# 《误差理论与测量平差基础》设计报告



## 所属学院： 土木工程学院

**专业班级： 测绘01班**

**指导老师： 刘文谷**

# 一.设计名称

基于c#的水准网间接平差程序

# 二.程序说明

该程序为基于c#的界面交互式程序，可以通过该程序计算出单线式具有n个未知点的导线网模型，且可以通过界面快速给出平差值和精度评定。

# 三.程序设计思路

1.根据已知条件，输入观测数n，必要观测数t，已知点数s，形成的闭合环数d。由输入者建立附和与闭合线路，得到方程组，再根据方程组向数据框中依次输入由各个未知点高程，已知点，高差观测值，观测路程，再读入程序内部并显示。

2.再向程序框中输入组成闭合环的高差观测值的系数（1或-1），输入待评定的点以及高差段，点击精度评定，讨论其全中误差是否超限（没有往返测）。

3.最后在语句框中输出精度评定结果和改正参数值。

# 四.程序内容

代码如下：

矩阵运算：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace pingcha

{

class Matrix

{

double[,] A;

int m, n;

string name;

public Matrix(int am, int an)

{

m = am;

n = an;

A = new double[m, n];

name = "Result";

}

public Matrix(int am, int an, string aName)

{

m = am;

n = an;

A = new double[m, n];

name = aName;

}

public Matrix(int am,int an,double[,] B)

{

m = am;

n = an;

A = B;

}

public int getM

{

get { return m; }

}

public int getN

{

get { return n; }

}

public double[,] Detail

{

get { return A; }

set { A = value; }

}

public string Name

{

get { return name; }

set { name = value; }

}

}

/\*\*\*\*\*\*\*\*\*\*\*矩阵通用操作打包\*\*\*\*\*\*\*\*\*\*\*\*\*/

class MatrixOperator

{

#region 矩阵加法

public static Matrix MatrixAdd(Matrix Ma, Matrix Mb)

{

int m = Ma.getM;

int n = Ma.getN;

int m2 = Mb.getM;

int n2 = Mb.getN;

if ((m != m2) || (n != n2))

{

Exception myException = new Exception("数组维数不匹配");

throw myException;

}

Matrix Mc = new Matrix(m, n);

double[,] c = Mc.Detail;

double[,] a = Ma.Detail;

double[,] b = Mb.Detail;

for (int i = 0; i < m; i++)

for (int j = 0; j < n; j++)

c[i, j] = a[i, j] + b[i, j];

return Mc;

}

#endregion

#region 矩阵减法

public static Matrix MatrixSub(Matrix Ma, Matrix Mb)

{

int m = Ma.getM;

int n = Ma.getN;

int m2 = Mb.getM;

int n2 = Mb.getN;

if ((m != m2) || (n != n2))

{

Exception myException = new Exception("数组维数不匹配");

throw myException;

}

Matrix Mc = new Matrix(m, n);

double[,] c = Mc.Detail;

double[,] a = Ma.Detail;

double[,] b = Mb.Detail;

for (int i = 0; i < m; i++)

for (int j = 0; j < n; j++)

c[i, j] = a[i, j] - b[i, j];

return Mc;

}

#endregion

#region 矩阵乘法

public static Matrix MatrixMulti(Matrix Ma, Matrix Mb)

{

int m = Ma.getM;

int n = Ma.getN;

int m2 = Mb.getM;

int n2 = Mb.getN;

if (n != m2)

{

Exception myException = new Exception("数组维数不匹配");

throw myException;

}

Matrix Mc = new Matrix(m, n2);

double[,] c = Mc.Detail;

double[,] a = Ma.Detail;

double[,] b = Mb.Detail;

for (int i = 0; i < m; i++)

for (int j = 0; j < n2; j++)

{

c[i, j] = 0;

for (int k = 0; k < n; k++)

c[i, j] += a[i, k] \* b[k, j];

}

return Mc;

}

#endregion

#region 矩阵数乘

public static Matrix MatrixSimpleMulti(double k, Matrix Ma)

{

int m = Ma.getM;

int n = Ma.getN;

