# 报告文档

## 一、程序优化性说明

1、用户交互界面说明

首先是程序主界面，包括打开文件功能，大地线长度计算功能，打开报告功能，清除数据功能，中间为数据框，用来展示读取的数据。

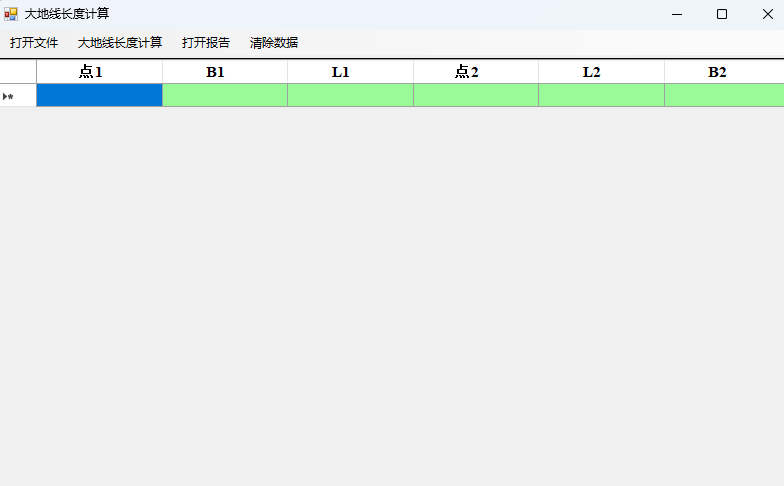


图1 软件主界面

点击打开报告按钮出现计算报告界面，用来展示计算数据，并在右上角有保存结果按钮。

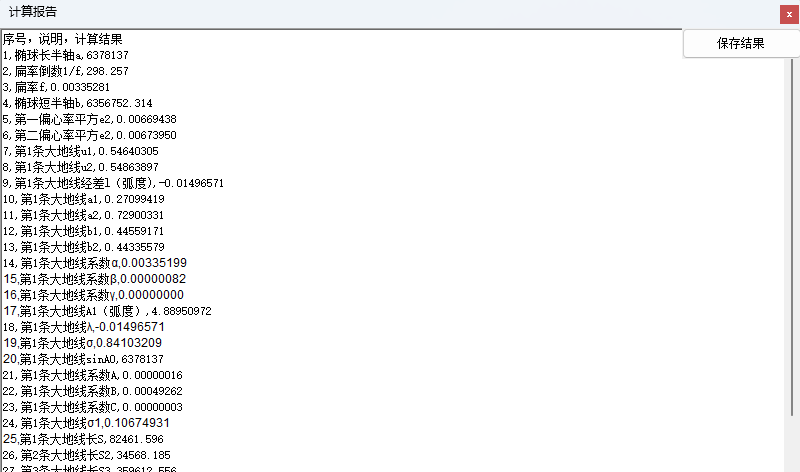


图2 计算报告界面

2、程序运行过程说明

点击打开文件按钮，选择数据进行读取，结果如图3所示。

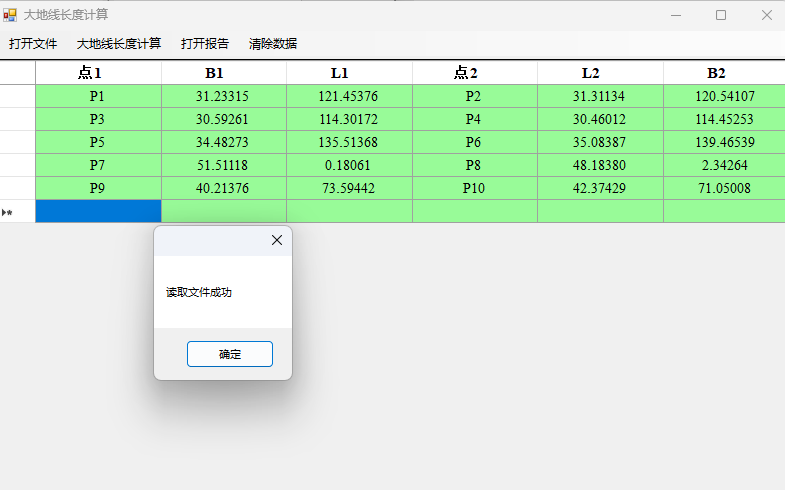


图3 打开文件

点击大地线长度计算按钮，出现计算成功提示框。



图4 计算成功

3、程序运行结果

计算成功后点击打开报告按钮，可查看计算结果。

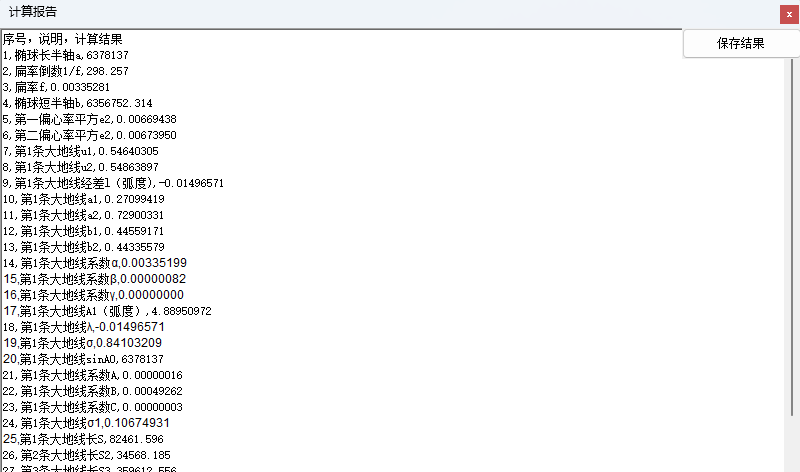


图5 运行结果

## 二、程序规范性说明

1、程序功能与结构设计说明

