

武汉大学

WUHAN UNIVERSITY



坐标转换与坐标系转换 编程实验报告

院 系 测 绘 学 院

学 号 2021302141134

年 级 2 0 2 1 级

班 级 2 1 0 1 班

姓 名 刘 恒 祯

时 间 二〇二二年十一月

目录

一、四参数坐标转换（小角度）	1
1、算法原理及流程	1
1) 算法原理	1
2) 算法流程	2
2、关键中间结果及最终结果	4
1) 程序运行结果	4
2) 特定点运行结果	4
3) 窗体运行界面	5
全部数据请详见电子文档“四参数计算结果.txt”。	5
二、七参数/六参数坐标转换（小角度）	5
1、算法原理及流程	5
1) 算法原理	5
2) 算法流程	6
2、关键中间结果及最终结果	9
1) 程序运行结果	9
2) 特定点运行结果	9
3) 窗体运行界面	10
全部数据请详见电子文档“七参数计算结果.txt”。	10
三、地心地固系与大地坐标系转换	10
1、算法原理及流程	10
1) 算法原理	10
2) 算法流程	11
2、关键中间结果及最终结果	14
1) 程序调试结果	14
2) 大地坐标系结果（前 10 个点）	14
3) 大地坐标系结果（全部 14435 个点）	15
四、地心地固系与站心坐标系转换	16
1、算法原理及流程	16
1) 算法原理	16
2) 算法流程	17
2、关键中间结果及最终结果	19
1) 程序调试结果	19
2) 站心坐标系结果（前 10 个点）	20
3) 站心坐标系结果（全部 14435 个点）	20
五、 作业感想	22

一、四参数坐标转换（小角度）

1、算法原理及流程

1) 算法原理

四参数坐标转换广泛运用于测绘的各项作业中。由于我国以 54 北京坐标系为主，其他地方有建有相应的地方独立系统，故在进行测量工作时，常常需要进行两个不同坐标系之间的转换。该类型的转换为同一个椭球系统的不同坐标系的转换，亦为二维平面坐标转换，可以推导基本数学模型如下：

$$\begin{bmatrix} X_B \\ Y_B \end{bmatrix} = \begin{bmatrix} X_A \\ Y_A \end{bmatrix} + \begin{bmatrix} 1 & 0 & Y_A & X_A \\ 0 & 1 & -X_A & Y_A \end{bmatrix} \begin{bmatrix} DX \\ DY \\ DR \\ DK \end{bmatrix} \quad (2.1)$$

上式中 (X_A, Y_A) 为源坐标系坐标， (X_B, Y_B) 为目标坐标系坐标， DX 、 DY 为平移参数， DR 为旋转参数， DK 为尺度因子。不难发现可以将该式子变形成为间接平差的形式，模型如下：

$$\begin{bmatrix} v_A \\ v_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & Y_A & X_A \\ 0 & 1 & -X_A & Y_A \end{bmatrix} \begin{bmatrix} DX \\ DY \\ DR \\ DK \end{bmatrix} - \begin{bmatrix} X_B - X_A \\ Y_B - Y_A \end{bmatrix} \quad (2.2)$$

因此必要观测量为 4，而要求中给出的参数点也有四个，即得到 8 个参数，故多余观测量为 4，其中的 B，L 矩阵如下

$$\text{B 矩阵: } \begin{bmatrix} 1 & 0 & Y_A & X_A \\ 0 & 1 & -X_A & Y_A \end{bmatrix} \quad (2.3) \quad \text{L 矩阵: } \begin{bmatrix} X_B - X_A \\ Y_B - Y_A \end{bmatrix} \quad (2.4)$$

最终带入 4 个转换公共点，得到的 B 矩阵是一个 8×4 的矩阵，而 L 是一个 8×1 的矩阵，根据间接平差的公式，求出 X 矩阵，即可得到要求取的四参数。

$$X = (B^T P B)^{-1} B^T P L \quad (2.5)$$

同时运用如下的间接平差单位权中误差公式，利用 V 矩阵和 n，t 即可求得验后单位权中误差。

$$\sigma = \sqrt{\frac{V^T P V}{n - t}} \quad (2.6)$$

解算完成后，就可以利用得到的四参数开始进行估计，利用 2.1 式，将原来

origin1 文件中的坐标加上有关自己学号的坐标代入计算，即可得到估计矩阵。

2) 算法流程

【注】：由于先完成的是七参数转换，四参数转换实际上不需要改动七参数的代码，直接默认所有 Z 分量等于 0 即可，算出的答案一定是正确的。不过，为了满足编程实习任务的流程要求，我还是按四参数的要求修改了相应的 B、1 矩阵等，也比较容易。在项目文件里能够看到我注释更改的内容，实际上两种方法我都跑通过了，最后留的是实习要求的标准四参数流程。

(1) C# WinForm 窗体设计

- ①使用 GroupBox、RadioButton、Label 控件提供选择“四参数 XOY”功能；
- ②RichTextBox 提供关于转换前后坐标数据、验核数据、特定点计算结果的显示，在此，将其“ReadOnly”属性设置为“True”，避免了对窗体内文本的误操作；
- ③几个 Button 控件关联“Click”事件，完成打开坐标文件、计算数据、清除全部内容（含释放内存、空间的选定状态）、保存坐标文件等功能；
- ④另外的 Label、GroupBox、textBox 提供简洁规范参数输出显示功能。

(2) 数据载入

使用 OpenFileDialog，StreamReader，Split 方法读入 txt，实现整行数据分割读入，用 list 分别存储点名、X0、Y0、Z0 坐标，同时在相应 RichTextBox 显示初始坐标数据。

其中：

- ①对于 new 的 OpenFileDialog 的对象“file”，限定“file.InitialDirectory”为“StartupPath”，“file.RestoreDirectory”为“True”，以及文件后缀类型“fileFilter”为“All Files (*.*)|*.txt|Dat Files (*.dat)|*.dat|Text Files (*.txt)|*.txt”，优化了寻找坐标文件的便捷性；
- ②用 list 的 Add 方法，避免了选择数组方法对于设置数组大小的限制，list 容量自动扩充，索引方便。
- ③载入 list 数据的同时，用 string 类型的 str 接收来自原始 txt 的全部数据信息，继而在 RichTextBox 中得到显示。

(3) 数据计算与显示

①定义了 bool 变量 j0 与 j，在 RadioButton 选定且数据成功载入之后，根据 if 的条件判断为真才可以进行数据计算，避免报错；

②根据“算法原理”，调用 matrix.cs，定义 Matrix_l、Matrix_B、Matrix_BT、Matrix_P、Matrix_BTP、Matrix_BTPB、Matrix_BTPB_l、Matrix_BTPB_lBT、Matrix_BTPB_lBTP、Matrix_BTPB_lBTPl、Matrix_Bxx、Matrix_V、Matrix_VT、Matrix_VTP、Matrix_VTPV，利用 matrix.cs 中的 matToStr 矩阵的显示、matTran 矩阵的转置、matInv 矩阵的求逆、matAdd 矩阵加法、matSub 矩阵减法、matMul 矩阵乘法，求得 V 矩阵，以及 Sigma (σ)，进而分别得到相应的参数，进行公共点坐标残差、验核点坐标残差、验后单位权中误差的计算，分别用（一维）矩阵等存储，并根据 list 中的数据量、求解的参数数量得到并记录条件数和冗余度，并在相应的 RichTextBox 输出。

③用 ToString 的 Substring 方法，将最后一组原始坐标数据小数点后三位改为自己的学号后三位存在 Matrix_Test 矩阵，继而在定义并对 Matrix_bb 矩阵赋值后，根据已解算的 Matrix_BTPB_lBTPl (\hat{x})，调用 matrix.cs 进行矩阵计算估计解算转换后的坐标存储到 Matrix_TargetPoint 中。

