
武汉大学本科生课程作业



二〇二〇年四月

目录

一、	程序背景.....	1
二、	算法原理及流程.....	1
	2.1 整体流程图.....	1
	2.2 读取省界文件程序原理.....	1
	2.3 利用省界文件寻找格网边界程序原理.....	2
	2.4 构建格网原理.....	3
	2.5 输出用于计算的格网点程序原理.....	3
	2.6 将格网点文件代入 EGM2008 中计算高程异常.....	4
	2.7 用 GAMIT 软件绘制高程异常图.....	5
三、	关键中间结果.....	5
	3.1 格线网边界及行列数.....	5
	3.2 高程异常结果.....	5
四、	最终结果.....	9
	4.1 高程异常图.....	9
五、	作业感想.....	9
	5.1 小组分工.....	9
	5.2 个人感想.....	10
六、	matlab 实验代码.....	10
	6.1 编程背景.....	10
	6.2 Matlab 代码.....	11

一、程序背景

高程异常是似大地水准面至椭球面的高度。计算高程异常的常用方法是用水准测量的方法联测 GPS 网中若干 GPS 点的正常高，然后根据 GPS 的大地高求出公共点的高程异常。但实际使用时不方便，费时费力，精度不高。如果使用 EGM2008 求得大地水准面差距之差能以较高精度将 GPS 测定大地高差转换为正常高差。求得大地高后可绘制该省高程异常图。

