EE1103: Numerical Methods

Programming Assignment # 5

Nachiket Dighe, EE21B093

March 19, 2022

Contents

List of Figures

List of Tables

1 Problem 1

Curve Fitting using Least Square Regression for straight Line.

1.1 Approach

In this problem, we use the *For Loops* and *Arrays* to calculate the *Least square Error* and finding the **Straight Line** which is best fit for the given Data Points.

1.2 Algorithm

The Pseudo-Code for Fitting the given Data using **Linear Regression** is given in Algorithm 1

Algorithm 1: Fitting the Given Data using Linear Regression.

$x \leftarrow inputdata, y \leftarrow outputdata$
$n \leftarrow 11$
$X \leftarrow [x_0, x_1, \cdots, x_{10}], Y \leftarrow [y_0, y_1, \cdots, y_{10}]$
$sum_{xy} \leftarrow 0, sum_x \leftarrow 0, sum_y \leftarrow 0, sum_{x2} \leftarrow 0$
$S_r \leftarrow 0, S_t \leftarrow 0$
for $i = 0$ *to* $n - 1$ **do**
    $sum_x \leftarrow sum_x + x_i$
    $sum_y \leftarrow sum_y + y_i$
    $sum_{xy} \leftarrow sum_{xy} + x_i y_i$
    $sum_{x2} \leftarrow sum_{x2} + x_i^2$
end
$a_1 = \frac{n sum_{xy} - sum_x sum_y}{n sum_{x2} - (sum_x)^2}$
$a_0 = \frac{sum_y}{n} - a_1 \frac{sum_x}{n}$
for $i = 0$ *to* $n - 1$ **do**
    $S_r \leftarrow S_r + (y_i - a_0 - a_1 x_i)^2$
    $S_t \leftarrow S_t + (y_i - y_m)^2$
end
$S_{y/x} \leftarrow \sqrt{\frac{S_r}{n-2}}$
$r^2 \leftarrow \frac{S_t - S_r}{S_t}$
$r \leftarrow \sqrt{\frac{S_t - S_r}{S_t}}$

1.3 Results

The Straight Line which is Best fit for the Given Data Set is:

$$f(x) = 31.05899 - 0.780546x \tag{1}$$

The data as discrete points and the regression line (as a continuous line) in the same set of axes is given in Fig. 1.

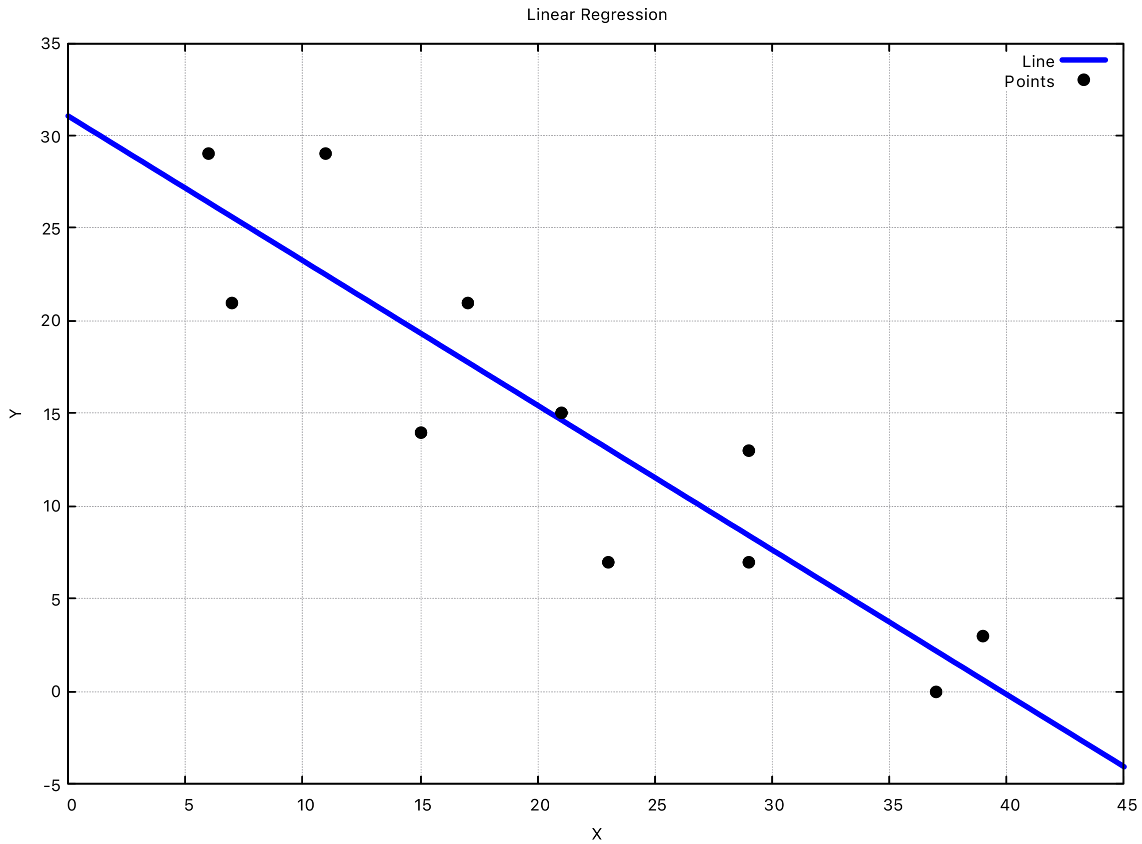

Figure 1: Linear Regression.

