

Magnetometer based Navigation Algorithm for a four-wheel Omni drive using concepts of Machine Learning

Anmol Popli

(Member of Team Robocon, IIT ROORKEE)

Objective:

Autonomously moving a four-wheel Omni drive on a defined trajectory using magnetometer.

Approach:

Machine Learning is used to get an equation of the desired trajectory by training the dataset (magnetometer's readings) obtained by moving the bot on the desired path manually.

1. **Obtaining the dataset (magnetometer readings):**

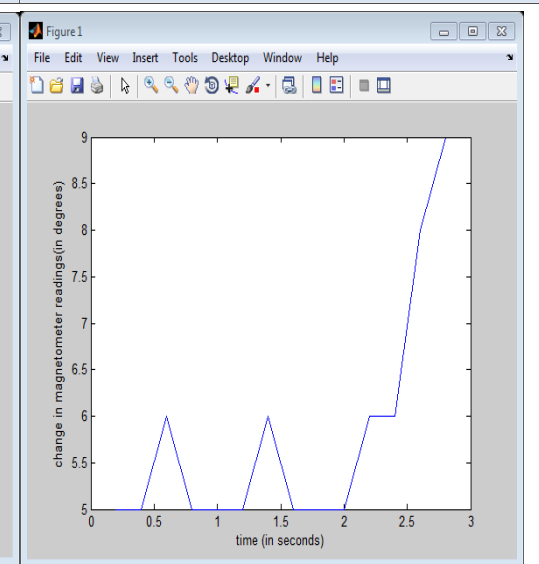
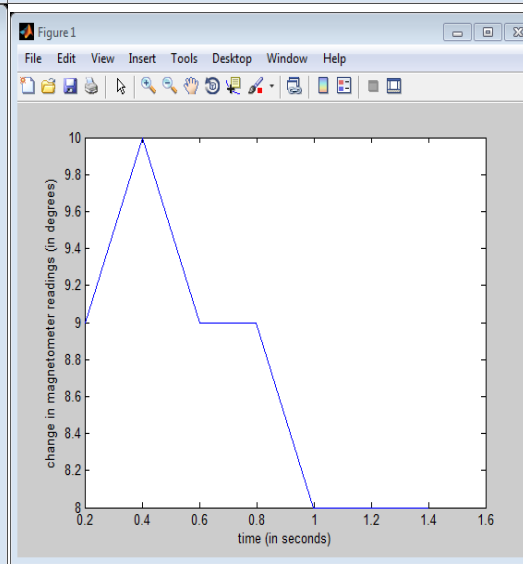
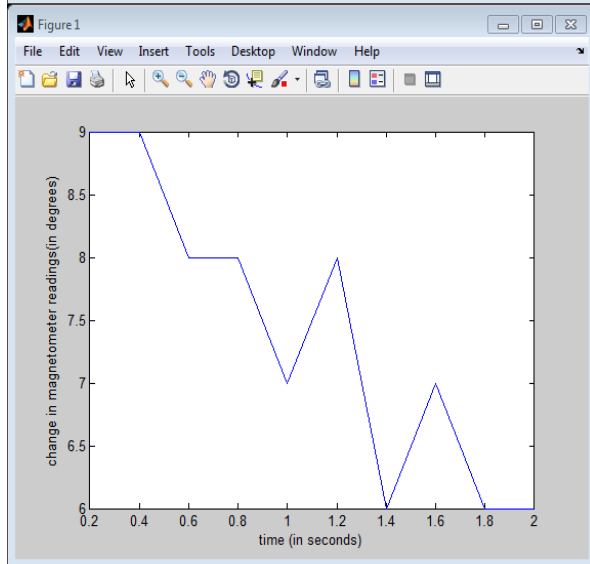
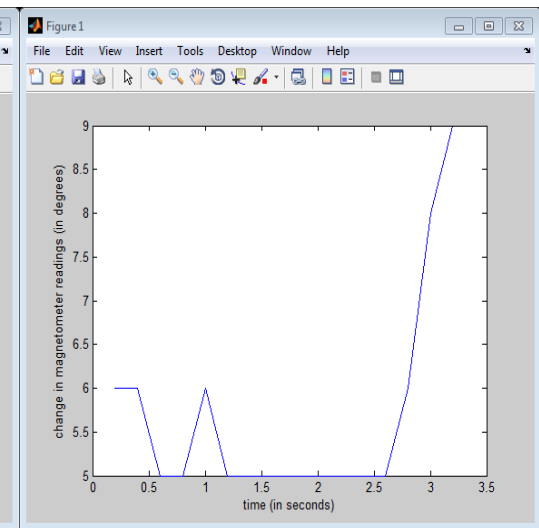
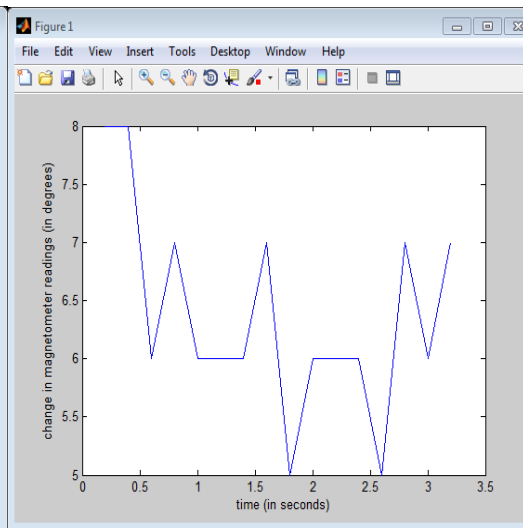
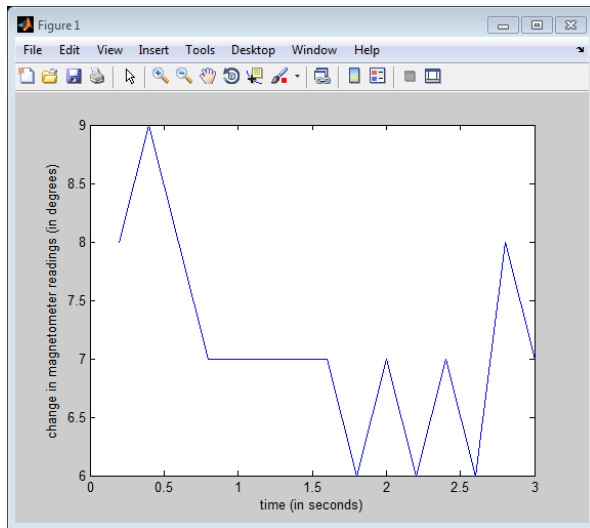
- First, the bot was made to move on a circle of defined radius manually.
- Readings of magnetometer were taken after every 0.2 seconds.
- Every run consists of 15-20 readings, i.e. the bot was made to move manually for about 3-4 seconds.
- Around 18 such runs were taken and data collected is shown below.

