报告文档

1. 程序优化性说明
   1. 用户交互界面说明

程序采用C# WinForm框架进行开发，界面风格采用标准Windows应用程序，界面从上到下、从左到右依次为：标题栏、菜单栏、工具栏、数据展示区、报告预览区、状态栏。整个界面布局清晰，设计操作人性化。具体如图 1所示。

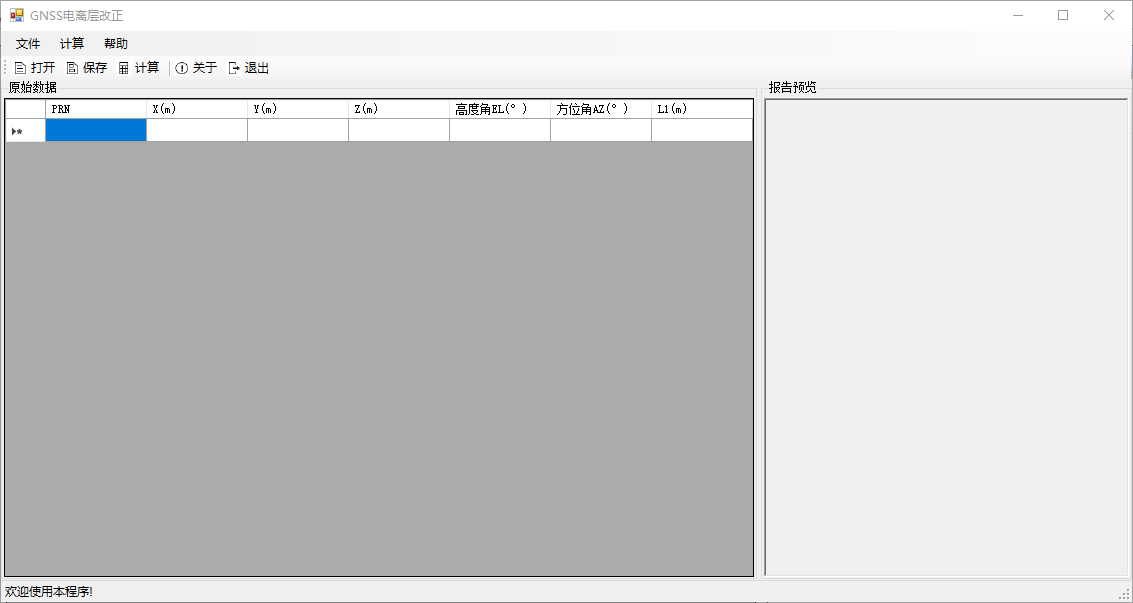


图 1 程序界面