Table 1: Linear Regression for the given Data set.

Intercept(a_0)	**Slope**(a_1)	**r**	**Standard Error**	S_r	S_t
31.05899	-0.780546	0.901489	4.476306	180.355831	962.727356

Table 2: Linear Regression after adding another Data point.

Intercept(a_0)	**Slope**(a_1)	**r**	**Standard Error**	S_r	S_t
27.966326	-0.682770	0.815879	5.726786	327.960785	980.916626

1.4 Inferences

We deduce the following inferences in this experiment:

- In Linear Regression, we say that the Curve is the Best fit for the Given Data points Only when the **Correlation Coefficient** is approximately equal to **1**.This is because, we try to *Minimize the Error* in Linear Regression i.e. we try to make $S_r = 0$.

- Since,from table 1 we see that the Correlation Coeffiecient(r) is **0.901489**, so Linear Regression is not considered the best Fit for the given data points as it is not close to 1.

- When we consider an additional data point(10,10), we can clearly see from Fig 1 that the additional measurement is **Faulty**.This is because the Correlation Coeffiecient *Falls down* to **0.815879** due to which it is **not** the correct measurement.

1.5 Code

The code used for the experiments is mentioned in Listing 1

```c
//Least Square Regression to Fit a Straight line.
#include<stdio.h>
#include<math.h>
int main()
{
    float x[11]={6,7,11,15,17,21,23,29,29,37,39};
    float y[11]={29,21,29,14,21,15,7,7,13,0,3};
    float a0,a1,sum_yx,r;
    float sum_xi=0,sum_yi=0,sum_xiyi=0,sum_x2=0,sum_t=0,sum_r=0;
    int i,n=11;
    for(i=0;i<n;i++)//Loop to calculate summations necessary to find a0 and a1
    {
        sum_xi+=x[i];
        sum_yi+=y[i];
        sum_xiyi+=x[i]*y[i];
        sum_x2+=x[i]*x[i];
    }

    a1=(n*sum_xiyi-sum_xi*sum_yi)/(n*sum_x2-sum_xi*sum_xi);//Slope of the line
    a0=(sum_yi-a1*sum_xi)/n;//Intercept of the line

    for(i=0;i<n;i++)//This Loop to calculate Sr and St
    {
        sum_r+=pow((y[i]-a0-a1*x[i]),2);
        sum_t+=pow(y[i]-sum_yi/n,2);
    }

    sum_yx=sqrt(sum_r/(n-2));//Standard Error of Estimate

    r=sqrt((sum_t-sum_r)/sum_t);//Correlation Coefficient

    printf("The intercept is(a0): %f\n",a0);
    printf("The Slope is(a1): %f\n",a1);
    printf("Standard Error of the Estimate is: %f\n",sum_yx);
    printf("r^2 is: %f\n",r*r);
    printf("Correlation Coefficient r is: %f\n",r);
```

```
}
```

Listing 1: Code to Find Straight line fit for the given Data points.

2 Problem 2

Curve Fitting using Least Square Regression using various Curves.

2.1 Approach

In this problem, we use the *For Loops*,*Gauss Elimination* and *Arrays* to calculate the *Least square Error* and finding the Curves which are best fit for the given Data Points.

2.2 Algorithm

The Pseudo-Code for Least Square regression to fit Various curves is given in Algorithm 2 ,Algorithm 3 ,Algorithm 4 and Algorithm 5.

Algorithm 2: Fitting the Given Data using Linear Regression.

$x \leftarrow inputdata, y \leftarrow outputdata$
$n \leftarrow 10$
$X \leftarrow [x_0, x_1, \cdots, x_9], Y \leftarrow [y_0, y_1, \cdots, y_9]$
$sum_{xy} \leftarrow 0, sum_x \leftarrow 0, sum_y \leftarrow 0, sum_{x2} \leftarrow 0$
$S_r \leftarrow 0, S_t \leftarrow 0$
for $i = 0$ *to* $n - 1$ **do**
  $sum_x \leftarrow sum_x + x_i$
  $sum_y \leftarrow sum_y + y_i$
  $sum_{xy} \leftarrow sum_{xy} + x_i y_i$
  $sum_{x2} \leftarrow sum_{x2} + x_i^2$
end
$a_1 = \frac{n sum_{xy} - sum_x sum_y}{n sum_{x2} - (sum_x)^2}$
$a_0 = \frac{sum_y}{n} - a_1 \frac{sum_x}{n}$
for $i = 0$ *to* $n - 1$ **do**
  $S_r \leftarrow S_r + (y_i - a_0 - a_1 x_i)^2$
  $S_t \leftarrow S_t + (y_i - y_m)^2$
end
$S_{y/x} \leftarrow \sqrt{\frac{S_r}{n-2}}$
$r^2 \leftarrow \frac{S_t - S_r}{S_t}$
$r \leftarrow \sqrt{\frac{S_t - S_r}{S_t}}$

Algorithm 3: Fitting the Given Data using Power regression.