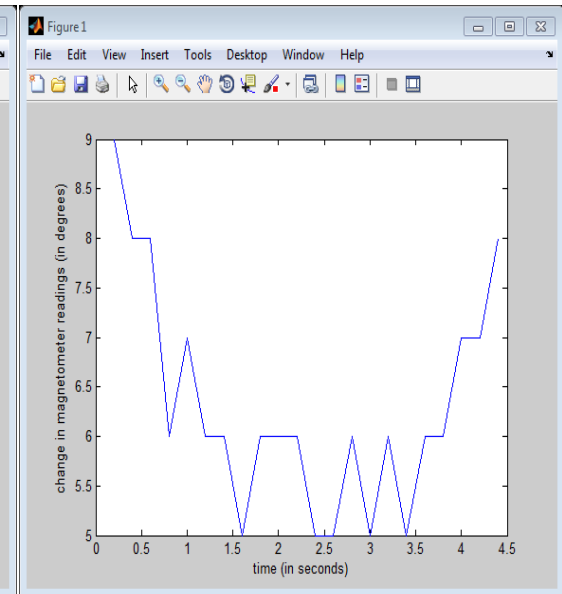
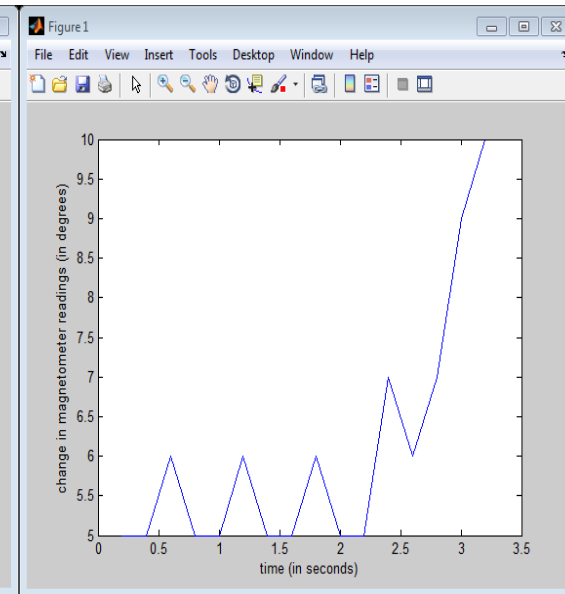
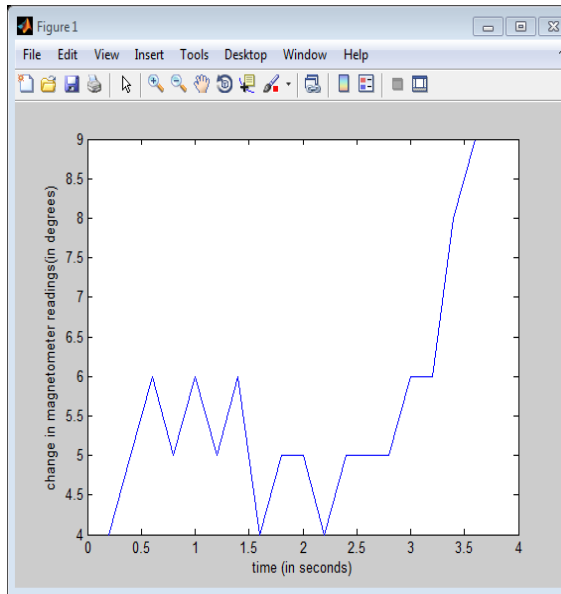
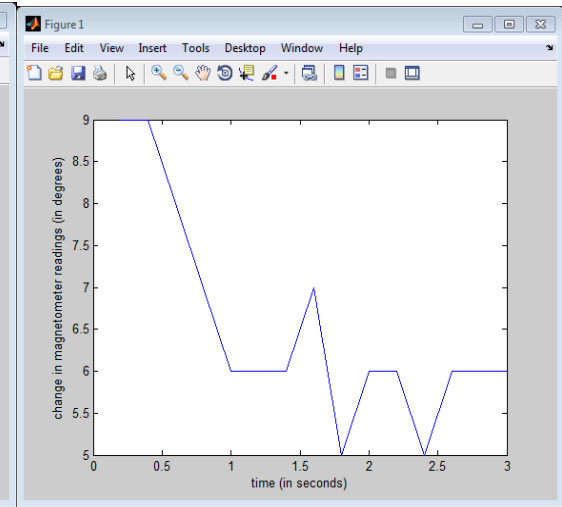
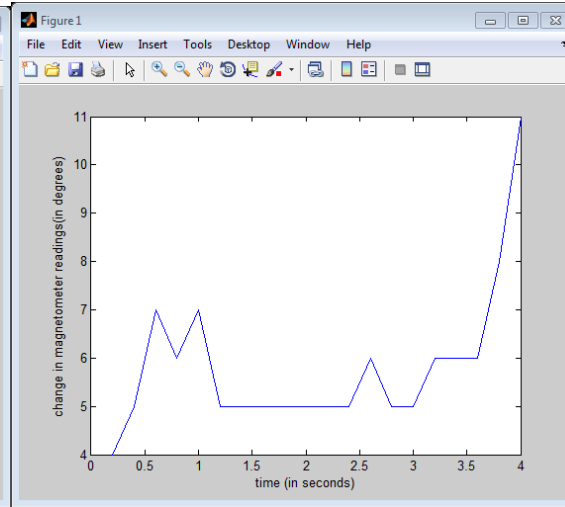
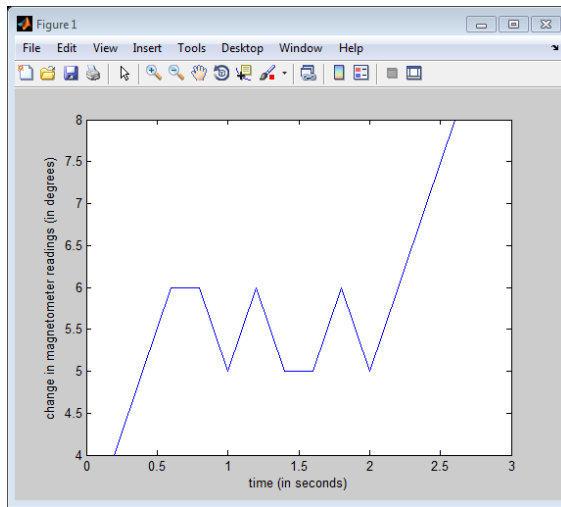
Book1 - Microsoft Excel (Product Activation Failed)																			
File Home Insert Page Layout Formulas Data Review View Add-Ins																			
Clipboard Font Alignment Number Styles Cells Editing																			
Q2 run 15																			
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Time (in seconds)	change in magnetometer's reading after every 0.2 seconds (in degrees)																	
2		run 1	run2	run3	run4	run5	run6	run7	run8	run 9	run10	run 11	run 12	run 13	run 14	run 15	run 16	run 17	
3	0.2	10	2	2	11	11	13	1	2	4	6	3	4	2	8	7	2	5	
4	0.4	16	6	5	26	25	26	5	4	5	10	5	5	5	15	16	5	13	
5	0.6	13	6	7	27	30	29	5	5	7	11	6	6	5	13	22	6	17	
6	0.8	12	5	8	22	25	25	6	6	6	10	6	5	6	12	27	6	22	
7	1	11	5	11	16	20	18	5	6	7	9	7	6	5	12	22	6	29	
8	1.2	10	6	12	13	14	15	5	5	5	9	9	5	5	10	18	6	25	
9	1.4	8	5	16	11	12	11	5	6	5	8	12	6	6	9	15	6	19	
10	1.6	8	5	20	11	10	10	6	5	5	7	11	4	5	8	11	7	17	
11	1.8	6	5	28	11	10	11	5	5	5	6	14	5	5	8	11	8	13	
12	2	7	5	32	9	8	9	5	6	5	6	15	5	6	6	9	8	11	
13	2.2	6	5	26	9	9	10	5	5	5	6	18	4	5	7	10	11	11	
14	2.4	6	5	19	8	8	9	6	6	5	7	23	5	5	6	9	11	10	
15	2.6	6	5	16	8	7	9	6	7	6	5	23	5	7	6	8	17	9	
16	2.8	7	5	13	7	7	8	8	8	5	6	18	5	6	5	8	20	10	
17	3	5	6	11	8	7	8	9	11	5	6	16	6	7	6	8	27	8	
18	3.2	6	8	11	6	7	8	11	13	6	5	15	6	9	6	7	25	8	
19	3.4	6	9	11	7	7		14	19	6	6	13	8	10	6	7	23	7	
20	3.6	6	12	11	6	6		22		6	6	11	9	14	5	6	19	7	
21	3.8	5	18	9	6	7		27		8	6	11	11	17	5	6	17	7	
22	4	7	30	9		6		34		11		12	14	25	6	7	15	7	
23	4.2	6		9		7		28				10	19		5	5	15	7	
24	4.4	7				6						11	22		6	6	14	6	
25	4.6					8						9	28		5	6	14	6	