Matrix Mc = new Matrix(m, n);

double[,] c = Mc.Detail;

double[,] a = Ma.Detail;

for (int i = 0; i < m; i++)

for (int j = 0; j < n; j++)

c[i, j] = a[i, j] \* k;

return Mc;

}

#endregion

#region 矩阵转置

public static Matrix MatrixTrans(Matrix MatrixOrigin)

{

int m = MatrixOrigin.getM;

int n = MatrixOrigin.getN;

Matrix MatrixNew = new Matrix(n, m);

double[,] c = MatrixNew.Detail;

double[,] a = MatrixOrigin.Detail;

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

c[i, j] = a[j, i];

return MatrixNew;

}

#endregion

#region 矩阵求逆（伴随矩阵法）

public static Matrix MatrixInvByCom(Matrix Ma)

{

double d = MatrixOperator.MatrixDet(Ma);

if (d == 0)

{

Exception myException = new Exception("没有逆矩阵");

throw myException;

}

Matrix Ax = MatrixOperator.MatrixCom(Ma);

Matrix An = MatrixOperator.MatrixSimpleMulti((1.0 / d), Ax);

return An;

}

//对应行列式的代数余子式矩阵

public static Matrix MatrixSpa(Matrix Ma, int ai, int aj)

{

int m = Ma.getM;

int n = Ma.getN;

if (m != n)

{

Exception myException = new Exception("矩阵不是方阵");

throw myException;

}

int n2 = n - 1;

Matrix Mc = new Matrix(n2, n2);

double[,] a = Ma.Detail;

double[,] b = Mc.Detail;

//左上

for (int i = 0; i < ai; i++)

for (int j = 0; j < aj; j++)

{

b[i, j] = a[i, j];

}

//右下

for (int i = ai; i < n2; i++)

for (int j = aj; j < n2; j++)

{

b[i, j] = a[i + 1, j + 1];

}

//右上

for (int i = 0; i < ai; i++)

for (int j = aj; j < n2; j++)

{

b[i, j] = a[i, j + 1];

}

//左下

for (int i = ai; i < n2; i++)

for (int j = 0; j < aj; j++)

{

b[i, j] = a[i + 1, j];

}

//符号位

if ((ai + aj) % 2 != 0)

{

for (int i = 0; i < n2; i++)

b[i, 0] = -b[i, 0];

}

return Mc;

}

//矩阵的行列式,矩阵必须是方阵

public static double MatrixDet(Matrix Ma)

{

int m = Ma.getM;

int n = Ma.getN;

if (m != n)

{

Exception myException = new Exception("数组维数不匹配");

throw myException;

}

double[,] a = Ma.Detail;

if (n == 1) return a[0, 0];

double D = 0;

for (int i = 0; i < n; i++)

{

D += a[1, i] \* MatrixDet(MatrixSpa(Ma, 1, i));

}

return D;

}

//矩阵的伴随矩阵

public static Matrix MatrixCom(Matrix Ma)

{

int m = Ma.getM;

int n = Ma.getN;

Matrix Mc = new Matrix(m, n);

double[,] c = Mc.Detail;

double[,] a = Ma.Detail;

for (int i = 0; i < m; i++)

for (int j = 0; j < n; j++)

c[i, j] = MatrixDet(MatrixSpa(Ma, j, i));

return Mc;

}

#endregion

}

}

平差函数：

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace pingcha

{

struct point

{

public string \_ID;

public string ID

{

get { return \_ID; }

set { \_ID = value; }

}

public double \_X { get; set; }//间接平差的参数设置

public double X0 { get; set; }

//public double x;//参数的改正值

//public double h;//点的高程观测值

//public double h\_cored;//改正值

//public double X;//点的XY的坐标

//public double Y;

}

class gaochenp

{

public static void pingc(ref point[] points,Matrix P,Matrix B,Matrix L,out Matrix v,out Matrix Nbb)//X0为已知，h为已知，H为已知。（待输入值）

//P为n\*n的对角阵，B为n\*t

{

int n = B.getM;

int t = B.getN;