$x \leftarrow inputdata, y \leftarrow outputdata$
$n \leftarrow 10$
$X \leftarrow [x_0, x_1, \cdots, x_9], Y \leftarrow [y_0, y_1, \cdots, y_9]$
$sum_{xy} \leftarrow 0, sum_x \leftarrow 0, sum_y \leftarrow 0, sum_{x2} \leftarrow 0$
$S_r \leftarrow 0, S_t \leftarrow 0$
for $i = 0$ *to* $n - 1$ **do**
  $sum_x \leftarrow sum_x + \log x_i$
  $sum_y \leftarrow sum_y + \log y_i$
  $sum_{xy} \leftarrow sum_{xy} + \log x_i \log y_i$
  $sum_{x2} \leftarrow sum_{x2} + \log {x_i}^2$
end
$a_1 = \frac{nsum_{xy} - sum_x sum_y}{nsum_{x2} - (sum_x)^2}$
$a_0 = \frac{sum_y}{n} - a_1 \frac{sum_x}{n}$
for $i = 0$ *to* $n - 1$ **do**
  $S_r \leftarrow S_r + (\log y_i - a_0 - a_1 \log x_i)^2$
  $S_t \leftarrow S_t + (\log y_i - \log y_m)^2$
end
$S_{y/x} \leftarrow \sqrt{\frac{S_r}{n-2}}$
$r^2 \leftarrow \frac{S_t - S_r}{S_t}$
$r \leftarrow \sqrt{\frac{S_t - S_r}{S_t}}$

Algorithm 4: Fitting the Given Data using Saturation Growth-rate equation.

$x \leftarrow inputdata, y \leftarrow outputdata$
$n \leftarrow 10$
$X \leftarrow [x_0, x_1, \cdots, x_9], Y \leftarrow [y_0, y_1, \cdots, y_9]$
$sum_{xy} \leftarrow 0, sum_x \leftarrow 0, sum_y \leftarrow 0, sum_{x2} \leftarrow 0$
$S_r \leftarrow 0, S_t \leftarrow 0$
for $i = 0$ *to* $n - 1$ **do**
  $sum_x \leftarrow sum_x + \frac{1}{x_i}$
  $sum_y \leftarrow sum_y + \frac{1}{y_i}$
  $sum_{xy} \leftarrow sum_{xy} + \frac{1}{x_i y_i}$
  $sum_{x2} \leftarrow sum_{x2} + \frac{1}{x_i^2}$
end
$a_1 = \frac{n sum_{xy} - sum_x sum_y}{n sum_{x2} - (sum_x)^2}$
$a_0 = \frac{sum_y}{n} - a_1 \frac{sum_x}{n}$
for $i = 0$ *to* $n - 1$ **do**
  $S_r \leftarrow S_r + (y_i - a_0 - \frac{a_1}{x_i})^2$
  $S_t \leftarrow S_t + (\frac{1}{y_i} - \frac{1}{y_m})^2$
end
$S_{y/x} \leftarrow \sqrt{\frac{S_r}{n-2}}$
$r^2 \leftarrow \frac{S_t - S_r}{S_t}$
$r \leftarrow \sqrt{\frac{S_t - S_r}{S_t}}$

Algorithm 5: Fitting the Given Data using Quadractic Regression.

$x \leftarrow inputdata, y \leftarrow outputdata$
$n \leftarrow 10$
$X \leftarrow [x_0, x_1, \cdots, x_9], Y \leftarrow [y_0, y_1, \cdots, y_9]$
$sum_{xy} \leftarrow 0, sum_x \leftarrow 0, sum_y \leftarrow 0, sum_{x2} \leftarrow 0$
$sum_{x3} \leftarrow 0, sum_{x4} \leftarrow 0, sum_{x2y} \leftarrow 0,$
$S_r \leftarrow 0, S_t \leftarrow 0$
for $i = 0$ *to* $n - 1$ **do**
  $sum_x \leftarrow sum_x + x_i$
  $sum_y \leftarrow sum_y + y_i$
  $sum_{x2} \leftarrow sum_{xy} + x_i^2$
  $sum_{x3} \leftarrow sum_{x2} + x_i^3$
  $sum_{x4} \leftarrow sum_{xy} + x_i^4$
  $sum_{xy} \leftarrow sum_{xy} + x_i y_i$
  $sum_{x2y} \leftarrow sum_{xy} + x_i^2 y_i$
end
$na_0 + a_1 sum_x + a_2 sum_{x2} = sum_y$
$a_0 sum_x + a_1 sum_{x2} + a_2 sum_{x3} = sum_{xy}$
$a_0 sum_{x2} + a_1 sum_{x3} + a_2 sum_{x4} = sum_{x2y}$
Using Gauss Elimination we can calculate a0,a1,a2
for $i = 0$ *to* $n - 1$ **do**
  $S_r \leftarrow S_r + (y_i - a_0 - a_1 x_i - a_2 x_i^2)^2$
  $S_t \leftarrow S_t + (y_i - y_m)^2$
end
$S_{y/x} \leftarrow \sqrt{\frac{S_r}{n-3}}$
$r^2 \leftarrow \frac{S_t - S_r}{S_t}$
$r \leftarrow \sqrt{\frac{S_t - S_r}{S_t}}$

