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。

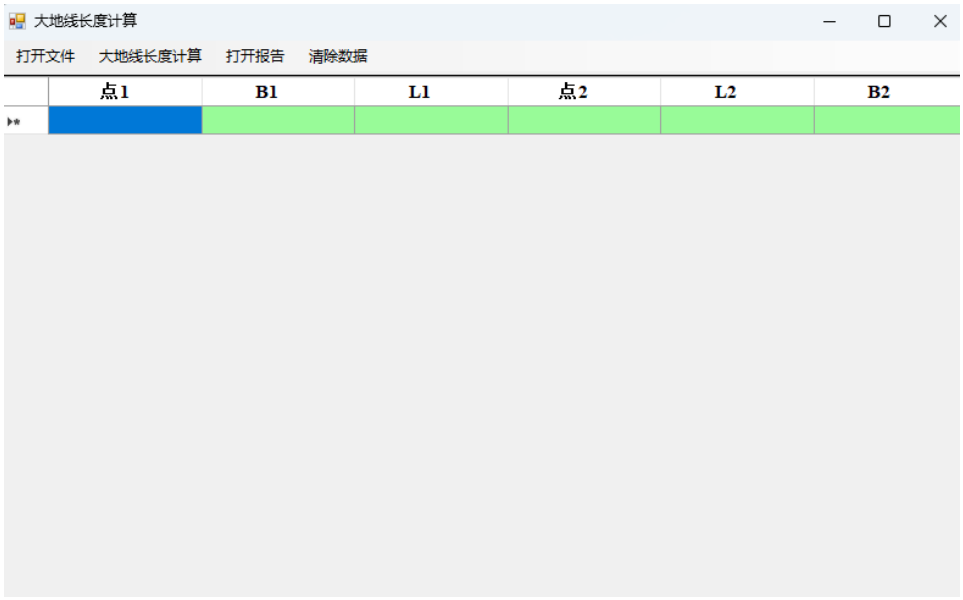

图 1 软件主界面

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。

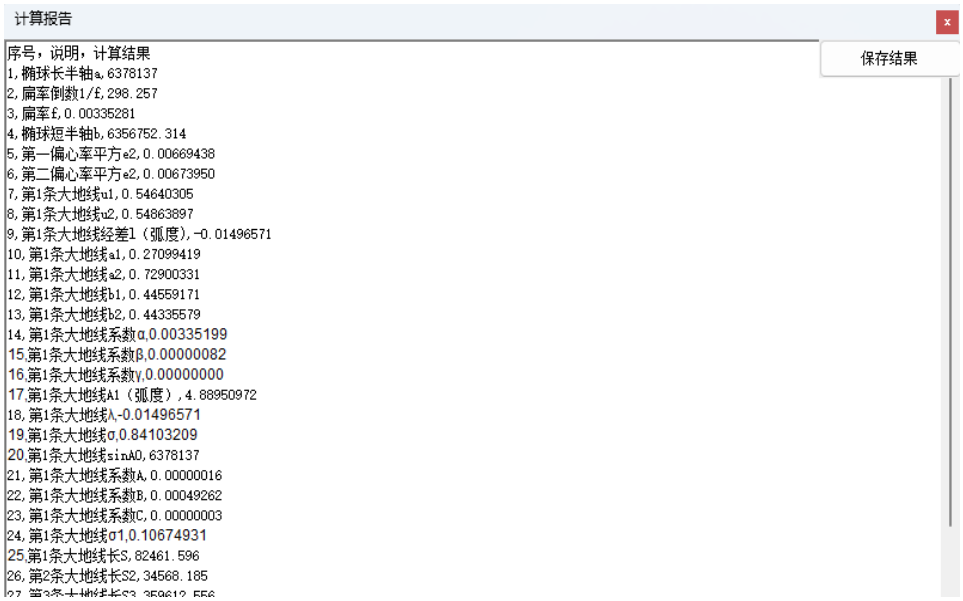

图 2 计算报告界面

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。