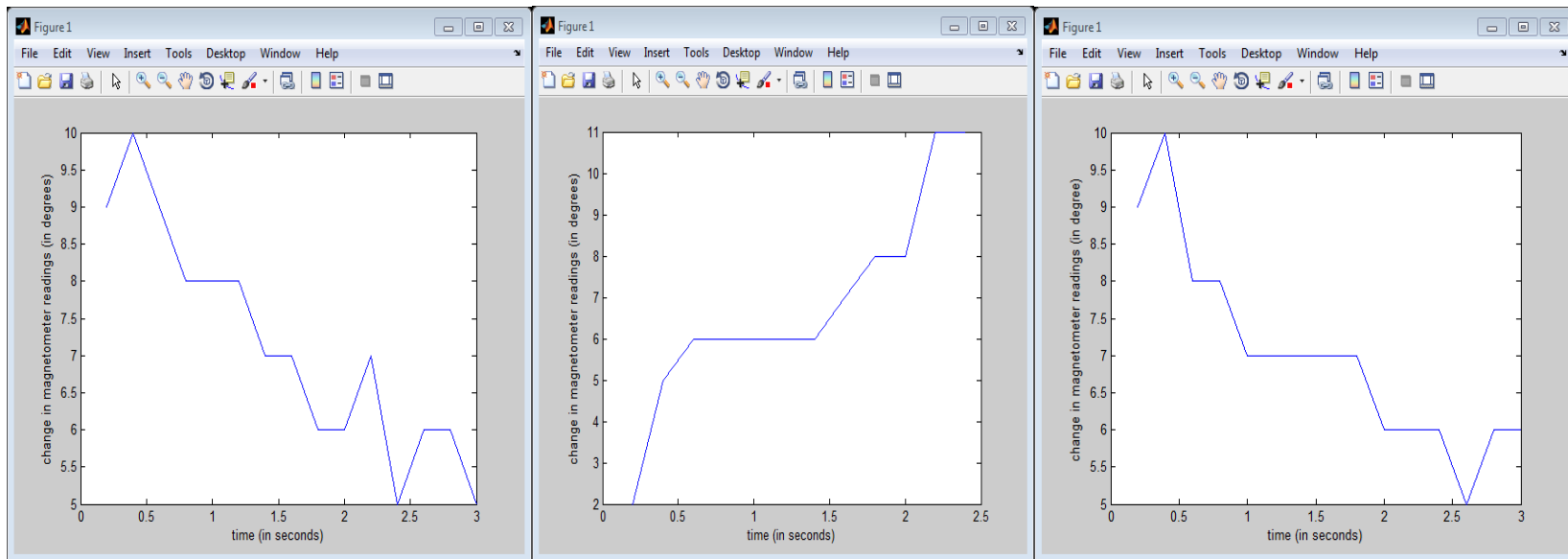
- The above data contains certain spikes at the beginning of the run because the motors doesn't respond to the given PWM instantaneously.
- So, a modified dataset was created as shown below.