2.3 Results

Using Least Square Regression the curves obtained are:

- (a)Linear Regression: $F(x) = 20.6 + 0.494545x$
- (a)Power Regression: $F(x) = 9.952803x^{0.385081}$
- (a)Saturation-growth rate Regression: $F(x) = \frac{50.092125x}{9.891373+x}$
- (a)Quadratic Regression: $F(x) = 11.76684 + 1.377877x - 0.016061x^2$

Table 3: Linear Regression for the given Data set.

Curves	**r**	**Standard Error**	S_r	S_t
Linear	0.915692	3.485033	97.163620	601.599976
Power	0.977375	0.064791	0.033583	0.750675
Parabola	0.989941	1.311619	12.042413	601.599976
Saturation	0.996758	0.000937	0.000007	0.001086

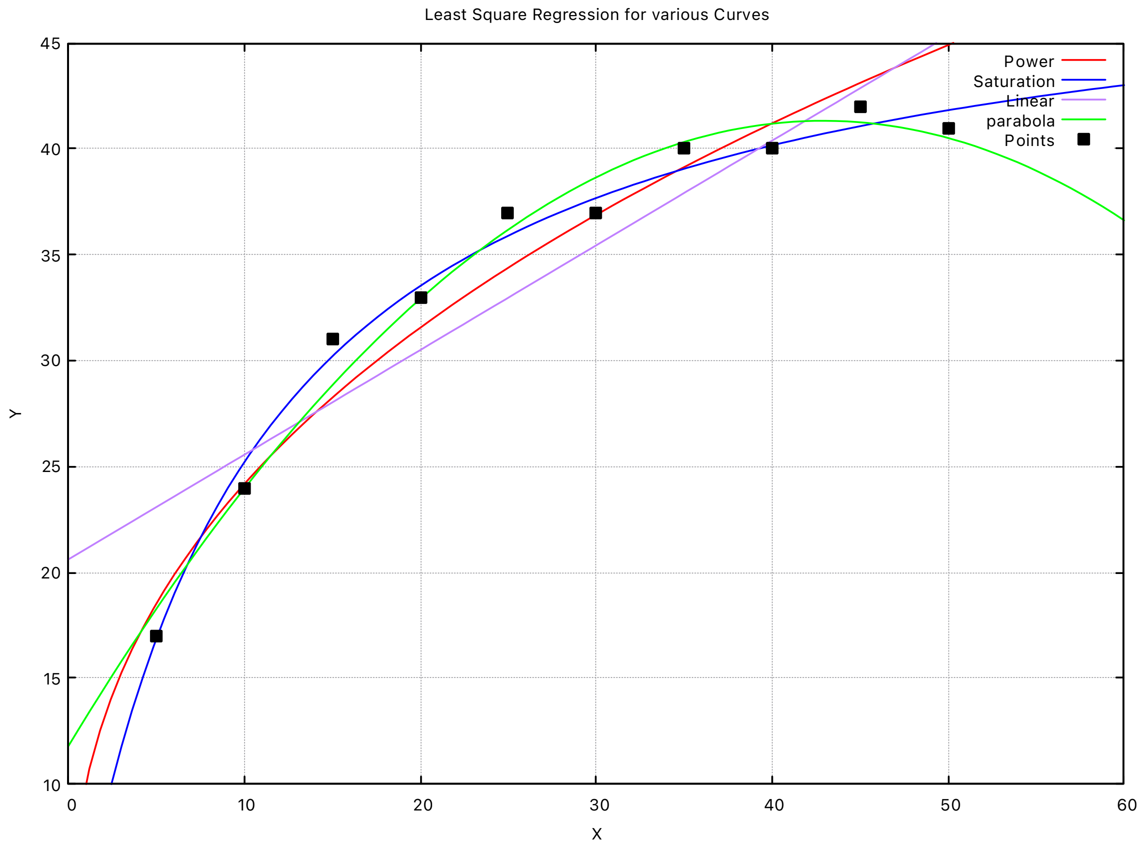

Figure 2: Least Square Regression to fit 4 curves

2.4 Inference

We deduce the following inferences in this experiment:

- In Regression, we say that the Curve is the Best fit for the Given Data points Only when the Correlation Coefficient is approximately equal to 1. This is because, we try to Minimize the Error in Regression i.e. we try to make $S_r = 0$.
- As seen from Table 3,The Correlation Coeffiecient varies as shown below:

$$Linear \leq Power \leq Parabola \leq Saturation$$

- Thus, we can say that the **Saturation-growth rate** Equation is the best fit for the given Data points as its Correlation Coefficient is nearly equal to 1.

