

# PYTHON与空间信息处理

## Geo-Processing with Python

宋现锋，博士  
中国科学院研究生院  
2009. 09 .17

# 1 Python语言

- 1.1 历史 (源于WIKI)
  - **Guido van Rossum**  
1960-01-31, Netherlands  
University of Amsterdam  
Google Computer programmer  
Award for the Advancement of Free Software (2001)  
More, [www.python.org/~guido/](http://www.python.org/~guido/)
  - **Language Development**  
1989 -  
Monty Python's Flying Circus  
[Summary Table](#)



## History and License

Release	Year	Owner	GPL
0.9 - 1.2	1991-1995	CWI (Stichting Mathematisch Centrum)	yes
1.3 - 1.6	1995-2000	CNRI (Corporation for National Research Initiatives)	yes
2.0	2000	BeOpen.com	no
2.1	2001	PSF (Python Software Foundation)	no
2.0.1	2001	PSF	yes
2.2	2001	PSF	yes
2.3	2002-2003	PSF	yes
2.4	2004-2005	PSF	yes
2.5	2006	PSF	yes
2.6	2008	PSF	yes



- 1.2 Python特点
  - 语言定位  
脚本语言 (script language) / 胶水语言 (glue language)
  - 兼容性  
windows, mac, linux/unix, palm os, win-ce  
Cython(C/C++), Jython(Java), IronPython(.net)
  - 扩充性  
标准库 (standard library)  
SWIG (simplified wrapper & interface generator)  
(another interface definition language)

- 1.3 GIS Programming with Python?

- GIS开发语言选择原则

- 基础专业工具包 (GIS)

- 程序运行速度 (用户)

- 系统集成方案 (开发者)

- Python与GIS

- [Open Source Geo-Processing with Python](#)

- [What is inside ARCGIS ?](#)

# Open Source Geo-Processing with Python

- **osgeo**      **gdal/ogr**      #栅格矢量基础格式读写
- **pgdb**      **postgres/postgis**      #空间数据库方式读写
- **cx\_Oracle**      **oracle/oracle\_spatial**
- ---
- **pyproj**      **proj4**      #地图投影
- ---
- **shapely**      **ogc simple feature**      #空间操作如: **buffer**
- **networkx**      **boost graph library**      #图论拓扑
- ---
- **numpy**      **matrix**      #数据矩阵分析 [数组]
- **SciPy**      **scientific tools for python**      #数学计算包
- ---
- **PyQt4**      **Qt**      #GUI
- **qgis**      **QGIS**      #GRASS GIS Desktop
- **matplotlib**      **2d/3d graphic**      #可视化
- **Py/VTK**      **VTK**

# What is inside ARCGIS ?

- Python [2006]
  - AML -> Avenue -> VB -> Python
- Numpy
  - Geoprocessing - (栅格数据) 矩阵分析
- GDAL/OGR
  - 空间数据格式读写

# 2 PYTHON GIS软件工具包

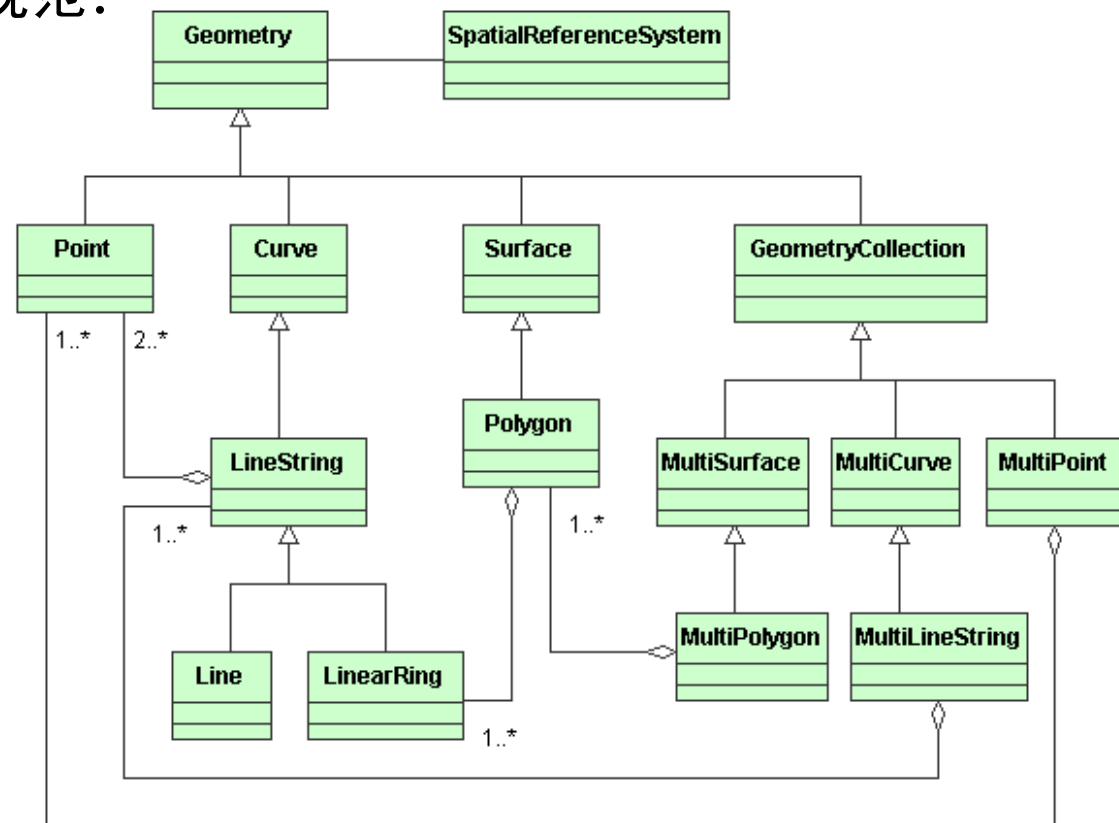
## 2.1 GDAL/OGR

(1) OGC Simple Feature规范:

点、线、多边形定义

WKT/WKB表达方式

空间操作规则



## (2) OGR实例代码 – 读取点文件(X,Y)坐标

```
#!C:\Program Files\pythonxy\python\python.exe
#-*- coding:gb2312 -*-
import os, ogr

class ARCVIEW_SHAPE:
    #inititalize
    def __init__(self):
    #read shape file
    def read_shp(self,file):
    #write shape file
    def write_shp(self,file,tuple):

#main function
if __name__ == "__main__":
    test = ARCVIEW_SHAPE()
    xy_tuple = test.read_shp('Borehole.shp')
    test.write_shp('Borehole_xy.shp', xy_tuple)
```

### (3) OGR实例结果

**View1**

- Ctour7.shp
- Borehole\_xy.shp
- Borehole.shp

**Attributes of Borehole.shp**

Shape	Id	Label	Z
Point	1106	14-11	-448.5
Point	1130	14-15-5	-433.7
Point	1151	14-15-7	-555.7
Point	1154	14-15-11	-394.9
Point	1158	补18-4	-425.6
Point	1165	补18-7	-508.6

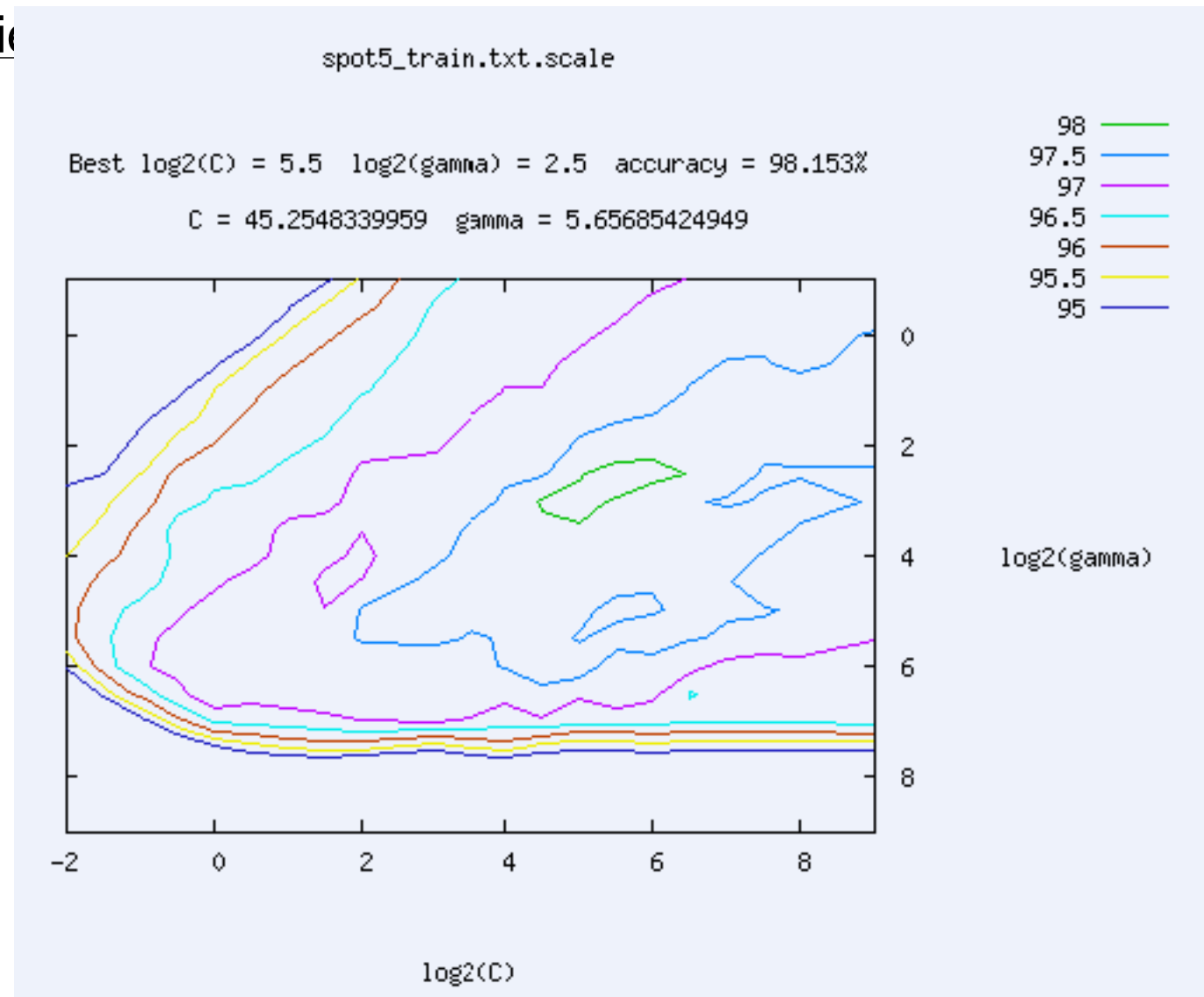
**Attributes of Borehole\_xy.shp**

Shape	Id	Label	Z	X	Y
Point	1101	13-14-12	-553.3	-26364422.76	-29645947.53
Point	1106	14-11	-448.5	-26364159.65	-29645432.93
Point	1130	14-15-5	-433.7	-26364710.01	-29645363.58
Point	1151	14-15-7	-555.7	-26365400.10	-29646001.95
Point	1154	14-15-11	-394.9	-26364162.78	-29644577.22
Point	1158	补18-4	-425.6	-26364675.55	-29644766.48
Point	1165	补18-7	-508.6	-26365795.56	-29645773.88
Point	1171	15-7	-496.6	-26365487.56	-29644863.86
Point	1174	15-12	-549.1	-26365592.53	-29644609.75

## (4)利用LIBSVM工具进行遥感图像分类 - GDAL

<http://www.csic>

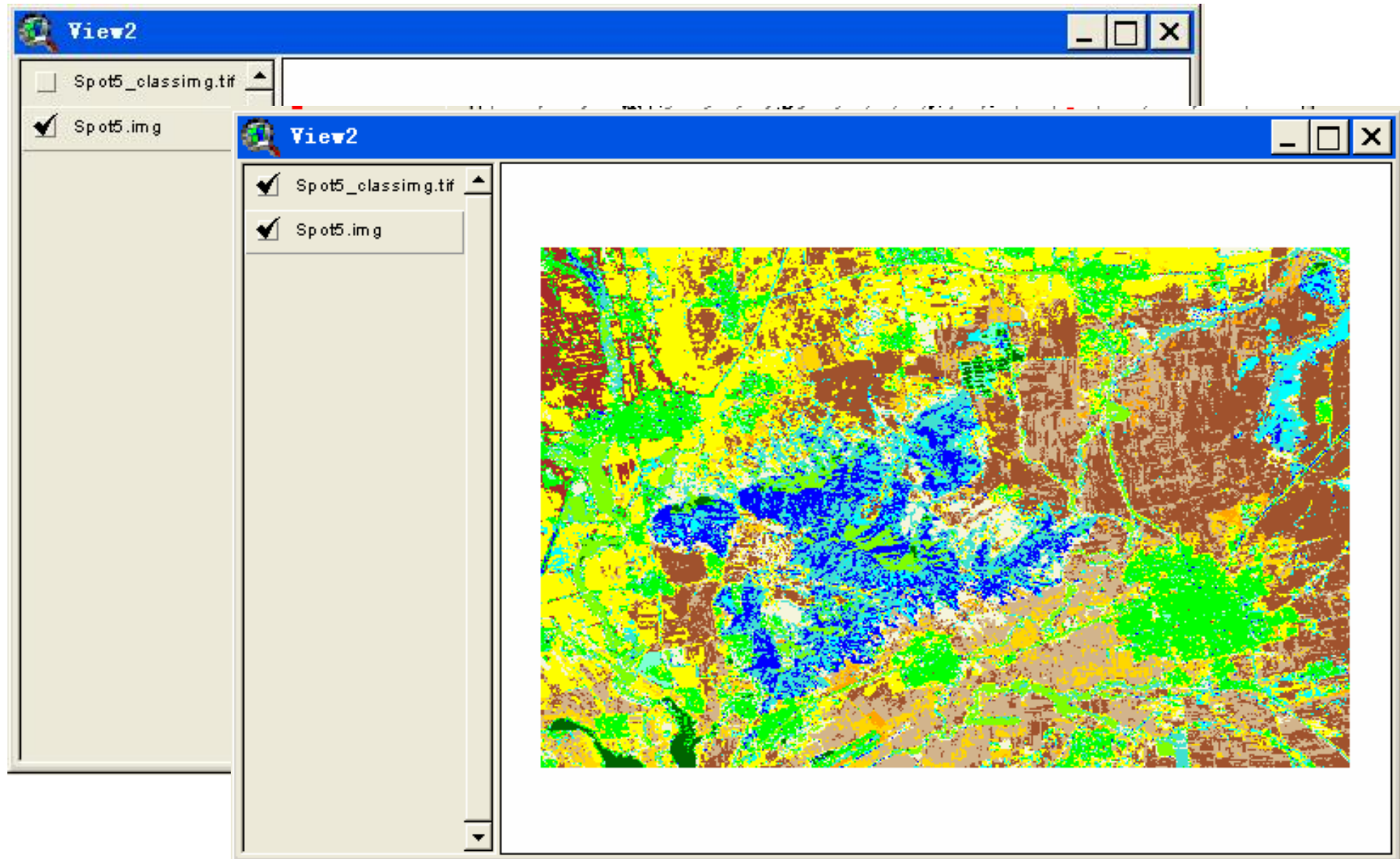
- sample.py
- dump.py
- [easy.py](#)
- [grid.py](#)
- restore.py



