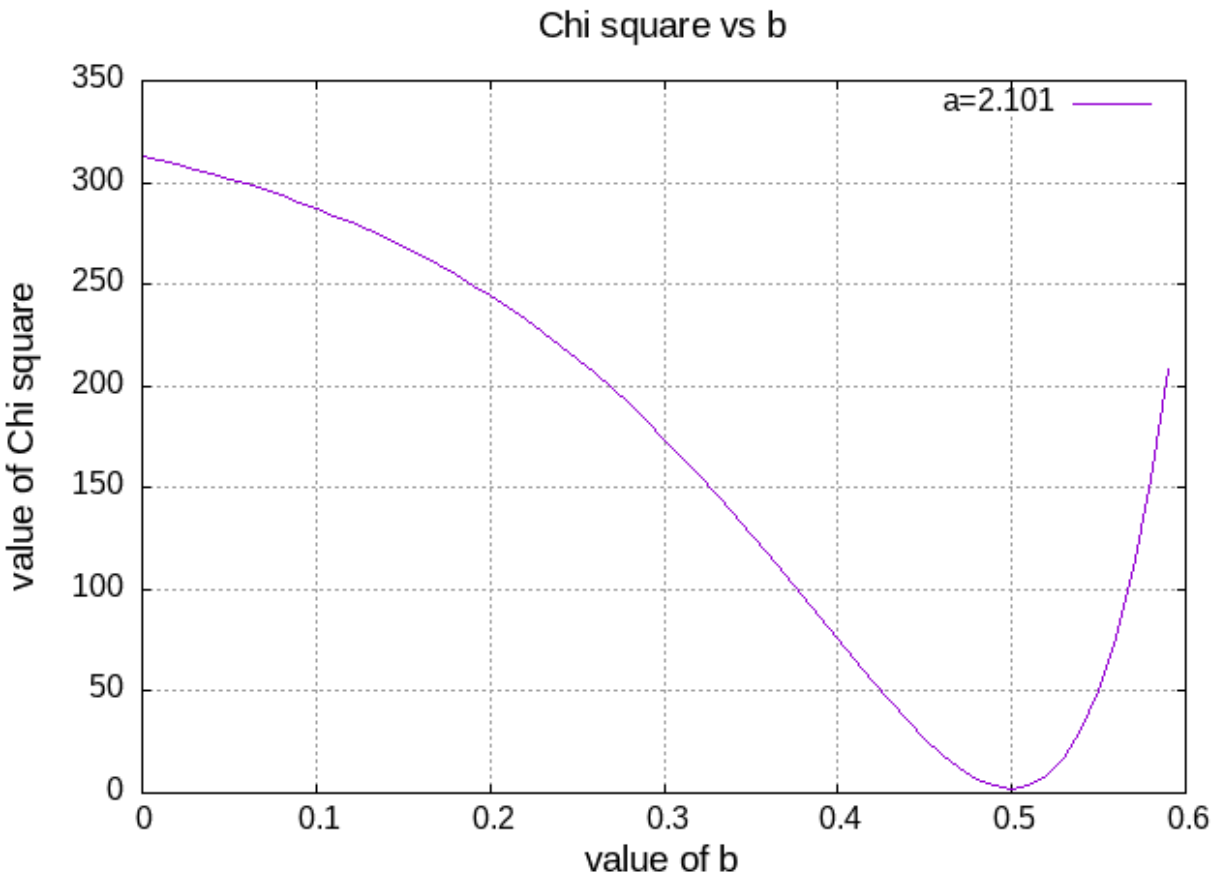
The coefficients a0=2.151576 a1=0.488720

PROBLEM 3 (a) :

In [ ]:

```
#include<stdio.h>
#include<math.h>

double x[]={1,2,3,4,5,6,7};
double y[]={4,5,8,16,30,38,70};
double sig[]={2,2,3,3,4,5,5};
double a=2.101;
int n=7;
// returning location of min value of an array
int point(int size,double array[])
{
  int index=0,i;
  double n;
  if (size!=1) {
    n=array[0];
    for (i=1;i<size;i++) {
        if (array[i]<n) {
            n=array[i];
            index=i;
        }
    }
  }
  return index;
}
// returning Chi square (X2) value
```

```c
double chisq(double b){
    double sum=0;
    int i;
    for(i=0;i<n;i++)
        sum=sum+pow((y[i]-a*exp(b*x[i]))/sig[i],2);
    return sum;
}

int main()
{
  double b,h=0.0001;   //earlier h=0.001

  // for plotting X2 vs b
  FILE*fp=NULL;
  fp=fopen("forb.txt","w");
  for (b=0;b<0.6;b+=0.01) {
    //printf("%lf\t%lf\n",b,chisq(b));
    fprintf(fp,"%lf\t%lf\n",b,chisq(b));
  }
  int steps=0;
  // calculating the total steps
  for (b=0.45;b<0.55;b+=h)
    steps++;
  // for minimum X2
  double chi[steps],B[steps];
  int i=0;
  for (b=0.45;b<0.55;b+=h) {
    B[i]=b;
    chi[i]=chisq(b);
    //printf("%d\t%.4lf\t%lf\n",i,B[i],chi[i]);
    i++;
  }
  // index of min chi square value
  int index=point(steps,chi);
  printf("value of minimum chi square: %lf\n",chi[index]);
  printf("for min Chi square and given a= %.3lf value of b= %.3lf\n",a,B[index])

  // for minimum X2+1
  for (int i=0;i<steps;++i) {
      if (fabs((chi[index]+1)-chi[i])<=0.011) {
          printf("for value of X2+1= %lf value of b= %.4lf\n",chi[i],B[i]);
      }
  }
  // for minimum X2+4
  for (int i=0;i<steps;++i) {
      if (fabs((chi[index]+4)-chi[i])<=0.01) {
          printf("for value of X2+4= %lf value of b= %.4lf\n",chi[i],B[i]);
      }
  }
  // for minimum X2+1
  for (int i=0;i<steps;++i) {
      if (fabs((chi[index]+9)-chi[i])<=0.03) {
          printf("for value of X2+9= %lf value of b= %.4lf\n",chi[i],B[i]);
      }
  }
}
```