Time (in seconds)	run 1	run2	run 3	run 4	run 5	run 6	run 7	run 8	run 9	run 10	run 11	run 12	run 13	run 14	run 15
0.2	8	6	9	8	9	5	4	4	9	4	5	9	9	2	9
0.4	8	6	9	9	10	5	5	5	9	5	5	8	10	5	10
0.6	6	5	8	8	9	6	6	7	8	6	6	8	9	6	8
0.8	7	5	8	7	9	5	6	6	7	5	5	6	8	6	8
1	6	6	7	7	8	5	5	7	6	6	5	7	8	6	7
1.2	6	5	8	7	8	5	6	5	6	5	6	6	8	6	7
1.4	6	5	6	7	8	6	5	5	6	6	5	6	7	6	7
1.6	7	5	7	7		5	5	5	7	4	5	5	7	7	7
1.8	5	5	6	6		5	6	5	5	5	6	6	6	8	7
2	6	5	6	7		5	5	5	6	5	5	6	6	8	6
2.2	6	5		6		6	6	5	6	4	5	6	7	11	6
2.4	6	5		7		6	7	5	5	5	7	5	5	11	6
2.6	5	5		6		8	8	6	6	5	6	5	6		5
2.8	7	6		8		9		5	6	5	7	6	6		6
3	6	8		7				5	6	6	9	5	5		6
3.2	7	9						6		6	10	6			
3.4								6		8		5			
3.6								6		9		6			
3.8								8				6			
4								11				7			
4.2												7			
4.4												8			

- Graphs/plots of these dataset were made using Matlab and are shown below.







2. Curve fitting using machine learning

- Regression algorithm is used to find the best fit to the above plotted graphs.
- Code is written in c++ to find which degree polynomial is most close to the above drawn plots and find its equation.
- Code for linear fit, quadratic fit and cubic fit is written in c++ and is shown below. **(No inbuilt libraries are used for regression algorithm)**