//BtPB

var Bt = MatrixOperator.MatrixTrans(B); //nt

Nbb = MatrixOperator.MatrixMulti(MatrixOperator.MatrixMulti(Bt, P), B); //tt

var W = MatrixOperator.MatrixMulti(MatrixOperator.MatrixMulti(Bt, P), L); //n1

var \_x = MatrixOperator.MatrixMulti(MatrixOperator.MatrixInvByCom(Nbb), W); //t1

v = MatrixOperator.MatrixSub(MatrixOperator.MatrixMulti(B, \_x),L); //n1

//var \_L = MatrixOperator.MatrixAdd(L, \_x);

var \_\_x = new double[t,1];

var \_\_v = v.Detail;

\_\_x = \_x.Detail;

for(int i=0;i<t;i++)

{

points[i].\_X = points[i].X0 + \_\_x[i, 0];

}

}

public static void jianyan(Matrix P,Matrix v,int t,double sumS,Matrix Nbb,out double etea)

{

int n = P.getM;

var Q = MatrixOperator.MatrixInvByCom(P);

var vt = MatrixOperator.MatrixTrans(v);

var etea1 = MatrixOperator.MatrixMulti(MatrixOperator.MatrixMulti(vt, P), v);

var \_etea = MatrixOperator.MatrixSimpleMulti(1000 / t, etea1);//单位权中误差

var InNbb = MatrixOperator.MatrixInvByCom(Nbb);

var \_InNbb = InNbb.Detail;

var \_Q = Q.Detail;

var etea\_ = \_etea.Detail;

double sumQ = 0;

etea = etea\_[0, 0];

for (int i = 0;i < n;i++)

{

sumQ += \_Q[i, i];

}

var km\_etea = MatrixOperator.MatrixSimpleMulti(sumQ,\_etea);

double[,] km\_etea1 = km\_etea.Detail;

//检验是否超限

//if(Math.Sqrt(km\_etea1[0,0]) > 10\*Math.Sqrt(sumQ))

//{

// MessageBox.Show("超限，超过四等规定每公里水准路线要求");

//}

//else if(Math.Sqrt(km\_etea1[0, 0]) > 3 \* Math.Sqrt(sumQ))

//{

// MessageBox.Show("超过三等规定每公里路线要求");

//}

}

public static void liangdian( Matrix B,Matrix Nbb,int t,int \_A,int \_B,int C,out double QABJ,out double QA,out double QB)//

{

var InNbb = MatrixOperator.MatrixInvByCom(Nbb);

var \_InNbb = InNbb.Detail;

Matrix hab = new Matrix(t, 1);

Matrix Qhh = MatrixOperator.MatrixMulti(MatrixOperator.MatrixMulti(B,InNbb),MatrixOperator.MatrixTrans(B));

double[,] \_hab = new double[t, 1];

for(int i=0;i<t;i++)

{

\_hab[i,0] = 0;

}

if (\_A < t)

\_hab[\_A, 0] = 1;

if (\_B < t)

\_hab[\_B, 0] = 1;

hab.Detail = \_hab;

//Qab = MatrixOperator.MatrixMulti(MatrixOperator.MatrixTrans(hab), InNbb);

//Qab = MatrixOperator.MatrixMulti(Qab, hab);

//var \_Qab = Qab.Detail;

//QABJ = \_Qab[0, 0];

var \_Qab = Qhh.Detail;

QABJ = \_Qab[C-1, C-1];

QA = \_InNbb[\_A-1, \_A-1];

QB = \_InNbb[\_B-1, \_B-1];

}

public static void quanchang(int d,int n,double[,] S,double[,] h,double[,] D,out double wucha)

{

var \_D = new Matrix(d, n, D);

var \_h = new Matrix(n, 1, h);

var Sumh = MatrixOperator.MatrixMulti(\_D, \_h);//d1矩阵

var subD = new double[d, n];

for(int i=0;i<d;i++)

{

for(int j=0;j<n;j++)

{

subD[i, j] = Math.Abs(D[i, j]);