Fri Mar 26 10:11:18 2021

OUTPUT :

value of minimum chi square: 2.371192

for min Chi square and given a= 2.101 value of b= 0.500

for value of X2+1= 3.360827 value of b= 0.4911

for value of X2+1= 3.369117 value of b= 0.5082

for value of X2+4= 6.370244 value of b= 0.4817

for value of X2+4= 6.363288 value of b= 0.5161

for value of X2+9= 11.392470 value of b= 0.4717

for value of X2+9= 11.344297 value of b= 0.5236

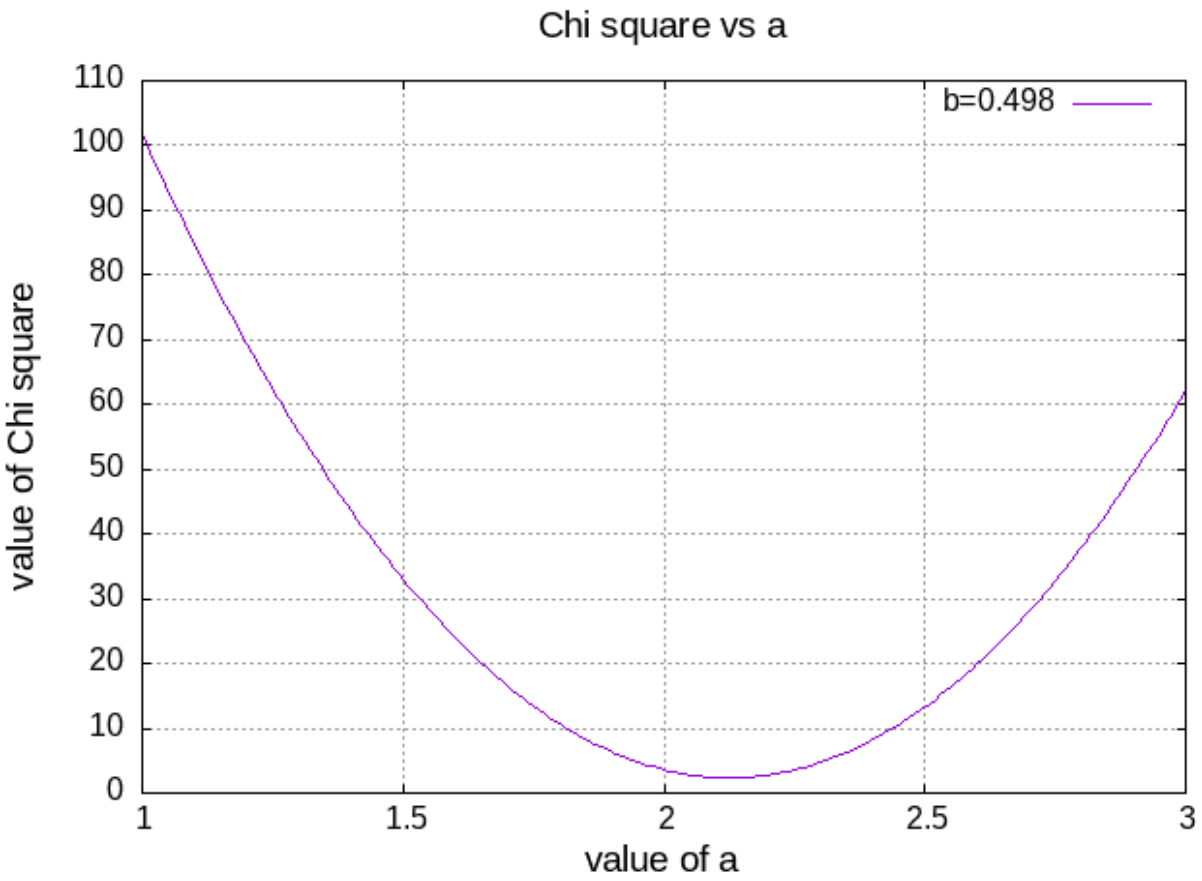
PROBLEM 3 (b) :