2.5 Code

The code used for the experiments is mentioned in Listing 2.

```
//Least Square Regression to Fit a 4 Curves.
#include<stdio.h>
#include<math.h>
void powerfunc(float x[],float y[])
```

```
{
    float a0,a1,sum_yx,sum_r=0,r;
    float sum_xi=0,sum_yi=0,sum_xiyi=0,sum_x2=0,sum_t=0;
    int i,n=10;
    for(i=0;i<n;i++)//Loop to calculate summations necessary to find a0 and a1
    {
        sum_xi+=log(x[i]);
        sum_yi+=log(y[i]);
        sum_xiyi+=log(x[i])*log(y[i]);
        sum_x2+=log(x[i])*log(x[i]);
    }

    a1=(n*sum_xiyi-sum_xi*sum_yi)/(n*sum_x2-sum_xi*sum_xi);//Slope of the line
    a0=(sum_yi-a1*sum_xi)/n;//Intercept of the line

    for(i=0;i<n;i++)//This Loop to calculate Sr and St
    {
        sum_r+=pow(log(y[i])-a0-a1*log(x[i]),2);
        sum_t+=pow(log(y[i])-sum_yi/n,2);
    }

    sum_yx=sqrt(sum_r/(n-2));//Standard Error of Estimate

    r=sqrt((sum_t-sum_r)/sum_t);//Correlation Coefficient

    printf("\nUsing Power Method we get:\n");
    printf("a: %f\n",exp(a0));
    printf("b: %f\n",a1);
    printf("Standard Error of the Estimate is: %f\n",sum_yx);
    printf("Correlation Coefficient r is: %f\n",r);
    printf("The Function is: F(x) = %f*x^%f\n",exp(a0),a1);
}

void satfunc(float x[],float y[])
{
    float a0,a1,r;
    float sum_xi=0,sum_yi=0,sum_xiyi=0,sum_x2=0,sum_t=0,sum_yx,sum_r=0;
    int i,n=10;
    for(i=0;i<n;i++)//Loop to calculate summations necessary to find a0 and a1
    {
        sum_xi+=1/x[i];
        sum_yi+=1/y[i];
        sum_xiyi+=1/(x[i]*y[i]);
```

```
        sum_x2+=1/(x[i]*x[i]);
    }

    a1=(n*sum_xiyi-sum_xi*sum_yi)/(n*sum_x2-sum_xi*sum_xi);//Slope of the line
    a0=(sum_yi-a1*sum_xi)/n;//Intercept of the line

    for(i=0;i<n;i++)//This Loop to calculate Sr and St
    {
        sum_r+=pow((1/y[i]-a0-a1/x[i]),2);
        sum_t+=pow(1/y[i]-sum_yi/n,2);
    }

    sum_yx=sqrt(sum_r/(n-2));//Standard Error of Estimate

    r=sqrt((sum_t-sum_r)/sum_t);//Correlation Coefficient

    printf("\n\nUsing Saturation Method we get:\n");
    printf("alpha: %f\n",1/a0);
    printf("beta: %f\n",a1/a0);
    printf("Standard Error of the Estimate is: %f\n",sum_yx);
    printf("Correlation Coefficient r is: %f\n",r);
    printf("The Function is: F(x) = %f*x/(%f+x)\n",1/a0,a1/a0);
}

void linefunc(float x[],float y[])
{
    float a0,a1,sum_yx,r;
    float sum_xi=0,sum_yi=0,sum_xiyi=0,sum_x2=0,sum_t=0,sum_r=0;
    int i,n=10;
    for(i=0;i<n;i++)//Loop to calculate summations necessary to find a0 and a1
    {
        sum_xi+=x[i];
        sum_yi+=y[i];
        sum_xiyi+=x[i]*y[i];
        sum_x2+=x[i]*x[i];
    }

    a1=(n*sum_xiyi-sum_xi*sum_yi)/(n*sum_x2-sum_xi*sum_xi);//Slope of the line
    a0=(sum_yi-a1*sum_xi)/n;//Intercept of the line

    for(i=0;i<n;i++)//This Loop to calculate Sr and St
    {
        sum_r+=pow((y[i]-a0-a1*x[i]),2);
        sum_t+=pow(y[i]-sum_yi/n,2);
```

```c
    }

    sum_yx=sqrt(sum_r/(n-2));//Standard Error of Estimate

    r=sqrt((sum_t-sum_r)/sum_t);//Correlation Coefficient

    printf("\n\nUsing Linear Method we get:\n");
    printf("The intercept is(a0): %f\n",a0);
    printf("The Slope is(a1): %f\n",a1);
    printf("Standard Error of the Estimate is: %f\n",sum_yx);
    printf("Correlation Coefficient r is: %f\n",r);
    printf("The Function is: F(x) = %f+%f*x\n",a0,a1);
}

int main()
{
    float x[10]={5,10,15,20,25,30,35,40,45,50};
    float y[10]={17,24,31,33,37,37,40,40,42,41};
    float a0,a1,a2,sum_yx,r;
    float sum_x=0,sum_y=0,sum_t=0,sum_r=0;
    float sum_xy=0,sum_x2=0,sum_x3=0,sum_x4=0,sum_x2y=0,sum_x3y=0;
    int i,j,k,n=10;
    powerfunc(x,y);
    satfunc(x,y);
    linefunc(x,y);
    for(i=0;i<n;i++)//Loop to calculate summations necessary to find
    ↪  a0 and a1
    {
        sum_x+=x[i];
        sum_y+=y[i];
        sum_xy+=x[i]*y[i];
        sum_x2+=x[i]*x[i];
        sum_x3+=x[i]*x[i]*x[i];
        sum_x4+=x[i]*x[i]*x[i]*x[i];
        sum_x2y+=x[i]*x[i]*y[i];
        sum_x3y+=x[i]*x[i]*x[i]*y[i];

    }

    float A[10][10]={{n,sum_x,sum_x2,sum_y},
                     {sum_x,sum_x2,sum_x3,sum_xy},
                     {sum_x2,sum_x3,sum_x4,sum_x2y}};
    float v[4],c,sum=0;

    for(j=0; j<=2; j++) /* loop for the generation of upper
    ↪  triangular matrix*/
    {
```

```
        for(i=0; i<=2; i++)
        {
            if(i>j)
            {
                c=A[i][j]/A[j][j];
                for(k=0; k<=3; k++)
                {
                    A[i][k]=A[i][k]-c*A[j][k];
                }
            }
        }
    }
    v[2]=A[2][3]/A[2][2];
    /* this loop is for backward substitution*/
    for(i=1; i>=0; i--)
    {
        sum=0;
        for(j=i+1; j<=2; j++)
        {
            sum=sum+A[i][j]*v[j];
        }
        v[i]=(A[i][3]-sum)/A[i][i];
    }
    a0=v[0];
    a1=v[1];
    a2=v[2];

    for(i=0;i<n;i++)//This Loop to calculate Sr and St
    {
        sum_r+=pow((y[i]-a0-a1*x[i]-a2*x[i]*x[i]),2);
        sum_t+=pow(y[i]-sum_y/n,2);
    }

    sum_yx=sqrt(sum_r/(n-3));//Standard Error of Estimate

    r=sqrt((sum_t-sum_r)/sum_t);//Correlation Coefficient

    printf("\n\nUsing Quadratic Regression we get:\n");
    printf("a0: %f\n",a0);
    printf("a1: %f\n",a1);
    printf("a2: %f\n",a2);
    printf("Standard Error of the Estimate is: %f\n",sum_yx);
    printf("Correlation Coefficient r is: %f\n",r);
    printf("The Function is: F(x) = %f+%f*x+%f*x*x\n",a0,a1,a2);

}
```