④4 个 textBox 中分别对应 Matrix_BTPB_lBTPl (\hat{x}) 中 4 个解算好的参数，分别在窗体中的“输出窗口”的 GroupBox 中显示。

[注：原本有 7 个 textbox，但因为三个参数为 0 不需要考虑。]

(4) 数据的清除

① 在载入数据，点击“Compute”按钮计算得到结果后，如果想选择另一个 RadioButton 的功能重新计算，那么必须释放 list 与其他计算存储变量，（包含 bool j, j0）中存储的数据，以及控件的数据与状态，继而重新进行（2）-（3）的数据计算与显示操作。

②对应的方法是 radioButton.Checked = false; richTextBox.Clear(); textBox.Clear(); list_xxx.Clear(); j = false;

(5) 数据计算结果保存

①点击“Save” Button，依然采用（radioButton1.Checked == true && j == true）作为是否执行结果保存命令的条件。

②使用 SaveFileDialog，StreamWriter，方法写入 txt 文件文本，同时对于 new 的 SaveFileDialog 的对象“file”，限定“file.InitialDirectory”为“StartupPath”，“file.RestoreDirectory”为“True”，便于保存至当前根目录，以及文件后缀类型“fileFilter”设定为“All Files (*.*)|*. *|Dat Files (*.dat)|*. dat|Text Files (*.txt)|*. txt”，通过赋值“file.FilterIndex = 3”，默认文件以“.txt”文本文件类型存储，优化了存储坐标文件的便捷性。

2、关键中间结果及最终结果

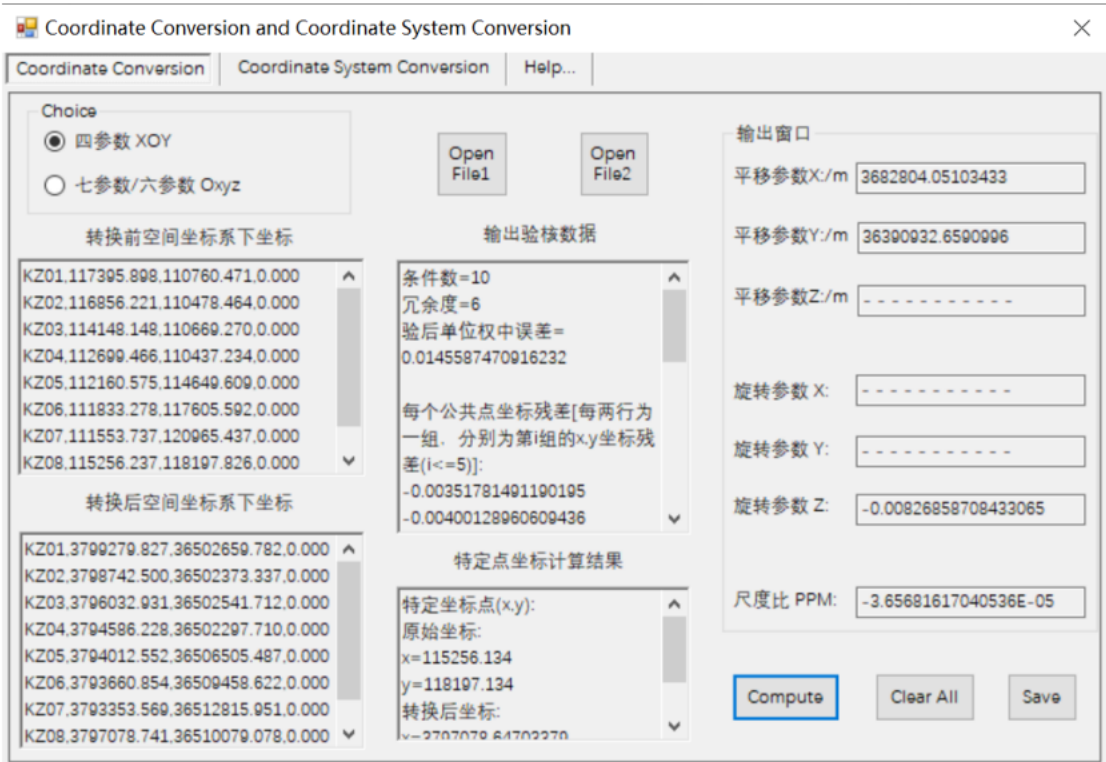
1) 程序运行结果

点号	参数值	公共点坐标残差 (Δx , Δy)	检核点坐标残差 (Δx , Δy)	条件数与 冗余度	单位权中误差
1	平移参数 DX/m	-0.003518	-0.046582	n=10	0.0145587471
	3682804.05103433	-0.004001	0.031677	r=6	
2	平移参数 DY/m	-0.001983	-0.073531		
	36390932.65909960	0.018055	0.113405		
3	旋转参数 DR	0.015350	0.003308		
	-0.00826859	0.014030	0.091055		
4	尺度比 DK/PPM	0.007935			
	-0.000036568	0.009962			
5		-0.017748			
		-0.001944			

2) 特定点运行结果

特定转换点	X	Y	Z
源坐标	115256.134	118197.134	0.000
目的坐标	3797078.64703379	36510078.4762286	0.000

3) 窗体运行界面



全部数据请详见电子文档“四参数计算结果.txt”。



四参数计算结果.txt

二、七参数/六参数坐标转换（小角度）

1、算法原理及流程

1) 算法原理

七参数坐标转换模型通常运用于两个不同的三维空间直角坐标系之间转换。我国常用的系统由北京 1954、西安 1980、WCG-84 和 CGCS2000，而这些坐标系统之间的三维坐标转换常用的便是七参数模型。作为一个三维坐标转换模型，其基本的数学模型如下：

$$\begin{bmatrix} X_B \\ Y_B \\ Z_B \end{bmatrix} = \begin{bmatrix} X_A \\ Y_A \\ Z_A \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 & -Z_A & Y_A & X_A \\ 0 & 1 & 0 & Z_A & 0 & -X_A & Y_A \\ 0 & 0 & 1 & -Y_A & X_A & 0 & Z_A \end{bmatrix} \begin{bmatrix} DX \\ DY \\ DZ \\ RX \\ RY \\ RZ \\ DK \end{bmatrix} \quad (3.1)$$

上式中 (X_B, Y_B, Z_B) 为目标坐标系坐标， (X_A, Y_A, Z_A) 为源坐标系坐标， DX 、 DY 、

DZ 为平移参数, RX 、 RY 、 RZ 为旋转参数, DK 为尺度因子。该式子也可以转换为间接平差的模型, 变换后的形式如下:

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & -Z_A & Y_A & X_A \\ 0 & 1 & 0 & Z_A & 0 & -X_A & Y_A \\ 0 & 0 & 1 & -Y_A & X_A & 0 & Z_A \end{bmatrix} \begin{bmatrix} DX \\ DY \\ DZ \\ RX \\ RY \\ RZ \\ DK \end{bmatrix} - \begin{bmatrix} X_B - X_A \\ Y_B - Y_A \\ Z_B - Z_A \end{bmatrix} \quad (3.2)$$

七参数模型中通常至少需要知道三个公共已知点, 才能解算出转换参数, 因为我们用了四组观测值, 所以存在多余观测, 条件数 $n=12$, 冗余度 $r=5$ 。同时我们可以列出 B 矩阵和 L 矩阵, 其式如下:

$$\text{B 矩阵: } \begin{bmatrix} 1 & 0 & 0 & 0 & -Z_A & Y_A & X_A \\ 0 & 1 & 0 & Z_A & 0 & -X_A & Y_A \\ 0 & 0 & 1 & -Y_A & X_A & 0 & Z_A \end{bmatrix} \quad (3.3) \quad \text{L 矩阵: } \begin{bmatrix} DX \\ DY \\ DZ \\ RX \\ RY \\ RZ \\ DK \end{bmatrix} \quad (3.4)$$