* 1. 程序运行过程说明

1. 点击【打开】，选择原始数据文件，点击【确定】后将读取数据并展示到界面。

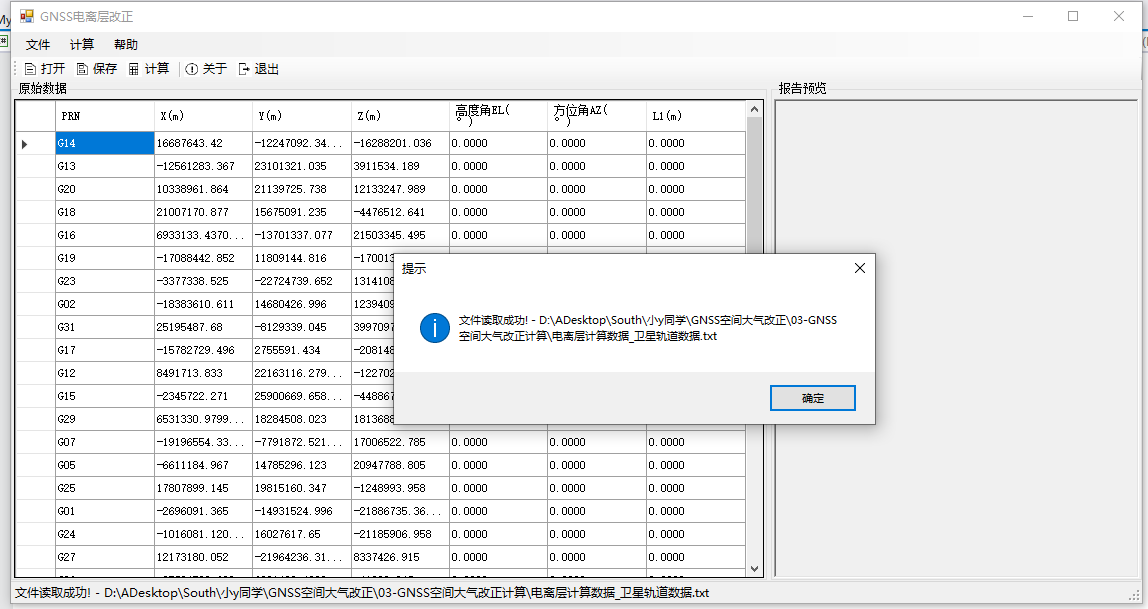


图 2 打开数据文件

1. 点击【计算】，程序将根据读取的文件进行电离层有延迟的计算，并生成报告预览。

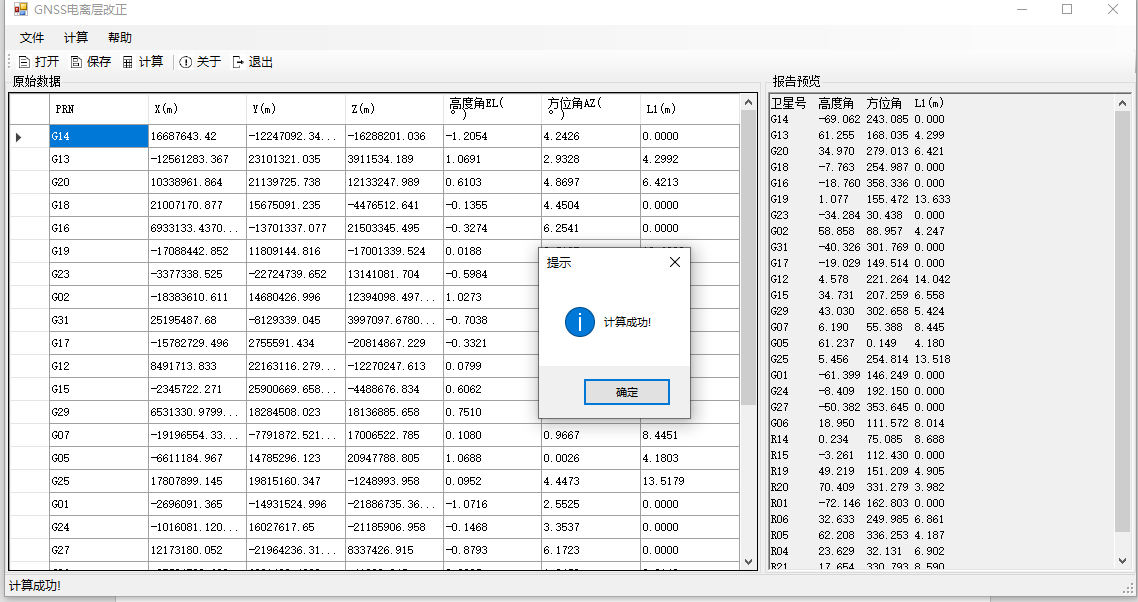


图 3 计算成功