Listing 2: Curve Fitting using various Curves.

3 Problem 3

Curve fitting using Interpolation and Polynomial Regression

3.1 Approach

In this problem, we use *For loops* and *Arrays* to find a Curve which passes through all the given data Points using **Newton divided Difference Interpolation**.

3.2 Algorithm

The Pseudo-Code for Curve Fitting using Polynomial Regression and Interpolation are given in Algorithm 6 and Algorithm 7.

Algorithm 6: Fitting the Given Data using Polynomial Regression.

$x \leftarrow inputdata, y \leftarrow outputdata$
$n \leftarrow 6$
$X \leftarrow [x_0, x_1, \cdots, x_5], Y \leftarrow [y_0, y_1, \cdots, y_5]$
$sum_{xy} \leftarrow 0, sum_x \leftarrow 0, sum_y \leftarrow 0, sum_{x2} \leftarrow 0$
$sum_{x3} \leftarrow 0, sum_{x4} \leftarrow 0, sum_{x2y} \leftarrow 0,$
$S_r \leftarrow 0, S_t \leftarrow 0$
for $i = 0$ *to* $n - 1$ **do**
    $sum_x \leftarrow sum_x + x_i$
    $sum_y \leftarrow sum_y + y_i$
    $sum_{x2} \leftarrow sum_{xy} + x_i^2$
    $sum_{x3} \leftarrow sum_{x2} + x_i^3$
    $sum_{x4} \leftarrow sum_{xy} + x_i^4$
    $sum_{xy} \leftarrow sum_{xy} + x_i y_i$
    $sum_{x2y} \leftarrow sum_{xy} + x_i^2 y_i$
end
$na_0 + a_1 sum_x + a_2 sum_{x2} = sum_y$
$a_0 sum_x + a_1 sum_{x2} + a_2 sum_{x3} = sum_{xy}$
$a_0 sum_{x2} + a_1 sum_{x3} + a_2 sum_{x4} = sum_{x2y}$
Using Gauss Elimination we can calculate a0,a1,a2
for $i = 0$ *to* $n - 1$ **do**
    $S_r \leftarrow S_r + (y_i - a_0 - a_1 x_i - a_2 x_i^2)^2$
    $S_t \leftarrow S_t + (y_i - y_m)^2$
end
$S_{y/x} \leftarrow \sqrt{\frac{S_r}{n-3}}$
$r^2 \leftarrow \frac{S_t - S_r}{S_t}$
$r \leftarrow \sqrt{\frac{S_t - S_r}{S_t}}$

Algorithm 7: Fitting the Given Data using Interpolation.

