|  |
| --- |
| 测绘技能 |
| 报告文档 |
| 第三次模拟 |

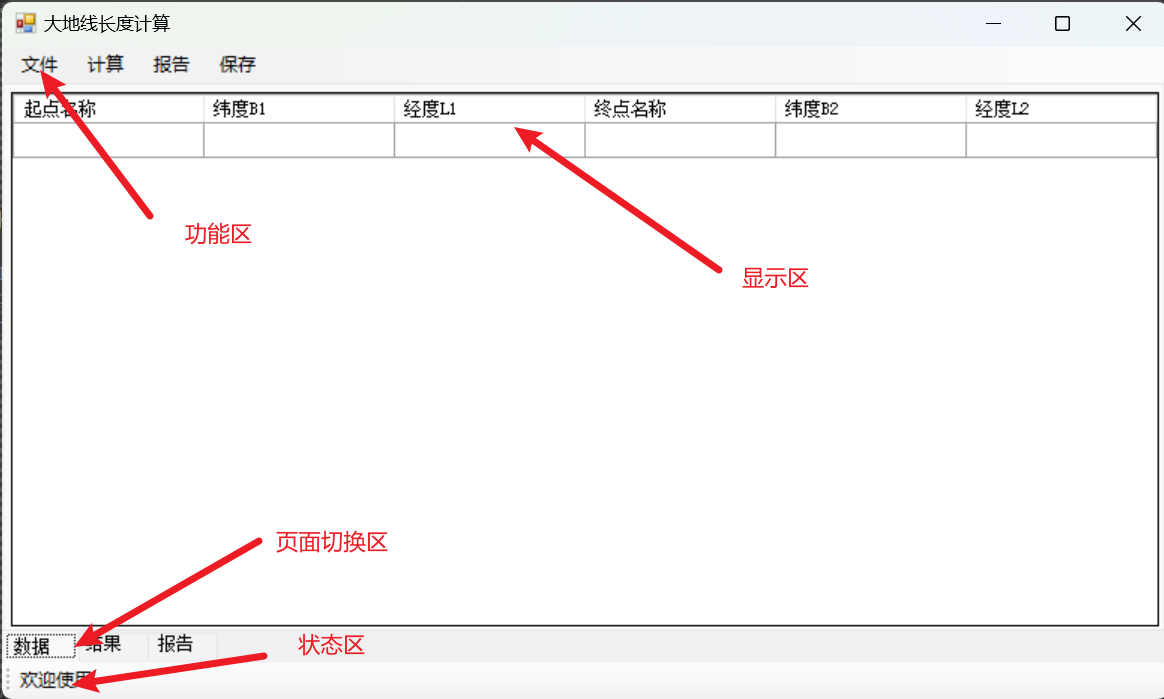
|  |
| --- |
| 2024-7-17 |

# **一、程序优化性说明**

## 1．用户交互界面说明

### 1.1初始界面