In [ ]:

```
#include<stdio.h>
#include<math.h>

double x[]={1,2,3,4,5,6,7};
double y[]={4,5,8,16,30,38,70};
double sig[]={2,2,3,3,4,5,5};
double b=0.498;
int n=7;
// returning location of min value of array
int point(int size,double array[])
{
  int index=0,i;
  double n;
  if (size!=1) {
    n=array[0];
```

```
    for (i=1;i<size;i++) {
        if (array[i]<n) {
            n=array[i];
            index=i;
        }
    }
  }
  return index;
}
// returning Chi square (X2) value
double chisq(double a){
    double sum=0;
    int i;
    for(i=0;i<n;i++)
        sum=sum+pow((y[i]-a*exp(b*x[i]))/sig[i],2);
    return sum;
}

int main()
{
  double a,h=0.001;

  // for plotting X2 vs a
  FILE*fp=NULL;
  fp=fopen("fora.txt","w");
  for (a=1;a<3;a+=0.01) {
    //printf("%lf\t%lf\n",a,chisq(b));
    fprintf(fp,"%lf\t%lf\n",a,chisq(a));
  }
  int steps=0;
  // calculating the total steps
  for (a=1;a<2.5;a+=0.001)
    steps++;
  // for minimum X2
  double chi[steps],A[steps];
  int i=0;
  for (a=1;a<2.5;a+=h) {
    A[i]=a;
    chi[i]=chisq(a);
    //printf("%d\t%.3lf\t%.3lf\n",i,A[i],chi[i]);
    i++;
  }
  // index of min chi square value
  int index=point(steps,chi);
  printf("value of minimum chi square: %lf\n",chi[index]);
  printf("for min Chi square and given b= %.3lf value of a= %.3lf\n",b,A[index])

  // for minimum X2+1
  for (int i=0;i<steps;++i) {
      if (fabs((chi[index]+1)-chi[i])<=0.01) {
          printf("for value of X2+1= %lf value of a= %.3lf\n",chi[i],A[i]);
      }
  }
  // for minimum X2+4
  for (int i=0;i<steps;++i) {
      if (fabs((chi[index]+4)-chi[i])<=0.015) {
          printf("for value of X2+4= %lf value of a= %.3lf\n",chi[i],A[i]);
      }
  }
  // for minimum X2+1
  for (int i=0;i<steps;++i) {
```

```
      if (fabs((chi[index]+9)-chi[i])<=0.025) {
          printf("for value of X2+9= %lf value of a= %.3lf\n",chi[i],A[i]);
      }
  }
}
```



Fri Mar 26 10:11:58 2021

OUTPUT :

value of minimum chi square: 2.368703

for min Chi square and given b= 0.498 value of a= 2.126

for value of X2+1= 3.370157 value of a= 2.013

for value of X2+1= 3.363949 value of a= 2.239

for value of X2+4= 6.368310 value of a= 1.900

for value of X2+4= 6.355894 value of a= 2.352

for value of X2+9= 11.363163 value of a= 1.787

for value of X2+9= 11.344539 value of a= 2.465

PROBLEM 3 (c) :

In [ ]:

```
#include<stdio.h>
#include<math.h>

double x[]={1,2,3,4,5,6,7};
double y[]={4,5,8,16,30,38,70};
double sig[]={2,2,3,3,4,5,5};
int n=7;
// returning location of min value of an array
int point(int size,double array[])
{
```

```
  int index=0,i;
  double n;
  if (size!=1) {
    n=array[0];
    for (i=1;i<size;i++) {
        if (array[i]<n) {
            n=array[i];
            index=i;
        }
    }
  }
  return index;
}
// returning Chi square (X2) value
double chisq(double a,double b){
    double sum=0;
    int i;
    for(i=0;i<n;i++){
        sum=sum+pow((y[i]-a*exp(b*x[i]))/sig[i],2);
    }
    return sum;
}
int main()
{
  double a,b;
  int steps=0;
  for (a=1;a<3;a+=0.01) {
      for (b=0;b<0.6;b+=0.01) {
        steps++;
      }
  }
  // for plotting X2 vs b
  FILE*fp=NULL;
  fp=fopen("forab.txt","w");

  double A[steps],B[steps],chi[steps];
  int i=0;
  for (a=1;a<3;a+=0.01) {
      for (b=0;b<0.6;b+=0.01) {
        A[i]=a;
        B[i]=b;
        chi[i]=chisq(a,b);
        //printf("%d\t%.3lf\t%.3lf\t%lf\n",i,A[i],B[i],chisq(a,b));
        fprintf(fp,"%.3lf\t%.3lf\t%lf\n",A[i],B[i],chisq(a,b));
        i++;
      }
  }
  // index of min chi square value
  int index=point(i,chi);
  printf("value of minimum chi square: %lf\n",chi[index]);
  printf("for min chi square, value of a= %.2lf\tb= %.2lf\n",A[index],B[index]);

  // for minimum X2+1
  for (int i=0;i<steps;++i) {
      if (fabs((chi[index]+1)-chi[i])<=0.009) {
          printf("for value of X2+1= %lf value a= %.2lf b= %.2lf\n",chi[i],A[i],
      }
  }
  // for minimum X2+4
  for (int i=0;i<steps;++i) {
      if (fabs((chi[index]+4)-chi[i])<=0.02) {
```

```
            printf("for value of X2+4= %lf value a= %.2lf b= %.2lf\n",chi[i],A[i],
        }
    }
    // for minimum X2+9
    for (int i=0;i<steps;++i) {
        if (fabs((chi[index]+9)-chi[i])<=0.02) {
            printf("for value of X2+9= %lf value a= %.2lf b= %.2lf\n",chi[i],A[i],
        }
    }
}
```

OUTPUT :

value of minimum chi square: 2.371418

for min chi square, value of a= 2.10 b= 0.50

for value of X2+1= 3.366575 value a= 1.69 b= 0.53

for value of X2+1= 3.363586 value a= 2.48 b= 0.48

for value of X2+4= 6.388617 value a= 1.67 b= 0.55

for value of X2+4= 6.359131 value a= 2.42 b= 0.46

~~for value of X2+4= 6.356855 value a= 2.91 b= 0.46~~

for value of X2+9= 11.388157 value a= 1.33 b= 0.59

~~for value of X2+9= 11.356366 value a= 2.29 b= 0.51~~

for value of X2+9= 11.380273 value a= 2.97 b= 0.42

Strike values are not acceptable because χ^2_{min} doesn't lie between them.