最终带入 4 个转换公共点, 得到的 B 矩阵是一个 12×7 的矩阵, 而 L 是一个 7×1 的矩阵, 根据间接平差的公式, 求出 X 矩阵, 即可得到要求取的四参数。

$$X = (B^T P B)^{-1} B^T P L \quad (3.5)$$

同时运用如下的间接平差单位权中误差公式, 利用 V 矩阵和 n, t 即可求得验后单位权中误差。

$$\sigma = \sqrt{\frac{V^T P V}{n - t}} \quad (3.6)$$

解算完成后, 就可以利用得到的七参数开始进行估计, 利用 3.1 式, 将原来 origin2 文件中的坐标加上有关自己学号的坐标代入计算, 即可得到估计矩阵。

2) 算法流程

(1) C# WinForm 窗体设计

①使用 GroupBox、RadioButton、Label 控件提供选择“七参数/六参数 0xyz”功能;

②RichTextBox 提供关于转换前后坐标数据、验核数据、特定点计算结果的显示，在此，将其“ReadOnly”属性设置为“True”，避免了对窗体内文本的误操作；

③几个 Button 控件关联“Click”事件，完成打开坐标文件、计算数据、清除全部内容（含释放内存、空间的选定状态）、保存坐标文件等功能；

④另外的 Label、GroupBox、textBox 提供简洁规范的参数输出显示功能。

（2）数据载入

使用 OpenFileDialog, StreamReader, Split 方法读入 txt，实现整行数据分割读入，用 list 分别存储点名、X0、Y0、Z0 坐标，同时在相应 RichTextBox 显示初始坐标数据。

其中：

①对于 new 的 OpenFileDialog 的对象“file”，限定“file.InitialDirector”为“StartupPath”，“file.RestoreDirectory”为“True”，以及文件后缀类型“fileFilter”为“All Files(*.*)|*.txt|Dat Files(*.dat)|*.dat|Text Files(*.txt)|*.txt”，优化了寻找坐标文件的便捷性；

②用 list 的 Add 方法，避免了选择数组方法对于设置数组大小的限制，list 容量自动扩充，索引方便。

③载入 list 数据的同时，用 string 类型的 str 接收来自原始 txt 的全部数据信息，继而在 RichTextBox 中得到显示。

（3）数据计算与显示

①定义了 bool 变量 j0 与 j，在 RadioButton 选定且数据成功载入之后，根据 if 的条件判断为真才可以进行数据计算，避免报错；

②根据“算法原理”，调用 matrix.cs，定义 Matrix_l、Matrix_B、Matrix_BT、Matrix_P、Matrix_BTP、Matrix_BTPB、Matrix_BTPB_l、Matrix_BTPB_lBT、Matrix_BTPB_lBTP、Matrix_BTPB_lBTPl、Matrix_Bxx、Matrix_V、Matrix_VT、Matrix_VTP、Matrix_VTPV，利用 matrix.cs 中的 matToStr 矩阵的显示、matTran 矩阵的转置、matInv 矩阵的求逆、matAdd 矩阵加法、matSub 矩阵减法、matMul

矩阵乘法，求得 V 矩阵，以及 $\text{Sigma}(\sigma)$ ，进而分别得到相应的参数，进行公共点坐标残差、验核点坐标残差、验后单位权中误差的计算，分别用（一维）矩阵等存储，并根据 list 中的数据量、求解的参数数量得到并记录条件数和冗余度，并在相应的 RichTextBox 输出。

③用 ToString 的 Substring 方法，将最后一组原始坐标数据小数点后三位改为自己的学号后三位存在 Matrix_Test 矩阵，继而在定义并对 Matrix_bb 矩阵赋值后，根据已解算的 Matrix_BTPB_1BTP1 (\hat{x})，调用 matrix.cs 进行矩阵计算估计解算转换后的坐标存储到 Matrix_TargetPoint 中。

④7 个 textBox 中分别对应 Matrix_BTPB_1BTP1 (\hat{x}) 中 7 个解算好的参数，分别在窗体中的“输出窗口”的 GroupBox 中显示。

（4）数据的清除

① 在载入数据，点击“Compute”按钮计算得到结果后，如果想选择另一个 RadioButton 的功能重新计算，那么必须释放 list 与其他计算存储变量，（包含 bool j, j0）中存储的数据，以及控件的数据与状态，继而重新进行（2）-（3）的数据计算与显示操作。

②对应的方法是 `radioButton.Checked = false; richTextBox.Clear(); textBox.Clear(); list_xxx.Clear(); j = false;`

（5）数据计算结果保存

①点击“Save”Button，依然采用（`radioButton1.Checked == true && j == true`）作为是否执行结果保存命令的条件。

②使用 SaveFileDialog, StreamWriter，方法写入 txt 文件文本，同时对于 new 的 SaveFileDialog 的对象“file”，限定“file.InitialDirectory”为“StartupPath”，“file.RestoreDirectory”为“True”，便于保存至当前根目录，以及文件后缀类型“fileFilter”设定为“All Files (*.*)|*.*|Dat Files (*.dat)|*.dat|Text Files (*.txt)|*.txt”，通过赋值“file.FilterIndex = 3”，默认文件以“.txt”文本文件类型存储，优化了存储坐标文件的便捷性。

2、关键中间结果及最终结果

1) 程序运行结果

点号	参数值	公共点坐标残差 ($\Delta x, \Delta y, \Delta z$)	检核点坐标残差 ($\Delta x, \Delta y, \Delta z$)	条件数与冗余度	单位权中误差
1	DX:273.18952063	-0.007591	0.007620	n=12	0.0360744272503
	DY:55.15872763	0.004946	0.028811	r=5	
	DZ:117.42078992	-0.035298	0.046039		
2	RX:0.00001482455	0.006900	0.017410		
	RY:0.00001583661	-0.002123	0.034264		
	RZ:-0.0000222871	-0.013493	0.057219		
3	DK:0.00000543233	0.028324			
		0.011711			
		0.054729			
4		-0.027633			
		-0.014533			
		-0.005938			

2) 特定点运行结果

转换点	X	Y	Z
待求点坐标	-2100573.134	5496675.134	2894377.134
转换后坐标	-2100469.42738492	5496729.37096009	2894381.37541947

3) 窗体运行界面

全部数据请详见电子文档“七参数计算结果.txt”。

三、地心地固系与大地坐标系转换



七参数计算结果.txt

1、算法原理及流程

1) 算法原理

大地坐标与空间直角坐标之间的关系式为：

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (N + H)\cos B \cos L \\ (N + H)\cos B \sin L \\ [N(1 - e^2) + H]\sin B \end{bmatrix} \quad \text{其中, } N = \frac{a}{\sqrt{1 - e^2 \sin^2 B}}$$

$$\Rightarrow \begin{cases} \tan B = \frac{Z + Ne^2 \sin B}{\sqrt{X^2 + Y^2}} \\ H = \frac{Z}{\sin B} - N(1 - e^2) \\ H = \frac{\sqrt{X^2 + Y^2}}{\cos B} - N \\ L = \arctan \frac{Y}{X} \end{cases}$$

$$\begin{aligned} \tan B &= \frac{Z + Ne^2 \sin B}{\sqrt{X^2 + Y^2}} = \frac{Z}{\sqrt{X^2 + Y^2}} + \frac{Ne^2 \sin B}{\sqrt{X^2 + Y^2}} \\ &= \frac{Z}{\sqrt{X^2 + Y^2}} + \frac{ae^2}{\sqrt{X^2 + Y^2}} \frac{\tan B}{\sqrt{(1 - e^2) \tan^2 B + 1}} \end{aligned}$$

$$t_{i+1} = \tan B, t_0 = \frac{Z}{\sqrt{X^2 + Y^2}}, k = \frac{ae^2}{\sqrt{X^2 + Y^2}}, p = 1 - e^2$$

$$t_{i+1} = t_0 + k \frac{t_i}{\sqrt{pt_i^2 + 1}}, |t_{i+1} - t_i| \leq \varepsilon$$

