Numerical Method

Traffic queue assignment

Rayhan Faris Jufahri
19/444315/PA/19377

Task description

We are given two days to assess and record the length of a traffic queue in 1 hour period. To carry out this assignment, we need to find out the length of traffic queue (in meters) right before the red light expires, and also record the time it occurred so that we have the x-value (the time) and y-value (traffic queue length) data. Since the red light expires every 2 minutes, there will be 30 data for a single day. The first day observation was conducted on 23th of April 2021 while the second day observation was conducted on 30st of April 2021. The result of the observations are:

Day One[] =
{34,37,28,16,44,36,37,43,50,22,13,28,41,10,14,27,41,27,23,37,12,19,18,30,33,31,13,24,18,36}

Day Two[] =
{12,21,40,36,34,33,17,29.5,28,13,11.5,12,23,27,17,22.5,30.5,26,22,19,13,3,2,49,10,28,43,20,9,
30}

And x[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}

The codes below are built according to the lecture notes taken and doing calculations by hand. Codes are accessible through the URL provided at the end of each section and explanation.

Linear Regression

```
#include <iostream>
#include <iomanip>
#include <math.h>

using namespace std;

int f(int x, float a, float b){
        return a*x + b;
}

// Linear function y=mx+b
void LinearRegression(float data[], int mins[], int arraySize, string name){
        float sumX = 0;
        float sumY = 0;
        float sumXY = 0;
        float sumXSquared = 0;

        float a = 0;
        float b = 0;
```

```cpp
	for(int i=0; i<arraySize; i++){
		if(mins[i] % 2 == 1){	// regression data
			sumX += mins[i];
			sumY += data[i];
			sumXY += mins[i]*data[i];
			sumXSquared += mins[i]*mins[i];
		}
	}

	// find a and b constants
	a = (((arraySize/2)*sumXY)-(sumX*sumY))/(((arraySize/2)*sumXSquared)-
(sumX*sumX));
	b = (sumY/(arraySize/2))-(a*(sumX/(arraySize/2)));

	// formatting output
	if(b < 0){
		cout << name << "\ty = " << setw(5) << a << "x " << setw(5) << b;
	} else {
		cout << name << "\ty = " << setw(5) << a << "x + " << setw(5) << b;
	}

	// calculating error with Et and E
	float E = 0;
	for(int i=0; i<arraySize; i++){
		if(mins[i] % 2 == 0){ // testing data
			E += pow((data[i] - (sumY/(arraySize/2))),2);
		}
	}

	// Correlation Coefficient calculation
	float Et = 0;
	for(int i=0; i<arraySize; i++){
		if(mins[i] % 2 == 0){ //testing data
			Et += pow(data[i] - f(data[i], a, b),2);
		}
	}

	// substitute to the R^2 error formula
	float r = 1 - sqrt(E/Et);

	cout << " with r = " <<  r << " where ";

	// Correlation coefficient relationship with the data
	// source from https://keisan.casio.com/exec/system/14059931777261
	if(r > 0.7 && r <=1) {
		cout << "the function has a strong correlation with the data";
	} else if(r > 0.4 && r <= 0.7){
		cout << "the function has a moderate correlation with the data";
	} else if(r > 0.2 && r <= 0.4){
		cout << "the function has a weak correlation with the data";
	} else {
		cout << "the function has no correlation with the data";
	}
```

```
	cout << "\n\n";
}

int main(){
	int arraySize;

	float dayOne[] = {34,37,28,16,44,36,37,43,50,22,13,28,41,10,14,27,41,27,23,37,12,19,18,30,33,31,13,24,18,36}; // y
	float dayTwo[] = {12,21,40,36,34,33,17,29.5,28,13,11.5,12,23,27,17,22.5,30.5,26,22,19,13,3,2,49,10,28,43,20,9,30}; // y
	int mins[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}; // x


	arraySize = sizeof(dayOne) / sizeof(dayOne[0]);

	LinearRegression(dayOne, mins, arraySize, "Day One");
	LinearRegression(dayTwo, mins, arraySize, "Day Two");

	return 0;
}
```

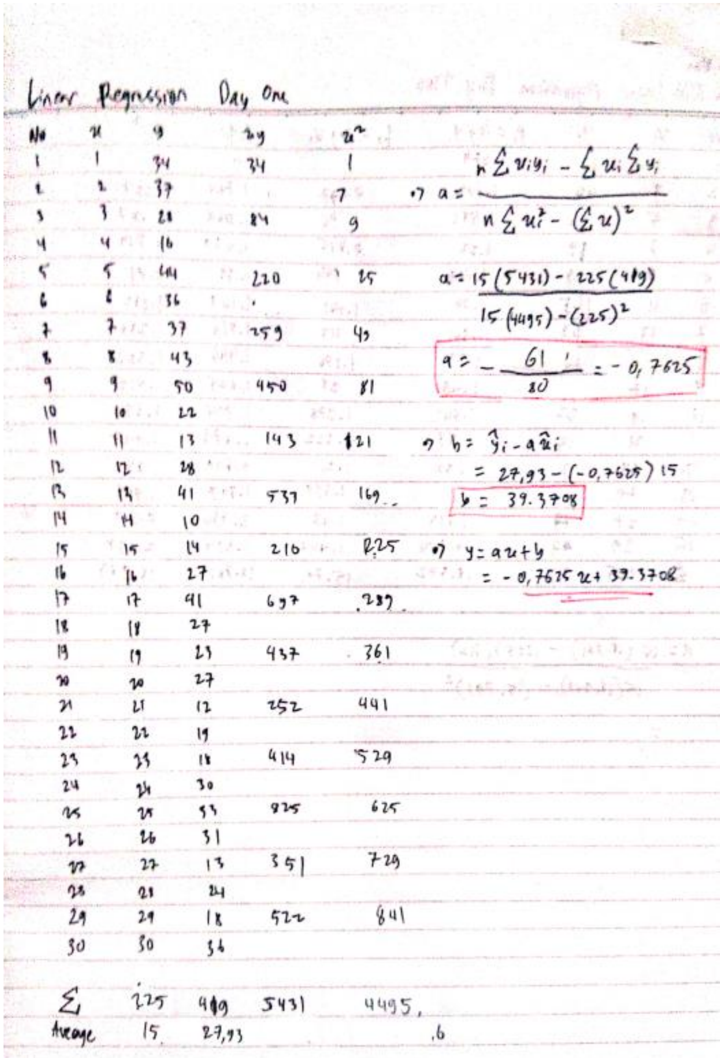
Code URL: https://onlinegdb.com/rJdX6AzOO

```
C:\Users\Dell\Documents\Dev C++\Code\trafficLinearRegression.exe
Day One y = -0.7625x + 39.3708 with r = 0.537126 where the function has a moderate correlation with the data

Day Two y = -0.35625x + 26.1437 with r = 0.310759 where the function has a weak correlation with the data


--------------------------------
Process exited after 0.01923 seconds with return value 0
Press any key to continue . . .
```