软件运行后,初始界面如下图,包括功能区,显示区,页面切换区和状态区,共同构成本次软件的界面。

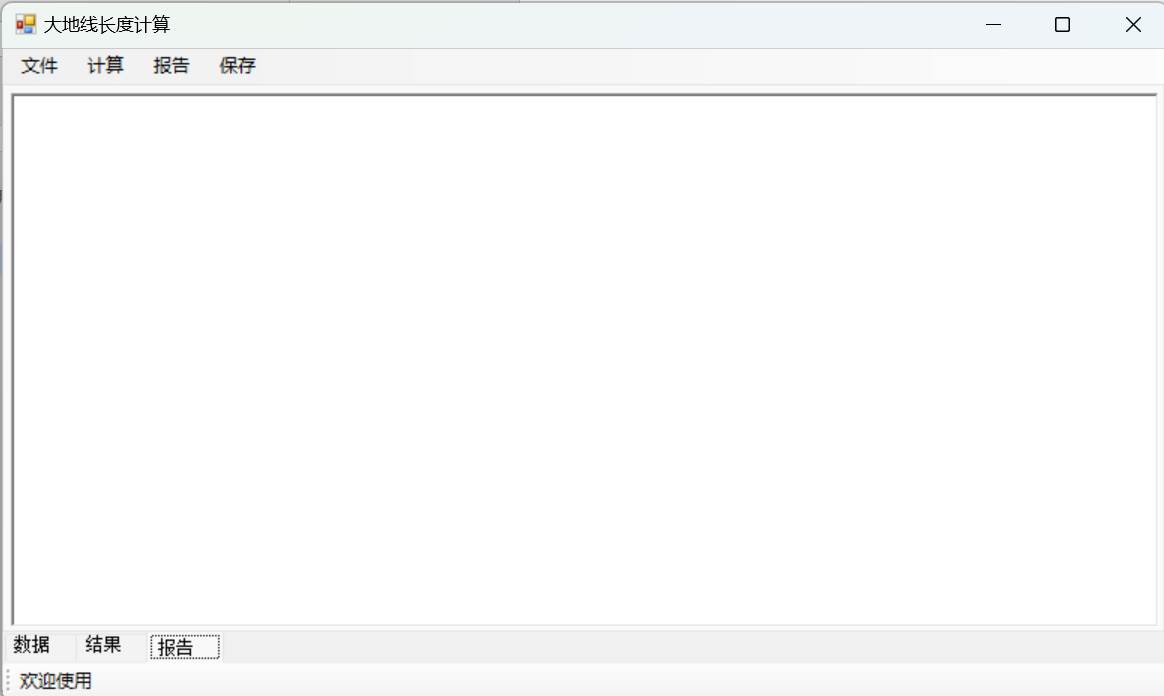


### 1.2其他页面



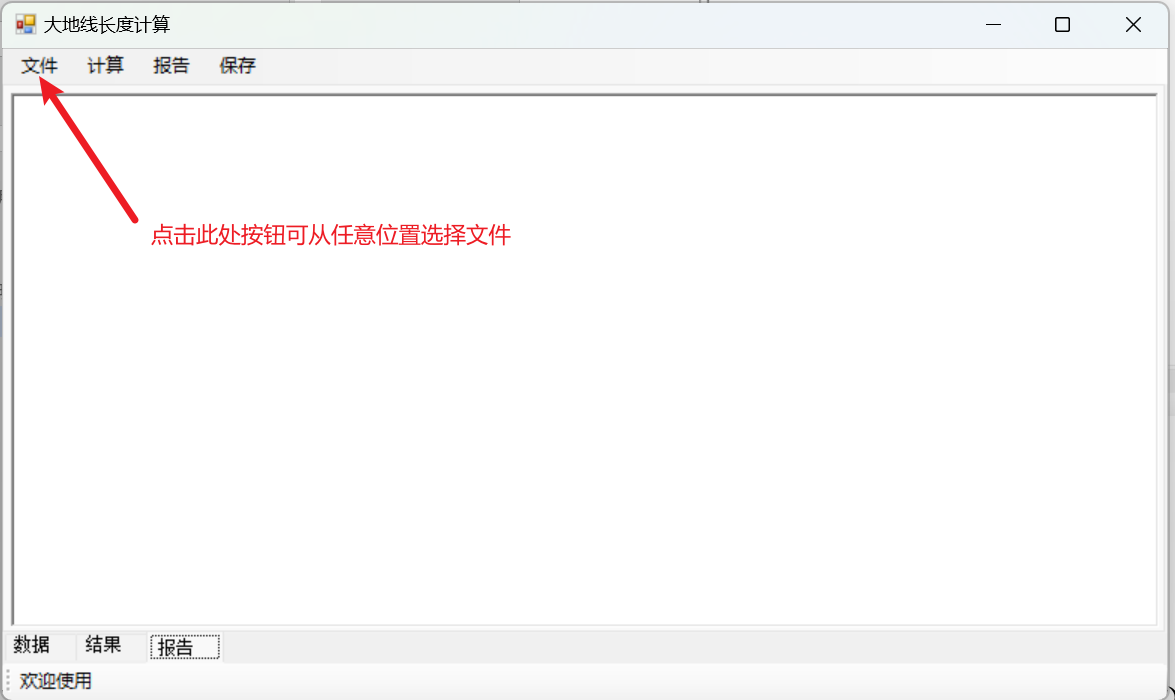
首先是结果显示页面，页面通过使用表格将结果显示在其中，非常方便，同时增加可观行。