PROBLEM 4 (a) : for (σ_1) upto order 7

In [ ]:

```
#include<stdio.h>
#include<math.h>
// function to solve AX=b for X
void gausspivot(int n,double A[n][n+1],double x[]){
    int i,j,k;
    for(i=0;i<n-1;i++){
        //Partial Pivoting
        for(k=i+1;k<n;k++){
            //If the diagonal element is less than the terms below it
            if(fabs(A[i][i])<fabs(A[k][i])){
                //Swap the rows in the matrix
                for(j=0;j<=n;j++){
                    double temp;
                    temp=A[i][j];
                    A[i][j]=A[k][j];
                    A[k][j]=temp;
                }
            }
        }
        //Begin the Gauss Elimination
        for(k=i+1;k<n;k++){
            double term;
            term=A[k][i]/A[i][i];
            for(j=0;j<=n;j++){
                A[k][j]=A[k][j]-term*A[i][j];
            }
        }
    }
```

```
    //Start with the back-substitution
    for(i=n-1;i>=0;i--){
        x[i]=A[i][n];
        for(j=i+1;j<n;j++){
            x[i]=x[i]-A[i][j]*x[j];
        }
        x[i]=x[i]/A[i][i];
    }
    // printing the x array
    for(i=0;i<n;i++) {
        printf(" a[%d]= %.3lf\n",i,x[i]);
    }
}
int main()
{
    int n,N=11;   // no of datapoints
     // order of matrix or n-1 is the order of polynomial
    printf("enter the order of augmented matrix:");
    scanf("%d",&n);
    int i,j,k;
    double x[]={0,0.2,0.4,0.6,0.8,1,1.2,1.4,1.6,1.8,2};
    double y[]={6.33,6.51,6.43,5.85,4.71,3.13,1.53,0.64,1.58,5.91,15.71};
    double sigma[]={0.95,0.98,0.96,0.88,0.71,0.47,0.23,0.09,0.24,0.89,2.35};
    double A[n][n+1];
    // part A of augmented matrix [A:b]
    for(i=0;i<n;i++) {
        for(j=0;j<n;j++) {
            double weight,sum=0,w=0;
            for(k=0;k<N;k++) {
                weight=1/pow(sigma[k],2);
                sum+=pow(x[k],i+j)*weight;
                w+=weight;
            }
            A[i][j]=sum/w;
        }
    }
    // part b of augmented matrix [A:b]
    for(i=0;i<n;i++) {
        double weight,sum=0,w=0;
        for(k=0;k<N;k++) {
            weight=1/pow(sigma[k],2);
            sum=sum+(pow(x[k],i)*y[k]*weight);
            w+=weight;
        }
        A[i][n]=sum/w;
    }
    double a[n],yf[N]; //yf[N] is fitted values
    // finding the coefficients
    printf("The coefficients for order [%d] are:\n",n-1);
    gausspivot(n,A,a);

    FILE*fp=NULL;
    fp=fopen("4.txt","w");
    // defining the polynomial
    k=0;
    while(k<N) {
        double sum=0;
        for (i=0;i<n;++i)
            sum += a[i]*pow(x[k],i);
        yf[k]=sum;
        fprintf(fp,"%.3lf\t%lf\n",x[k],yf[k]);
```

```
        k++;
    }
    // calculating the chi square
    double chi=0;
    for (i=0;i<N;++i)
        chi += 1/pow(sigma[i],2)*(yf[i]-y[i])*(yf[i]-y[i]);
    printf("Chi square for order [%d]:%lf\n",n-1,chi);
}
```

OUTPUT :

The coefficients for order [1] are:
a[0]= 5.846
a[1]= -3.466
Chi square [1]:165.419862

The coefficients for order [2] are:
a[0]= 9.590
a[1]= -12.657
a[2]= 4.620
Chi square [2]:110.920798

The coefficients for order [3] are:
a[0]= 5.154
a[1]= 17.113
a[2]= -31.924
a[3]= 12.439
Chi square [3]:12.564621

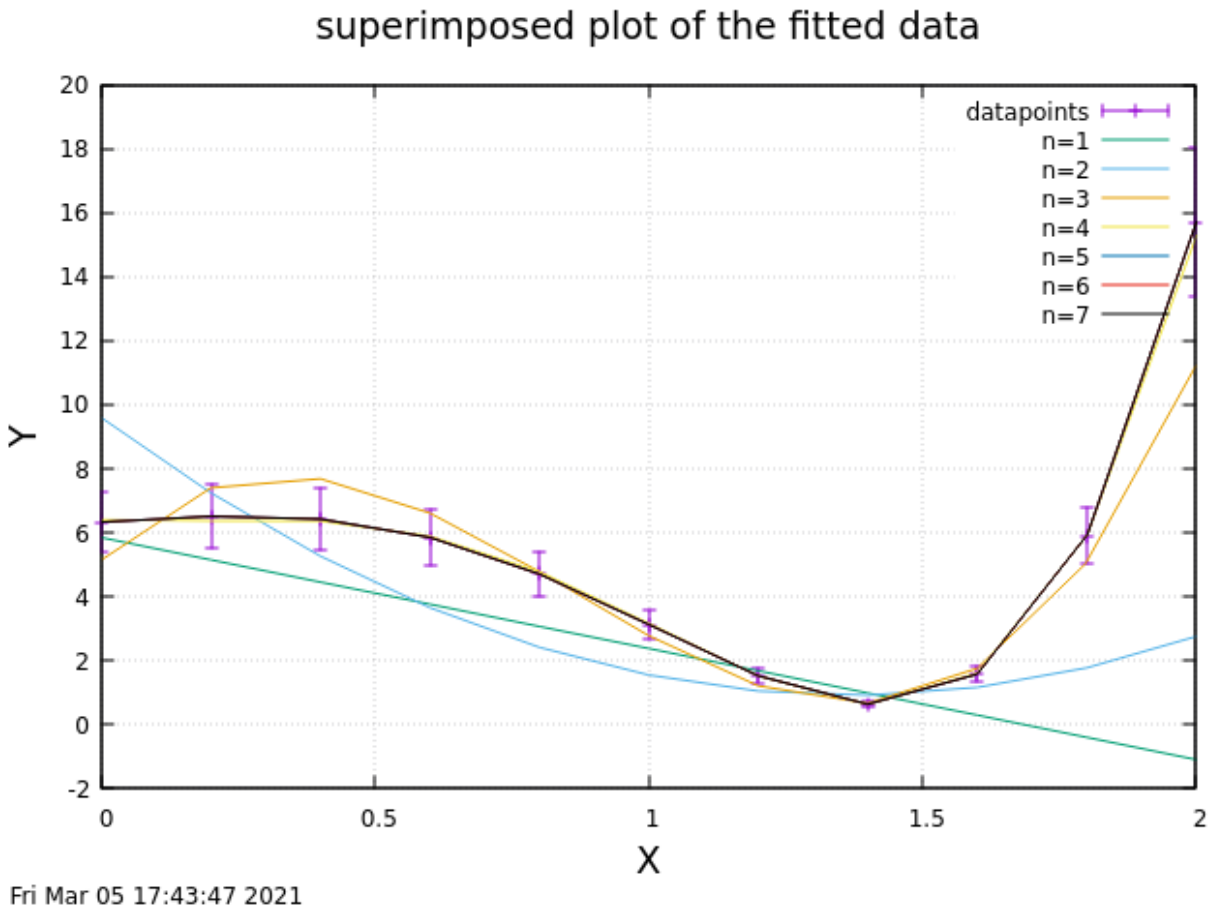
The coefficients for order [4] are:
a[0]= 6.412
a[1]= -1.604
a[2]= 10.616
a[3]= -20.703
a[4]= 8.452
Chi square [4]:0.116348