As we can see, we are able to find the two constants (slope and y-intercept) of both data sample (day-one and day-two) and create a linear regression. Using the methods of constructing the table of data get the slope value using mathematical proof and y-intercept based of the new slope constant, we are able to create a function that is approximately over the data.

We also need to consider the r constants, which is the R^2 formula, commonly used to indicate whether the function that we have built using the table has a correlation with the data. The closer the constant r to 1, the better and accurate, while the closer to 0, then the weaker correlation the function has on the given data.

As we can see the r constants says that the first linear regression has moderate correlation with the data, so for any value of x the y value approximation has moderate error margin with respect to the original data. However, for the second function of Day Two has a weak correlation coefficient with the data it has, so any given value of x, the y error margin is significantly larger than the original data.



Linear Regression Day One

No	x	y	xy	x^2
1	1	34	34	1
2	2	37		7
3	3	28	84	9
4	4	16		
5	5	44	220	25
6	6	36		
7	7	37	259	49
8	8	43		
9	9	50	450	81
10	10	22		
11	11	13	143	121
12	12	28		
13	13	41	537	169
14	14	10		
15	15	14	210	225
16	16	27		
17	17	41	697	.289
18	18	27		
19	19	23	437	361
20	20	27		
21	21	12	252	441
22	22	19		
23	23	18	414	529
24	24	30		
25	25	33	825	625
26	26	31		
27	27	13	351	729
28	28	24		
29	29	18	522	841
30	30	36		
Σ	225	419	5431	4495
Average	15	27,93		