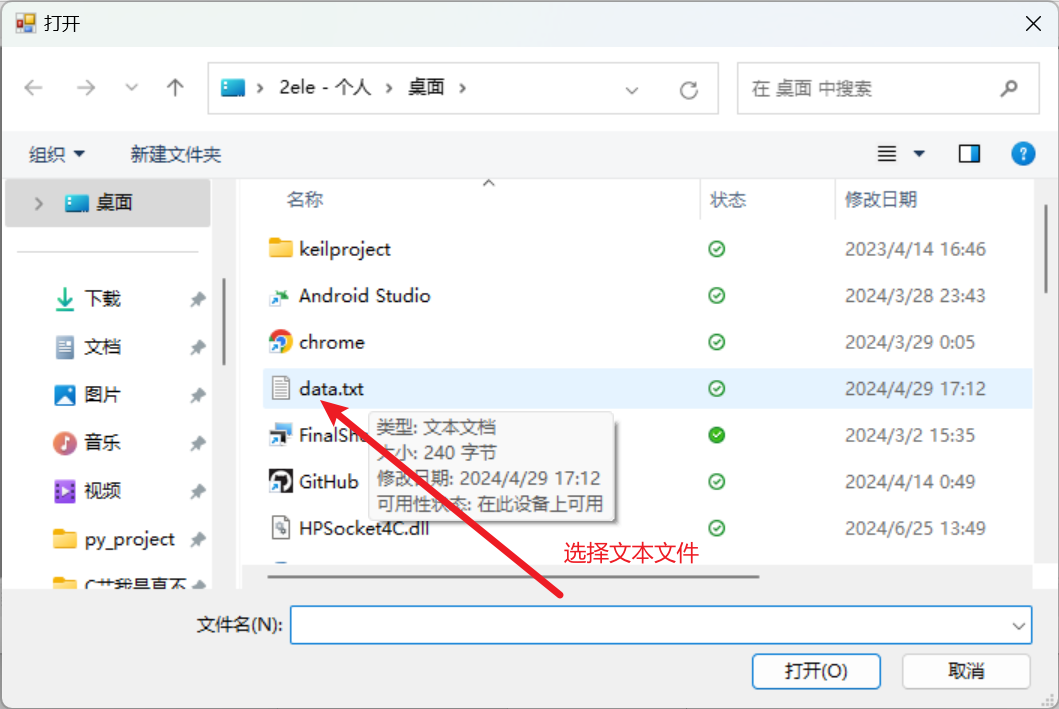
其次是报告页面，通过文本将信息显示出来。



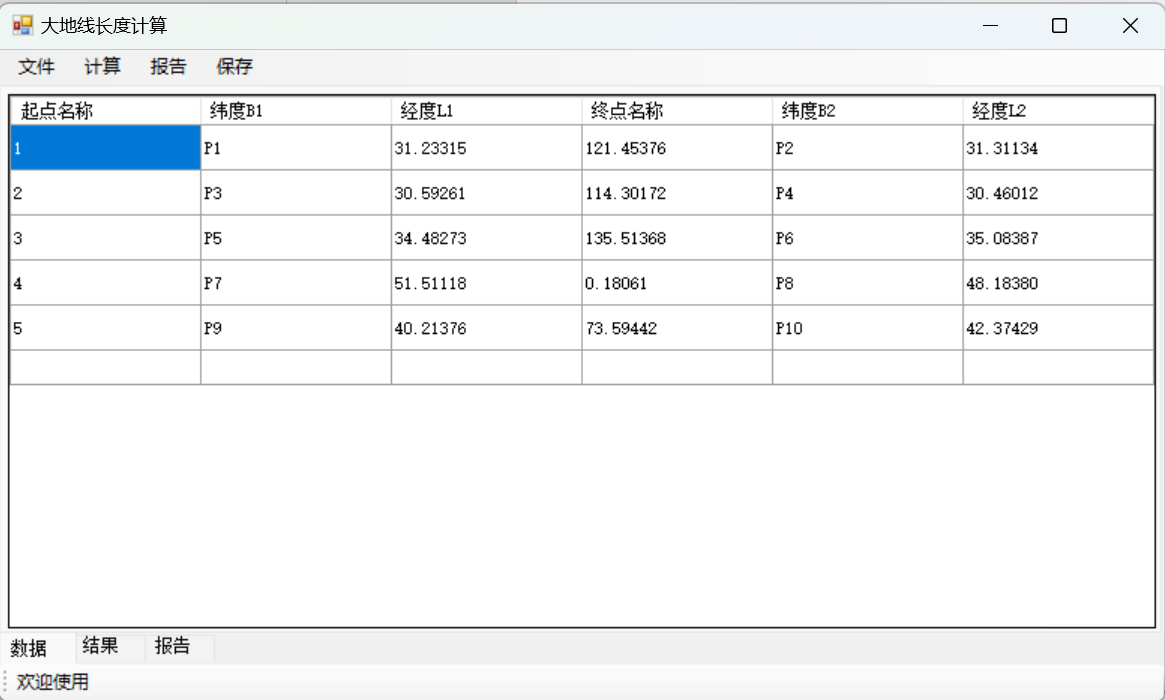
## **2．程序运行过程说明**

### 2.1打开文件