二、算法原理及流程

2.1 整体流程图



2.2 读取省界文件程序原理

利用 C++ 的文件流，创建两个数组，分别用来储存文件中的经度和纬度。

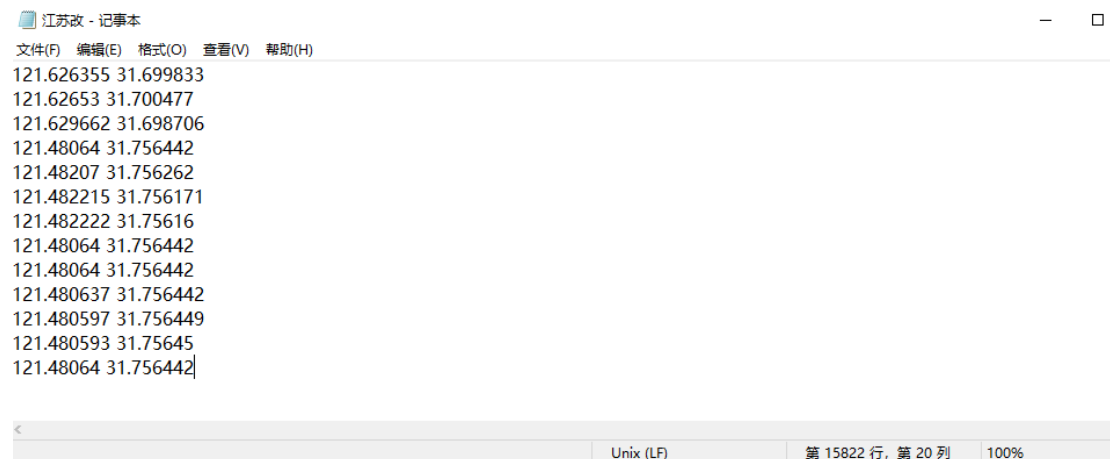


图 2.1 用于读取的省界文件

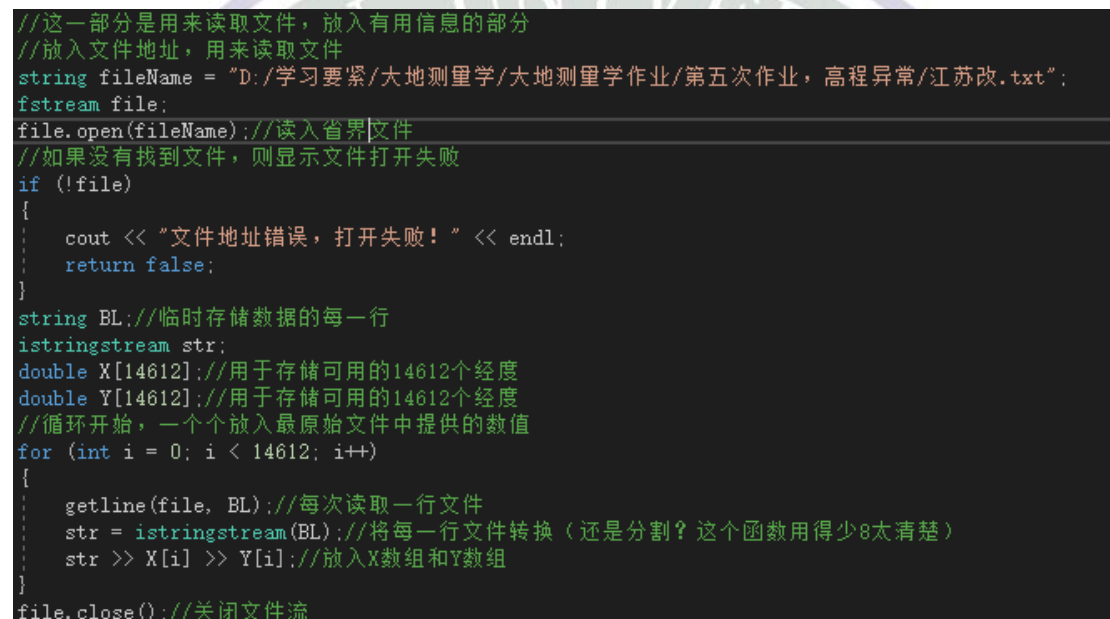


图 2.2 读取文件的相关代码

2.3 利用省界文件寻找格网边界程序原理

得到储存的省界文件后，需要找到能囊括所有边界点的最小矩形的四个边。

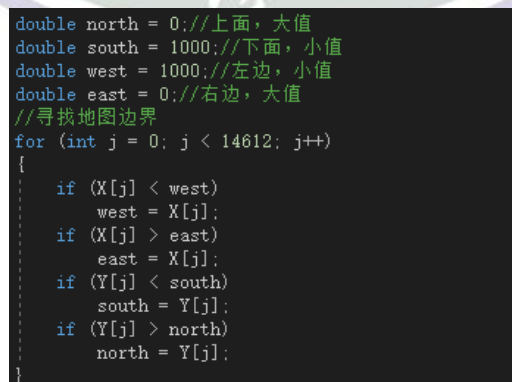


图 2.3 寻找格网边界相关代码

2.4 构建格网原理

因为要求以 0.5 度为一格，故取寻找到格网的最西边和最南边的数值，每次加上 0.5，得到格网究竟应该是一个几行几列的格网。

```
int Row = 0;
int Column = 0;
for(int i=0;i<1000;i++)//随便给定的一个比较大的值，因为行数列数一定小于这个数
{
    if (west < east)//如果西还没有东大，就加上 0.5,同时计数加上1
    {
        west = west + 0.5;
        Column = Column + 1;
    }
    if(south<north)//如果南还没有北大，就加上 0.5,同时计数加上1
    {
        south = south + 0.5;
        Row = Row + 1;
    }
}
```

图 2.4 构建格网相关代码

2.5 输出用于计算的格网点程序原理

因为 EGM2008 软件实际上是一个利用经度纬度计算改点高程异常的软件，想要得到格网点中每一点的高程异常，就需要输出格网点中每一点的经度纬度坐标用于计算。

```
double Result[130][2];
for (int i = 0; i < 130; i++)//循环130次，每一次放进一个点
{
    Result[i][0] = west + 0.5 * (i % 13);
    Result[i][1] = south + 0.5 * (int(i / 13));
}
//输出文件
ofstream file_result;//创建输出文件流
//创建输出文件地址
file_result.open("D:/学习要紧/大地测量学/大地测量学作业/第五次作业，高程异常/new_result.txt");//自行修改文件路径
//开始输出二维数组里的数值
for (int i = 0; i < 130; i++)
{
    file_result.flags(ios::fixed);
    file_result.precision(6);//设置保留小数位数
    file_result << Result[i][0] << "\t" << Result[i][1] << "\n";
}
//关闭输出文件流
file_result.close();
```