$$\Rightarrow a = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x)^2}$$

$$a = \frac{15(5431) - 225(419)}{15(4495) - (225)^2}$$

$$\boxed{a = -\frac{61}{80} = -0,7625}$$

$$\Rightarrow b = \hat{y}_i - a\hat{x}_i$$

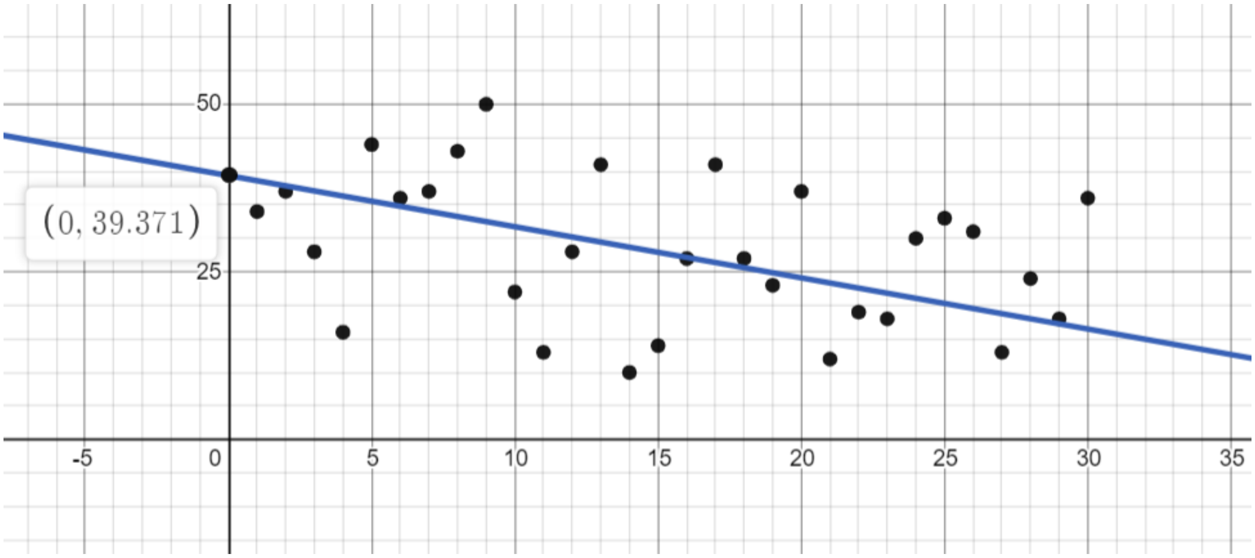
$$= 27,93 - (-0,7625)\,15$$

$$\boxed{b = 39.3708}$$

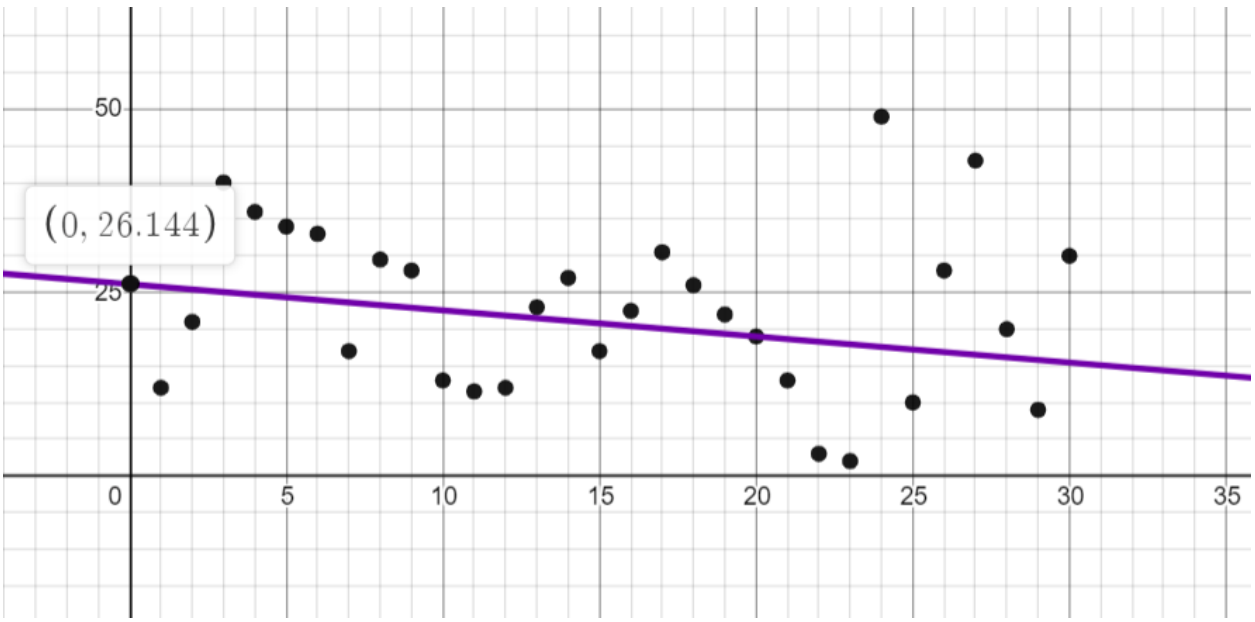
$$\Rightarrow y = ax + b$$

$$= -0,7625\,x + 39.3708$$

Day One Linear Regression graph



Day Two linear regression



Non-Linear Regression

```
#include <iostream>
#include <iomanip>
#include <math.h>

using namespace std;

float f(int x, float a, float b){
        return a*pow(x, b);
}
```

```
// y = ax^b
void NonLinearRegression(float data[], int mins[], int arraySize, string name){
	float sumX = 0;
	float sumY = 0;
	float sumQ = 0;
	float sumP = 0;
	float sumPQ = 0;
	float sumQSquared = 0;
	float sumPSquared = 0;
	float tempP = 0;
	float tempQ = 0;

	float a = 0;
	float b = 0;
	float c = 0;

	for(int i=0; i<arraySize; i++){
		if(mins[i] % 2 == 1){ // regression data
			sumX += mins[i];
			sumY += data[i];
			tempP = log10(data[i]);
			tempQ = log10(mins[i]);
			sumP += tempP;
			sumQ += tempQ;
			sumPQ += tempP * tempQ;
			sumQSquared += tempQ * tempQ;
			sumPSquared += tempP * tempP;
		}
	}

	b = (((arraySize/2)*sumPQ)-(sumP*sumQ))/(((arraySize/2)*sumQSquared)-
(sumQ*sumQ));
	c = (sumP/(arraySize/2))-(b*(sumQ/(arraySize/2)));
	a = pow(10, c);
	cout << name << "\ty = " << setw(5) << a << "x^" << setw(5) << b << endl;

	float error = 0;
	float calcError;
	for(int i=0; i<arraySize; i++){
		if(mins[i] % 2 == 0){ // testing data
			calcError = fabs((data[i]-f(mins[i], a, b))/data[i])*100;
			error += calcError;
			cout << "f(" << mins[i] << ") = "<< f(mins[i], a, b) <<
"\terror: " << calcError << "%" << endl;
		}
	}
	cout << "Average error: " << error/(arraySize/2) << endl;

	cout << endl;
}

int main(){
	int arraySize;
```

```
        float dayOne[] = {34,37,28,16,44,36,37,43,50,22,13,28,41,10,14,27,41,27,23,37,12,19,18,30,33,31,13,24,18,36}; // y
        float dayTwo[] = {12,21,40,36,34,33,17,29.5,28,13,11.5,12,23,27,17,22.5,30.5,26,22,19,13,3,2,49,10,28,43,20,9,30}; // y
        int mins[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}; // x

        arraySize = sizeof(dayOne) / sizeof(dayOne[0]);

        NonLinearRegression(dayOne, mins, arraySize, "Day One");
        NonLinearRegression(dayTwo, mins, arraySize, "Day Two");

}
```