首先设计存储数据类，包括椭球类，点类，大地线类。

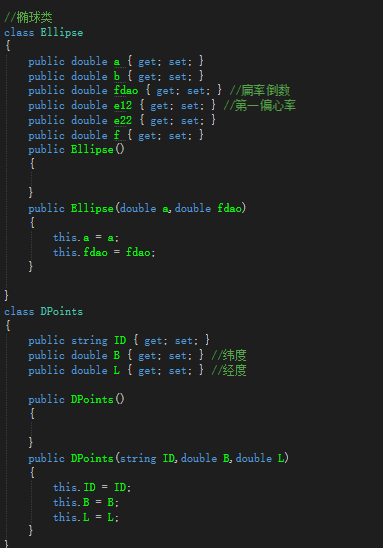


图6 椭球类和点类

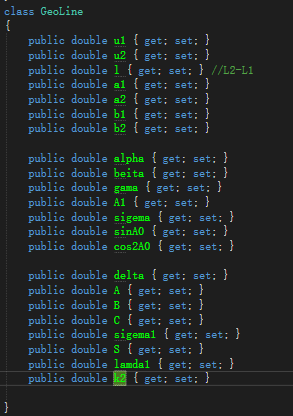


图7 大地线类

之后进行Alogrithm算法类的设计，包含一些数据格式转换，算法的设计。

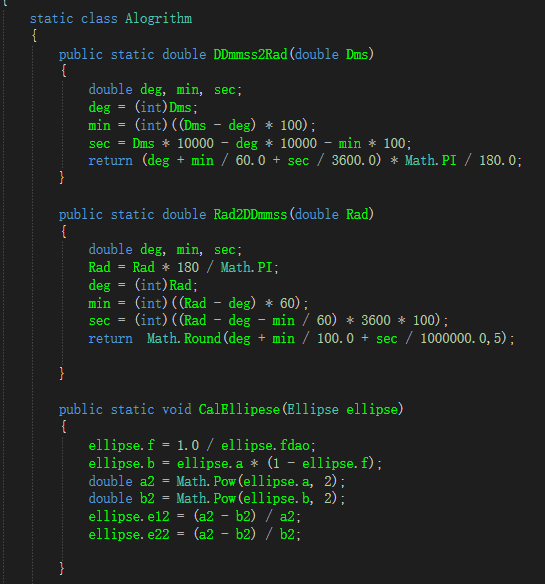


图8 算法类

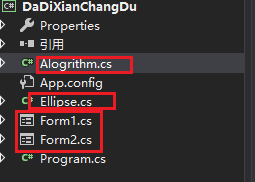


图9 程序总体设计

此程序Form1，Form2为窗体类，Form1为主界面，Form2为数据展示界面

。Ellipse.cs中存放椭球类，大地线类，点类。

2、核心算法源码

Alogrithm.CalEllipese(ellipse);

foreach(var point in dPoints)

{

point.B = Alogrithm.DDmmss2Rad(point.B);

point.L = Alogrithm.DDmmss2Rad(point.L);