Code for linear fit:

```
#include<iostream>
using namespace std;
int main()
{
    double x_1[]={16.0,8.0,8.0,6.0,7.0,6.0,6.0,6.0,7.0,5.0,6.0,6.0,6.0,5.0,7.0,6.0,7.0};
    double x_2[]={16.0,6.0,6.0,5.0,5.0,6.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,6.0,8.0,9.0};
```

```

double x_3[]={10.0,9.0,9.0,8.0,8.0,7.0,8.0,6.0,7.0,6.0,6.0};
double x_4[]={15.0,8.0,9.0,8.0,7.0,7.0,7.0,7.0,6.0,7.0,6.0,7.0,6.0,8.0,7.0};
double x_5[]={7.0,9.0,10.0,9.0,9.0,8.0,8.0,8.0};
double x_6[]={14.0,5.0,5.0,6.0,5.0,5.0,5.0,6.0,5.0,5.0,5.0,6.0,6.0,8.0,9.0};
double x_7[]={13.0,4.0,5.0,6.0,6.0,5.0,6.0,5.0,5.0,6.0,5.0,6.0,7.0,8.0};
double x_8[]={20.0,4.0,5.0,7.0,6.0,7.0,5.0,5.0,5.0,5.0,5.0,5.0,6.0,5.0,5.0,6.0,6.0,6.0,8.0,11.0};
double x_9[]={15.0,9.0,9.0,8.0,7.0,6.0,6.0,6.0,7.0,5.0,6.0,6.0,5.0,6.0,6.0,6.0};
double x_10[]={18.0,4.0,5.0,6.0,5.0,6.0,5.0,6.0,4.0,5.0,5.0,4.0,5.0,5.0,5.0,6.0,6.0,8.0,9.0};
double x_11[]={16.0,5.0,5.0,6.0,5.0,5.0,6.0,5.0,5.0,6.0,5.0,5.0,7.0,6.0,7.0,9.0,10.0};
double x_12[]={22.0,9.0,8.0,8.0,6.0,7.0,6.0,6.0,5.0,6.0,6.0,6.0,5.0,5.0,6.0,5.0,6.0,5.0,6.0,6.0,7.0,7.0,8.0};
double x_13[]={15.0,9.0,10.0,9.0,8.0,8.0,8.0,7.0,7.0,6.0,6.0,7.0,5.0,6.0,6.0,5.0};
double x_14[]={12.0,2.0,5.0,6.0,6.0,6.0,6.0,6.0,7.0,8.0,8.0,11.0,11.0};
double x_15[]={15.0,9.0,10.0,8.0,8.0,7.0,7.0,7.0,7.0,6.0,6.0,6.0,5.0,6.0,6.0};

```

```

double theta_1=1;
double theta_0=1;
double alpha=0.1;
int iterations=5000;
double sum1=0,sum2=0;
for(int i1=1;i1<=iterations;i1++)
{
    sum1=0;sum2=0;
    for(int i=1;i<=int(x_1[0]);i++)
    {
        sum1=sum1+(theta_1*0.2*i+theta_0-x_1[i])*0.2*i;
        sum2=sum2+(theta_1*0.2*i+theta_0-x_1[i]);
    }
    for(int i=1;i<=int(x_2[0]);i++)
    {
        sum1=sum1+(theta_1*0.2*i+theta_0-x_2[i])*0.2*i;
    }
}

```

```
    sum2=sum2+(theta_1*0.2*i+theta_0-x_2[i]);
}
for(int i=1;i<=int(x_3[0]);i++)
{
    sum1=sum1+(theta_1*0.2*i+theta_0-x_3[i])*0.2*i;
    sum2=sum2+(theta_1*0.2*i+theta_0-x_3[i]);
}
for(int i=1;i<=int(x_4[0]);i++)
{
    sum1=sum1+(theta_1*0.2*i+theta_0-x_4[i])*0.2*i;
    sum2=sum2+(theta_1*0.2*i+theta_0-x_4[i]);
}
for(int i=1;i<=int(x_5[0]);i++)
{
    sum1=sum1+(theta_1*0.2*i+theta_0-x_5[i])*0.2*i;
    sum2=sum2+(theta_1*0.2*i+theta_0-x_5[i]);
}
for(int i=1;i<=int(x_6[0]);i++)
{
    sum1=sum1+(theta_1*0.2*i+theta_0-x_6[i])*0.2*i;
    sum2=sum2+(theta_1*0.2*i+theta_0-x_6[i]);
}
for(int i=1;i<=int(x_7[0]);i++)
{
    sum1=sum1+(theta_1*0.2*i+theta_0-x_7[i])*0.2*i;
    sum2=sum2+(theta_1*0.2*i+theta_0-x_7[i]);
}
for(int i=1;i<=int(x_8[0]);i++)
{
    sum1=sum1+(theta_1*0.2*i+theta_0-x_8[i])*0.2*i;
```



```

        sum2=sum2+(theta_1*0.2*i+theta_0-x_8[i]);
    }
    for(int i=1;i<=int(x_9[0]);i++)
    {
        sum1=sum1+(theta_1*0.2*i+theta_0-x_9[i])*0.2*i;
        sum2=sum2+(theta_1*0.2*i+theta_0-x_9[i]);
    }
    for(int i=1;i<=int(x_10[0]);i++)
    {
        sum1=sum1+(theta_1*0.2*i+theta_0-x_10[i])*0.2*i;
        sum2=sum2+(theta_1*0.2*i+theta_0-x_10[i]);
    }
    for(int i=1;i<=int(x_11[0]);i++)
    {
        sum1=sum1+(theta_1*0.2*i+theta_0-x_11[i])*0.2*i;
        sum2=sum2+(theta_1*0.2*i+theta_0-x_11[i]);
    }
    for(int i=1;i<=int(x_12[0]);i++)
    {
        sum1=sum1+(theta_1*0.2*i+theta_0-x_12[i])*0.2*i;
        sum2=sum2+(theta_1*0.2*i+theta_0-x_12[i]);
    }
    sum2=sum2+theta_0;
    theta_1=theta_1-2*alpha/224.0*sum1;
    theta_0=theta_0-2*alpha/224.0*sum2;

}
cout<<"predicted values...."<<endl;
for(int i=1;i<20;i++)