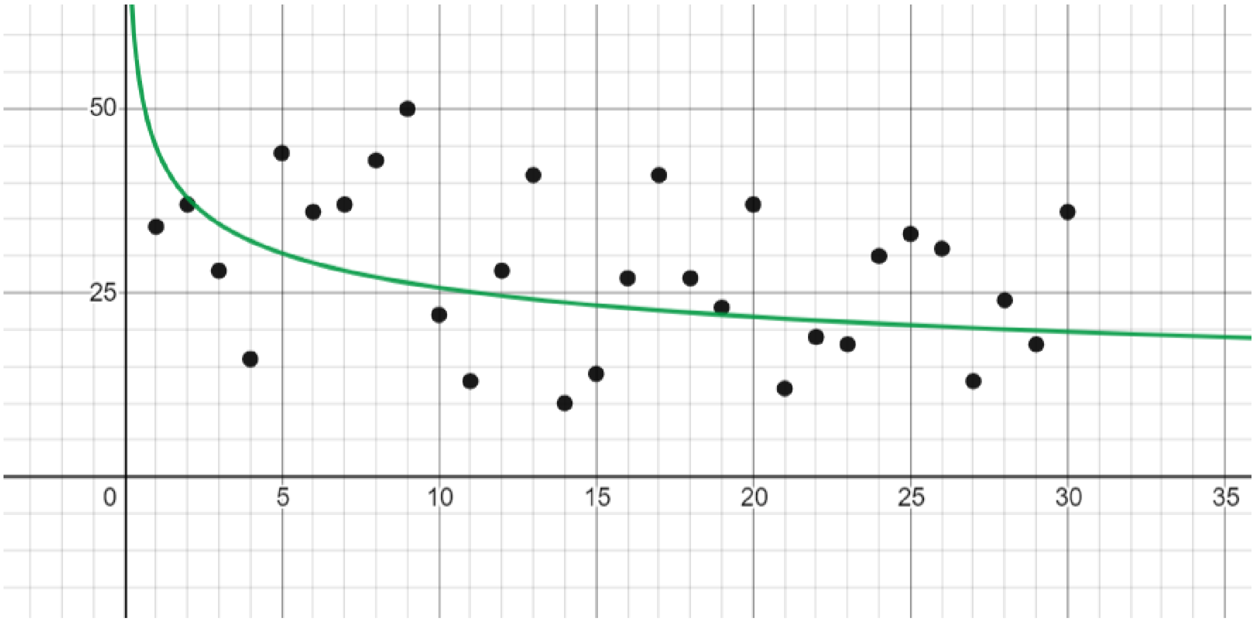
```
C:\Users\Dell\Documents\Dev C++\Code\trafficNonLinearRegression.exe
Day One y = 44.8257x^-0.241525
f(2) = 37.9158  error: 2.47526%
f(4) = 32.0711  error: 100.445%
f(6) = 29.0793  error: 19.2242%
f(8) = 27.1274  error: 36.9131%
f(10) = 25.7041 error: 16.8366%
f(12) = 24.5967 error: 12.1545%
f(14) = 23.6978 error: 136.978%
f(16) = 22.9457 error: 15.0159%
f(18) = 22.3022 error: 17.3994%
f(20) = 21.7418 error: 41.2384%
f(22) = 21.247  error: 11.8264%
f(24) = 20.8052 error: 30.6495%
f(26) = 20.4068 error: 34.1716%
f(28) = 20.0448 error: 16.48%
f(30) = 19.7135 error: 45.2401%
Average error: 35.8032

Day Two y = 27.1735x^-0.198812
f(2) = 23.6754  error: 12.7401%
f(4) = 20.6276  error: 42.7011%
f(6) = 19.0301  error: 42.3332%
f(8) = 17.9722  error: 39.0774%
f(10) = 17.1923 error: 32.2484%
f(12) = 16.5803 error: 38.1689%
f(14) = 16.0798 error: 40.445%
f(16) = 15.6586 error: 30.4063%
f(18) = 15.2962 error: 41.1686%
f(20) = 14.9791 error: 21.1627%
f(22) = 14.6979 error: 389.931%
f(24) = 14.4459 error: 70.5187%
f(26) = 14.2178 error: 49.2222%
f(28) = 14.0098 error: 29.9508%
f(30) = 13.819  error: 53.9367%
Average error: 62.2674
```

Code URL: https://onlinegdb.com/rJL9xkXOu

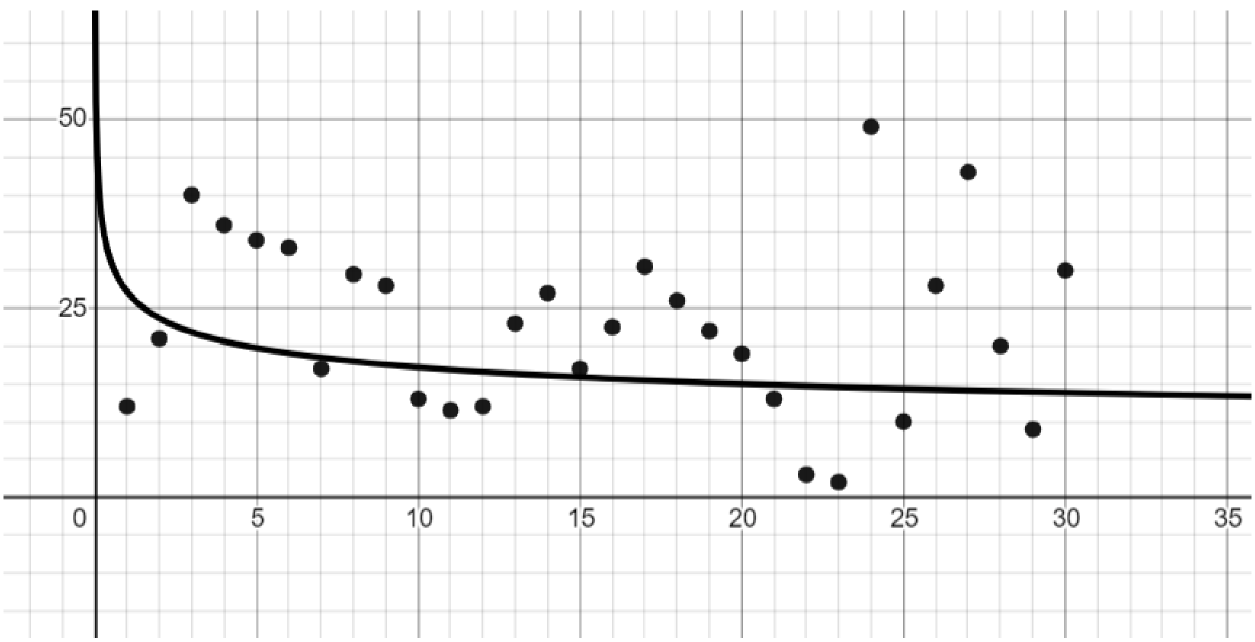
As we can see from the output above, we are able to find constants necessary for building a non-linear function that correlates with the given data of Day One and Day Two traffic queue. For the error finding, we use the even x values for testing our data. These testing data are then used to calculate the error margin of the original data with the data from the non-linear function.

As we can at the bottom of each day, the average error that we are getting is 30% for the first day and 60% for the second day.