2) 算法流程

(1) C# WinForm 窗体设计

- ①使用 GroupBox、RadioButton、Label 控件提供选择“XYZ to BLH”功能；
- ②RichTextBox 提供关于转换前后坐标数据、坐标点计算结果的显示，在此，将其“ReadOnly”属性设置为“True”，避免了对窗体内文本的误操作；
- ③几个 Button 控件关联“Click”事件，完成打开坐标文件、计算数据、清除全部内容（含释放内存、空间的选定状态）、保存坐标文件等功能；

(2) 数据载入

- ①在 Point.cs 内定义了“XYZPoint”“BLHPoint”“NEUPoint”三个类，并用属性的{get; set;}来声明了关于每个点的“name”“X”“Y”“Z”数据类型为 string 和 double；判断数据是否载入的 bool 类型变量 j 值赋为“True”。

②在 Formula.cs 内定义了 WGS84 椭球的 a、e、f、b 等参数，以及为计算 B 而设置的微小量 epsilon=0.0000000000000001、pi=3.14159265358979323846；使用 OpenFileDialog, StreamReader, Split 方法读入 txt，实现整行数据分割读入，用 list<XYZPoint>类型的实例化对象 XPoint 列表，分别存储点名、X、Y、Z 坐标，同时在相应 RichTextBox 显示初始坐标数据。

③定义了以 out List<XYZPoint> Xpoint 为形参的 Readfile 函数，对于 new 的 OpenFileDialog 的对象“file”，限定“file.InitialDirector”为“StartupPath”，“file.RestoreDirectory”为“True”，以及文件后缀类型“fileFilter”为“All Files(*.*)|*.dat|Dat Files(*.dat)|*.dat|Text Files(*.txt)|*.txt”，优化了寻找坐标文件的便捷性；

④在 Form.cs 里面的关于“OpenFile”的 Button_Click () 事件响应函数中，实例化对象 XPoint 后，直接调用 formula.Readfile(out Xpoint);所有数据点的坐标信息都存在 XPoint 列表中，便于 Count 和索引。

(3) 数据的计算与显示

①“Compute”Button 与 Button_Click () 事件响应函数相关联，将 bool 类型的 jj 与 j1 作为是否执行 Compute 指令的条件，只有在 RadioButton 选定以及坐标文件打开数据成功载入之后才能执行 Compute，避免了报错。

②在 Formula.cs 中，定义了 xyz2blh(List<XYZPoint> Xpoint, out List<BLHPoint> Bpoint)函数，并且在 foreach 的每一次循环内，用迭代的方式（用到了 epsilon）解出 B，与解算出的 L、H 一起，带出 Bpoint 列表。[结合算法原理中 B 的迭代求解]

③使用 new 方法，实例化 List<BLHPoint>类的 BPoint，然后再调用 formula.xyz2blh(Xpoint, out Bpoint)函数完成对 Bpoint 数据的存储，此时将 XYZ 转为 BLH 坐标。

④string.Format () 方法控制输出空格与对齐方式，再结合 string 类型的格式控制，foreach 遍历每一个坐标点，将处理结果保存至 report1, report2 中。Report1 是 XYZ 原始坐标,Report2 是转化的 BLH 大地坐标。后在两个 RichTextBox 中分别显示计算前后数据。

(4) 数据的清除

写这个清除函数并不容易，要充分考虑到预先定义的 XPoint、BPoint、NPoint 是否实例化的问题，在此使用了 if 条件的嵌套，利用了 RadioButton 的状态、jj 与 j1 的值来进行判断因而得以解决。

方法为清除 RadioButton 的选定状态，清除 RichTextBox 内容，将 bool 类型的 jj 与 j1 重置为 false。

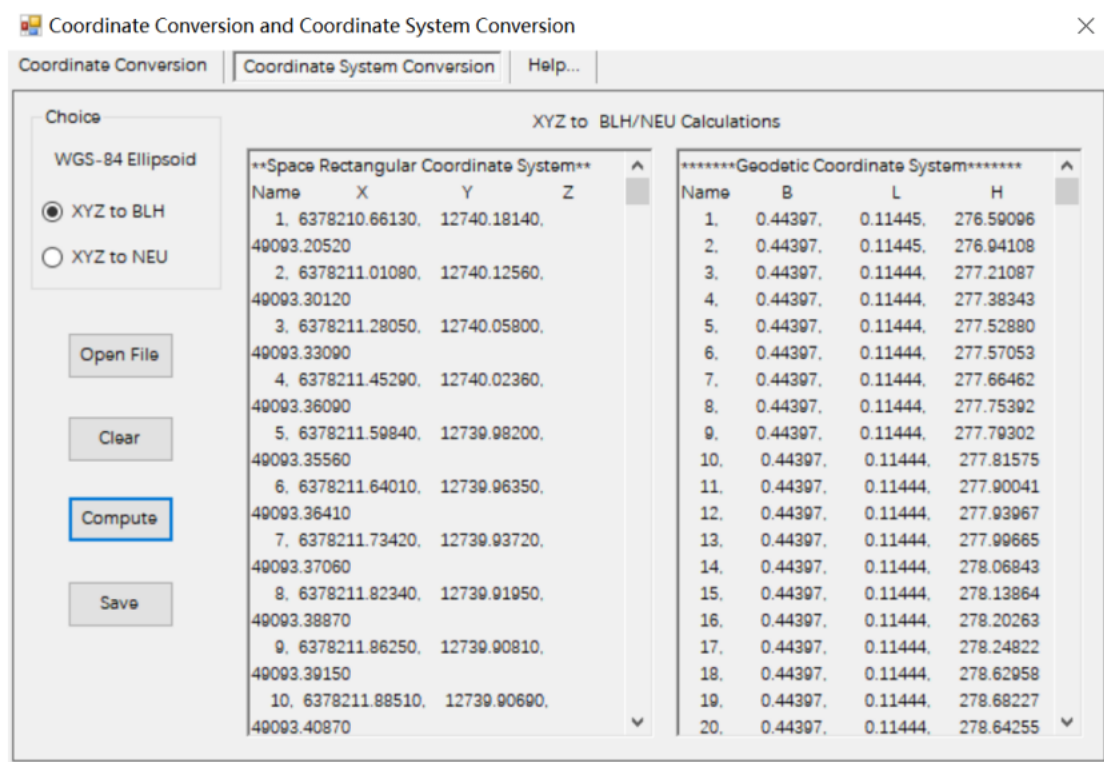
(5) 数据计算结果保存

① 点击“Save”Button，采用 (radioButton3.Checked == true && jj == true) 且 j1 为 false 时作为是否执行结果保存命令的条件。

② 使用 SaveFileDialog，StreamWriter，方法将两个转换前后的 RichTextBox 内容写入 txt 文件文本，同时对于 new 的 SaveFileDialog 的对象 “file”，限定 “file.InitialDirectory” 为 “StartupPath”，“file.RestoreDirectory” 为 “True”，便于保存至当前根目录，以及文件后缀类型 “fileFilter” 设定为 “All Files (*.*)|*.*.|Dat Files (*.dat)|*.dat|Text Files (*.txt)|*.txt”，通过赋值 “file.FilterIndex = 3”，默认文件以 “.txt” 文本文件类型存储，优化了存储坐标文件的便捷性。