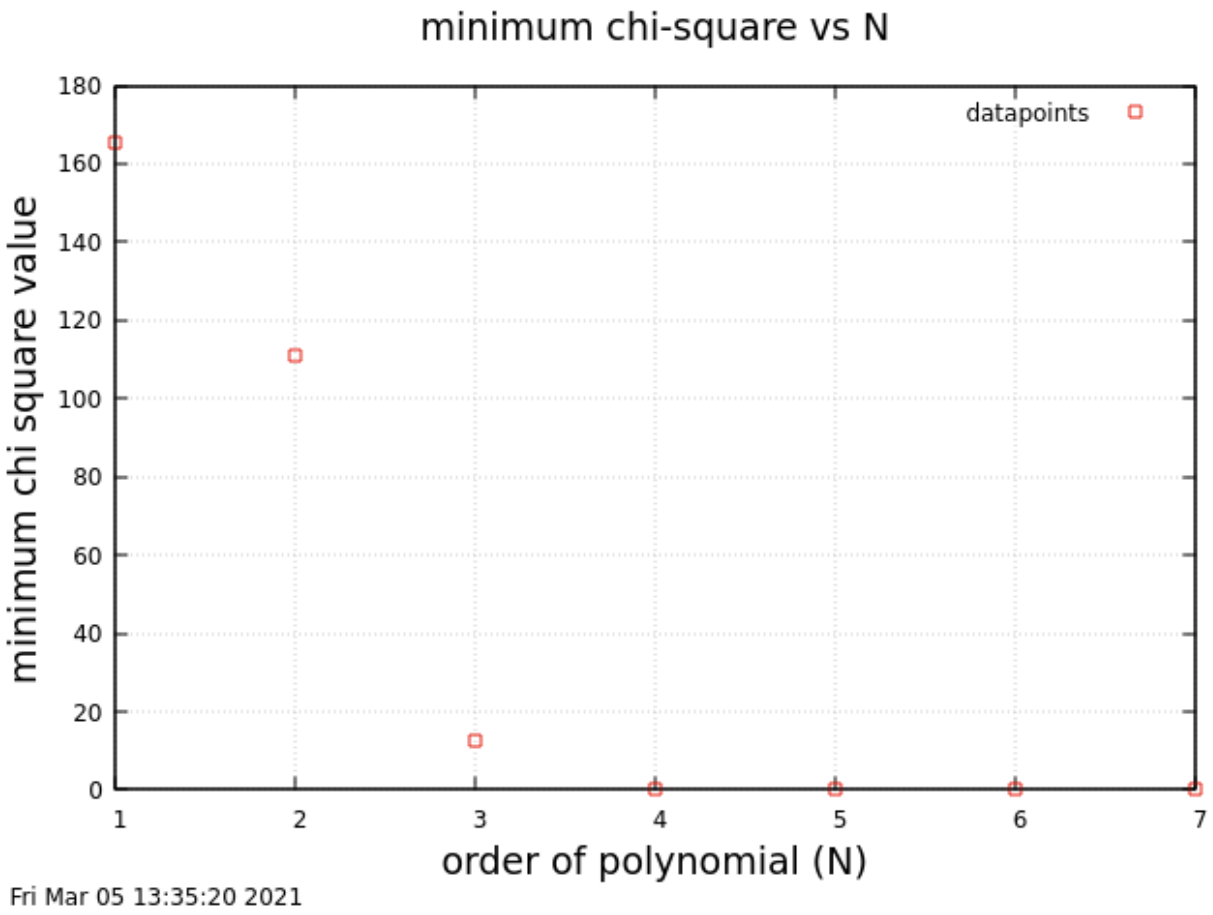
The coefficients for order [5] are:
a[0]= 6.330
a[1]= 1.055
a[2]= 0.714
a[3]= -7.476
a[4]= 1.002
a[5]= 1.506
Chi square [5]:0.000051

The coefficients for order [6] are:
a[0]= 6.330
a[1]= 1.012
a[2]= 0.955

a[3]= -7.956
a[4]= 1.442
a[5]= 1.316
a[6]= 0.031
Chi square [6]:0.000038

The coefficients for order [7] are:
a[0]= 6.330
a[1]= 1.068
a[2]= 0.520
a[3]= -6.729
a[4]= -0.224
a[5]= 2.497
a[6]= -0.390
a[7]= 0.060
Chi square [7]:0.000029





PROBLEM 4 (a) : for (σ_2) upto order 7

Changed the array for sigma in the above code.