#### (4) GDAL实例代码 – 由ASCII数据创建TIFF (索引颜色型TIFF文件)

```
#COLOR TABLE
ct = gdal.ColorTable()
ctmap = {1:(255,0,0,255),2:(0,255,0,255), ... ...}
for key in ctmap.keys():
    ct.SetColorEntry( key, ctmap[key])
#OPEN
driver = gdal.GetDriverByName("GTiff" )
tiff = driver.Create('my.tif', 100, 150, 1, gdal.GDT_Byte)
tiff.GetRasterBand(1).SetRasterColorTable( ct )
tiff.SetGeoTransform((506804.5, 10.0, 0.0, 4537178.5, 0.0, -10.0) )
#WRITE
fh = open('my.txt','r')
for j in range(100):
    vals = [int(fh.readline()) for i in range(150)]
    buff = struct.pack('150B',vals)
    tiff.GetRasterBand(1).WriteRaster(0,j,150,1,buff,150,1,gdal.GDT_Byte)
fh.close()
#CLOSE
tiff.GetRasterBand(1).FlushCache()
tiff.Destroy()
```

## (5) GDAL实例结果



## 2.2 数据库PostgreSQL/PostGIS

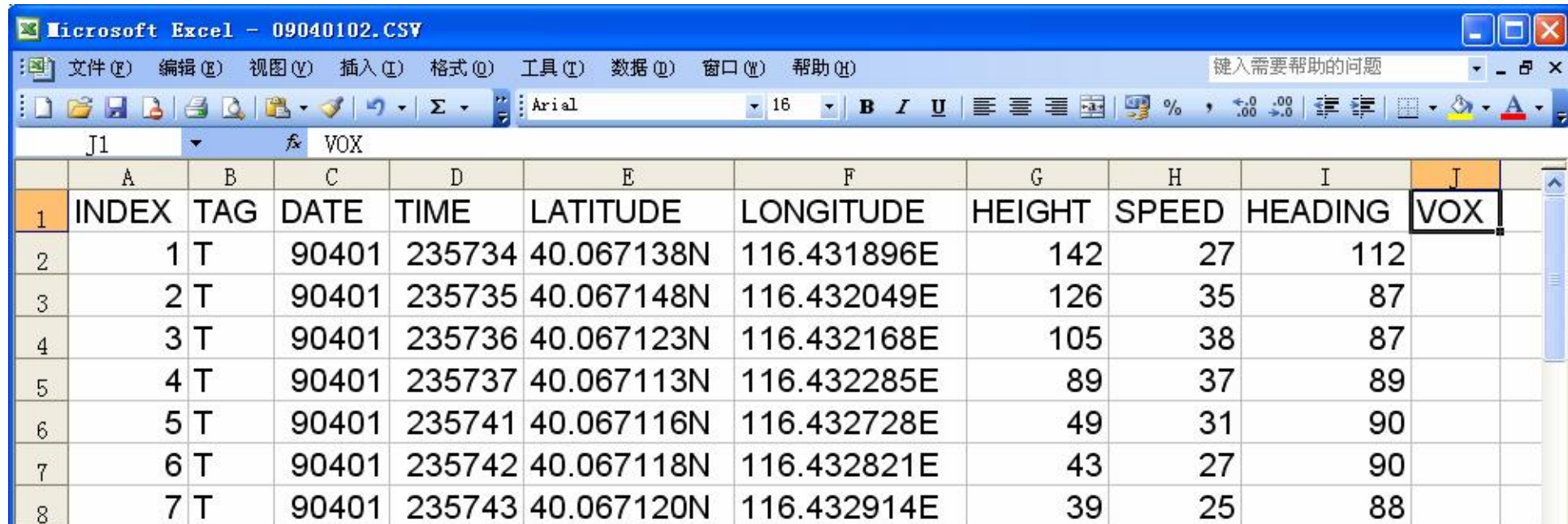


- Berkeley, 对象关系型数据库
- PostGIS空间数据引擎
- PgRoute图论拓扑实现
- PostgreSQL vs. MySQL

## #GPS航迹数据PostgreSQL存储

```
CREATE TABLE trace (  
    time    timestamp without time zone,  
    latitude float,  
    longitude float,  
    height  float,  
    speed   float,  
    heading float  
);
```

#GPS数据表



	A	B	C	D	E	F	G	H	I	J
1	INDEX	TAG	DATE	TIME	LATITUDE	LONGITUDE	HEIGHT	SPEED	HEADING	VOX
2	1	T	90401	235734	40.067138N	116.431896E	142	27	112	
3	2	T	90401	235735	40.067148N	116.432049E	126	35	87	
4	3	T	90401	235736	40.067123N	116.432168E	105	38	87	
5	4	T	90401	235737	40.067113N	116.432285E	89	37	89	
6	5	T	90401	235741	40.067116N	116.432728E	49	31	90	
7	6	T	90401	235742	40.067118N	116.432821E	43	27	90	
8	7	T	90401	235743	40.067120N	116.432914E	39	25	88	