2、关键中间结果及最终结果

1) 程序调试结果



2) 大地坐标系结果（前 10 个点）

*****Geodetic Coordinate System*****

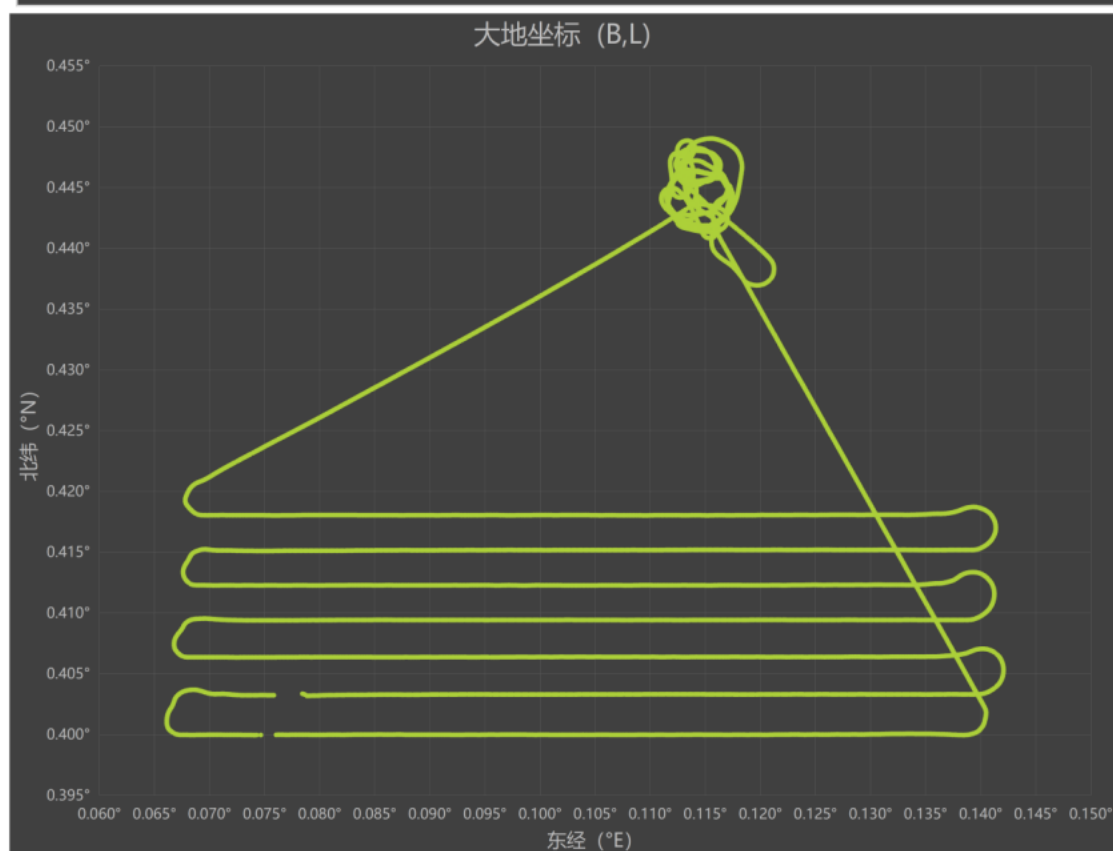
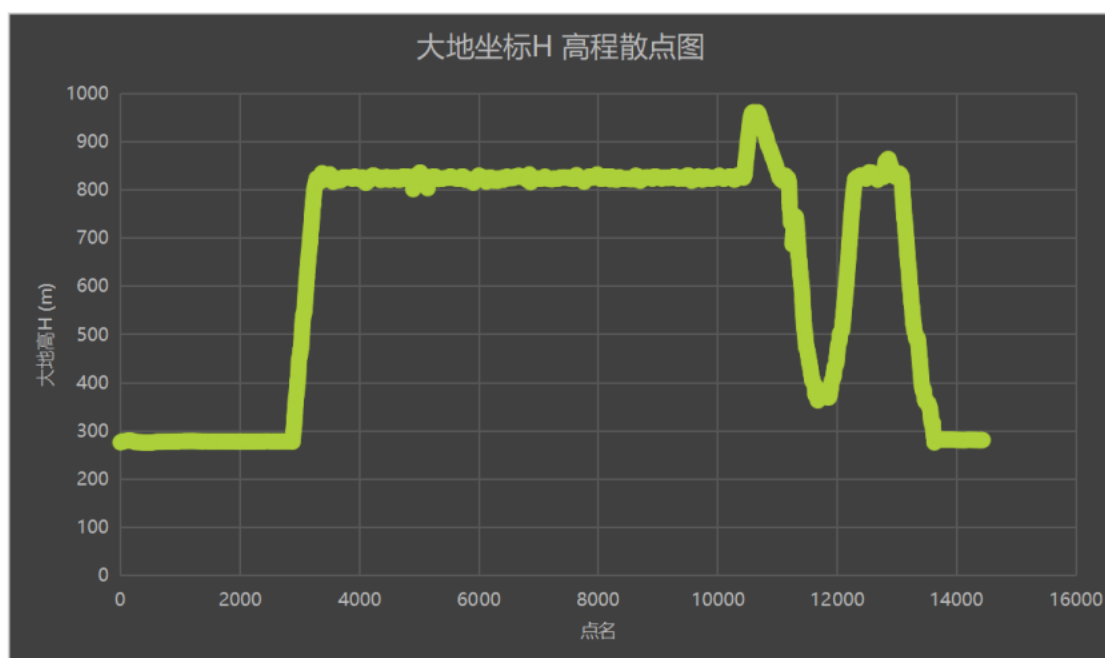
Name	B	L	H
1,	0.44397,	0.11445,	276.59096
2,	0.44397,	0.11445,	276.94108
3,	0.44397,	0.11444,	277.21087
4,	0.44397,	0.11444,	277.38343
5,	0.44397,	0.11444,	277.52880
6,	0.44397,	0.11444,	277.57053
7,	0.44397,	0.11444,	277.66462
8,	0.44397,	0.11444,	277.75392
9,	0.44397,	0.11444,	277.79302
10,	0.44397,	0.11444,	277.81575

3) 大地坐标系结果（全部 14435 个点）

应用场景猜想：

像无人机摄影测量由一个位置附近起飞调整姿态环绕升空，后平行往返扫描，再绕回降落。

[辅助论据：WGS84 的长半轴 $a=6378137$ ，数据“XYZ2BLHNEU.txt”的 x 坐标范围为 63782**, 63787**, 也就是说数据点不仅在地球表面，有一定海拔，而且集中在赤道附近（从原始 xyz 数据，或者转换为 blh 都能看得出）]





XYZ2BLH.txt

全部数据请详见电子文档“XYZ2BLH.txt”。

四、地心地固系与站心坐标系转换

1、算法原理及流程

1) 算法原理

以测站 P 点为原点，P 点法线方向为 z^* 轴（指向天顶为正），子午线方向为 x^* 轴， y^* 轴与 x^* 轴、 z^* 轴垂直，构成左手坐标系。这种坐标系就称为法线站心直角坐标系，或称为站心椭球坐标系。

若设 P 点的大地经纬度为 (L, B)，则可推导出法线站心直角坐标系与相应的地心直角坐标系之间的换算关系：

$$\begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix}_{PQ} = \begin{bmatrix} -\sin B \cos L & -\sin B \sin L & \cos B \\ -\sin L & \cos L & 0 \\ \cos B \cos L & \cos B \sin L & \sin B \end{bmatrix} \begin{bmatrix} X_Q - X_P \\ Y_Q - Y_P \\ Z_Q - Z_P \end{bmatrix}$$

其中，我们记

$$\begin{bmatrix} N \\ E \\ U \end{bmatrix}_Q = \begin{bmatrix} x^* \\ y^* \\ z^* \end{bmatrix}_{PQ}$$