OUTPUT :

The coefficients for order [1] are:
a[0]= 6.602
a[1]= -3.995
Chi square [1]:572.957309

The coefficients for order [2] are:
a[0]= 6.563
a[1]= -3.470
a[2]= -0.360
Chi square [2]:569.274419

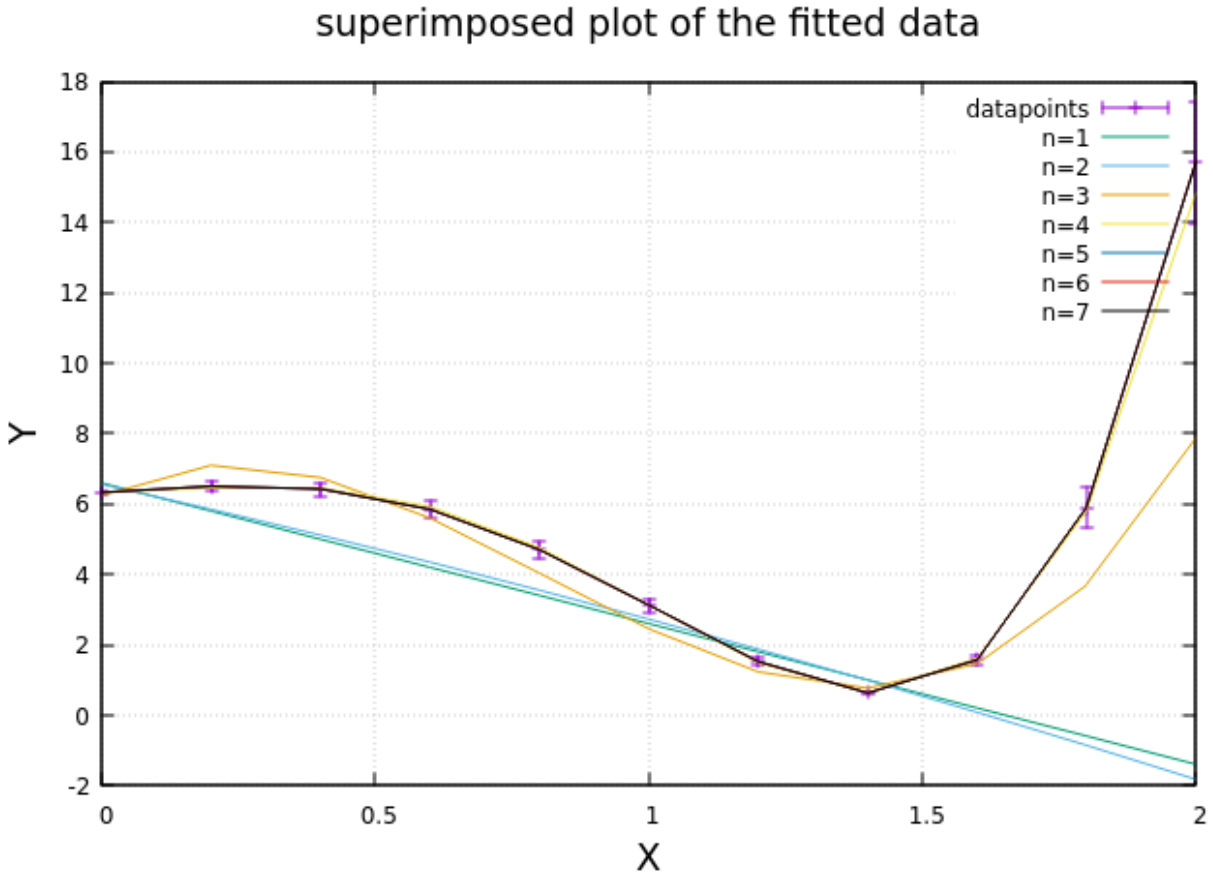
The coefficients for order [3] are:
a[0]= 6.246
a[1]= 7.909
a[2]= -19.848
a[3]= 8.149
Chi square [3]:97.430383

The coefficients for order [4] are:
a[0]= 6.335
a[1]= -0.152
a[2]= 7.045
a[3]= -17.780
a[4]= 7.678
Chi square [4]:0.874686