首先通过导入文件，可查看具体要算的大地线，然后即可在表格中查看具体信息，包括起点名称，纬度B1，经度L1,终点名称,纬度B2,经度L2。



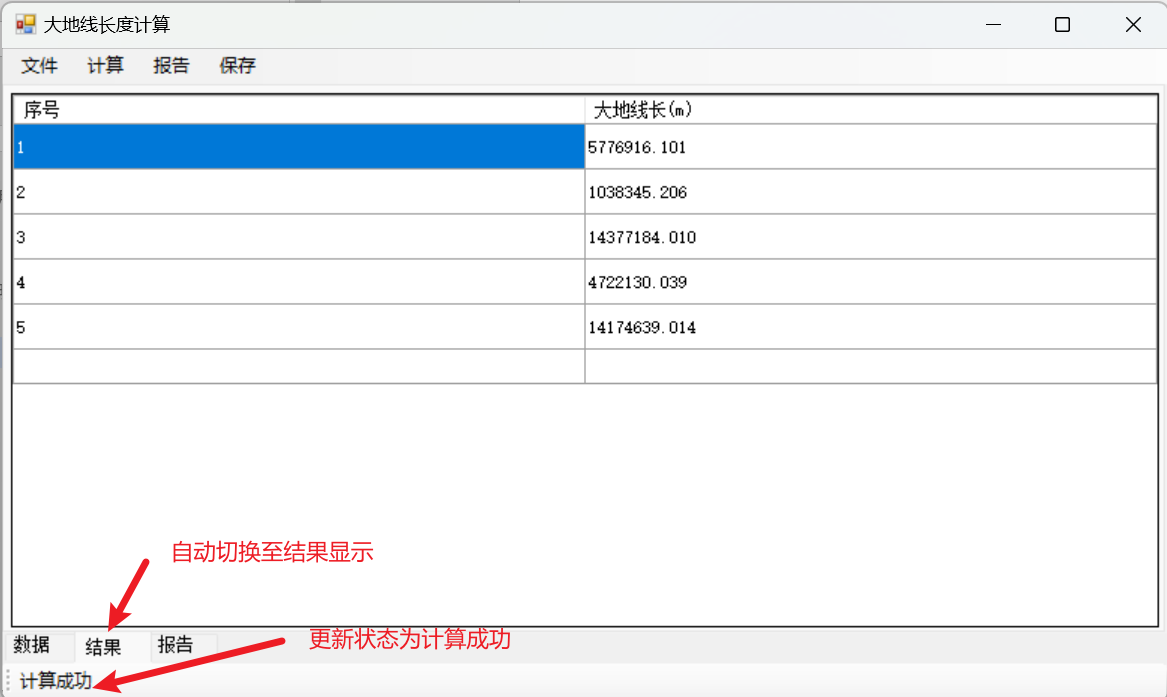
### 2.2计算大地线长