2) 算法流程

(1) C# WinForm 窗体设计

- ①使用 GroupBox、RadioButton、Label 控件提供选择“XYZ to NEU”功能;
- ②RichTextBox 提供关于转换前后坐标数据、坐标点计算结果的显示,在此,将其“ReadOnly”属性设置为“True”,避免了对窗体内文本的误操作;
- ③几个 Button 控件关联“Click”事件,完成打开坐标文件、计算数据、清除全部内容(含释放内存、空间的选定状态)、保存坐标文件等功能;

(2) 数据载入

①在 Point.cs 内定义了“XYZPoint”“BLHPoint”“NEUPoint”三个类,并用属性的{get; set;}来声明了关于每个点的“name”“X”“Y”“Z”数据类型为 string 和 double;判断数据是否载入的 bool 类型变量 j 值赋为“True”。

②在 Formula.cs 内定义了 WGS84 椭球的 a、e、f、b 等参数,以及为计算 B 而设置的微小量 epsilon=0.0000000000000001、pi=3.14159265358979323846;使用 OpenFileDialog,StreamReader,Split 方法读入 txt,实现整行数据分割读入,用 list<XYZPoint>类型的实例化对象 XPoint 列表,分别存储点名、X、Y、Z 坐标,同时在相应 RichTextBox 显示初始坐标数据。

③定义了以 out List<XYZPoint> Xpoint 为形参的 Readfile 函数,对于 new 的 OpenFileDialog 的对象“file”,限定“file.InitialDirectory”为“StartupPath”,“file.RestoreDirectory”为“True”,以及文件后缀类型“fileFilter”为“All Files (*.*)|*. *|Dat Files (*.dat)|*.dat|Text Files (*.txt)|*.txt”,优化了寻找坐标文件的便捷性;

④在 Form.cs 里面的关于“OpenFile”的 Button_Click() 事件响应函数中,实例化对象 XPoint 后,直接调用 formula.Readfile(out Xpoint);所有数据点的坐标信息都存在 XPoint 列表中,便于 Count 和索引。

(3) 数据的计算与显示

①“Compute”Button 与 Button_Click() 事件响应函数相关联,将 bool 类型的 jj 与 j1 作为是否执行 Compute 指令的条件,只有在 RadioButton 选定以

及坐标文件打开数据成功载入之后才能执行 Compute，避免了报错。

②在 Formula.cs 中，定义了 xyz2blh(List<XYZPoint> Xpoint, out List<BLHPoint> Bpoint)函数，并且在 foreach 的每一次循环内，用迭代的方式（用到了 epsilon）解出 B，与解算出的 L、H 一起，带出 Bpoint 列表。[结合算法原理中 B 的迭代求解]；

③同时还定义了 blh2neu(List<XYZPoint> Xpoint, List<BLHPoint> Bpoint, int num, out List<NEUPoint> Npoint)函数，将 XYZ 坐标经过第[num]点转换为 NEU 站心坐标，num 为站心坐标点号，注意到需要用 Bpoint 作为形参，这与“算法原理”中的变换矩阵含站心点的 (B, L, H) 坐标是一致的，必须求得站心点的 BLH 大地坐标才能求得变换矩阵，继而完成 XYZ 至 NEU 的坐标系转换。

④定义 int 类型变量 num=1，意为第 1 点为站心的站心标号，可以改动 num 来完成改变站心点的功能；使用 new 方法，实例化 List<BLHPoint>类的 BPoint，然后再调用 formula.xyz2blh(Xpoint, out Bpoint)函数完成对 Bpoint 数据的存储，此时将 XYZ 转为 BLH 坐标；继而调用 formula.blh2neu(Xpoint, Bpoint, num, out Npoint)函数，传出 Npoint 列表即将其实例化，储存计算完的 NEU 坐标数据。

⑤string.Format()方法控制输出空格与对齐方式，再结合 string 类型的格式控制，foreach 遍历每一个坐标点，将处理结果保存至 report1, report2, report3 中。Report1 是 XYZ 原始坐标，Report2 是转化的 BLH 大地坐标，Report3 是转化的 NEU 站心坐标。后在两个 RichTextBox 中分别显示计算前后数据，即将 report1 与 report3 显示。

（4）数据的清除

写这个清除函数并不容易，要充分考虑到预先定义的 XPoint、BPoint、NPoint 是否实例化的问题，在此使用了 if 条件的嵌套，利用了 RadioButton 的状态、jj 与 j1 的值来进行判断因而得以解决。

方法为清除 RadioButton 的选定状态，清除 RichTextBox 内容，将 bool 类型的 jj 与 j1 重置为 false。

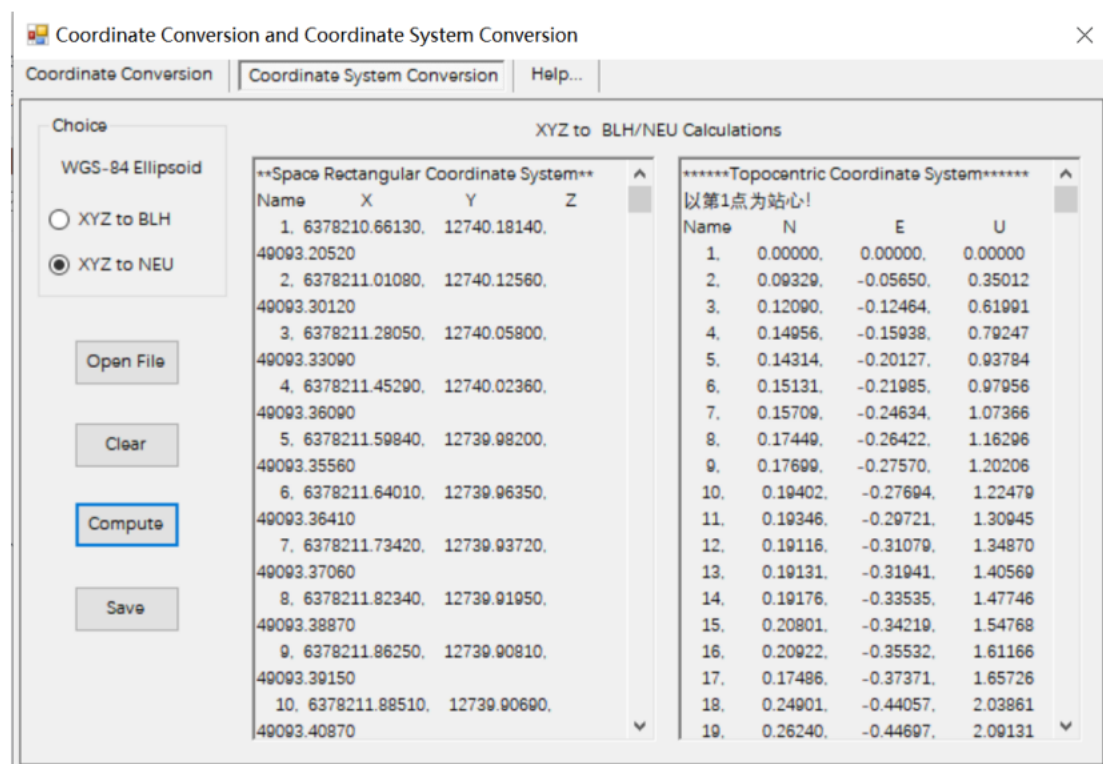
(5) 数据计算结果保存

① 点击“Save”Button, 采用 (radioButton3.Checked == true && jj == true) 且 j1 为 false 时作为是否执行结果保存命令的条件。

② 使用 SaveFileDialog, StreamWriter, 方法将两个转换前后的 RichTextBox 内容写入 txt 文件文本, 同时对于 new 的 SaveFileDialog 的对象 “file”, 限定 “file.InitialDirectory” 为 “StartupPath”, “file.RestoreDirectory” 为 “True”, 便于保存至当前根目录, 以及文件后缀类型 “fileFilter” 设定为 “All Files (*.*)|*.txt|Dat Files (*.dat)|*.dat|Text Files (*.txt)|*.txt”, 通过赋值 “file.FilterIndex = 3”, 默认文件以 “.txt” 文本文件类型存储, 优化了存储坐标文件的便捷性。