}

}

var \_subD = new Matrix(d, n, subD);

var \_S = new Matrix(n, 1, S);

var SumS = MatrixOperator.MatrixMulti(\_subD, \_S);//d1矩阵

var \_SumS = SumS.Detail;

var \_SumH = Sumh.Detail;

double detadeta = 0;

for(int i=0;i<d;i++)

{

detadeta += \_SumH[i,0] \* \_SumH[i,0] \* 1000 / ( \_SumS[i,0] \* d) ;

}

wucha = Math.Sqrt(detadeta);

}

}

}

窗口函数：

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace pingcha

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

int n = int.Parse(textBox\_n.Text);

int t = int.Parse(textBox\_t.Text);

int s = int.Parse(textBox1.Text);

for (int i = 0; i < t; i++)

{

data\_income.Columns.Add("参数", "第" + (i + 1) + "个参数点点");

}

for(int i = 0; i < s;i++)

{

data\_income.Columns.Add("已知点", "第" + (i + 1) + "个已知点");

}

for(int i = t+s;i<t+s+2;i++)

{

if (i == t+s)

data\_income.Columns.Add("起点观测值", "高程观测值的负数");

//if (i == t+s+1)

// data\_income.Columns.Add("终点观测值", "终点观测值");

if (i == t+s+1)

data\_income.Columns.Add("观测路线长", "路程的观测距离");

}

data\_income.Rows.Add(n);

for(int i = 0;i<t;i++)

{

data\_B.Columns.Add("第" + (i + 1) + "个参数的系数", "第" + (i + 1) + "个参数的系数");

}

for(int i=0;i<n;i++)

{

data\_quanchang.Columns.Add("高程", "第" + i + "个高差观测值");

}

data\_quanchang.Rows.Add(n);

}

private void button2\_Click(object sender, EventArgs e)

{

int n = int.Parse(textBox\_n.Text);

int t = int.Parse(textBox\_t.Text);

int s = int.Parse(textBox1.Text);

var b = new double[n, t+s+2];

var B = new double[n, t];

var S = new double[n, 1];

var l = new double[n, 1];

var x0 = new double[t, 1];

for(int i=0;i<n;i++)

{

for(int j=0;j<t+s+2;j++)

{

b[i, j] = double.Parse(data\_income.Rows[i].Cells[j].Value.ToString());

}

}

//预处理 获得B,S,L

for(int i=0;i<n;i++)

{

for(int j=0;j<t;j++)

{

if (b[i, j] != 0)

x0[j,0] = Math.Abs(b[i, j]);

if (b[i, j] > 0)

B[i, j] = 1;

else if (b[i, j] < 0)

B[i, j] = -1;

else

B[i, j] = 0;

}

S[i, 0] = b[i, t + s + 1];

l[i, 0] = 0;

for(int j=0; j<t+s+1; j++)

{

l[i, 0] += b[i, j];

}

}

for(int i=0;i<n;i++)

{

data\_S.Rows.Add();

data\_l.Rows.Add();

data\_S.Rows[i].Cells[0].Value = S[i, 0];

data\_l.Rows[i].Cells[0].Value = l[i, 0];

}

for(int i=0;i<n;i++)

{

data\_B.Rows.Add();

for(int j=0;j<t;j++)

{

data\_B.Rows[i].Cells[j].Value = B[i, j];

}

}

for(int i=0;i<t;i++)

{

data\_X0.Rows.Add();

data\_X0.Rows[i].Cells[0].Value = x0[i, 0];

}

}

private void Button3\_Click(object sender, EventArgs e)

{

int n = int.Parse(textBox\_n.Text);

int t = int.Parse(textBox\_t.Text);

int d = int.Parse(textBox4.Text);

var B = new double[n, t];

var S = new double[n, 1];

var L = new double[n, 1];

var h = new double[n, 1];

int s = int.Parse(textBox1.Text);

var C = int.Parse(textBox3.Text);

var D = new double[d, n];

double SumS = 0;

point[] points = new point[t];

for(int i=0;i<n;i++)

{

L[i, 0] = double.Parse(data\_l.Rows[i].Cells[0].Value.ToString());