**#PYTHON CODE**

**import pgdb**

**conn = pgdb.connect(host="localhost:5432",database='gps',user='me')**

**cur = conn.cursor()**

**cols = [time, latitude, longitude, height,speed,heading]**

**vals = ['2009/04/01 23:57:34', '40.067138','116.431896','142','27','112']**

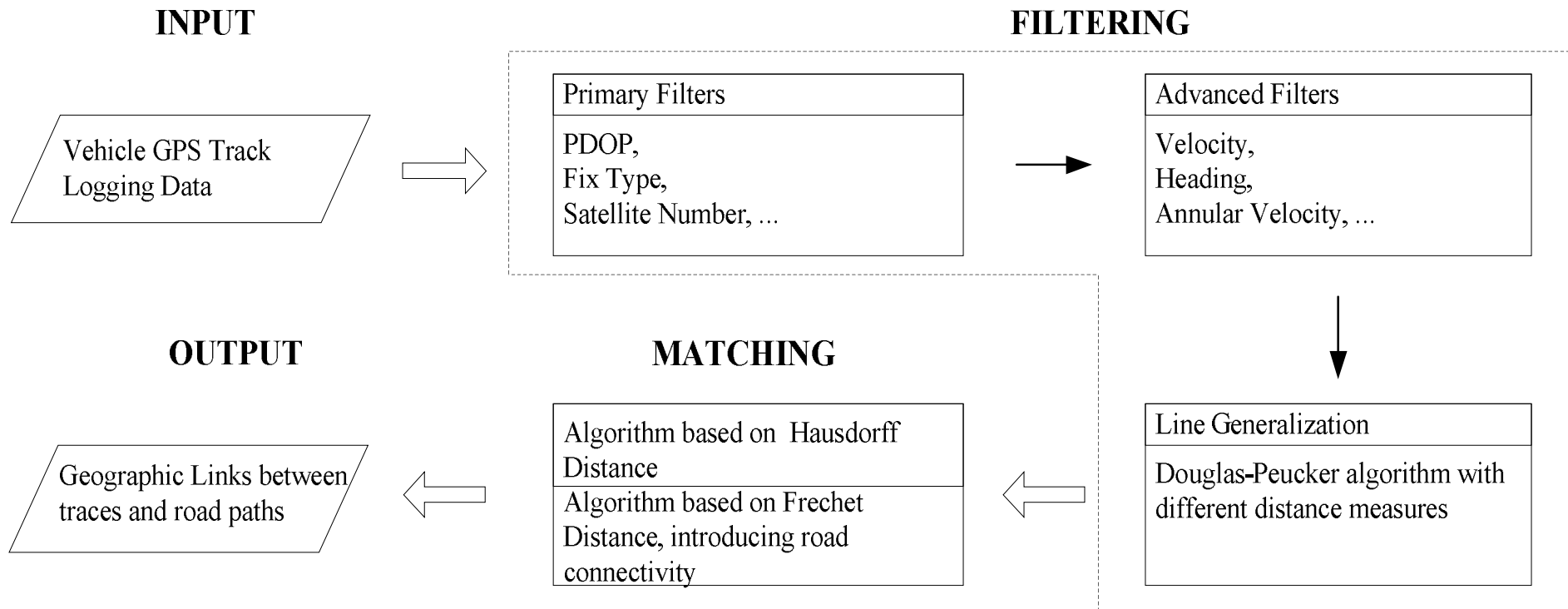
**sql = 'insert into trace (%s) values (%s)' % (','.join(cols), ','.join(vals))**