Day One non-linear regression



Day Two non-linear regression



Linear & Non-linear interpolation

```
#include <iostream>
#include <iomanip>
#include <math.h>
#include <algorithm>

using namespace std;

// Below are the equations with its constance
// this is hardcoded from the calculation
// done on linear and non linear regression
// changes to these functions are required to
// when changing the x and y data

// Linear regression
float f1(float x){
        return -0.7625 * x + 39.3708;
}

float f2(float x){
        return -0.35625 * x + 26.1437;
}

// non linear regression
float f3(float x){
        return 44.8257 * pow(x, -0.241525);
}

float f4(float x){
        return 27.1735* pow(x, -0.198812);
}

void linearInterpolation(float data[], int mins[], int arraySize, int x, float x1,
float x2, string name){
        float ypLinearReg1, ypNonLinearReg1;
        float ypLinearReg2, ypNonLinearReg2;
        float calcErrorF1, calcErrorF2, calcErrorF3, calcErrorF4;
        float error = 0;

        if(name == "Day One" && x > x1 && x2 > x1 && binary_search(mins, mins +
arraySize, x1) && binary_search(mins, mins + arraySize, x2)){
                ypLinearReg1 = ((f1(x2)-f1(x1))/(x2-x1)*(x-x1))+f1(x1);
                ypNonLinearReg1 = ((f3(x2)-f3(x1))/(x2-x1)*(x-x1))+f3(x1);
                cout << "=== "<< name << " Linear Regression with linear
interpolation ===\n" << "f(x) = -0.7625x + 39.3708\nfind f(" << x << ") where x1 = "
<< x1 << " and x2 = " << x2 << " using linear interpolation is " << ypLinearReg1 << "
with error: " << (fabs(data[x]-f1(x))/data[x])*100 << "%" << endl;

                // Average error finding
                for(int i=0; i<arraySize; i++){
                        if(mins[i] % 2 == 0){
                                error += (fabs(data[i]-f1(i))/data[i])*100;
```

```
                }
        }

        error /= (arraySize/2);
        cout << "Average error: " << error << endl;

        cout << "=== "<< name << " Non Linear Regression with linear
interpolation ===\n" << "f(x) = 44.8257x^-0.241525\nfind f(" << x << ") where x1 = "
<< x1 << " and x2 = " << x2 << " using linear interpolation is " << ypNonLinearReg1 <<
" with error: " << (fabs(data[x]-f3(x))/data[x])*100 << "%" << endl;

        // Average error finding
        for(int i=0; i<arraySize; i++){
                if(mins[i] % 2 == 0){
                        error += (fabs(data[i]-f3(i))/data[i])*100;
                }
        }

        error /= (arraySize/2);
        cout << "Average error: " << error << endl;
    } else if(name == "Day Two" && x > x1 && x2 > x1 && binary_search(mins, mins +
arraySize, x1) && binary_search(mins, mins + arraySize, x2)){
        ypLinearReg2 = ((f2(x2)-f2(x1))/(x2-x1)*(x-x1))+f2(x1);
        ypNonLinearReg2 = ((f4(x2)-f4(x1))/(x2-x1)*(x-x1))+f4(x1);
        cout << "\n=== " << name << " Linear Regression with linear
interpolation ===\n" << "f(x) = -0.35625x + 26.1437\nfind f(" << x << ") where x1 = "
<< x1 << " and x2 = " << x2 << " using linear interpolation is " << ypLinearReg2 << "
with error: " << (fabs(data[x]-f2(x))/data[x])*100 << "%" << endl;

        // Average error finding
        for(int i=0; i<arraySize; i++){
                if(mins[i] % 2 == 0){
                        error += (fabs(data[i]-f2(i))/data[i])*100;
                }
        }

        error /= (arraySize/2);
        cout << "Average error: " << error << endl;

        cout << "=== " << name << " Non Linear Regression with linear
interpolation ===\n" << "f(x) = 27.1735x^-0.198812\nfind f(" << x << ") where x1 = "
<< x1 << " and x2 = " << x2 << " using linear interpolation is " << ypNonLinearReg2 <<
" with error: " << (fabs(data[x]-f4(x))/data[x])*100 << "%" << endl;

        // Average error finding
        for(int i=0; i<arraySize; i++){
                if(mins[i] % 2 == 0){
                        error += (fabs(data[i]-f4(i))/data[i])*100;
                }
        }

        error /= (arraySize/2);
        cout << "Average error: " << error << endl;
```

```cpp
	} else {
		cout << "\nMake sure the x value in f(x) is within the interval of x1 and x2" << endl;
		return;
	}
	cout << endl;
}

int main(){
	int arraySize, arraySize2;
	float x1, x2;
	int x;

	float dayOne[] = {34,37,28,16,44,36,37,43,50,22,13,28,41,10,14,27,41,27,23,37,12,19,18,30,33,31,13,24,18,36}; // y
	float dayTwo[] = {12,21,40,36,34,33,17,29.5,28,13,11.5,12,23,27,17,22.5,30.5,26,22,19,13,3,2,49,10,28,43,20,9,30}; // y
	int mins[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}; // x

	cout << "Enter X, X1 and X2:\n";
	cin >> x >> x1 >> x2;

	arraySize = sizeof(dayOne) / sizeof(dayOne[0]);

	linearInterpolation(dayOne, mins, arraySize, x, x1, x2, "Day One");
	linearInterpolation(dayTwo, mins, arraySize, x, x1, x2, "Day Two");
}
```

C:\Users\Dell\Documents\Dev C++\Code\linearInterpolation.exe

```
Enter X, X1 and X2:
15 3 25
=== Day One Linear Regression with linear interpolation ===
f(x) = -0.7625x + 39.3708
find f(15) where x1 = 3 and x2 = 25 using linear interpolation is 27.9333 with error: 3.45667%
Average error: 40.5362
=== Day One Non Linear Regression with linear interpolation ===
f(x) = 44.8257x^-0.241525
find f(15) where x1 = 3 and x2 = 25 using linear interpolation is 26.8637 with error: 13.6807%
Average error: 40.3947


=== Day Two Linear Regression with linear interpolation ===
f(x) = -0.35625x + 26.1437
find f(15) where x1 = 3 and x2 = 25 using linear interpolation is 20.7999 with error: 7.55578%
Average error: 66.7965
=== Day Two Non Linear Regression with linear interpolation ===
f(x) = 27.1735x^-0.198812
find f(15) where x1 = 3 and x2 = 25 using linear interpolation is 17.7439 with error: 29.5076%
Average error: 67.6904


--------------------------------
Process exited after 14.05 seconds with return value 0
Press any key to continue . . .
```