S[i, 0] = double.Parse(data\_S.Rows[i].Cells[0].Value.ToString());

h[i, 0] = double.Parse(data\_income.Rows[i].Cells[t + s].Value.ToString());

SumS += S[i, 0];

for(int j=0;j<t;j++)

{

B[i, j] = int.Parse(data\_B.Rows[i].Cells[j].Value.ToString());

}

}

for(int i=0;i<t;i++)

{

points[i].X0 = double.Parse(data\_X0.Rows[i].Cells[0].Value.ToString());

}

var P = new double[n, n];

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

P[i, j] = 0;

if (i == j)

P[i, j] = 1 / S[i, 0];

}

}

for(int i=0;i<d;i++)

{

for(int j=0;j<n;j++)

{

D[i, j] = int.Parse(data\_quanchang.Rows[i].Cells[j].Value.ToString());

}

}

Matrix \_P = new Matrix(n,n,P);

Matrix \_B = new Matrix(n,t,B);

Matrix \_L = new Matrix(n,1,L);

Matrix V = new Matrix(n, 1);

Matrix Nbb;double etea;

double QAB, QA, QB;

int A\_ = int.Parse(textBox\_A.Text);

int B\_ = int.Parse(textBox\_b.Text);

gaochenp.pingc(ref points, \_P, \_B, \_L,out V,out Nbb);

gaochenp.jianyan(\_P, V, t, SumS,Nbb,out etea);

gaochenp.liangdian(\_B,Nbb, t, A\_, B\_,C, out QAB, out QA, out QB);

double quanwucha;

gaochenp.quanchang(d, n, S, h, D,out quanwucha);

data\_result.Columns.Add("X", "参数的改正值");

//data\_result.Columns.Add("H", "高差的校正值");

data\_result.Rows.Add(n);

for(int i =0;i<t;i++)

{

data\_result.Rows[i].Cells[0].Value = points[i].\_X;

}

if (quanwucha > 10)

MessageBox.Show("全长中误差超限");

else

MessageBox.Show("全长中误差未超限");

//var \_V = V.Detail;

//for(int i = 0;i<n;i++)

//{

// data\_result.Rows[i].Cells[1].Value =double.Parse(data\_income.Rows[i].Cells[n + s].Value.ToString())+\_V[i,0];

//}

textBox2.Text = A\_.ToString() + "点的高程中误差为" + (Math.Sqrt(etea \* QA).ToString()) + "\n " + B\_.ToString() + "点的高程中误差为" + (Math.Sqrt(etea \* QB)).ToString() + "\n " + A\_ + "与" + B\_ + "之间的高程中误差为" + (Math.Sqrt(etea \* QAB)).ToString();