**cur.execute(sql)**

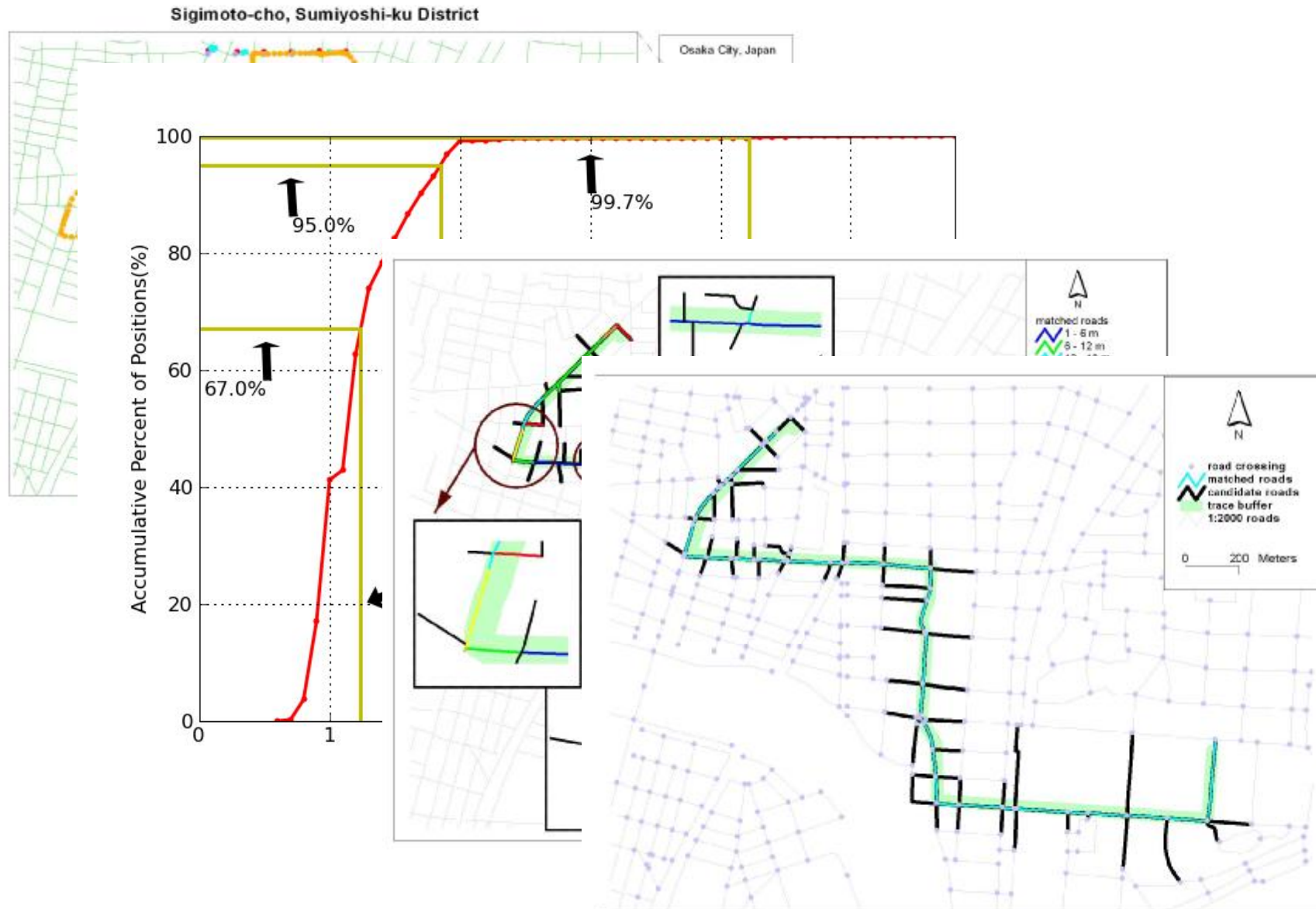
**cur.close()**

**conn.close()**

# #GPS航迹配准 - networkx



# #GPS航迹配准 - networkx



## 2.3 地图投影PROJ4

### #EPSG CODE理解

**<2204> +proj=lcc**

**+lat\_1=35.25 +lat\_2=36.416666666666666**

**+lat\_0=34.666666666666666 +lon\_0=-86**

**+x\_0=609601.2192024384 +y\_0=30480.06096012192**

**+ellps=clrk66 +datum=NAD27**

**+to\_meter=0.3048006096012192 +no\_defs**

## #PYTHON CODE

```
from pyproj import Proj
#parameters
params = {'proj':'merc', 'a':6378137, 'b':6378137,
          'lat_ts':0, 'lon_0':0, 'x_0':0, 'y_0':0, 'k':1.0, 'units':'m'}

#computation
proj900913 = Proj(params)
print proj900913(117,40)

#####
#<900913> +proj=merc +a=6378137 +b=6378137
#      +lat_ts=0.0 +lon_0=0.0
#      +x_0=0.0 +y_0=0
#      +k=1.0 +units=m
#      +nadgrids=@null +no_defs
```

## #结果

```
D:\GISB_20090908>python proj.py
```

```
proj4 library version = 4.6
```

```
+a=6378137 +b=6378137 +k=1.0 +lat_ts=0 +nadgrids=@null
```

```
+lon_0=0 +proj=merc +x_0=0 +units=m +y_0=0
```

```
(117,40)
```

```
(13024380.422813008, 4865942.279503176)
```

## 2.4 数学工具应用 (scipy, cvxopt)

### #混合像元分解

### (二次规规划分解方法-quadratic program)

线性模型基于如下假设：混合像元在某一波段的反射率是各端元组分在该波段反射率的线性组合，其中，权重系数就是各端元组分所占像元面积的百分数，表示为：

$$G_i = \sum_{j=1}^n (r_{ij} F_j) + \epsilon_i \quad (7)$$

式中： $\sum_{j=1}^n F_j = 1, 0 \leq F_j \leq 1$ ； $n$  为端元组分的数目； $G_i$  为混合像元在第  $i$  波段的反射值； $r_{ij}$  为混合像元内第  $j$  端元组分在第  $i$  波段的反射值； $F_j$  为混合像元内第  $j$  端元组分的丰度； $\epsilon_i$  为第  $i$  光谱波段的残差。

#PYTHON CODE – 最优化工具包python cvxopt / scipy

```
from cvxopt import matrix
```

```
from cvxopt import solvers
```

```
#
```

```
Q = 2*matrix([ [2, .5], [0.5, 1] ])
```

```
p = matrix([1.0, 1.0])
```

```
G = matrix([[[-1.0,0.0],[0.0,-1.0]])
```

```
h = matrix([0.0,0.0])
```

```
A = matrix([1.0, 1.0], (1,2))
```

```
b = matrix(1.0)
```

```
sol=solvers.qp(Q, p, G, h, A, b)
```

```
#
```

```
print sol['x']
```