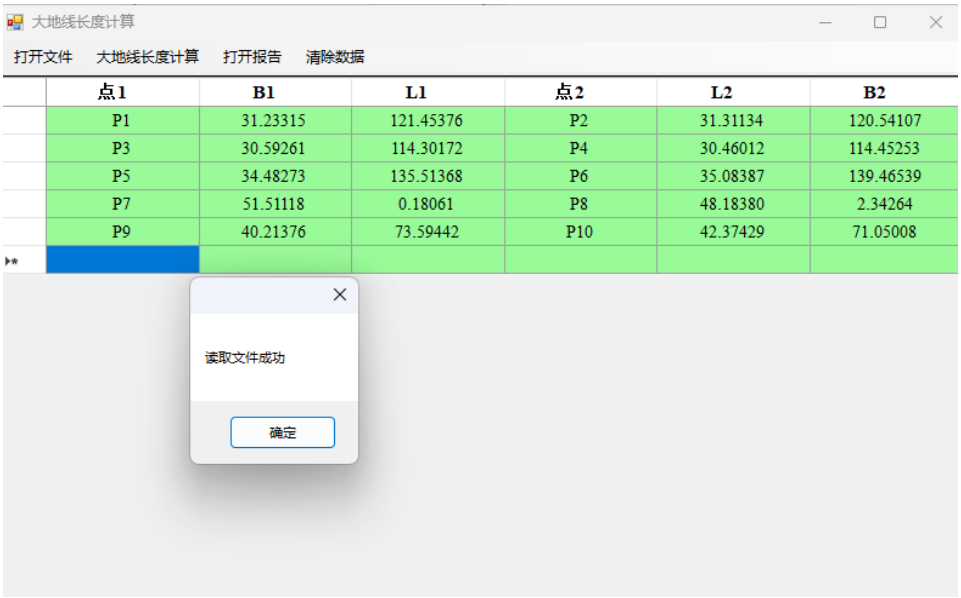

图 3 打开文件

点击大地线长度计算按钮，出现计算成功提示框。

点1	B1	L1	点2	L2	B2
P1	31.23315	121.45376	P2	31.31134	120.54107
P3	30.59261	114.30172	P4	30.46012	114.45253
P5	34.48273	135.51368	P6	35.08387	139.46539
P7	51.51118	0.18061	P8	48.18380	2.34264
P9	40.21376	73.59442	P10	42.37429	71.05008