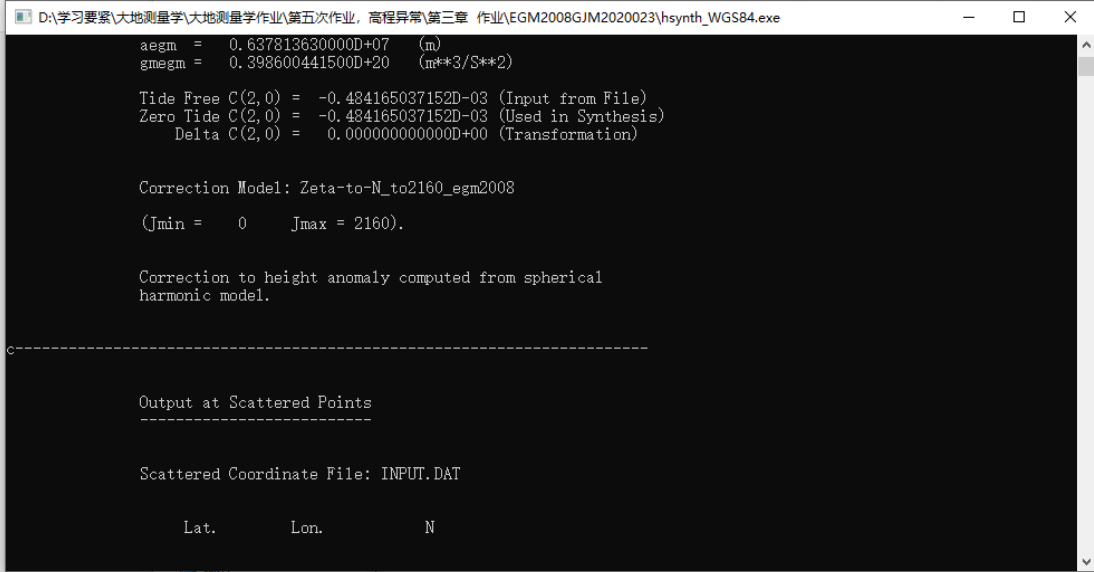
图 2.5 输出用于计算的格网点相关代码

因为这个时候已经确定了应该是有多少行多少列，也已经知道了最小的经度和纬度，就像一个坐标轴，已经知道了原点的位置和坐标轴尺度，就可以开始算其中

每一个点的坐标了。我创建用于储存结果的二维数组，这里的基础思想是用数字的序数(因为一共有130个数，每个数对应的序号是0到129),算得这个数的坐标，承接上面坐标轴的思想，先找出坐标，再来换算，这里序号走的顺序是由西到东，由南向北，而且是先由西到东，再由南向北，简单来说就是一行一行向上计数，寻找坐标的思想是取余数和取整数，比如说第二个点，第二个点的序数是1(0开始)，那么这个数的坐标就应该是(1, 0)，不难看出横坐标就是序数和13的余数，对应下面的式子就是 $(i \% 13)$ ，纵坐标就是序数和13的取整(抹掉后面的，不进行四舍五入)对应下面的式子就是 $\text{int}(i / 13)$ ，matlab这里多一个部分，是因为matlab从1开始，每到13会取余为0，但实际上横坐标应该是13，所以每到13需要额外计算，因为matlab是1到0(1 2 3 4 5 6 7 8 9 10 11 12 0)，C++却正好避开了这个问题，它是0到12，没有“转向”的问题。

2.6 将格网点文件代入EGM2008中计算高程异常

得到格网点中每个点的经度纬度之后直接带入EGM2008，按照教学视频中步骤计算即可。



```
D:\学习资源\大地测量学\大地测量学作业第五次作业_高程异常\第三章_作业\EGM2008\JM2020023\hsynth_WGS84.exe

aegm = 0.637813630000D+07 (m)
gmegm = 0.398600441500D+20 (m**3/S**2)

Tide Free C(2,0) = -0.484165037152D-03 (Input from File)
Zero Tide C(2,0) = -0.484165037152D-03 (Used in Synthesis)
Delta C(2,0) = 0.000000000000D+00 (Transformation)

Correction Model: Zeta-to-N_to2160_egm2008
(Jmin = 0 Jmax = 2160).

Correction to height anomaly computed from spherical
harmonic model.

-----
Output at Scattered Points
-----

Scattered Coordinate File: INPUT.DAT

Lat. Lon. N
```