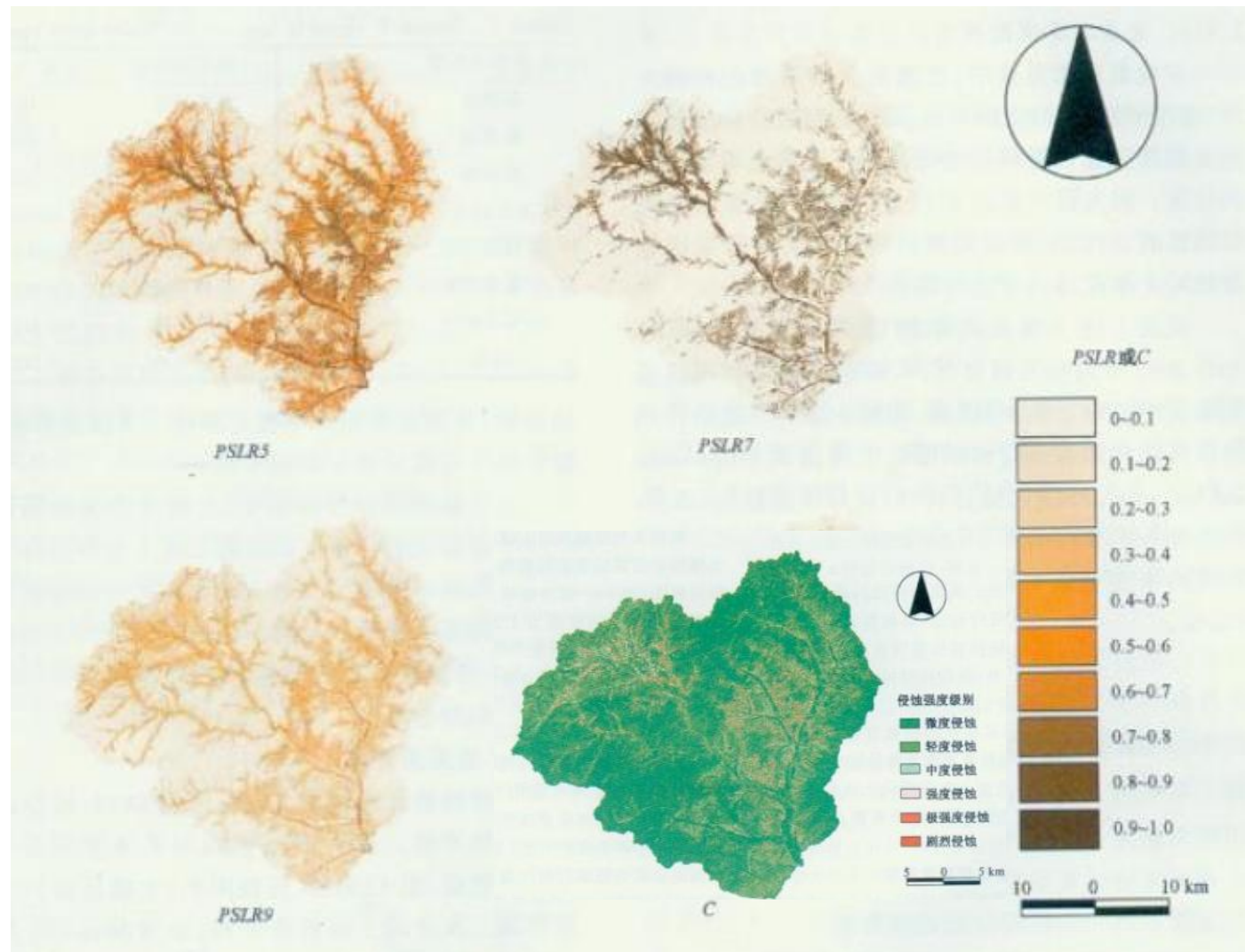
minimize  $2x_1^2 + x_2^2 + x_1x_2 + x_1 + x_2$