}

for(int i =0;i<dPoints.Count;i+=2)

{

GeoLine geoLine = new GeoLine();

geoLine.u1 = Math.Atan(Math.Sqrt(1 - ellipse.e12) \* Math.Tan(dPoints[i].B));

geoLine.u2 = Math.Atan(Math.Sqrt(1 - ellipse.e12) \* Math.Tan(dPoints[i+1].B));

geoLine.l = dPoints[i+1].L - dPoints[i].L;

geoLine.a1 = Math.Sin(geoLine.u1) \* Math.Sin(geoLine.u2);

geoLine.a2 = Math.Cos(geoLine.u1) \* Math.Cos(geoLine.u2);

geoLine.b1 = Math.Cos(geoLine.u1) \* Math.Sin(geoLine.u2);

geoLine.b2 = Math.Sin(geoLine.u1) \* Math.Cos(geoLine.u2);

geoLines.Add(geoLine);

}

//计算起点大地方位角

foreach(var geoline in geoLines)

{

double delta1 = 0;

geoline.delta = 0;

do

{

delta1 = geoline.delta;

double lamda = geoline.l + geoline.delta;

double p = Math.Cos(geoline.u2) \* Math.Sin(lamda);

double q = geoline.b1 - geoline.b2 \* Math.Cos(lamda);

geoline.A1 = Alogrithm.FangWei(p, q);

double Sinsigema = p \* Math.Sin(geoline.A1) + q \* Math.Cos(geoline.A1);

double Cossigema = geoline.a1 + geoline.a2 \* Math.Cos(lamda);

geoline.sigema = Math.Atan(Sinsigema/Cossigema);

if(Cossigema>0)

{

geoline.sigema = Math.Abs(geoline.sigema);

}

else

{

geoline.sigema = Math.PI - Math.Abs(geoline.sigema);

}

geoline.sinA0 = Math.Cos(geoline.u1) \* Math.Sin(geoline.A1);

geoline.cos2A0 = 1 - Math.Pow(geoline.sinA0, 2);

geoline.sigema1 = Math.Atan(Math.Tan(geoline.u1) / Math.Cos(geoline.A1));

double e4 = Math.Pow(ellipse.e12, 2);

double e6 = Math.Pow(ellipse.e12, 3);

geoline.alpha = (ellipse.e12 / 2 + e4 / 8 + e6 / 16) - (e4 / 16 + e6 / 16) \* geoline.cos2A0 + (3 \* e6 / 128) \* Math.Pow(geoline.cos2A0, 2);

geoline.beita = (e4 / 16 + e6 / 16) \* geoline.cos2A0 - (e6 / 32) \* Math.Pow(geoline.cos2A0, 2);

geoline.gama = (e6 / 256) \* Math.Pow(geoline.cos2A0, 2);

geoline.delta = (geoline.alpha \* geoline.sigema + geoline.beita \* Math.Cos(geoline.sigema1 \* 2 + geoline.sigema) \* Math.Sin(geoline.sigema) + geoline.gama \* Math.Sin(2 \* geoline.sigema) \* Math.Cos(4 \* geoline.sigema1 + 2 \* geoline.sigema)) \* geoline.sinA0;

geoline.lamda1 = lamda;

}while (geoline.delta - delta1 >= 1e-10);

}

//计算大地线长度

foreach(var geoline in geoLines)

{

geoline.k2 = ellipse.e22 \* geoline.cos2A0;

double k2 = geoline.k2;

double k4 = Math.Pow(geoline.k2, 2);

double k6 = Math.Pow(geoline.k2, 3);

geoline.A = (1 - (k2 / 4) + 7 \* k4 / 64 - 15 \* k6 / 256) / ellipse.b;

geoline.B = (k2 / 4 - k4 / 8 + 37 \* k6 / 512);

geoline.C = k4 / 128 - k6 / 128;

geoline.sigema1 = Math.Atan(Math.Tan(geoline.u1) \* Math.Cos(geoline.A1));

double Xs = geoline.C \* Math.Sin(2 \* geoline.sigema) \* Math.Cos(4 \* geoline.sigema1 + 2 \* geoline.sigema);

geoline.S = (geoline.sigema - geoline.B \* Math.Sin(geoline.sigema) \* Math.Cos(2 \* geoline.sigema1 + 2 \* geoline.sigema) - Xs) / geoline.A;

}