2、关键中间结果及最终结果

1) 程序调试结果



2) 站心坐标系结果（前 10 个点）

*****Topocentric Coordinate System*****

以第 1 点为站心！

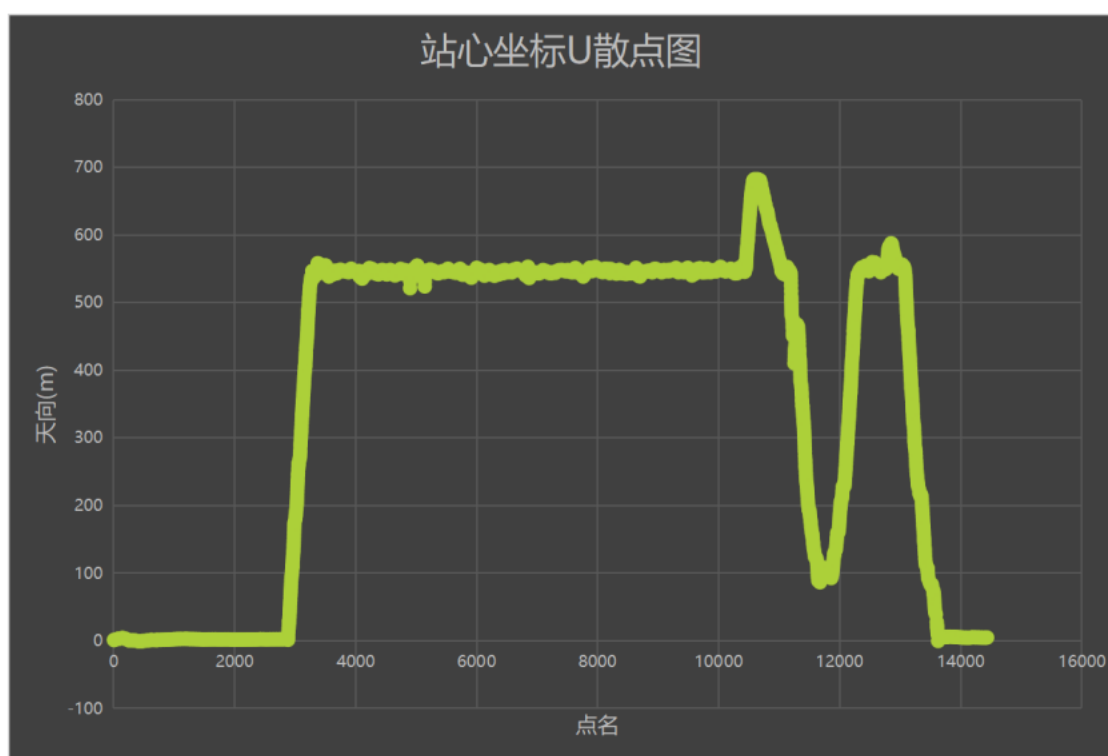
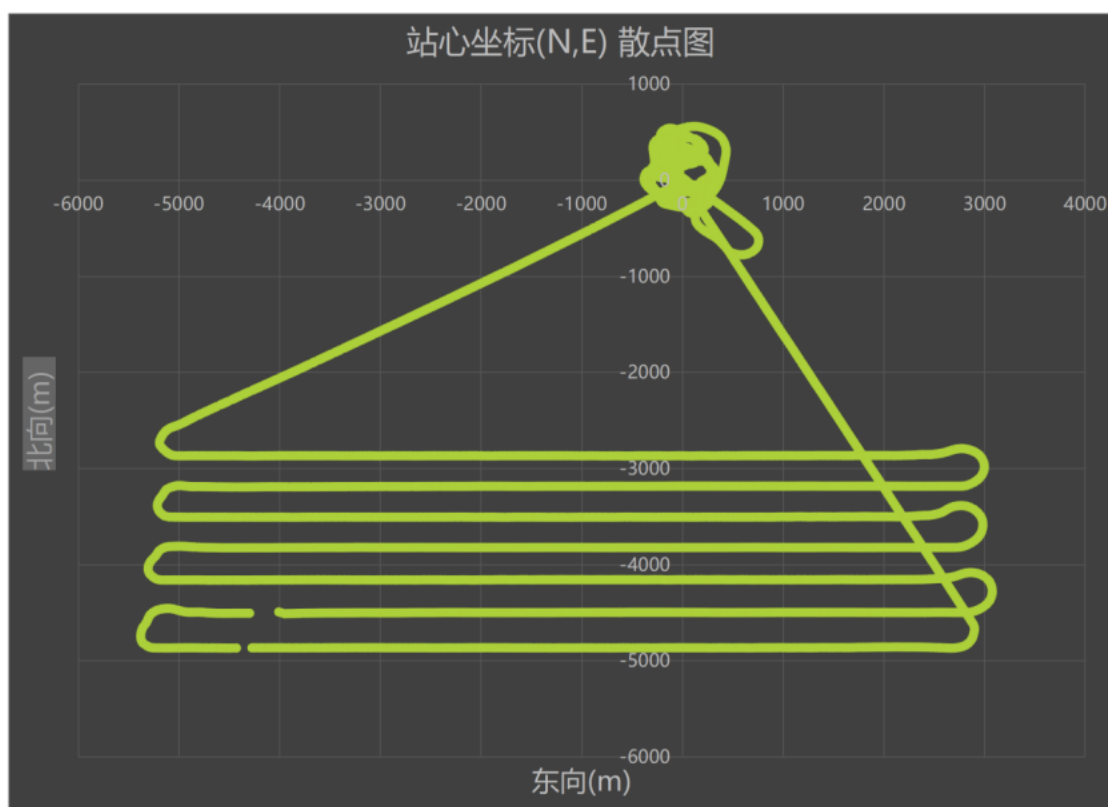
Name	N	E	U
1,	0.00000,	0.00000,	0.00000
2,	0.09329,	-0.05650,	0.35012
3,	0.12090,	-0.12464,	0.61991
4,	0.14956,	-0.15938,	0.79247
5,	0.14314,	-0.20127,	0.93784
6,	0.15131,	-0.21985,	0.97956
7,	0.15709,	-0.24634,	1.07366
8,	0.17449,	-0.26422,	1.16296
9,	0.17699,	-0.27570,	1.20206
10,	0.19402,	-0.27694,	1.22479