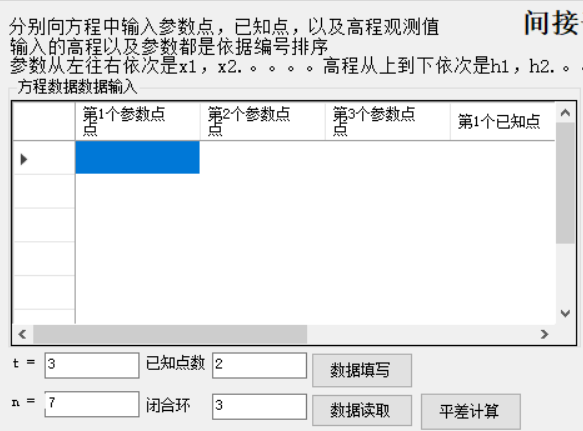
}

# 五．实例检验

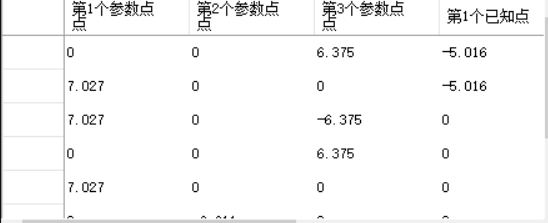
程序实例检验按照题目所给网进行精度评定

1. 根据已知条件可得必要观测数t=3，观测数n=7，已知点个数s=2

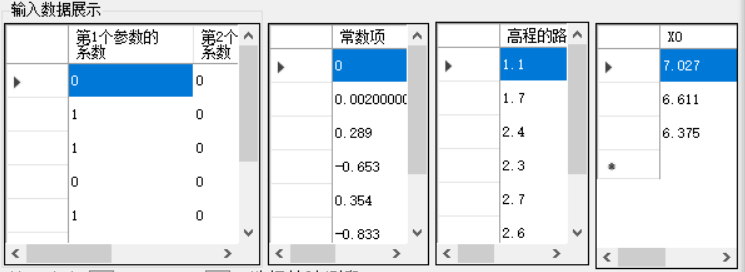
根据水准网特点可得组成其方程的各个待改正的未知点高程，已知高程点，以及观测值和路线



以下是输入结果



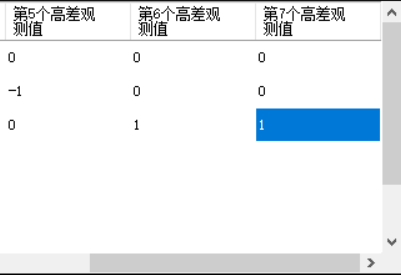
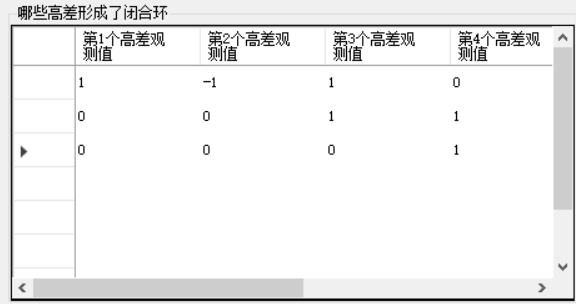
1. 根据输入的数据进行预处理并将预处理结果放在form上面。



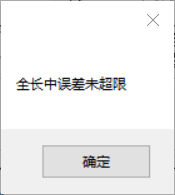
1. 再将精度评定需要的数据输入程序，本次精度评定是3,4点，在模型中对于x1，x3这里两个参数，两者间的高差编号为h3，所以向程序输入



有水准网可以得到三个闭合网，分别向闭合环框中输入其系数



1. 得到其全中误差是否超限的结果



1. 得到其最后精度评定的结果



程序运行完毕

# 六．设计总结

1.设计想法：

在上误差理论与平差基础时，间接平差可以说是非常方便的一个模型，之前用matlab做过，我感觉效果不是非常棒，数据输入十分麻烦，可以说完全没有体现程序的便捷。所以我就想设计一个可以用于界面交互的平差程序，可以直接在界面输入矩阵达到较为便捷的效果，或者有现有的txt文件可以被程序读取然后运行。

2.设计的过程

在做这个程序时，总得来说还是十分麻烦，6.22到6.29那周我都在搞矩阵运算以及平差模型设计的代码编写，期间请教兰老师了一些小问题，29号之后才开始设计界面，远远超出了我的计划时间，但是在第二周做得却十分顺利，调试的很快，几乎没有遇到什么问题，第三周老师给我提出了一些关于界面设计以及精度评定的想法后我对界面还有精度评定进行了修改，删除了偶然中误差，重新将全中误差引用了进来。

3.程序的优点缺点

这个程序优点很多，可以直接输入矩阵，结果简单清晰，适用于各种水准网平差（不包括往返测），可以说是一个适用性很广的模型。缺点也很明显，初次输入者容易不明白程序究竟要输入什么，程序的一些bug和expection也还没有太多地检查，容易产生运行错误。

4.对程序以后的改进想法

因为这个程序是输入的矩阵，我希望以后可以将它改成读取文件的程序，只要是满足一定数据结构的txt文本就可以进行精度评定计算。

再把模型的适用性做得更广一点点。

总得来说收获很多，我很喜欢编程，虽然有些时候回出bug，但是最后能做出结果我觉得很开心。