```

```

{
    cout<<int(theta_1*0.2*i+theta_0+0.5)<<endl;
}
/* cout<<theta_1<<endl;
   cout<<theta_0<<endl; */
system("pause");
return 0;
}

```

Code for quadratic fit:

```

#include<iostream>
using namespace std;
int main()
{
    double x_1[]={16.0,8.0,8.0,6.0,7.0,6.0,6.0,6.0,7.0,5.0,6.0,6.0,6.0,5.0,7.0,6.0,7.0};
    double x_2[]={16.0,6.0,6.0,5.0,5.0,6.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,6.0,8.0,9.0};
    double x_3[]={10.0,9.0,9.0,8.0,8.0,7.0,8.0,6.0,7.0,6.0,6.0};
    double x_4[]={15.0,8.0,9.0,8.0,7.0,7.0,7.0,7.0,6.0,7.0,6.0,7.0,6.0,8.0,7.0};
    double x_5[]={7.0,9.0,10.0,9.0,9.0,8.0,8.0,8.0};
    double x_6[]={14.0,5.0,5.0,6.0,5.0,5.0,5.0,6.0,5.0,5.0,5.0,6.0,6.0,8.0,9.0};
    double x_7[]={13.0,4.0,5.0,6.0,6.0,5.0,6.0,5.0,5.0,6.0,5.0,6.0,7.0,8.0};
    double x_8[]={20.0,4.0,5.0,7.0,6.0,7.0,5.0,5.0,5.0,5.0,5.0,5.0,6.0,5.0,5.0,6.0,6.0,8.0,11.0};
    double x_9[]={15.0,9.0,9.0,8.0,7.0,6.0,6.0,6.0,7.0,5.0,6.0,6.0,5.0,6.0,6.0,6.0};
    double x_10[]={18.0,4.0,5.0,6.0,5.0,6.0,5.0,6.0,4.0,5.0,5.0,4.0,5.0,5.0,5.0,6.0,6.0,8.0,9.0};
    double x_11[]={16.0,5.0,5.0,6.0,5.0,5.0,6.0,5.0,5.0,6.0,5.0,5.0,7.0,6.0,7.0,9.0,10.0};
    double x_12[]={22.0,9.0,8.0,8.0,6.0,7.0,6.0,6.0,5.0,6.0,6.0,6.0,5.0,5.0,6.0,5.0,6.0,6.0,7.0,7.0,8.0};
    double x_13[]={15.0,9.0,10.0,9.0,8.0,8.0,8.0,7.0,7.0,6.0,6.0,7.0,5.0,6.0,6.0,5.0};
    double x_14[]={12.0,2.0,5.0,6.0,6.0,6.0,6.0,6.0,7.0,8.0,8.0,11.0,11.0};
    double x_15[]={15.0,9.0,10.0,8.0,8.0,7.0,7.0,7.0,7.0,6.0,6.0,6.0,5.0,6.0,6.0};
    double theta_2=1.0;
}

```

```

double theta_1=1;
double theta_0=1;
double alpha=0.007;
int iterations=500000;
double sum1=0,sum2=0,sum3=0;
for(int i1=1;i1<=iterations;i1++)
{
    sum1=0;sum2=0;sum3=0;
    for(int i=1;i<=int(x_1[0]);i++)
    {
        sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_1[i])*0.2*i*0.2*i;
        sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_1[i])*0.2*i;
        sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_1[i]);
    }
    for(int i=1;i<=int(x_2[0]);i++)
    {
        sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_2[i])*0.2*i*0.2*i;
        sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_2[i])*0.2*i;
        sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_2[i]);
    }
    for(int i=1;i<=int(x_3[0]);i++)
    {
        sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_3[i])*0.2*i*0.2*i;
        sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_3[i])*0.2*i;
        sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_3[i]);
    }
    for(int i=1;i<=int(x_4[0]);i++)
    {
        sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_4[i])*0.2*i*0.2*i;
        sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_4[i])*0.2*i;
    }
}

```

```

    sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_4[i]);
}
for(int i=1;i<=int(x_5[0]);i++)
{
    sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_5[i])*0.2*i*0.2*i;
    sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_5[i])*0.2*i;
    sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_5[i]);
}
for(int i=1;i<=int(x_6[0]);i++)
{
    sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_6[i])*0.2*i*0.2*i;
    sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_6[i])*0.2*i;
    sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_6[i]);
}
for(int i=1;i<=int(x_7[0]);i++)
{
    sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_7[i])*0.2*i*0.2*i;
    sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_7[i])*0.2*i;
    sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_7[i]);
}
for(int i=1;i<=int(x_8[0]);i++)
{
    sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_8[i])*0.2*i*0.2*i;
    sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_8[i])*0.2*i;
    sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_8[i]);
}
for(int i=1;i<=int(x_9[0]);i++)
{
    sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_9[i])*0.2*i*0.2*i;
    sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_9[i])*0.2*i;