3) 站心坐标系结果（全部 14435 个点）

应用场景猜想：

像无人机摄影测量由一个位置附近起飞调整姿态环绕升空，后平行往返扫描，再绕回降落。

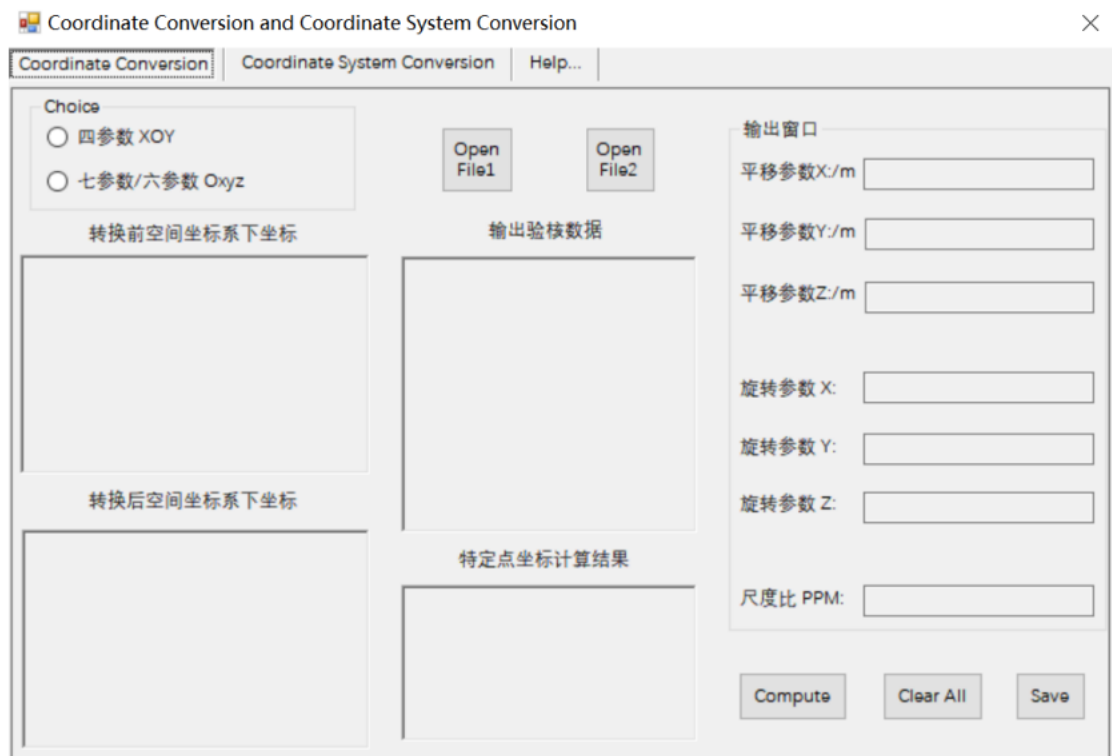
[辅助论据：WGS84 的长半轴 $a=6378137$ ，数据“XYZ2BLHNEU.txt”的 x 坐标范围为 63782**，63787**，也就是说数据点不仅在地球表面，有一定海拔，而且集中在赤道附近（从原始 xyz 数据，或者转换为 blh 都能看得出）]

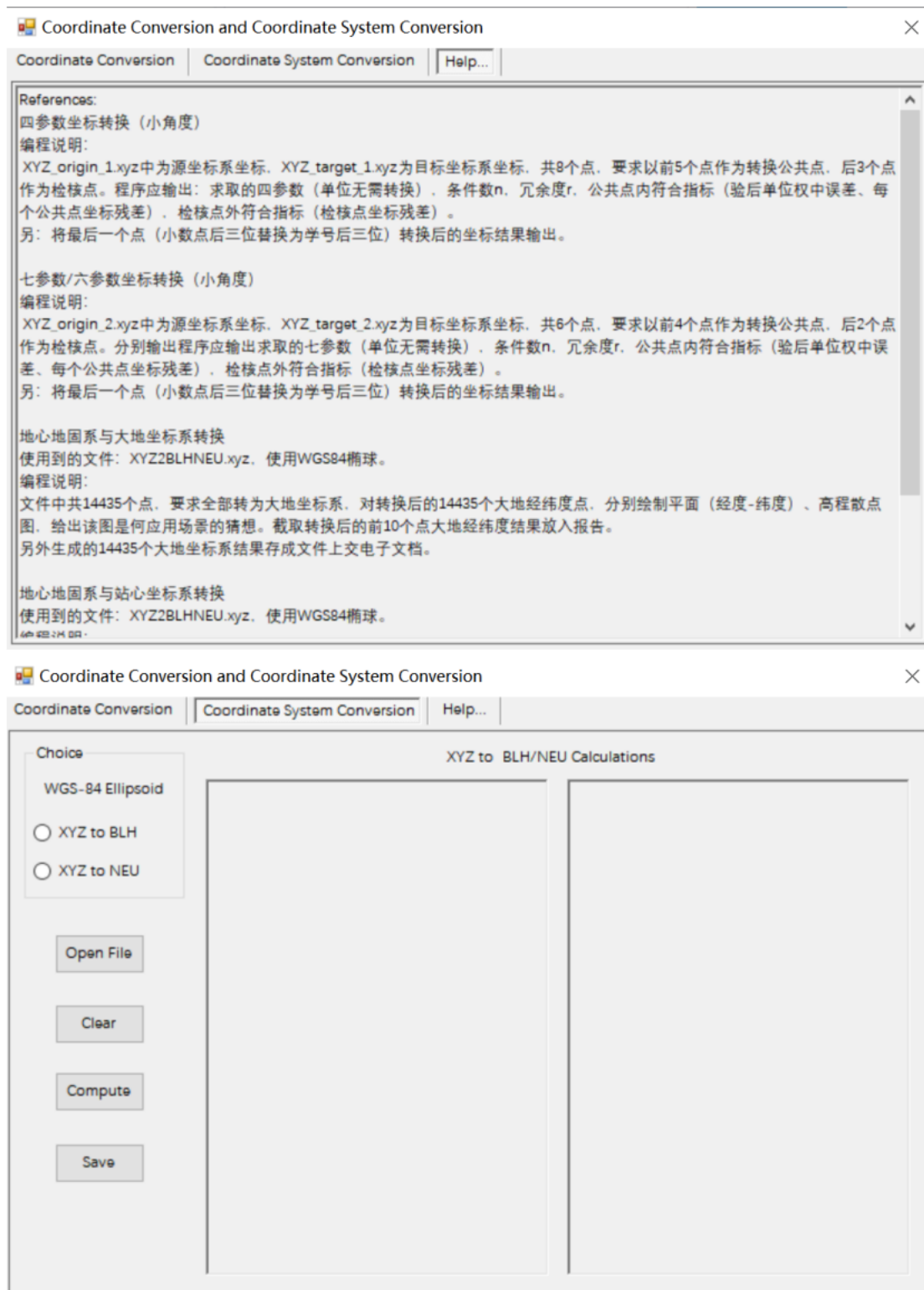


全部数据请详见电子文档“XYZ2NEU. txt”。  XYZ2NEU.txt

五、作业感想

既然选择使用 C#，就要优化窗体的交互性，消灭可能的交互性 bug，以及美观条理舒适的界面。为此更改了默认窗体的最大化最小化窗口，tabcontrol 改变了样式，改变字体与大小，合理安排控件间的相对位置等等。





回忆起大一下 C#网络程序设计编程。语法，变量声明，类的静态方法与非静态方法的实例化，自定义类的列表用法，文件的读取，Split 分割，事件的响应... 让我感到 C#模块化编程的魅力。我可以

用 `matrix.cs`/`Formula.cs`/`Point.cs`/`Form.cs` 四个 `cs` 文件构建窗体功能，完全的面向对象让我感受到了语言的逻辑与条理性，没有像 `C++` 一样的琐碎、冗杂。

大大小小分两周熬了 5 天半的夜，平均在两点半的夜里入睡，完全的自主写代码查阅资料，一些知识网络忘记了又重新找出《网络程序设计这本书》来，回忆一下李英冰老师的授课，收获颇多。没有求助学长姐的代价是连续熬了三个晚上写完了所有代码任务与窗体优化设计，一个中午一个下午完成图像可视化，两个晚上写报告。

我的成就感蛮大的，但窗体依然还有改进的地方，譬如 `UI` 可以更精美，代码可以更精简...

总的说来，通过此次编程，我发现了这两门重难点课程之间的联系与相同之处，将平差中学到的间接平差用在了本次编程程序的函数中。测绘的专业学科本质上就是相同的，尤其是在数据处理上，平差是核心，大地测量是方法和武器。这次编程作业让我更清晰的理解了这两门课的内在联系，让我更熟悉了流程、加深了对知识的理解与掌握。

谢谢老师与助教的辛勤付出，我会再接再厉的。

此致

敬礼！

学生：刘恒祯