Code URL: https://onlinegdb.com/S1UOV1mOd

For the linear interpolation, we observe for functions from the linear and non-linear function from the result of regression calculation. Linear interpolation needs 3 inputs that is necessary to carry out the interpolation calculation. The first input (x) is the x value for the function to process to get the approximated y value. Then we have the x1 and x2 input that will be used to interval at which x is in. A line will connect those two (x1 and x2) points such that its continuous. For finding the f(x) - where f(x) is a function from the regression – where x is the point in y such that it is close to the actual data that is being requested. For this example, we are interested in finding/approximate the value where x = 15 between the interval [3, 25]. By substituting the x value, we can get y value from the interpolation plus a margin of error. As we can see the error value varies (from small percentage to big percentage) from each points because coincidentally, the traffic queue data is not predictable and hard for a function to pass through all the given points.

We can see that we successfully calculate the interpolation for both the function from the linear and non-linear regression. We can conclude that the linear interpolation for the Day One linear function works better than the Day Two linear function.

Quadratic Interpolation

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <algorithm>

using namespace std;

// Constants are set to global variable
float b0, b1, b2;

// this is hardcoded from the calculation
// done on linear and non linear regression
// changes to these functions are required to
// when changing the x and y data


// Linear regression equation
float f1(float x){ // Day One
        return -0.7625 * x + 39.3708;
}

float f2(float x){ // Day Two
        return -0.35625 * x + 26.1437;
}

// non linear regression
float f3(float x){ // Day One
```

```
        return 44.8257 * pow(x, -0.241525);
}

float f4(float x){ // Day Two
        return 27.1735* pow(x, -0.198812);
}

// quadratic interpolation function for 2nd order
float fQ(float x, float x1, float x2, float x3){
        return b0+(b1*(x-x1))+(b2*(x-x1)*(x-x2));
}

// y = b= + b1(x-x0) + b2(x-x1)(x-x0)
void quadraticInterpolation(float data[], int mins[], int arraySize, int x, float x1,
float x2, float x3, string name){
        // Day One linear and nonlinear interpolation
        float error = 0;
        if(name == "Day One" && x > x1 && x2 > x1 && x3 > x2 && binary_search(mins,
mins + arraySize, x1) && binary_search(mins, mins + arraySize, x2) &&
binary_search(mins, mins + arraySize, x3)){
                b0 = f1(x1);
                b1 = (f1(x2)-f1(x1))/(x2-x1);
                b2 = (((f1(x3)-f1(x2))/(x3-x2))-((f1(x2)-f1(x1))/(x2-x1)))/(x3-x1);
                if(b2 < 1.e-7){
                        b2 = 0;
                }
                cout << "=== Day One Quadratic interpolation ===\nf(x): " << b0 <<
"+" << b1 << "(x-" << x1 << ")+" << b2 << "(x-" << x1 << ")(x-" << x2 << ") using
linear regression equation" << endl;
                cout << "f(" << x << ") = " << fQ(x, x1, x2, x3) << endl;
                cout << "Error: " << (fabs(fQ(x, x1, x2, x3)-data[x-1])/data[x-
1])*100 << "%" << endl;

                // Average error finding
                for(int i=0; i<arraySize; i++){
                        if(mins[i] % 2 == 0){
                                error += (fabs(fQ(x, x1, x2, x3)-
data[i])/data[i])*100;
                        }
                }

                error /= (arraySize/2);
                cout << "Average error: " << error << endl;

                // Quadratic Interpolation from non linear regression
                b0 = f3(x1);
                b1 = (f3(x2)-f3(x1))/(x2-x1);
                b2 = (((f3(x3)-f3(x2))/(x3-x2))-((f3(x2)-f3(x1))/(x2-x1)))/(x3-x1);
                if(b2 < 1.e-7){
                        b2 = 0;
                }
                cout << "f(x): " << b0 << "+" << b1 << "(x-" << x1 << ")+" << b2 <<
"(x-" << x1 << ")(x-" << x2 << ") using non linear regression equation" << endl;
```

```
                cout << "f(" << x << ") = " << fQ(x, x1, x2, x3) << endl;
                cout << "Error: " << (fabs(fQ(x, x1, x2, x3)-data[x-1])/data[x-
1])*100 << "%" << endl;

                // Average error finding
                for(int i=0; i<arraySize; i++){
                        if(mins[i] % 2 == 0){
                                error += (fabs(fQ(x, x1, x2, x3)-
data[i])/data[i])*100;
                        }
                }

                error /= (arraySize/2);
                cout << "Average error: " << error << endl;
        } else if(name == "Day Two" && x > x1 && x2 > x1 && x3 > x2 &&
binary_search(mins, mins + arraySize, x1) && binary_search(mins, mins + arraySize, x2)
&& binary_search(mins, mins + arraySize, x3)){
                // Day two linear and nonlinear interpolation
                b0 = f2(x1);
                b1 = (f2(x2)-f2(x1))/(x2-x1);
                b2 = (((f2(x3)-f2(x2))/(x3-x2))-((f2(x2)-f2(x1))/(x2-x1)))/(x3-x1);
                if(b2 < 1.e-7){
                        b2 = 0;
                }
                cout << "=== Day Two Quadratic interpolation ===\nf(x): " << b0 <<
"+" << b1 << "(x-" << x1 << ")+" << b2 << "(x-" << x1 << ")(x-" << x2 << ") using
linear regression equation" << endl;
                cout << "f(" << x << ") = " << fQ(x, x1, x2, x3) << endl;
                cout << "Error: " << (fabs(fQ(x, x1, x2, x3)-data[x-1])/data[x-
1])*100 << "%" << endl;

                // Average Error finding
                for(int i=0; i<arraySize; i++){
                        if(mins[i] % 2 == 0){
                                error += (fabs(fQ(x, x1, x2, x3)-
data[i])/data[i])*100;
                        }
                }

                error /= (arraySize/2);
                cout << "Average error: " << error << endl;