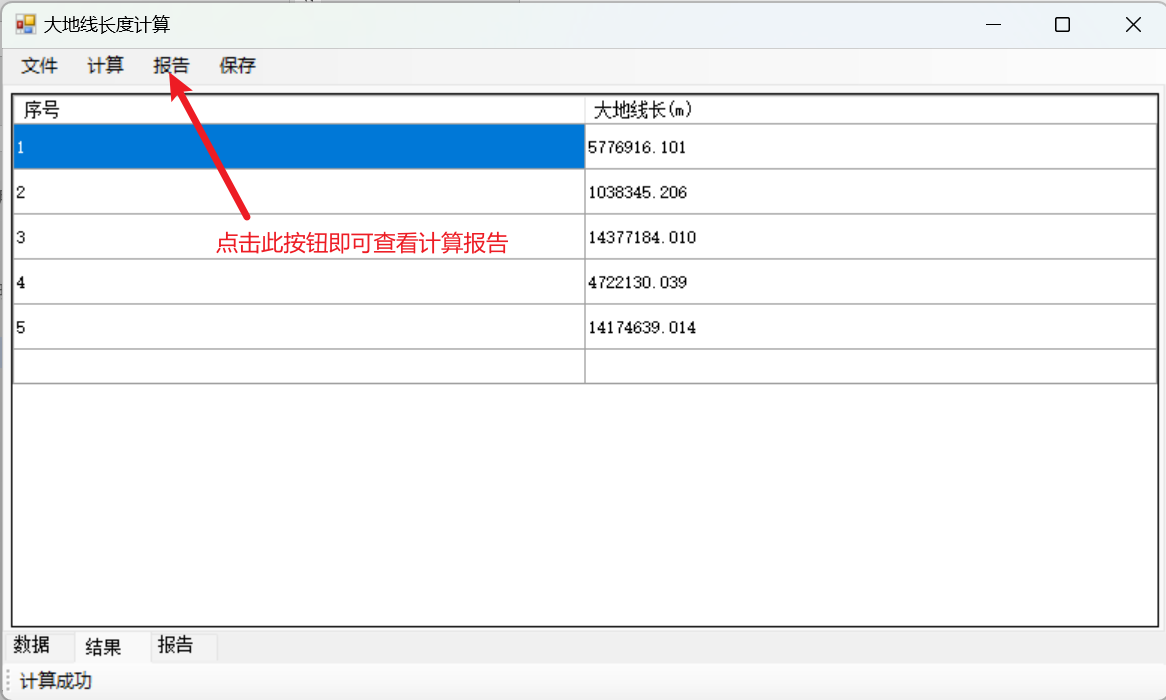
点击功能区的计算按钮即可计算

同时会切换界面至结果显示界面，可清晰看到计算结果

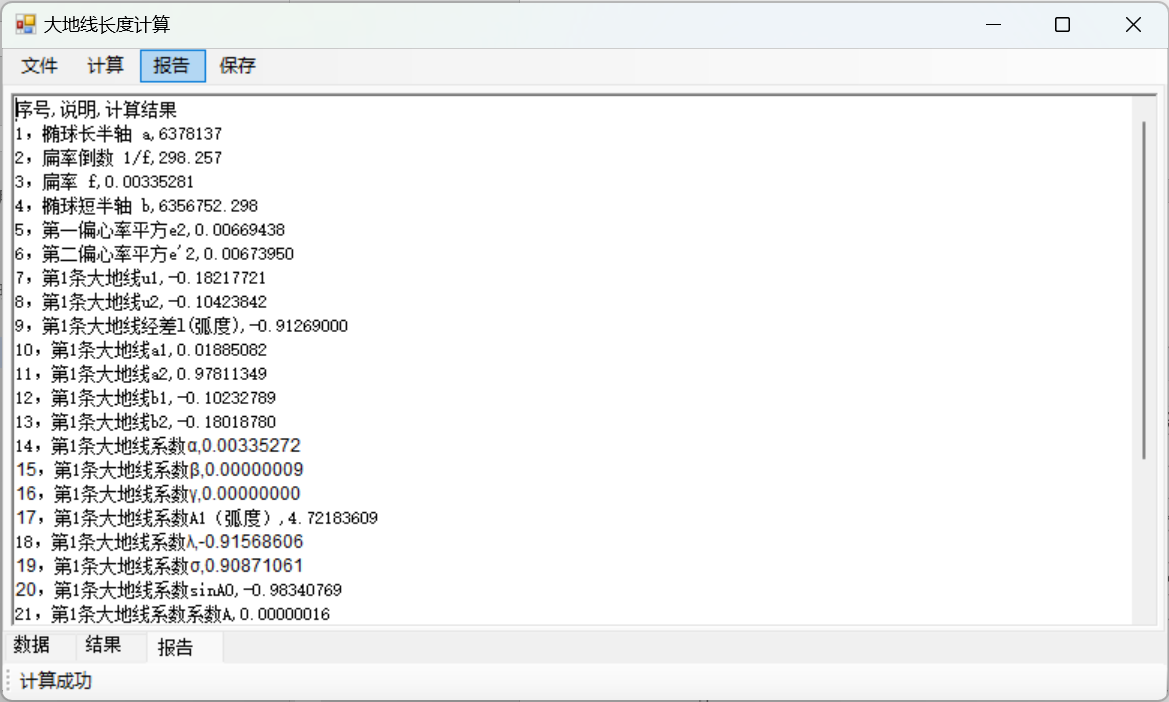
左下角的状态区也会改变，显示“计算成功”