```

```

        sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_9[i]);
    }
    for(int i=1;i<=int(x_10[0]);i++)
    {
        sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_10[i])*0.2*i*0.2*i;
        sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_10[i])*0.2*i;
        sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_10[i]);
    }
    for(int i=1;i<=int(x_11[0]);i++)
    {
        sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_11[i])*0.2*i*0.2*i;
        sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_11[i])*0.2*i;
        sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_11[i]);
    }
    for(int i=1;i<=int(x_12[0]);i++)
    {
        sum3=sum3+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_12[i])*0.2*i*0.2*i;
        sum1=sum1+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_12[i])*0.2*i;
        sum2=sum2+(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_12[i]);
    }
    sum2=sum2+theta_0;
    theta_2=theta_2-2*alpha/224.0*sum3;
    theta_1=theta_1-2*alpha/224.0*sum1;
    theta_0=theta_0-2*alpha/224.0*sum2;

}
cout<<"predicted values...."<<endl;
for(int i=1;i<20;i++)
{

```

```

        cout<<int(theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0+0.5)<<endl;
    }
    cout<<" "<<endl;
    cout<<theta_2<<endl;
    cout<<theta_1<<endl;
    cout<<theta_0<<endl;
    system("pause");
    return 0;
}

```

Code for cubic fit:

```

#include<iostream>
using namespace std;
int main()
{
    double x_1[]={16.0,8.0,8.0,6.0,7.0,6.0,6.0,6.0,7.0,5.0,6.0,6.0,6.0,5.0,7.0,6.0,7.0};
    double x_2[]={16.0,6.0,6.0,5.0,5.0,6.0,5.0,5.0,5.0,5.0,5.0,5.0,5.0,6.0,8.0,9.0};
    double x_3[]={10.0,9.0,9.0,8.0,8.0,7.0,8.0,6.0,7.0,6.0,6.0};
    double x_4[]={15.0,8.0,9.0,8.0,7.0,7.0,7.0,7.0,6.0,7.0,6.0,7.0,6.0,8.0,7.0};
    double x_5[]={7.0,9.0,10.0,9.0,9.0,8.0,8.0,8.0};
    double x_6[]={14.0,5.0,5.0,6.0,5.0,5.0,5.0,6.0,5.0,5.0,5.0,6.0,6.0,8.0,9.0};
    double x_7[]={13.0,4.0,5.0,6.0,6.0,5.0,6.0,5.0,5.0,6.0,5.0,6.0,7.0,8.0};
    double x_8[]={20.0,4.0,5.0,7.0,6.0,7.0,5.0,5.0,5.0,5.0,5.0,5.0,6.0,5.0,6.0,6.0,6.0,8.0,11.0};
    double x_9[]={15.0,9.0,9.0,8.0,7.0,6.0,6.0,6.0,7.0,5.0,6.0,6.0,5.0,6.0,6.0,6.0};
    double x_10[]={18.0,4.0,5.0,6.0,5.0,6.0,5.0,6.0,4.0,5.0,5.0,4.0,5.0,5.0,5.0,6.0,6.0,8.0,9.0};
    double x_11[]={16.0,5.0,5.0,6.0,5.0,5.0,6.0,5.0,5.0,6.0,5.0,5.0,7.0,6.0,7.0,9.0,10.0};
    double x_12[]={22.0,9.0,8.0,8.0,6.0,7.0,6.0,6.0,5.0,6.0,6.0,6.0,5.0,5.0,6.0,5.0,6.0,6.0,7.0,7.0,8.0};
    double x_13[]={15.0,9.0,10.0,9.0,8.0,8.0,8.0,7.0,7.0,6.0,6.0,7.0,5.0,6.0,6.0,5.0};
    double x_14[]={12.0,2.0,5.0,6.0,6.0,6.0,6.0,6.0,7.0,8.0,8.0,11.0,11.0};
    double x_15[]={15.0,9.0,10.0,8.0,8.0,7.0,7.0,7.0,7.0,6.0,6.0,6.0,5.0,6.0,6.0};
}

```

```

double theta_3=1;
double theta_2=1.0;
double theta_1=1;
double theta_0=1;
double alpha=0.00000007;
int iterations=500000;
double sum1=0,sum2=0,sum3=0,sum4=0;
for(int i1=1;i1<=iterations;i1++)
{
    sum1=0;sum2=0,sum3=0,sum4=0;
    for(int i=1;i<=int(x_1[0]);i++)
    {

        sum4=sum4+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_1[i])*0.2*i*0.2*i*0.2*i;
        sum3=sum3+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_1[i])*0.2*i*0.2*i;
        sum2=sum2+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_1[i])*0.2*i;
        sum1=sum1+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_1[i]);
    }
    for(int i=1;i<=int(x_2[0]);i++)
    {
        sum4=sum4+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_2[i])*0.2*i*0.2*i*0.2*i;
        sum3=sum3+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_2[i])*0.2*i*0.2*i;
        sum2=sum2+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_2[i])*0.2*i;
        sum1=sum1+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_2[i]);
    }
    for(int i=1;i<=int(x_3[0]);i++)
    {
        sum4=sum4+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_3[i])*0.2*i*0.2*i*0.2*i;
        sum3=sum3+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_3[i])*0.2*i*0.2*i;
        sum2=sum2+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_3[i])*0.2*i;
    }
}

```

```

    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_3[i]);
}
for(int i=1;i<=int(x_4[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_4[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_4[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_4[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_4[i]);
}
for(int i=1;i<=int(x_5[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_5[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_5[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_5[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_5[i]);
}
for(int i=1;i<=int(x_6[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_6[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_6[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_6[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_6[i]);
}
for(int i=1;i<=int(x_7[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_7[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_7[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_7[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_7[i]);
}