                // Quadratic Interpolation from non linear regression
                b0 = f4(x1);
                b1 = (f4(x2)-f4(x1))/(x2-x1);
                b2 = (((f4(x3)-f4(x2))/(x3-x2))-((f4(x2)-f4(x1))/(x2-x1)))/(x3-x1);
                if(b2 < 1.e-7){
                        b2 = 0;
                }
                cout << "f(x): " << b0 << "+" << b1 << "(x-" << x1 << ")+" << b2 <<
"(x-" << x1 << ")(x-" << x2 << ") using non linear regression equation" << endl;
                cout << "f(" << x << ") = " << fQ(x, x1, x2, x3) << endl;
```

```
		cout << "Error: " << (fabs(fQ(x, x1, x2, x3)-data[x-1])/data[x-
1])*100 << "%" << endl;

		// Average error finding
		for(int i=0; i<arraySize; i++){
			if(mins[i] % 2 == 0){
				error += (fabs(fQ(x, x1, x2, x3)-
data[i])/data[i])*100;
			}
		}

		error /= (arraySize/2);
		cout << "Average error: " << error << endl;
	} else {
		cout << "\nMake sure the x value in f(x) is within the interval of
x1, x2 & x3" << endl;
		return;
	}
	cout << endl;
}

int main(){
	int arraySize;
	float x1, x2, x3;
	int x;

	float dayOne[] =
{34,37,28,16,44,36,37,43,50,22,13,28,41,10,14,27,41,27,23,37,12,19,18,30,33,31,13,24,1
8,36}; // y
	float dayTwo[] =
{12,21,40,36,34,33,17,29.5,28,13,11.5,12,23,27,17,22.5,30.5,26,22,19,13,3,2,49,10,28,4
3,20,9,30}; // y
	int mins[] =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30}; //
x

	cout << "Enter X, X1, X2 and X3:\n";
	cin >> x >> x1 >> x2 >> x3;

	arraySize = sizeof(dayOne) / sizeof(dayOne[0]);

	quadraticInterpolation(dayOne, mins, arraySize, x, x1, x2, x3, "Day One");
	quadraticInterpolation(dayTwo, mins, arraySize, x, x1, x2, x3, "Day Two");
}
```

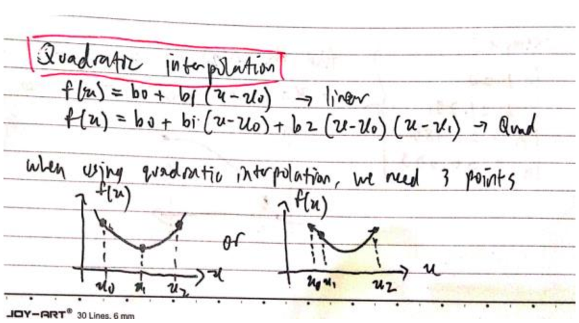
C:\Users\Dell\Documents\Dev C++\Code\qInterpolation.exe

```
Enter X, X1, X2 and X3:
25 1 2 5 30
=== Day One Quadratic interpolation ===
f(x): 38.6083+-0.762501(x-1)+3.12924e-007(x-1)(x-2) using linear regression equation
f(25) = 20.3085
Error: 38.4592%
Average error: 35.5981
f(x): 44.8257+-6.90985(x-1)+1.10018(x-1)(x-2) using non linear regression equation
f(25) = 486.287
Error: 1373.6%
Average error: 1881.55

=== Day Two Quadratic interpolation ===
f(x): 25.7875+-0.356251(x-1)+1.56462e-007(x-1)(x-2) using linear regression equation
f(25) = 17.2375
Error: 72.3752%
Average error: 64.8221
f(x): 27.1735+-3.49811(x-1)+0.545951(x-1)(x-2) using non linear regression equation
f(25) = 244.584
Error: 2345.84%
Average error: 1445.56