### 2.3查看计算报告



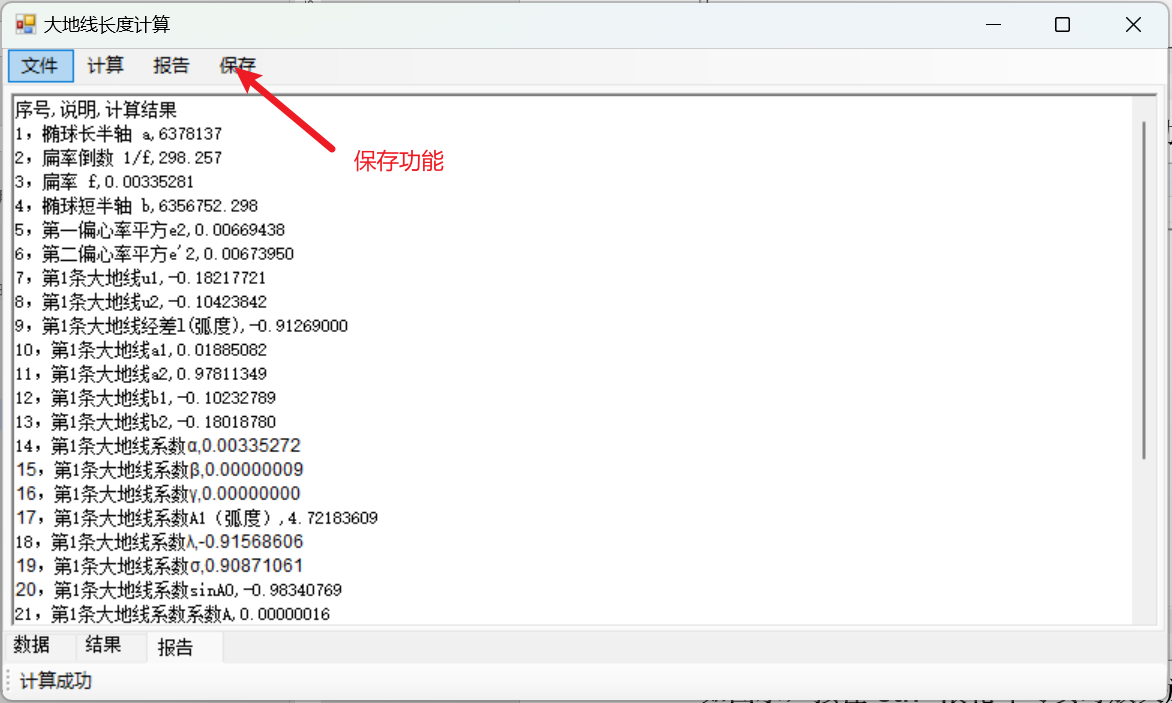
点击此按钮即可查看计算报告，会自动切换页面至报告显示界面



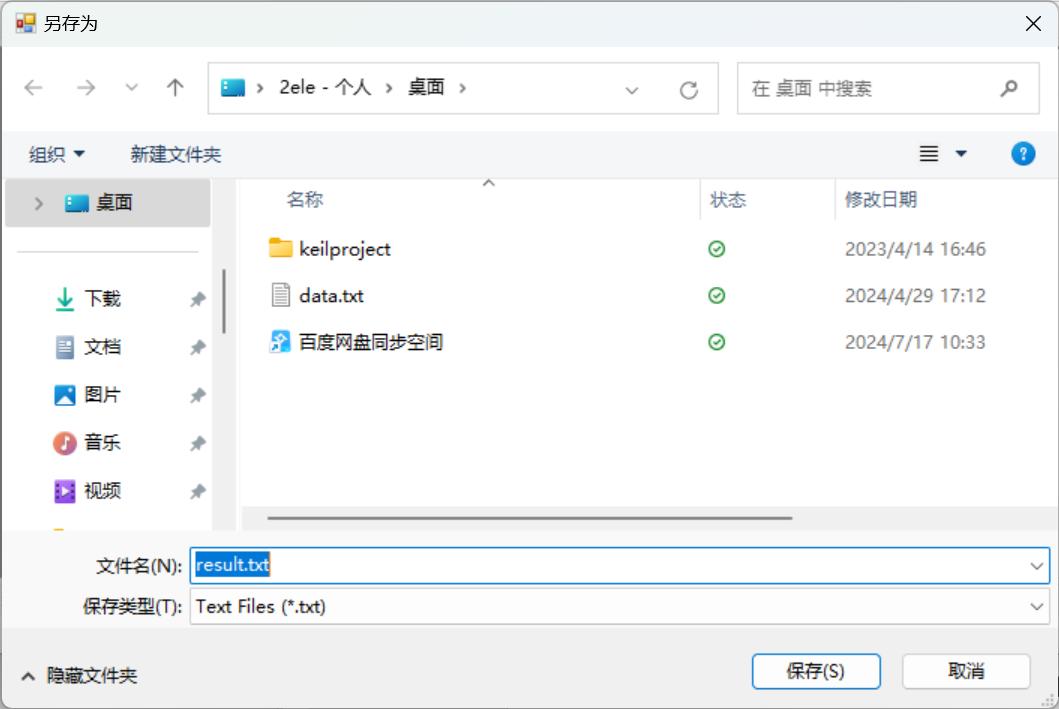
如图示，按住Ctrl+滚轮即可实时放大放小

### 2.4保存计算报告

点击功能区按钮“保存键”即可实现保存当前报告内容



同时会弹出运行框，自由选择保存位置，保存为“result.txt”文件



## 3．程序运行结果

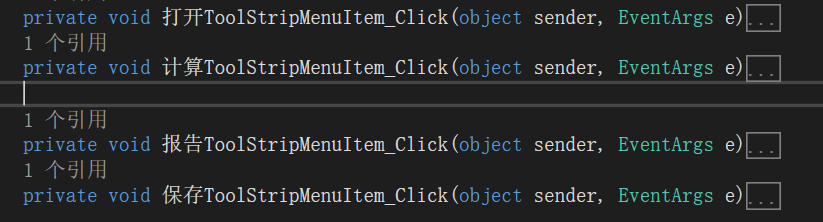
程序运行结果如下:

|  |
| --- |
| 序号,说明,计算结果  1，椭球长半轴 a,6378137  2，扁率倒数 1/f,298.257  3，扁率 f,0.00335281  4，椭球短半轴 b,6356752.298  5，第一偏心率平方e2,0.00669438  6，第二偏心率平方e'2,0.00673950  7，第1条大地线u1,-0.18217721  8，第1条大地线u2,-0.10423842  9，第1条大地线经差l(弧度),-0.91269000  10，第1条大地线a1,0.01885082  11，第1条大地线a2,0.97811349  12，第1条大地线b1,-0.10232789  13，第1条大地线b2,-0.18018780  14，第1条大地线系数?,0.00335272  15，第1条大地线系数?,0.00000009  16，第1条大地线系数?,0.00000000  17，第1条大地线系数A1（弧度）,4.72183609  18，第1条大地线系数?,-0.91568606  19，第1条大地线系数?,0.90871061  20，第1条大地线系数sinA0,-0.98340769  21，第1条大地线系数系数A,0.00000016  22，第1条大地线系数系数B,0.00005544  23，第1条大地线系数系数C,0.00000000  24，第1条大地线系数?1,-1.51956024  25,第1条大地线系数S , 5776916.101  26，第2条大地线系数S , 1038345.206  27，第3条大地线系数S , 14377184.010  28，第4条大地线系数S , 4722130.039  29，第5条大地线系数S , 14174639.014 |

二、程序规范性说明



通过定义多个类来实现功能封装, 通过全局变量来实现各个按钮之间的联系。其中Algo类负责大地线长计算的相关算法及公式的实现。Point类这个记录初始点的信息和结束点的信息,Myfunctions类实现角度的转化。