图 2.6 EGM2008 运行完成界面

2.7 用 GAMIT 软件绘制高程异常图

已经得到格网点和其中相应的坐标,利用 GAMIT 即可绘制最终的高程异常图片。

三、关键中间结果

3.1 格线网边界及行列数

边界西经	边界东经	边界南纬	边界北纬	格网行数	格网列数
116.355183	122.355183	30.760280	35.260280	10	13

3.2 高程异常结果

经度	纬度	高程异常
116.3552	30.76028	23.691
116.8552	30.76028	24.66
117.3552	30.76028	25.466
117.8552	30.76028	26.313
118.3552	30.76028	27.127
118.8552	30.76028	27.888
119.3552	30.76028	28.487
119.8552	30.76028	28.908
120.3552	30.76028	29.301
120.8552	30.76028	30.061
121.3552	30.76028	31.208
121.8552	30.76028	32.039
122.3552	30.76028	32.542
116.3552	31.26028	23.689
116.8552	31.26028	24.635
117.3552	31.26028	25.526
117.8552	31.26028	26.426
118.3552	31.26028	27.264
118.8552	31.26028	28.054
119.3552	31.26028	28.658
119.8552	31.26028	29.071
120.3552	31.26028	29.472
120.8552	31.26028	30.291
121.3552	31.26028	31.32
121.8552	31.26028	32.262

122.3552	31.26028	32.88
116.3552	31.76028	23.601
116.8552	31.76028	24.566
117.3552	31.76028	25.358
117.8552	31.76028	26.417
118.3552	31.76028	27.341
118.8552	31.76028	28.197
119.3552	31.76028	28.857
119.8552	31.76028	29.34
120.3552	31.76028	29.757
120.8552	31.76028	30.508
121.3552	31.76028	31.62
121.8552	31.76028	32.374
122.3552	31.76028	32.97
116.3552	32.26028	23.577
116.8552	32.26028	24.54
117.3552	32.26028	25.305
117.8552	32.26028	26.23
118.3552	32.26028	27.386
118.8552	32.26028	28.337
119.3552	32.26028	29.082
119.8552	32.26028	29.624
120.3552	32.26028	30.163
120.8552	32.26028	30.933
121.3552	32.26028	31.897
121.8552	32.26028	32.656
122.3552	32.26028	33.086
116.3552	32.76028	23.923
116.8552	32.76028	24.704
117.3552	32.76028	25.445
117.8552	32.76028	26.399
118.3552	32.76028	27.411
118.8552	32.76028	28.501
119.3552	32.76028	29.387
119.8552	32.76028	29.952
120.3552	32.76028	30.582
120.8552	32.76028	31.395
121.3552	32.76028	32.263
121.8552	32.76028	32.909
122.3552	32.76028	33.379
116.3552	33.26028	24.191
116.8552	33.26028	25.098
117.3552	33.26028	25.883
117.8552	33.26028	26.759

118.3552	33.26028	27.636
118.8552	33.26028	28.748
119.3552	33.26028	29.688
119.8552	33.26028	30.303
120.3552	33.26028	31.03
120.8552	33.26028	31.831
121.3552	33.26028	32.61
121.8552	33.26028	33.219
122.3552	33.26028	33.722
116.3552	33.76028	24.673
116.8552	33.76028	25.486
117.3552	33.76028	26.367
117.8552	33.76028	27.114
118.3552	33.76028	27.872
118.8552	33.76028	28.909
119.3552	33.76028	30.008
119.8552	33.76028	30.699
120.3552	33.76028	31.444
120.8552	33.76028	32.207
121.3552	33.76028	32.856
121.8552	33.76028	33.514
122.3552	33.76028	34.085
116.3552	34.26028	25.317
116.8552	34.26028	26.133
117.3552	34.26028	26.856
117.8552	34.26028	27.484
118.3552	34.26028	28.186
118.8552	34.26028	29.149
119.3552	34.26028	30.234
119.8552	34.26028	31.124
120.3552	34.26028	31.814
120.8552	34.26028	32.503
121.3552	34.26028	33.151
121.8552	34.26028	33.763
122.3552	34.26028	34.396
116.3552	34.76028	25.846
116.8552	34.76028	26.598
117.3552	34.76028	27.37
117.8552	34.76028	27.907
118.3552	34.76028	28.514
118.8552	34.76028	29.602
119.3552	34.76028	30.576
119.8552	34.76028	31.381
120.3552	34.76028	32.027

120.8552	34.76028	32.787
121.3552	34.76028	33.374
121.8552	34.76028	34.006
122.3552	34.76028	34.668
116.3552	35.26028	26.313
116.8552	35.26028	26.969
117.3552	35.26028	27.724
117.8552	35.26028	28.344
118.3552	35.26028	29.016
118.8552	35.26028	29.914
119.3552	35.26028	30.821
119.8552	35.26028	31.655
120.3552	35.26028	32.252
120.8552	35.26028	32.877
121.3552	35.26028	33.554
121.8552	35.26028	34.202
122.3552	35.26028	34.927



