# 实习三实习报告

## 实习要求

对以下数据进行最小二乘法计算

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| t | 20 | 25 | 30 | 35 | 40 | 45 | 50 | 55 | 60 |
| y | 805 | 985 | 1170 | 1365 | 1570 | 1790 | 2030 | 2300 | 2610 |

编程求出三次的表达式，并使用所得到的表达式通过带入t的实际的值计算结果，与实际的y进行比较，得出误差值输出出来，并通过软件画出模拟出来的散点图和趋势图。

## 算法描述

1. 输入数组x和y，将其中的每一个数的0,1,2,3次方储存到一个数组里，然后通过内积运算得到数组G。
2. 再通过内积运算利用y数组求出G。
3. 对于Ga=b方程组使用高斯消元法求解，实际上是扩充了一个三维数组方便计算，利用书上的公式以及循环即可求出a值。
4. 输出表达式，利用表达式带回计算误差。

## 程序清单

#include<iostream>

#include<cmath>

using namespace std;

double jieguo[4];

void jisuan(double x[],double y[])

{

int m=9,n=4;

double temp[11][5];

for(int i=1;i<=m;i++)

{

temp[i][1]=1;

temp[i][2]=x[i];

temp[i][3]=x[i]\*x[i];

temp[i][4]=x[i]\*x[i]\*x[i];

}//用于构造G

double G[5][5],b[5];

for(int i=1;i<=n;i++)

{

b[i]=0.0;

for(int k=1;k<=n;k++)

{

G[i][k]=0.0;

}

}//数组初始化

for(int i=1;i<=n;i++)

{

for(int j=1;j<=n;j++)

{

for(int k=1;k<=m;k++)

{

G[i][j]=G[i][j]+temp[k][i]\*temp[k][j];

}

}

}

for(int i=1;i<=n;i++)

{

for(int k=1;k<=m;k++)

{

b[i]=b[i]+temp[k][i]\*y[k];

}

}//至此完成了G和b的构造，接下来gauss计算即可

/\*for(int i=1;i<=n;i++)

{

for(int k=1;k<=n;k++)

{

cout<<G[i][k]<<" ";

}

cout<<b[i]<<endl;

}\*/

double G1[5][5][5],temp1[5][5];

double b1[5][5];

for(int i=1;i<=n;i++)

{

b1[1][i]=b[i];

for(int k=1;k<=n;k++)

{

G1[1][i][k]=G[i][k];

}

}

for(int k=1;k<=n-1;k++)

{

for(int i=k+1;i<=n;i++)

{

temp1[i][k]=G1[k][i][k]/G1[k][k][k];

}

for(int i=k+1;i<=n;i++)

{

b1[k+1][i]=b1[k][i]-temp1[i][k]\*b1[k][k];

}

for(int i=k+1;i<=n;i++)

{

for(int j=k+1;j<=n;j++)

{

G1[k+1][i][j]=G1[k][i][j]-temp1[i][k]\*G1[k][k][j];

}

}

}//消元计算

jieguo[4]=b1[4][4]/G1[4][4][4];

for(int i=n-1;i>=1;i--)

{

double sum=0.0;

for(int j=i+1;j<=n;j++)

{

sum=sum+G1[i][i][j]\*jieguo[j];

}

jieguo[i]=(b1[i][i]-sum)/G1[i][i][i];

}

}

double guji(double t)

{

double sum=0.0;

double h[5];

h[1]=1.0;

h[2]=t;

h[3]=t\*t;

h[4]=t\*t\*t;

for(int i=4;i>=1;i--)

{

sum=sum+h[i]\*jieguo[i];

}

return sum;

}

int main()

{

double x[]={0.0,20.0,25.0,30.0,35.0,40.0,45.0,50.0,55.0,60.0};

double y[]={0.0,805.0,985.0,1170.0,1365.0,1570.0,1790.0,2030.0,2300.0,2610.0};

jisuan(x,y);

/\*for(int i=1;i<=4;i++)

{

cout<<jieguo[i]<<endl;

}\*/

for(int i=4;i>=1;i--)

{

if(jieguo[i]>0&&i!=4)

{

cout<<"+";

}

if(i==1)

{

cout<<jieguo[i];

}

if(i!=1)

{

cout<<jieguo[i]<<"x^"<<(i-1);

}

}

cout<<endl;

for(int i=1;i<=9;i++)

{

cout<<x[i]<<": "<<guji(x[i])<<" 误差为 ："<<(y[i]-guji(x[i]))<<endl;

}

system("pause");

return 0;

}

运行结果：

运行的拟合结果较好，误差也比较小，计算结果较好。

## 体会与问题

1. 编程时注意数组默认从0开始，我一开始的时候默认其从一开始对于下标未做调整，导致结果出现乱码。
2. 这种通过求取矩阵并进行高斯消元的算法是具有普适性的，并不是像一些方法只适用于线性方程，这个方法同样适用于高次多项式方程或者其他方程。
3. Excel模拟方程的图像也很方便，画出散点图之后添加趋势线即可。
4. 求解矩阵方程的时候，可以选择高斯消元法也可以选择矩阵分解之后的解法，不过高斯消元法的实现起来更加方便，两者在时间渐进复杂度上一致，而在这种小规模数据的情况下时间基本一样，因而建议使用高斯消元法。