一、功能说明

1. 文件读取：程序支持打开并读取包含地理坐标信息的文本文件。

2. 数据展示：将读取的坐标数据展示在表格中，方便用户查看。

3. 大地测量距离计算：根据给定的椭球参数和坐标点，计算两点之间的大地测量距离。

4. 结果展示：将计算结果展示在另一个表格中，并支持导出报告。

5. 报告生成与保存：生成包含椭球参数、计算过程及结果的报告，并提供保存功能。

二、结构设计

1. 主窗体（Form1）：程序的主界面，包含菜单栏、状态栏、标签页控件等。

2. Algo类：封装大地测量距离计算算法，包括Bessal\_P方法。

3. MyFunctions类：提供辅助功能，如将度分秒格式转换为弧度。

4. Input结构体：用于存储输入参数，包括椭球参数和坐标点。

5. Point类：用于存储地理坐标点信息。

通过以上设计，程序实现了从文件读取坐标数据、计算大地测量距离、展示结果以及生成报告的功能，为大地测量工作提供了便捷的工具。

2．核心算法源码

### Form1.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.IO;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using static 第三次模拟.Algo;  namespace 第三次模拟  {  public partial class Form1 : Form  {  string[] all\_lines;  Input input = new Input();  Algo algo = new Algo();  MyFunctions my = new MyFunctions();  List<Input> list = new List<Input>();  public Form1()  {  InitializeComponent();  }  private void 打开ToolStripMenuItem\_Click(object sender, EventArgs e)  {  OpenFileDialog op = new OpenFileDialog();  if (op.ShowDialog() == DialogResult.OK)  {  all\_lines = File.ReadAllLines(op.FileName, Encoding.Default);  }  else  {  return;  }  try  {  string firstLine = all\_lines[0];  string[] firstLineParts = firstLine.Split(',');  input.a = int.Parse(firstLineParts[0]); // "6378137"  input.fdao = double.Parse(firstLineParts[1]); // "298.257222"  input.fdao = Math.Round(input.fdao, 3);  for (int i = 2; i < all\_lines.Length; i++)  {  string line = all\_lines[i];  string[] parts = line.Split(',');  if (parts.Length == 6)  {  Input inp = new Input  {  a = input.a,  fdao = input.fdao,  B1 = double.Parse(parts[1]),  L1 = double.Parse(parts[2]),  B2 = double.Parse(parts[4]),  L2 = double.Parse(parts[5]),  };  list.Add(inp);  dataGridView1.Rows.Add(i - 1, parts[0], parts[1], parts[2], parts[3], parts[4], parts[5]);  }  }  }  catch  {  MessageBox.Show("文件格式错误!");  toolStripLabel1.Text = "文件格式错误!";  return;  }  }  private void 计算ToolStripMenuItem\_Click(object sender, EventArgs e)  {  for (int i = 0; i < list.Count; i++)  {  Input item = list[i];  double S = algo.Bessal\_P(item);  toolStripLabel1.Text = "计算成功";  dataGridView2.Rows.Add(i + 1, S.ToString("F3"));  }  tabControl1.SelectedTab = tabPage2;  }  private void 报告ToolStripMenuItem\_Click(object sender, EventArgs e)  {  StringBuilder report = new StringBuilder();  report.AppendLine("序号,说明,计算结果");  foreach (var entry in GlobalVariables.Variables)  {  report.AppendLine(entry.Key + ',' + entry.Value.ToString());  }  int currentLine = 25;  for (int i = 0; i < dataGridView2.Rows.Count - 1; i++)  {  string coefficientS = dataGridView2.Rows[i].Cells[1].Value?.ToString() ?? "数据无效";  currentLine++;  report.AppendFormat("{0}，第{1}条大地线系数S , {2}\n", currentLine++, i + 1, coefficientS);  }  // 更新富文本框的内容  richTextBox1.Text = report.ToString();  tabControl1.SelectedTab = tabPage3;  }  private void 保存ToolStripMenuItem\_Click(object sender, EventArgs e)  {  SaveFileDialog saveFileDialog = new SaveFileDialog();  saveFileDialog.Filter = "Text Files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";  saveFileDialog.FileName = "result";  if (saveFileDialog.ShowDialog() == DialogResult.OK)  {  string filePath = saveFileDialog.FileName;  using (StreamWriter writer = new StreamWriter(filePath))  {  writer.Write(richTextBox1.Text);  }  }  }  }  } |