--------------------------------
Process exited after 9.527 seconds with return value 0
Press any key to continue . . .
```

Code URL: https://onlinegdb.com/r1pGHJmOO

For quadratic interpolation, we are interested in finding the approximation using a quadratic function at a given set of points such that we can make a line that connects those points and approximate the values that we are interested in. As we can see there are 4 inputs, the first one is the x value that we like to approximate using the quadratic interpolation and the x1, x2, x3 are like the linear interpolation – which is the interval/nodes/x value – that is helpful for finding the approximate at point x. As we can see the error margin is better when we use the Day One quadratic interpolation which has 38% error for f(25).

The formula is obtained through mathematical intuition below.



- find b_0 by putting x_0 & $f(x_0)$

$$f(x_0) = b_0 + b_1\underbrace{(x_0 - x_0)}_{0} + b_2\underbrace{(x_0 - x_0)}_{0}(x_0 - x_1)$$

$$\boxed{f(x_0) = b_0}$$

- find b_1 by putting x_1 to $f(x_1)$

$$f(x_1) = f(x_0) + b_1(x_1 - x_0) + b_2\underbrace{(x_1 - x_0)(x_1 - x_1)}_{0}$$

$$f(x_1) = f(x_0) + b_1(x_1 - x_0)$$

$$\boxed{b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}}$$

- find b_2 by putting x_2 to $f(x_2)$

$$f(x_2) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x_2 - x_0) + b_2(x_2 - x_0)(x_2 - x_1)$$

$$\boxed{b_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}}$$

General polynomial

$$f(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \dots + b_n(x - x_0)(x - x_1)\dots(x - x_{n-1})$$

where n is the highest power of the polynomial

$b_0 = f(x_0)$

$b_1 = f(x_1, x_0)$

$b_2 = f(x_2, x_1, x_0)$

$\vdots$

$b_n = f(x_n, x_{n-1}, \dots x_1, x_0)$

$$b_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$

Below are the handwritten calculation.

Quadratic interpolation

$f(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)$

$f(x_0) = b_0$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

$$b_2 = \frac{\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}}{x_2 - x_0}$$

$f(x) = -0.387987x + 34.0805$

→ Find estimation when $x = 6$ from $x_0 = 5$, $x_1 = 10$, $x_2 = 15$

$b_0 = f(x_0) = -0.387987(5) + 34.0805$

$= 32.140565$

$$b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{f(10) - f(5)}{10 - 5} = \frac{30.20063 - 32.140565}{5}$$

$= -0.387987$

$$b_2 = \frac{\frac{f(15) - f(10)}{15 - 10} - \frac{f(10) - f(5)}{10 - 5}}{15 - 5}$$

No

Date

$$= \frac{\frac{28.260695 - 30.20063}{5} - \frac{30.20063 - 32.140565}{5}}{10}$$

$$= \frac{-0.387987 - (-0.387987)}{10} = 0$$

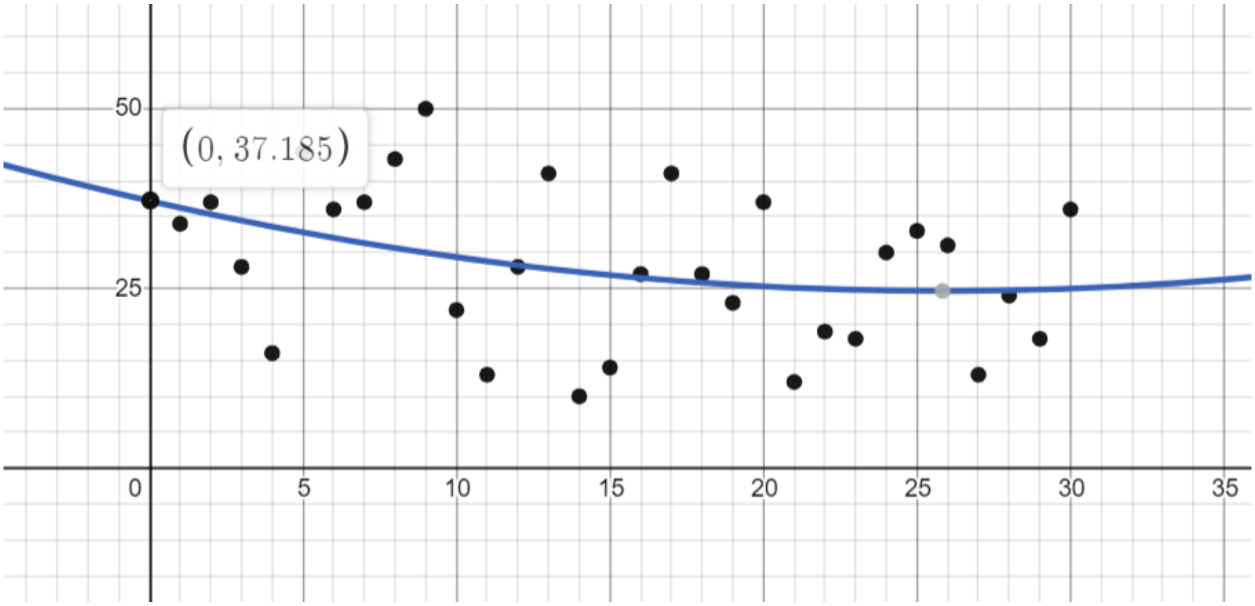
$f(x) = 32.140565 - 0.387987(x - 5)$

$f(6) = 32.140565 - 0.387987(6 - 5)$

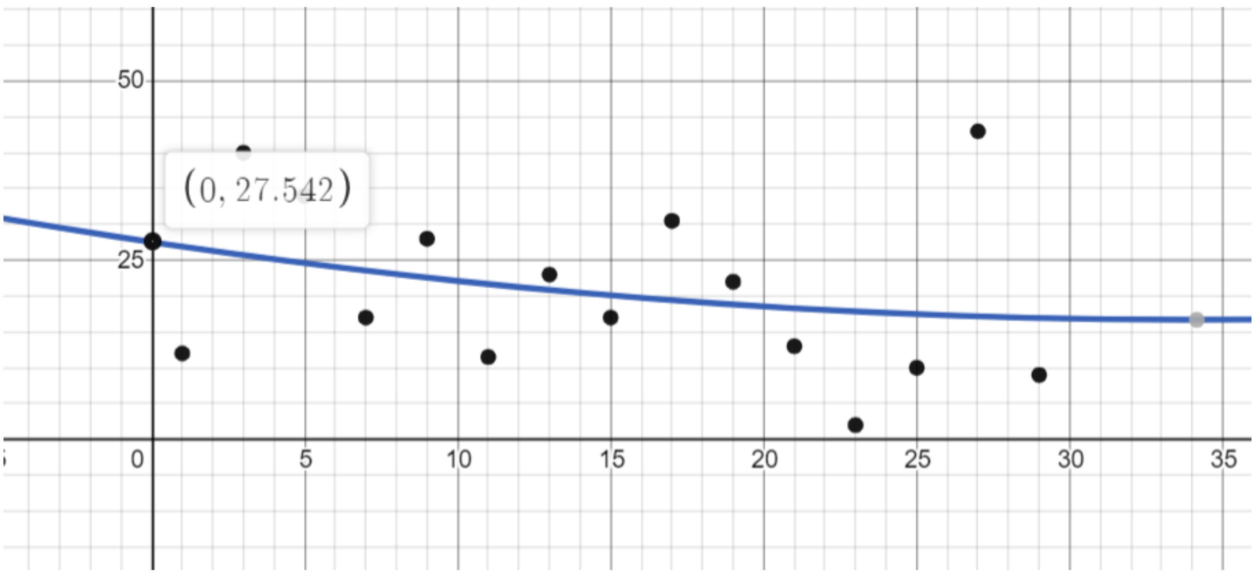
$= 31.752578$

$$\varepsilon = \frac{36 - 31.752578}{36} \times 100\% = \boxed{11.79\%}$$

Quadratic Interpolation Day One graph



Quadratic Interpolation Day Two graph



Attachments

*Day One documentation on 23 April 2021

*Day Two documentation 30 April 2021