1. 点击【保存】，选择保存文件路径，点击【确定】后程序将保存【报告预览区域】内容到文件。

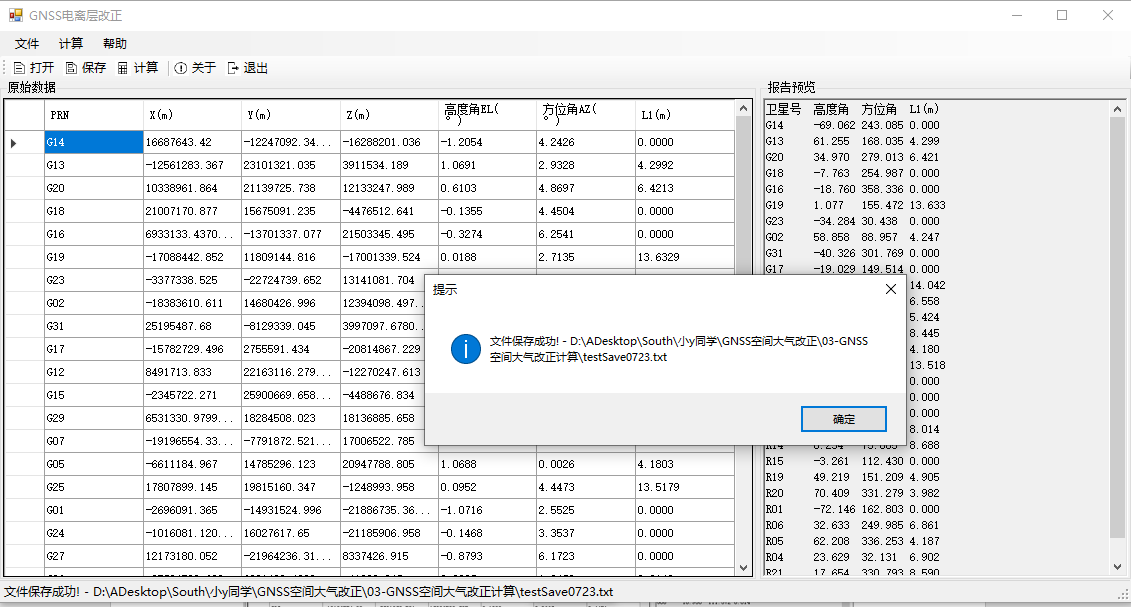


图 4 保存报告

1. 点击【退出】，经过二次确认后，程序退出。

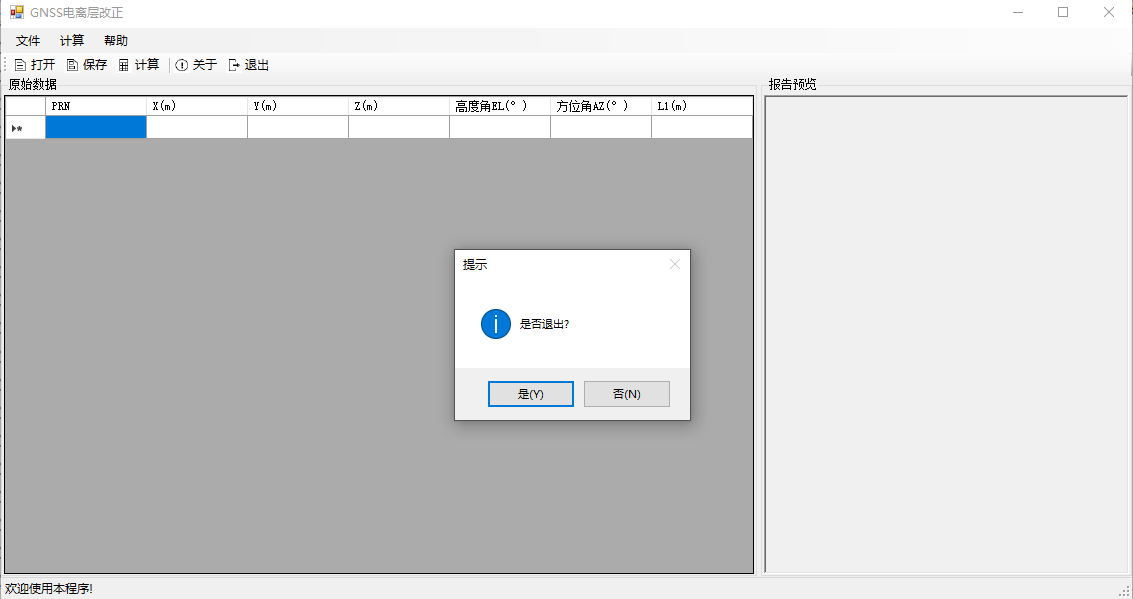


图 5 退出程序

* 1. 程序运行结果

卫星号 高度角 方位角 L1(m)