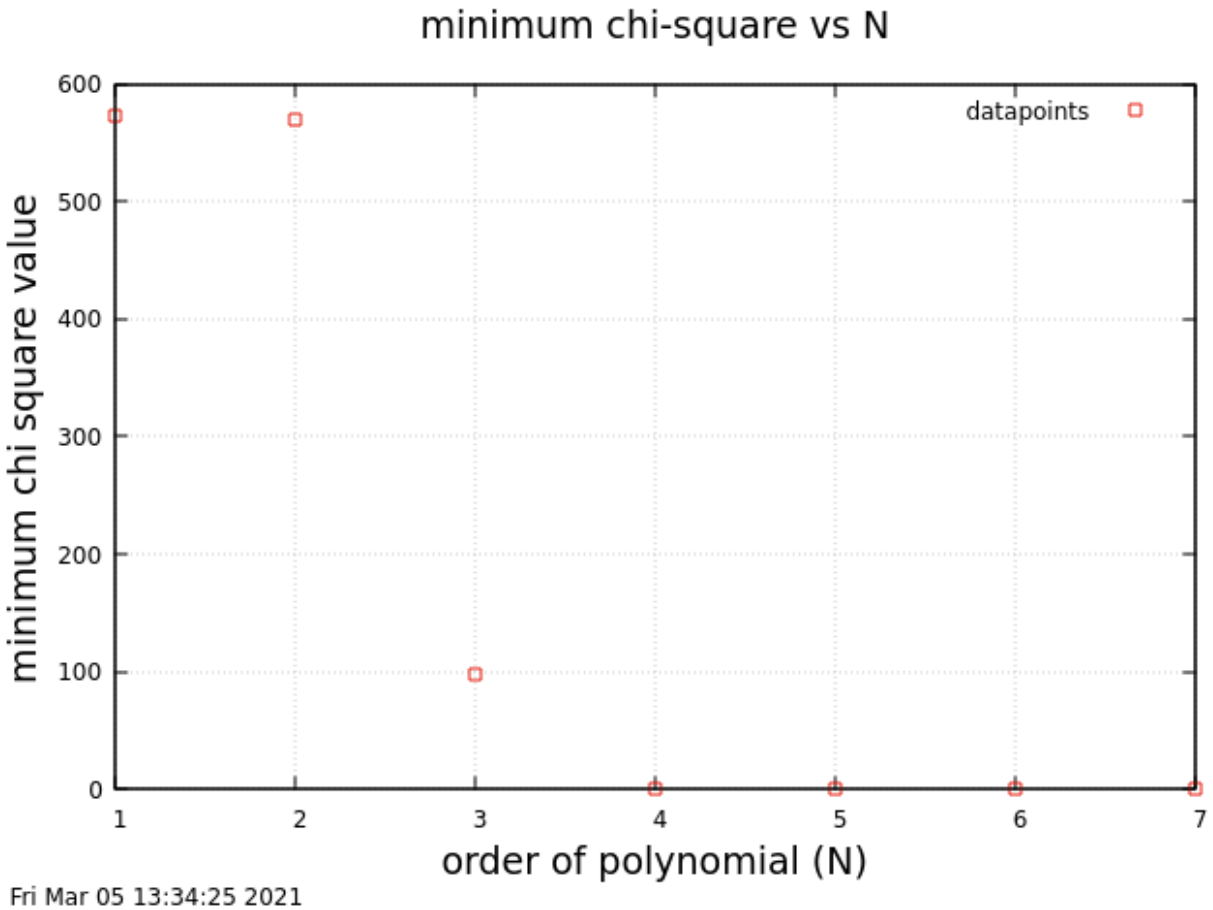
The coefficients for order [5] are:
a[0]= 6.330
a[1]= 1.038
a[2]= 0.785
a[3]= -7.575
a[4]= 1.059
a[5]= 1.494
Chi square [5]:0.000413

The coefficients for order [6] are:
a[0]= 6.330
a[1]= 1.023
a[2]= 0.891
a[3]= -7.823
a[4]= 1.317
a[5]= 1.371
a[6]= 0.022
Chi square [6]:0.000362

The coefficients for order [7] are:
a[0]= 6.330
a[1]= 1.031
a[2]= 0.818
a[3]= -7.595
a[4]= 0.980
a[5]= 1.627
a[6]= -0.075
a[7]= 0.014
Chi square [7]:0.000358