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。

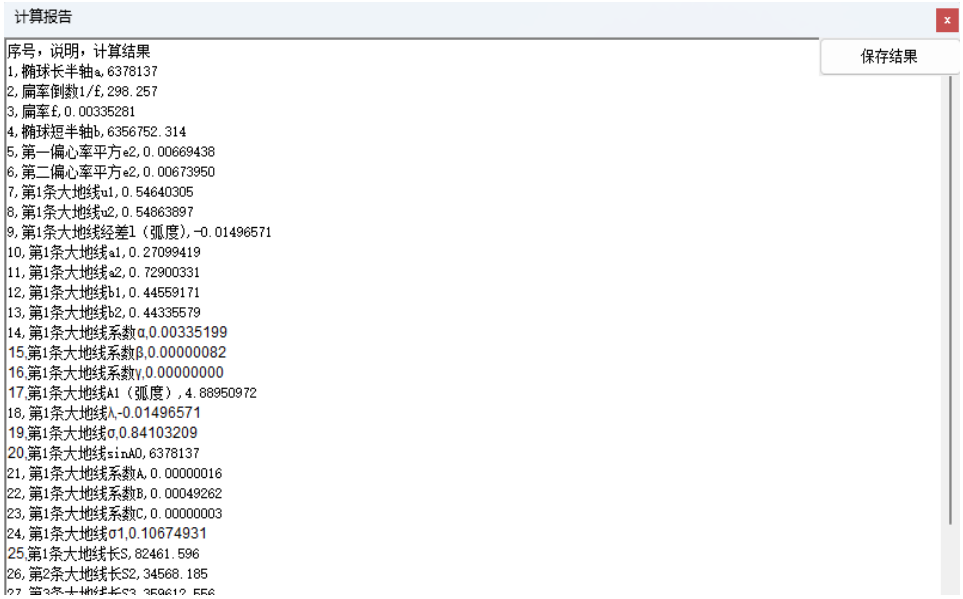

图 5 运行结果

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。

```
//椭球类
class Ellipse
{
    public double a { get; set; }
    public double b { get; set; }
    public double fdao { get; set; } //扁率倒数
    public double e12 { get; set; } //第一偏心率
    public double e22 { get; set; }
    public double f { get; set; }
    public Ellipse()
    {

    }
    public Ellipse(double a,double fdao)
    {
        this.a = a;
        this.fdao = fdao;
    }

}
class DPoints
{
    public string ID { get; set; }
    public double B { get; set; } //纬度
    public double L { get; set; } //经度

    public DPoints()
    {

    }
    public DPoints(string ID,double B,double L)
    {
        this.ID = ID;
        this.B = B;
        this.L = L;
    }
}
```

图 6 椭球类和点类

```
class GeoLine
{
    public double u1 { get; set; }
    public double u2 { get; set; }
    public double l { get; set; } //L2-L1
    public double a1 { get; set; }
    public double a2 { get; set; }
    public double b1 { get; set; }
    public double b2 { get; set; }

    public double alpha { get; set; }
    public double beita { get; set; }
    public double gama { get; set; }
    public double A1 { get; set; }
    public double sigema { get; set; }
    public double sinA0 { get; set; }
    public double cos2A0 { get; set; }

    public double delta { get; set; }
    public double A { get; set; }
    public double B { get; set; }
    public double C { get; set; }
    public double sigema1 { get; set; }
    public double S { get; set; }
    public double lamda1 { get; set; }
    public double k2 { get; set; }

}
```

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。

```
static class Alogrithm
{
    public static double DDmmss2Rad(double Dms)
    {
        double deg, min, sec;
        deg = (int)Dms;
        min = (int)((Dms - deg) * 100);
        sec = Dms * 10000 - deg * 10000 - min * 100;
        return (deg + min / 60.0 + sec / 3600.0) * Math.PI / 180.0;
    }

    public static double Rad2DDmmss(double Rad)
    {
        double deg, min, sec;
        Rad = Rad * 180 / Math.PI;
        deg = (int)Rad;
        min = (int)((Rad - deg) * 60);
        sec = (int)((Rad - deg - min / 60) * 3600 * 100);
        return  Math.Round(deg + min / 100.0 + sec / 1000000.0,5);

    }

    public static void CalEllipese(Ellipse ellipse)
    {
        ellipse.f = 1.0 / ellipse.fdao;
        ellipse.b = ellipse.a * (1 - ellipse.f);
        double a2 = Math.Pow(ellipse.a, 2);
        double b2 = Math.Pow(ellipse.b, 2);
        ellipse.e12 = (a2 - b2) / a2;
        ellipse.e22 = (a2 - b2) / b2;

    }
```

图 8 算法类

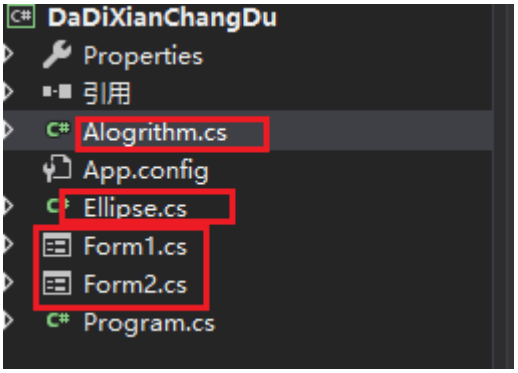

图 9 程序总体设计

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;
                }
```