```



```

for(int i=1;i<=int(x_8[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_8[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_8[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_8[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_8[i]);
}
for(int i=1;i<=int(x_9[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_9[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_9[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_9[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_9[i]);
}
for(int i=1;i<=int(x_10[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_10[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_10[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_10[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_10[i]);
}
for(int i=1;i<=int(x_11[0]);i++)
{
    sum4=sum4+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_11[i])*0.2*i*0.2*i*0.2*i;
    sum3=sum3+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_11[i])*0.2*i*0.2*i;
    sum2=sum2+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_11[i])*0.2*i;
    sum1=sum1+(theta_3*0.2*0.2*0.2*i*i+i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_11[i]);
}
for(int i=1;i<=int(x_12[0]);i++)
{

```

```

        sum4=sum4+(theta_3*0.2*0.2*0.2*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_12[i])*0.2*i*0.2*i*0.2*i;
        sum3=sum3+(theta_3*0.2*0.2*0.2*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_12[i])*0.2*i*0.2*i;
        sum2=sum2+(theta_3*0.2*0.2*0.2*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_12[i])*0.2*i;
        sum1=sum1+(theta_3*0.2*0.2*0.2*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0-x_12[i]);
    }
    sum2=sum2+theta_0;
    theta_3=theta_3-2*alpha/224.0*sum4;
    theta_2=theta_2-2*alpha/224.0*sum3;
    theta_1=theta_1-2*alpha/224.0*sum1;
    theta_0=theta_0-2*alpha/224.0*sum2;

}
cout<<"predicted values...."<<endl;
for(int i=1;i<20;i++)
{
    cout<<int(theta_3*0.2*0.2*0.2*i*i+theta_2*(0.2*i)*(0.2*i)+theta_1*0.2*i+theta_0+0.5)<<endl;
}
/* cout<<theta_2<<endl;
cout<<theta_1<<endl;
cout<<theta_0<<endl; */
system("pause");
return 0;
}

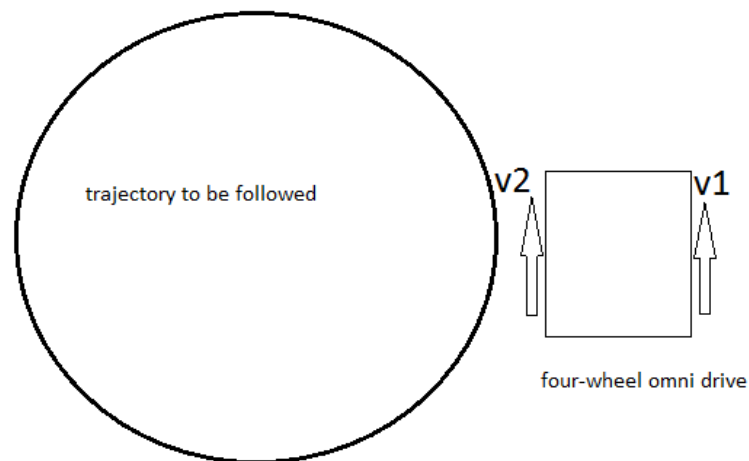
```

- Quadratic fit gave the closest approximation and its equation is $f(t)=0.398979*t^2-1.42948*t+7.08966$ where t =time in seconds and $f(t)$ is the change in magnetometer reading at time= t .
- This equation is used by arduino to make the bot move in desired trajectory.

3. Moving the bot on desired path using the equation obtained from above c++ code

- Rather than using the typical PID algorithm for navigation, new method is developed and is explained below:

Derivation:



- Omni chassis is made to follow the circle by keeping the two opposite facing wheels stall and moving the rest two wheels at velocity **v_1 and v_2** .
- Let the length of bot be ' l ' and radius of the trajectory is ' r '.
$$v_2/r = v_1/(r+l) \quad -1$$

After a time interval of $t=0.2$ seconds:

$$V_2 * t = r * (\text{change in magnetometer reading}) / 2/\pi \quad -2$$

From equation '1':

$$r = v_2 * l / (v_1 - v_2)$$

from equation '2':

$$v_2 \cdot t = v_2 \cdot l / (v_1 - v_2) \cdot (\text{change in magnetometer reading}) / 2 / \pi$$

dividing the entire equation by v_2 :

$$v_1 / v_2 = 1 + (\text{change in magnetometer reading}) \cdot c \quad \text{where } c = l / t / 2 / \pi \text{ is a constant}$$

$v_1 / v_2 = 1 + c \cdot (\text{change in magnetometer reading})$ is the equation we are using for moving the bot on a circle.

- At every time step, value of c is updated by the equation :
 $C(\text{present}) = c(\text{past}) \cdot (\text{value of the quadratic equation at time} = t) / (\text{present magnetometer reading} - \text{past magnetometer reading})$

And this time step is chosen to be 0.2 seconds.

- If $(\text{present magnetometer reading} - \text{past magnetometer reading}) > \text{desired change in magnetometer reading given by the quadratic equation}$, then value of v_1 is decreased keeping v_2 constant.
- If $(\text{present magnetometer reading} - \text{past magnetometer reading}) < \text{desired change in magnetometer reading given by the quadratic equation}$, then value of v_1 is increased keeping v_2 constant.
- If $(\text{present magnetometer reading} - \text{past magnetometer reading}) = \text{desired change in magnetometer reading given by the quadratic equation}$, then value of v_1 and v_2 is kept constant.
- This equation gives the bot an auto-correct feature which helps it during the motion

4. Conclusion

This approach is successfully implemented for navigation on a circular track but could be used for moving on any desired trajectory.

END