G14 -69.062 243.085 0.000

G13 61.255 168.035 4.299

G20 34.970 279.013 6.421

G18 -7.763 254.987 0.000

G16 -18.760 358.336 0.000

G19 1.077 155.472 13.633

G23 -34.284 30.438 0.000

G02 58.858 88.957 4.247

G31 -40.326 301.769 0.000

G17 -19.029 149.514 0.000

G12 4.578 221.264 14.042

G15 34.731 207.259 6.558

G29 43.030 302.658 5.424

G07 6.190 55.388 8.445

G05 61.237 0.149 4.180

G25 5.456 254.814 13.518

G01 -61.399 146.249 0.000

G24 -8.409 192.150 0.000

G27 -50.382 353.645 0.000

G06 18.950 111.572 8.014

R14 0.234 75.085 8.688

R15 -3.261 112.430 0.000

R19 49.219 151.209 4.905

R20 70.409 331.279 3.982

R01 -72.146 162.803 0.000

R06 32.633 249.985 6.861

R05 62.208 336.253 4.187

R04 23.629 32.131 6.902

R21 17.654 330.793 8.590

1. 程序规范性说明
   1. 程序功能与结构设计说明

程序可以实现从文件读取数据到界面，计算电离层延迟改正量，保存报告等功能。具体功能按菜单栏列出见表 1。

表 1 程序功能

|  |  |
| --- | --- |
| 选项 | 功能 |
| 打开 | 用户选择原始数据文件后读取文件数据并展示到界面 |
| 保存 | 用户选择保存路径后，保存报告内容到文件 |
| 打开数据文件夹 | 如果用户打开过数据文件，则打开文件所在文件夹，否则给出相应提示。 |
| 打开报告文件夹 | 如果用户保存过报告文件，则打开文件所在文件夹，否则给出相应提示。 |
| 计算 | 电离层延迟改正计算 |
| 关于 | 显示程序基本信息 |
| 帮助 | 打开帮助文档 |
| 退出 | 用户经二次确认后退出程序 |

程序共设计有MyData、MyFile、MySatellite、MyStation、MyTime四个类，具体类的功能如表 2所示。

表 2 类的功能

|  |  |
| --- | --- |
| 类名 | 功能简述 |
| MyData | 存放维持窗体运行的必要数据，例如inFile和outFile |
| MyFile | 读取数据到MyData |
| MySatellite | 存储单个卫星的信息，包括计算单个卫星的高度角、方位角和电离层延迟函数 |
| MyStation | 存储单个测站的信息，包括测站的地心坐标、大地坐标、观测的卫星列表 |
| MyTime | 存储时间信息 |

* 1. 核心算法源码

1. 计算卫星高度角和方位角代码：

/// <summary>

/// 计算该卫星的高度角和方位角

/// </summary>

/// <param name="Station">测站</param>

public void CalELAZ(MyStation Station)

{

//计算站心坐标

double sb = Math.Sin(Station.Bp);

double cb = Math.Cos(Station.Bp);

double sl = Math.Sin(Station.Lp);

double cl = Math.Cos(Station.Lp);

double dx = this.Xs - Station.Xp;

double dy = this.Ys - Station.Yp;

double dz = this.Zs - Station.Zp;

this.X = -sb \* cl \* dx - sb \* sl \* dy + cb \* dz;

this.Y = -sl \* dx + cl \* dy;

this.Z = cb \* cl \* dx + cb \* sl \* dy + sb \* dz;