Fri Mar 05 17:45:22 2021



Fri Mar 05 13:34:25 2021

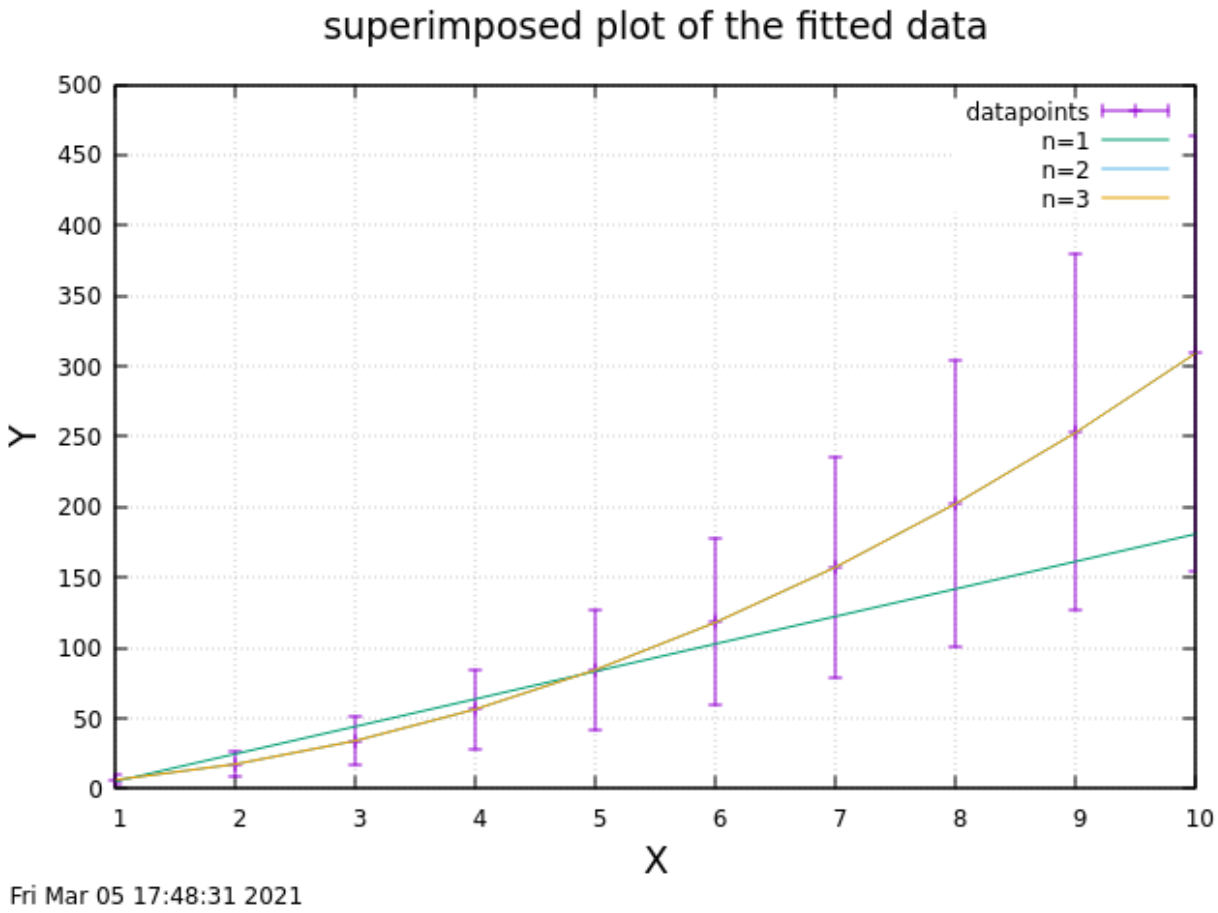
PROBLEM 4 (b) : Upto order 3

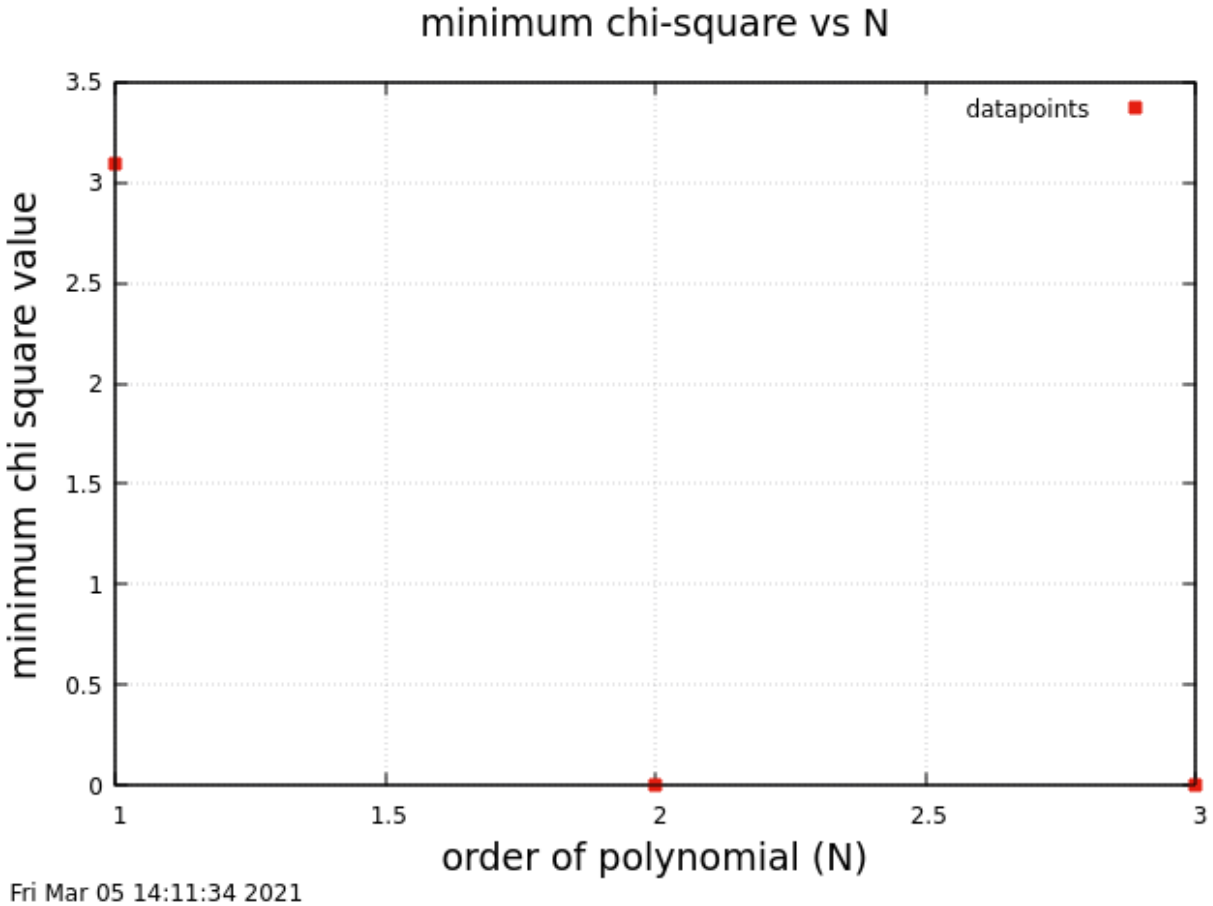
Change the array for x, y and σ in the above code.

OUTPUT : The coefficients for order [1] are:
a[0]= -14.308
a[1]= 19.528
Chi square [1]:3.096022

The coefficients for order [2] are:
a[0]= 0.974
a[1]= 2.568
a[2]= 2.828
Chi square [2]:0.000016

The coefficients for order [3] are:
a[0]= 0.963
a[1]= 2.585
a[2]= 2.822
a[3]= 0.000
Chi square [3]:0.000016





Fri Mar 05 14:11:34 2021