$x \leftarrow inputdata, y \leftarrow outputdata$
$n \leftarrow 6$
$X \leftarrow [x_0, x_1, \cdots, x_5], Y \leftarrow [y_0, y_1, \cdots, y_5]$
for $i = 0$ *to* n **do**
  $fdd_{i,0} = y_i$
end
for $j = 1$ *to* n **do**
  for $i = 0$ *to* $n - j$ **do**
    $fdd_{i,j} = (fdd_{i+1,j-1} - fdd_{i,j-1}/(x_{i+j} - x_i)$
  end
end
$xterm \leftarrow 1$
$yint_0 \leftarrow fdd_{0,0}$
for $order = 1$ *to* n **do**
  $xterm \leftarrow xterm * (x_i - x_{order-1})$
  $yint2 \leftarrow yint_{order-1} + fdd_{0,order} * xterm$
  $ea_{order-1} \leftarrow yint2 - yint_{order-1}$
  $yint_{order} \leftarrow yint2$
end

3.3 Results

The Equation of the Function obtained through Polynomial Regression is:

$$F(x) = 1.767245 - 0.049493x + 0.000548x^2$$

The value of μ at $T = 7.5\,\text{deg}$ is **1.4301** using Interpolation and using Polynomial Regression is **1.427** .

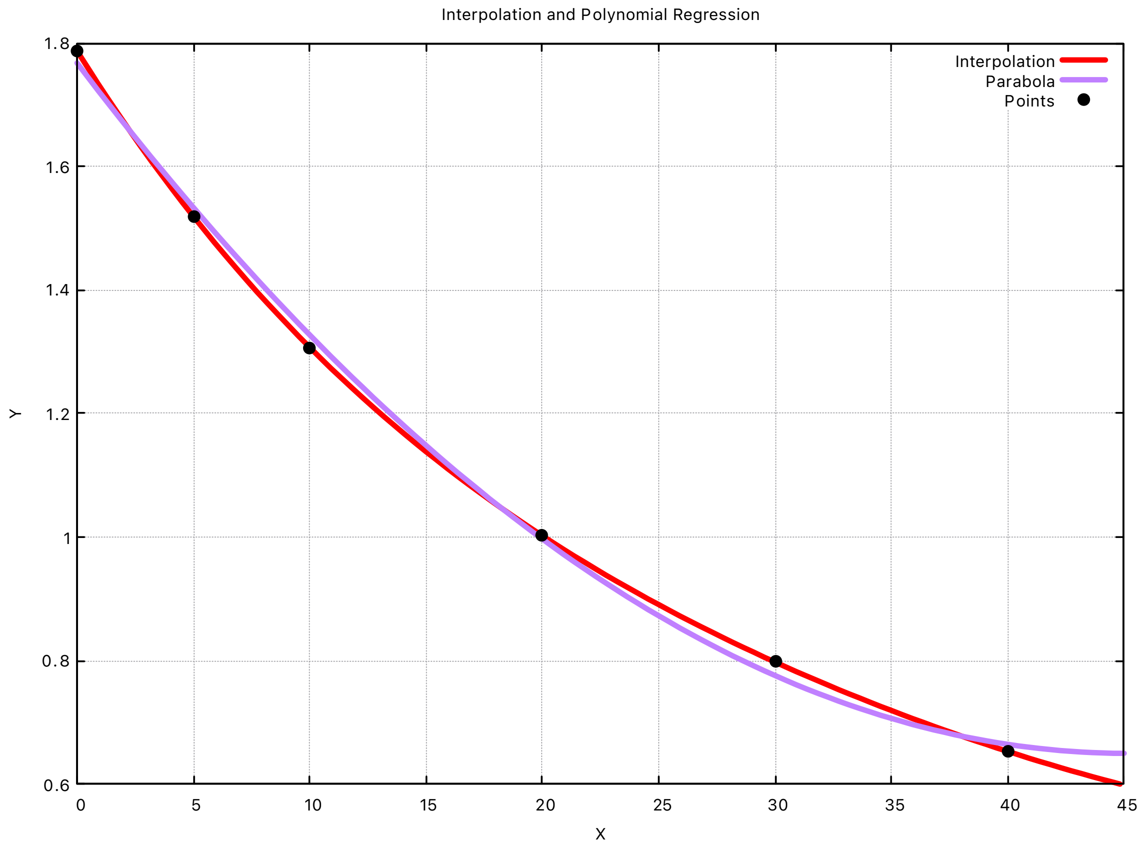

Figure 3: Interpolation and Polynomial Regression.

3.4 Inferences

We deduce the following inferences in this experiment:

- In Regression, we say that the Curve is the Best fit for the Given Data points Only when the Correlation Coefficient is approximately equal to 1. This is because, we try to Minimize the Error in Regression i.e. we try to make $S_r = 0$.
- Interpolation is a method of fitting the data points to represent the value of a function. It is used to construct new data points within the range of a discrete data set of known data points or can be used for determining a formula of the function that will pass from the given set of points (x,y).
- For Interpolation we assumed that there are no errors,but by Polynomial Regression we can see that there is an error in the data points.Thus, Doing Interpolation was not necessary.