四、最终结果

4.1 高程异常图

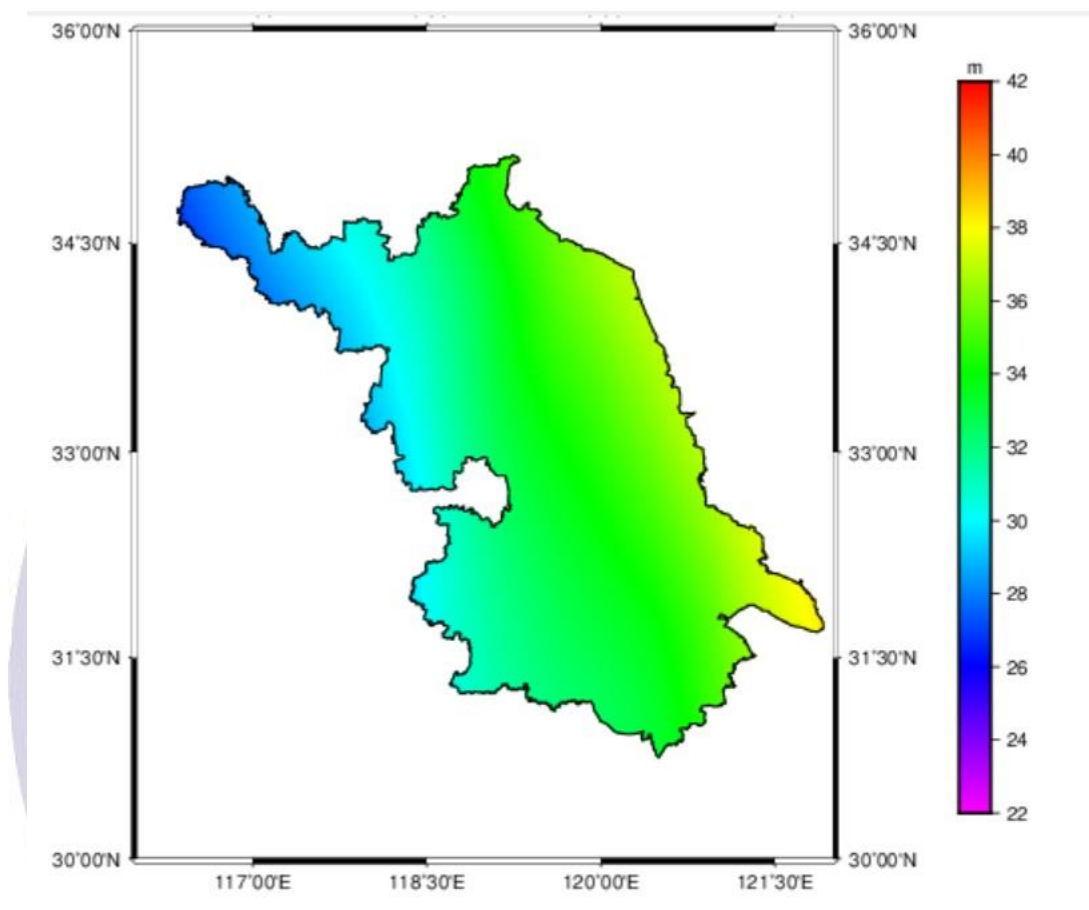


图 4.1 最终得出的高程异常图片

五、作业感想

5.1 小组分工

序号	姓名	任务
1	杨晓晨	C++代码编写
2	张雪晴	Gamit 软件成图
3	张洋	EGM2008 高程异常计算
4	胡俊东	MATLAB 代码编写

5.2 个人感想

本次作业是一次小组作业，我的主要任务是进行前期的一些实验性准备和主要完成代码中输出用于计算的格网点部分代码的编写。经历了本次作业，我对于 C++ 的使用更加熟练，能够更加从容地实现自己想要实现的功能。小组协作的联动也大大加快了完成的速度，在编程时大家可以相互分享思路，在分工时也因为人更多了，“单线程”变成了“多线程”，处理的效率明显增加。

六、matlab 实验代码

6.1 编程背景

在做程序的同时，我也用 matlab 实现了一遍前期整体数据的处理。因为 matlab 在对于相对规范的数据是较好处理，同时在运行的过程中容易跟进找到问题，所以在前期准备时我也同时写了一份 matlab 的代码进行试验。虽然 matlab 和 C++ 的默认起始点不同，matlab 默认从 1 开始，更符合人的逻辑，而 C++ 默认从 0 开始，有时会规避掉不少问题，两个各有千秋。