### Algo.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace 第三次模拟  {  class Algo  {  private static bool isFirstCall = true;  public struct Input  {  public double a;  public double fdao;  public double b;  public double e2;  public double epie2;  public double L1;  public double B1;  public double L2;  public double B2;  }  public static class GlobalVariables  {  public static Dictionary<string, string> Variables { get; set; }  }  public double Bessal\_P(Input input)  {  #region 变量  double u1;  double u2;  double L;  double a1;  double a2;  double b1;  double b2;  double p;  double q;  double lamda;  double sigema;  double A1;  double Sin\_sigema;  double Cos\_sigema;  double sin\_A0;  double sigma1;  double alpha = 0;  double beta;  double gamma;  double epsilon;  double k2;  double A = 0;  double B = 0;  double C = 0;  double xs;  double S;  double temp;  #endregion  input.b = input.a \* (1 - 1 / input.fdao);  input.e2 = ((input.a \* input.a) - (input.b \* input.b)) / (input.a \* input.a);  input.epie2 = (input.a \* input.a - input.b \* input.b) / (input.b \* input.b);  //辅助计算  u1 = Math.Atan(Math.Sqrt(1 - input.e2) \* Math.Tan(input.B1));  u2 = Math.Atan(Math.Sqrt(1 - input.e2) \* Math.Tan(input.B2));  L = input.L2 - input.L1;  a1 = Math.Sin(u1) \* Math.Sin(u2);  a2 = Math.Cos(u1) \* Math.Cos(u2);  b1 = Math.Cos(u1) \* Math.Sin(u2);  b2 = Math.Sin(u1) \* Math.Cos(u2);  //逐次趋近法  lamda = L;  temp = 0;  while (true)  {  p = Math.Cos(u2) \* Math.Sin(lamda);  q = b1 - b2 \* Math.Cos(lamda);  A1 = Math.Atan(p / q);  if (p > 0 && q > 0)  A1 = Math.Abs(A1);  else if (p > 0 && q < 0)  A1 = Math.PI - Math.Abs(A1);  else if (p < 0 && q < 0)  A1 = Math.Abs(A1) + Math.PI;  else  A1 = 2 \* Math.PI - Math.Abs(A1);  if (A1 < 0)  {  A1 = A1 + 2 \* Math.PI;  }  else if (A1 > 2 \* Math.PI)  {  A1 = A1 - 2 \* Math.PI;  }  Sin\_sigema = p \* Math.Sin(A1) + q \* Math.Cos(A1);  Cos\_sigema = a1 + a2 \* Math.Cos(lamda);  sigema = Math.Atan(Sin\_sigema / Cos\_sigema);  if (Cos\_sigema > 0)  {  sigema = Math.Abs(sigema);  }  else if (Cos\_sigema < 0)  {  sigema = Math.PI - Math.Abs(sigema);  }  sin\_A0 = Math.Cos(u1) \* Math.Sin(A1);  alpha = (input.e2 / 2 + input.e2 \* input.e2 / 8 + input.e2 \* input.e2 \* input.e2 / 16) -  (input.e2 \* input.e2 / 16 + input.e2 \* input.e2 \* input.e2 / 16) \* (1 - Math.Pow(sin\_A0, 2)) +  (3 \* Math.Pow(input.e2, 3) / 128) \* (1 - Math.Pow(sin\_A0, 2)) \* (1 - Math.Pow(sin\_A0, 2));  beta = (Math.Pow(input.e2, 2) / 16 + Math.Pow(input.e2, 3) / 16) \* (1 - Math.Pow(sin\_A0, 2)) -  (Math.Pow(input.e2, 3) / 32) \* (1 - Math.Pow(sin\_A0, 2)) \* (1 - Math.Pow(sin\_A0, 2));  gamma = Math.Pow(input.e2, 3) / 256 \* (1 - Math.Pow(sin\_A0, 2)) \* (1 - Math.Pow(sin\_A0, 2));  sigma1 = Math.Atan(Math.Tan(u1) / Math.Cos(A1));  epsilon = ((alpha \* sigema) + (beta \* Math.Cos(2 \* sigma1 + sigema) \* Math.Sin(sigema)) +  (gamma \* Math.Sin(2 \* sigema) \* Math.Cos(4 \* sigma1 + 2 \* sigema))) \* sin\_A0;  lamda = L + epsilon;  if (Math.Abs(epsilon - temp) <= 1e-10)  {  temp = epsilon;  break;  }  temp = epsilon;  }  //计算S  k2 = input.epie2 \* (1 - Math.Pow(sin\_A0, 2));  A = (1 - k2 / 4 + 7 \* k2 \* k2 / 64 - 15 \* k2 \* k2 \* k2 / 256) / input.b;  B = k2 / 4 - k2 \* k2 / 8 + 37 \* k2 \* k2 \* k2 / 512;  C = k2 \* k2 / 128 - k2 \* k2 \* k2 / 128;  sigma1 = Math.Atan(Math.Tan(u1) / Math.Cos(A1));  xs = C \* Math.Sin(2 \* sigema) \* Math.Cos(4 \* sigma1 + 2 \* sigema);  S = (sigema - B \* Math.Sin(sigema) \* Math.Cos(2 \* sigma1 + sigema) - xs) / A;  if (isFirstCall)  {  GlobalVariables.Variables = new Dictionary<string, string>  {  { "1，椭球长半轴 a", input.a.ToString("F0") },  { "2，扁率倒数 1/f",input.fdao.ToString("F3") },  { "3，扁率 f", (1 / input.fdao).ToString("F8") },  { "4，椭球短半轴 b", input.b.ToString("F3") },  { "5，第一偏心率平方e2", input.e2.ToString("F8") },  { "6，第二偏心率平方e'2", input.epie2.ToString("F8") },  { "7，第1条大地线u1", u1.ToString("F8") },  { "8，第1条大地线u2", u2.ToString("F8")},  { "9，第1条大地线经差l(弧度)", L.ToString("F8") },  { "10，第1条大地线a1", a1.ToString("F8") },  { "11，第1条大地线a2", a2.ToString("F8") },  { "12，第1条大地线b1", b1.ToString("F8") },  { "13，第1条大地线b2", b2.ToString("F8") },  { "14，第1条大地线系数α", alpha.ToString("F8") },  { "15，第1条大地线系数β", beta.ToString("F8") },  { "16，第1条大地线系数γ", gamma.ToString("F8") },  { "17，第1条大地线系数A1（弧度）", A1.ToString("F8") },  { "18，第1条大地线系数λ", lamda.ToString("F8") },  { "19，第1条大地线系数σ", sigema.ToString("F8") },  { "20，第1条大地线系数sinA0", sin\_A0.ToString("F8") },  { "21，第1条大地线系数系数A", A.ToString("F8") },  { "22，第1条大地线系数系数B", B.ToString("F8") },  { "23，第1条大地线系数系数C", C.ToString("F8") },  { "24，第1条大地线系数σ1", sigma1.ToString("F8") },  };  isFirstCall = false;  }  return S;  }  }  } |

### MyFunctions.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace 第三次模拟  {  class MyFunctions  {  public double ddmmssTorad(double ddmmss)  {  double degrees;  double minutes;  double seconds;  degrees = Math.Floor(ddmmss);  minutes = Math.Floor((ddmmss - degrees) \* 100);  seconds = (ddmmss - degrees - minutes / 100) \* 10000;  double rad = (degrees + minutes / 60.0 + seconds / 3600.0) \* (Math.PI / 180.0);  return rad;  }  }  } |

### Point.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace 第三次模拟  {  public class Point  {  public string name1;  public double B1;  public double B2;  public string name2;  public double L1;  public double L2;  }  } |