3.5 Code

The code used for the experiments are mentioned in Listing 3 and Listing 4

```
// Newton divided difference Interpolation.
#include<stdio.h>
```

```c

// Function to find the product term
float proterm(int i, float value, float x[])
{
        float pro = 1;
        for (int j = 0; j < i; j++) {
                pro = pro * (value - x[j]);
        }
        return pro;
}

// Function for calculating
// divided difference table
void dividedDiffTable(float x[], float y[][10], int n)
{
        for (int i = 1; i < n; i++) {
                for (int j = 0; j < n - i; j++) {
                        y[j][i] = (y[j][i - 1] - y[j + 1]
                                                [i - 1]) / (x[j] - x[i + j]);
                }
        }
}

// Function for applying Newton's
// divided difference formula
float applyFormula(float value ,float x[] ,float y[][10] ,int n)
{
        float sum = y[0][0];

        for (int i = 1; i < n; i++) {
        sum = sum + (proterm(i, value, x) * y[0][i]);
        }
        return sum;
}

// Function for displaying
// divided difference table
void printDiffTable(float y[][10],int n)
{
        for (int i = 0; i < n; i++) {
                for (int j = 0; j < n - i; j++) {
            printf("%.9f \t",y[i][j]);

                }
                printf("\n");
        }
```

```
}

int main()
{
        // number of inputs given
        int n = 6;
        float value, sum,y[10][10];
        float x[] = { 0,5,10,15,20,25 };
        // y[][] is used for divided difference
    y[0][0]=1.787;
    y[1][0]=1.519;
    y[2][0]=1.307;
    y[3][0]=1.002;
    y[4][0]=0.7975;
    y[5][0]=0.6529;

        // calculating divided difference table
        dividedDiffTable(x, y, n);

        // displaying divided difference table
        printDiffTable(y,n);

        // value to be interpolated
        value = 7.5;

        // printing the value
    printf("\nValue at %f is
    ↪   %.9f\n",value,applyFormula(value,x,y,n));
        return 0;
}
```

Listing 3: Curve Fitting using Interpolation.

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

int main()
{
    float x[10]={0,5,10,20,30,40};
    float y[10]={1.787,1.519,1.307,1.002,0.7975,0.6529};
    float a0,a1,a2,sum_yx,r;
    float sum_x=0,sum_y=0,sum_t=0,sum_r=0;
    float sum_xy=0,sum_x2=0,sum_x3=0,sum_x4=0,sum_x2y=0,sum_x3y=0;
    int i,j,k,n=6;
    for(i=0;i<n;i++)//Loop to calculate summations necessary to find
    ↪   a0 and a1
    {
```

```c
        sum_x+=x[i];
        sum_y+=y[i];
        sum_xy+=x[i]*y[i];
        sum_x2+=x[i]*x[i];
        sum_x3+=x[i]*x[i]*x[i];
        sum_x4+=x[i]*x[i]*x[i]*x[i];
        sum_x2y+=x[i]*x[i]*y[i];
        sum_x3y+=x[i]*x[i]*x[i]*y[i];

    }

    float A[10][10]={{n,sum_x,sum_x2,sum_y},
                    {sum_x,sum_x2,sum_x3,sum_xy},
                    {sum_x2,sum_x3,sum_x4,sum_x2y}};
    float v[4],c,sum=0;

    for(j=0; j<=2; j++) /* loop for the generation of upper
     ↪  triangular matrix*/
    {
        for(i=0; i<=2; i++)
        {
            if(i>j)
            {
                c=A[i][j]/A[j][j];
                for(k=0; k<=3; k++)
                {
                    A[i][k]=A[i][k]-c*A[j][k];
                }
            }
        }
    }
    v[2]=A[2][3]/A[2][2];
    /* this loop is for backward substitution*/
    for(i=1; i>=0; i--)
    {
        sum=0;
        for(j=i+1; j<=2; j++)
        {
            sum=sum+A[i][j]*v[j];
        }
        v[i]=(A[i][3]-sum)/A[i][i];
    }
    a0=v[0];
    a1=v[1];
    a2=v[2];

    for(i=0;i<n;i++)//This Loop to calculate Sr and St
```

```
    {
        sum_r+=pow((y[i]-a0-a1*x[i]-a2*x[i]*x[i]),2);
        sum_t+=pow(y[i]-sum_y/n,2);
    }

    sum_yx=sqrt(sum_r/(n-3));//Standard Error of Estimate

    r=sqrt((sum_t-sum_r)/sum_t);//Correlation Coefficient

    printf("\n\nUsing Polynomial Regression we get:\n");
    printf("a0: %f\n",a0);
    printf("a1: %f\n",a1);
    printf("a2: %f\n",a2);
    printf("Standard Error of the Estimate is: %f\n",sum_yx);
    printf("Correlation Coefficient r is: %f\n",r);
    printf("The Function is: F(x) = %f %f*x + %f*x*x\n",a0,a1,a2);

}
```

Listing 4: Curve Fitting using Polynomial regression.