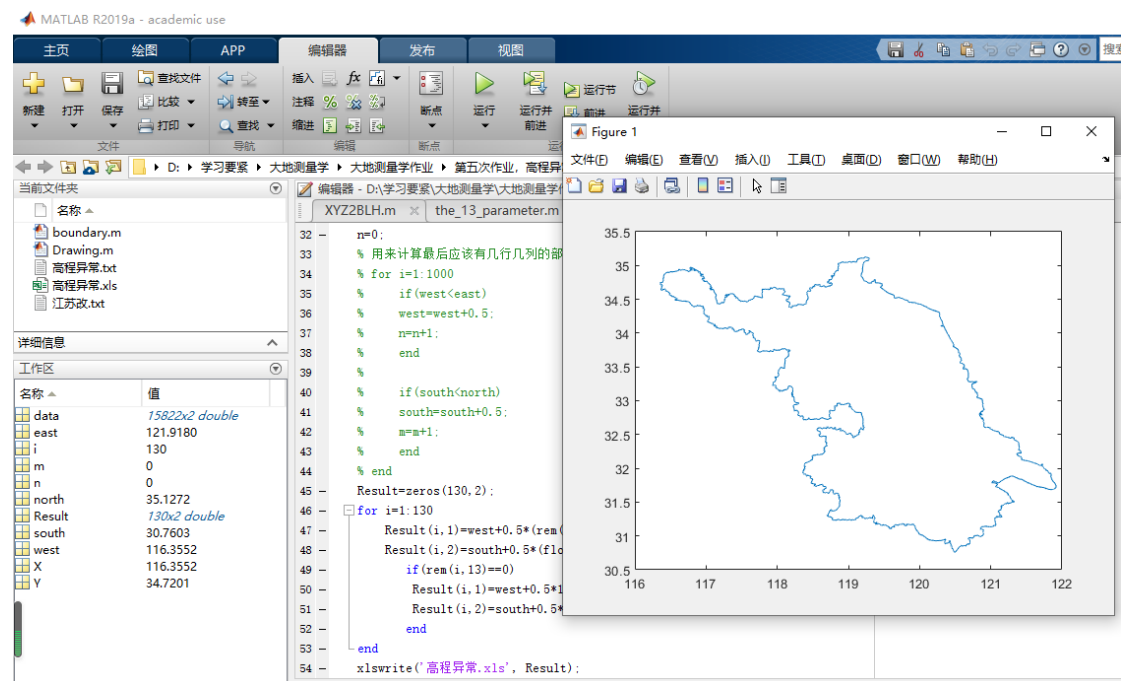


图 6.1 matlab 运行结果截图

6.2 Matlab 代码

1. %该程序是大地测量学高程异常作业
2. data=importdata('江苏改.txt');
3. % 实际的可用点只有 14612 个
4. i=1:14612;
5. X=data(i,1);
6. Y=data(i,2);
7. % 省界的画图
8. plot(X ,Y);
9. north=0;%上面，大值
10. south=1000;%下面，小值
11. west=1000;%左边，小值
12. east=0;%右边，大值
13. %寻找地图的边界应该在哪里
14. for i=1:14612
15. X=data(i,1);
16. Y=data(i,2);
17. if(X<west)
18. west=X;
19. end
20. if(X>east)
21. east=X;
22. end
23. if(Y<south)


```

24     south=Y;
25 end
26 if(Y>north)
27     north=Y;
28 end
29 end
30 %构建包括整个江苏的矩阵,m 是行数, n 是列数, 间隔都是 0.5
31 m=0;
32 n=0;
33 % 用来计算最后应该有几行几列的部分, 用完就可以注释掉
34 % for i=1:1000
35 %     if(west<east)
36 %         west=west+0.5;
37 %         n=n+1;
38 %     end
39 %
40 %     if(south<north)
41 %         south=south+0.5;
42 %         m=m+1;
43 %     end
44 % end
45 Result=zeros(130,2);
46 for i=1:130
47     Result(i,1)=west+0.5*(rem(i,13)-1);
48     Result(i,2)=south+0.5*(floor(i/13));
49     if(rem(i,13)==0)
50         Result(i,1)=west+0.5*12;
51         Result(i,2)=south+0.5*(floor(i/13)-1);
52     end
53 end
54 xlswrite('高程异常.xls', Result);

```