//计算卫星方位角A

this.AZ = Math.Atan2(this.Y, this.X) + (Y < 0 ? 1 : 0) \* 2 \* Math.PI;

//计算高度角E

this.EL = Math.Atan(Z / Math.Sqrt(X \* X + Y \* Y));

}

1. 计算电离层延迟代码：

/// <summary>

/// 计算电离层延迟

/// </summary>

/// <param name="Station"></param>

/// <param name="H0"></param>

public void IonoDelay(MyStation Station, double H0 = 350)

{

if (EL < 0)//高度角小于0没有延迟

{

this.L1D = 0;

this.L1T = 0;

return;

}

//穿刺点IP的坐标

double psi = 0.0137 / (EL / Math.PI + 0.11) - 0.022;

double phi = Station.Bp / Math.PI + psi \* Math.Cos(AZ);

if (phi > 0.416) phi = 0.416;

else if (phi < -0.416) phi = -0.416;

double lambda = Station.Lp / Math.PI + psi \* Math.Sin(AZ) / Math.Cos(phi \* Math.PI);

//穿刺点地磁纬度

double phi\_m = phi + 0.064 \* Math.Cos((lambda - 1.617) \* Math.PI);

//克罗布歇模型改正

double[] alpha = new double[] { 0.1397e-7, -0.7451e-8, -0.5960e-7, 0.1192e-6 };

double[] beta = new double[] { 0.1270e6, -0.1966e6, 0.6554e5, 0.2621e6 };

double A1 = 5e-9;

double A2 = alpha[0] + alpha[1] \* phi\_m + alpha[2] \* phi\_m \* phi\_m + alpha[3] \* phi\_m \* phi\_m \* phi\_m;

double A3 = 50400;

double A4 = beta[0] + beta[1] \* phi\_m + beta[2] \* phi\_m \* phi\_m + beta[3] \* phi\_m \* phi\_m \* phi\_m;

A2 = A2 < 0 ? 0 : A2;

A4 = A4 < 72000 ? 72000 : A4;

double F = 1 + 16 \* Math.Pow((0.53 - EL / Math.PI), 3);

//计算当地时间

double localHour = this.obsTime.hour + Station.Lp \* 180 / Math.PI / 15;

localHour = localHour > 24 ? localHour - 24 : localHour;

double T = obsTime.hour \* 3600.0 + obsTime.min \* 60.0 + obsTime.sec;

double t = 43200 \* lambda + T;

double flag = Math.Abs(2 \* Math.PI \* (t - A3)) / A4;

if (flag < 1.57)

this.L1T = F \* (A1 + A2 \* (1 - Math.Pow(flag, 2) / 2 + Math.Pow(flag, 4) / 24));

else

this.L1T = F \* A1;

this.L1D = this.L1T \* 299792458;

}

1. 读取数据文件代码：

/// <summary>

/// 打开文件

/// </summary>

/// <param name="myData"></param>

/// <returns></returns>

public static bool OpenFile(MyData myData)

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Title = "请选择原始数据文件";

ofd.Filter = "文本文件|\*.txt|所有文件|\*.\*";

if (ofd.ShowDialog() != DialogResult.OK) return false;

myData.inFile = ofd.FileName;

using (StreamReader sr = new StreamReader(myData.inFile))

{

myData.Station = new MyStation(-2225669.7744, 4998936.1598, 3265908.9678, 30 \* ToRad, 114 \* ToRad);

MyTime myTime = new MyTime();

while (!sr.EndOfStream)

{

string Line = sr.ReadLine();

if (Line[0] == '\*')//读取时间列

{

string[] Data = Line.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

myTime = new MyTime(Data[1], Data[2], Data[3], Data[4], Data[5], Data[6]);

}

else

{

string PRN = Line.Substring(0, 3);

string X = Line.Substring(3, 14);

string Y = Line.Substring(17, 14);

string Z = Line.Substring(31, 14);

myData.Station.Sats.Add(new MySatellite(myTime, PRN, X, Y, Z));

}

}

return true;

}

}