subject to  $x_1 \geq 0$

$x_2 \geq 0$

$x_1 + x_2 = 1$

## #土壤侵蚀模型中植被管理因子C值的遥感估算



## 2.5 Quantum GIS (QGIS, Qt包)

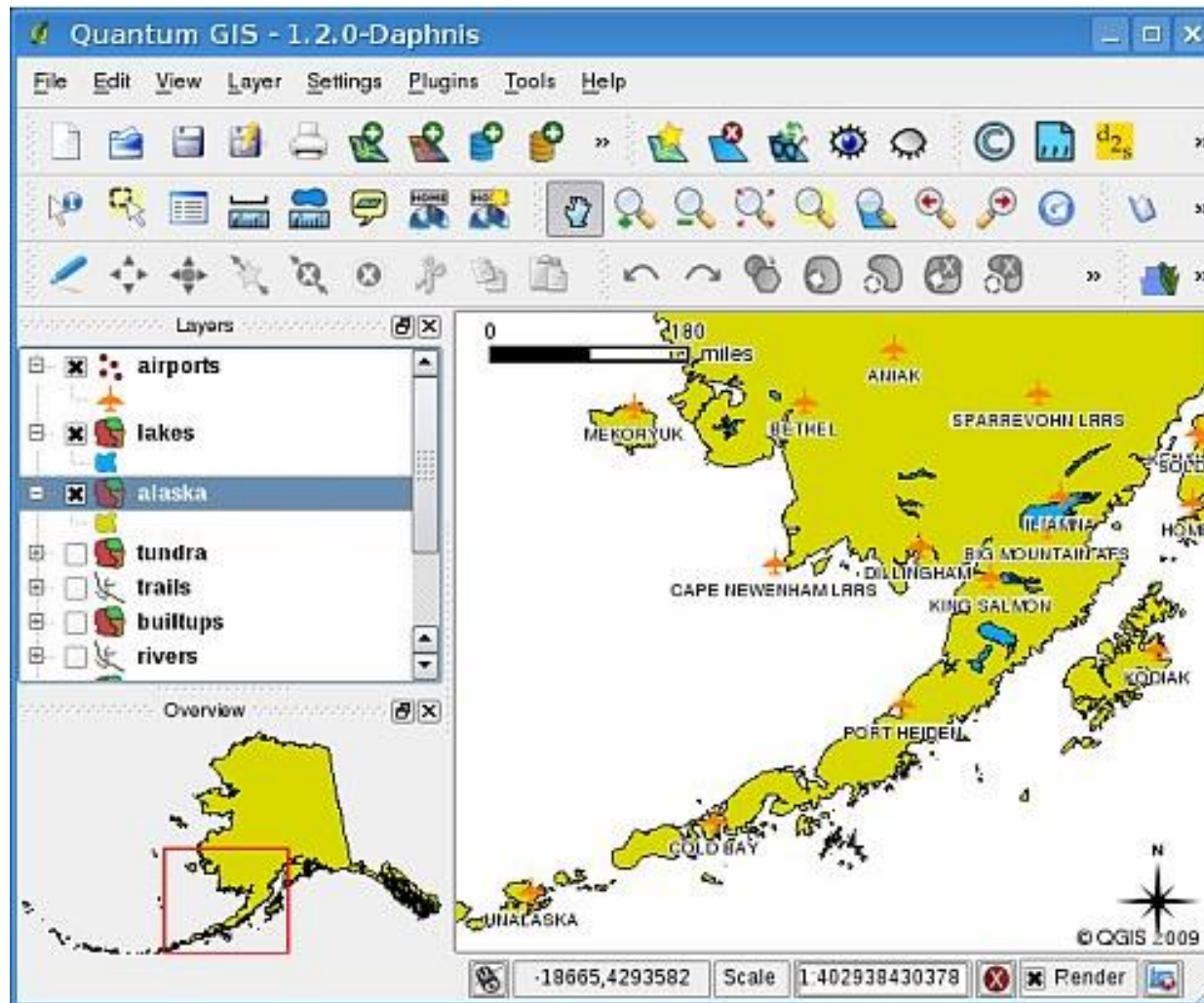
### #QGIS vs. GRASS GIS

QGIS --- 类似ARCVIEW or ARCMAP

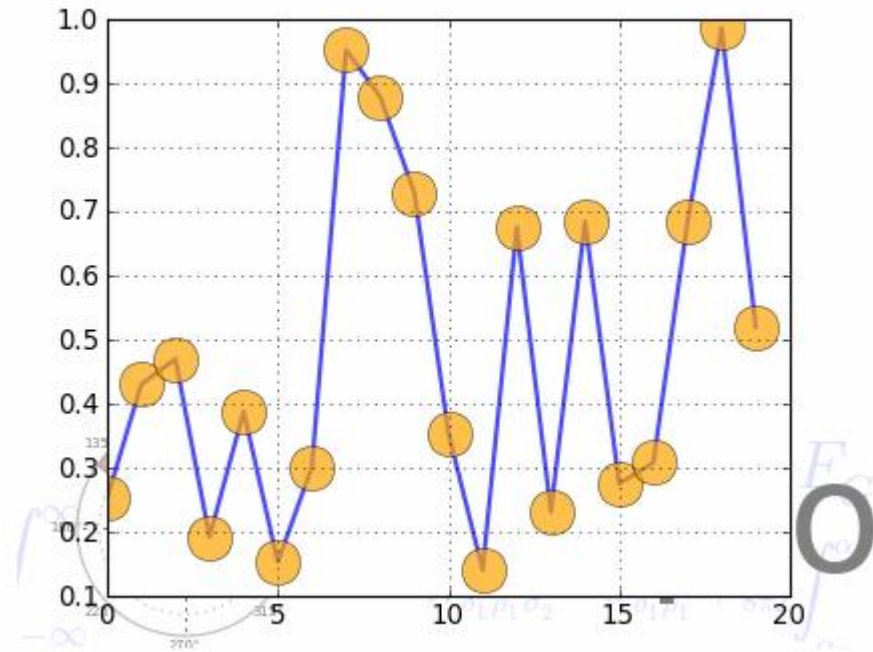
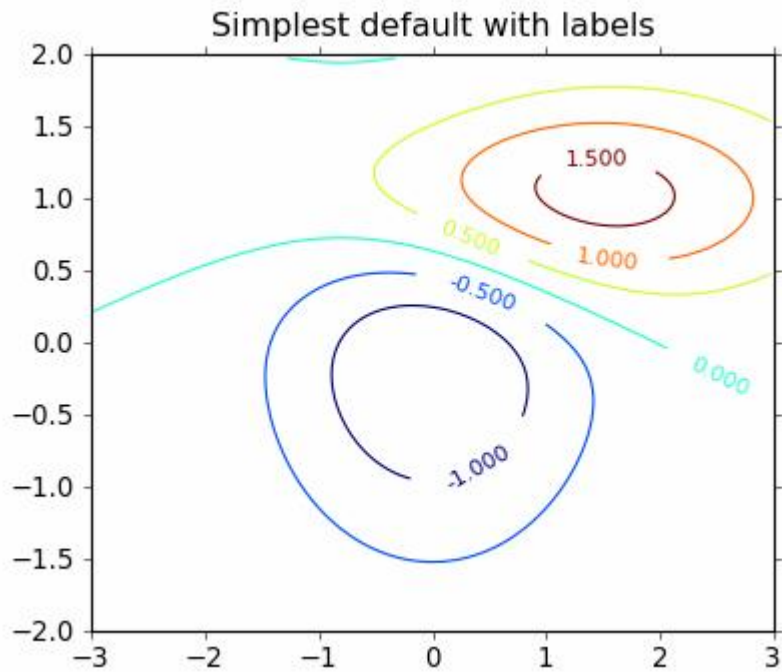
QGIS --- TOOLBOX ( GRASS GIS )

QGIS --- MAPFILE ( MAPSERVER )

# #QGIS效果



## 2.6 Matplotlib绘图包



## #PYTHON CODE - 3D双曲面

```
#D:\DIPMODEL\matplotlib-0.99.0.win32-py2.5\python>  
python d:\DIPMODEL\testdata\fig01.py
```

```
from mpl_toolkits.mplot3d import  
    axes3d  
import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib import cm
```

```
fig = plt.figure()  
ax = axes3d.Axes3D(fig)
```

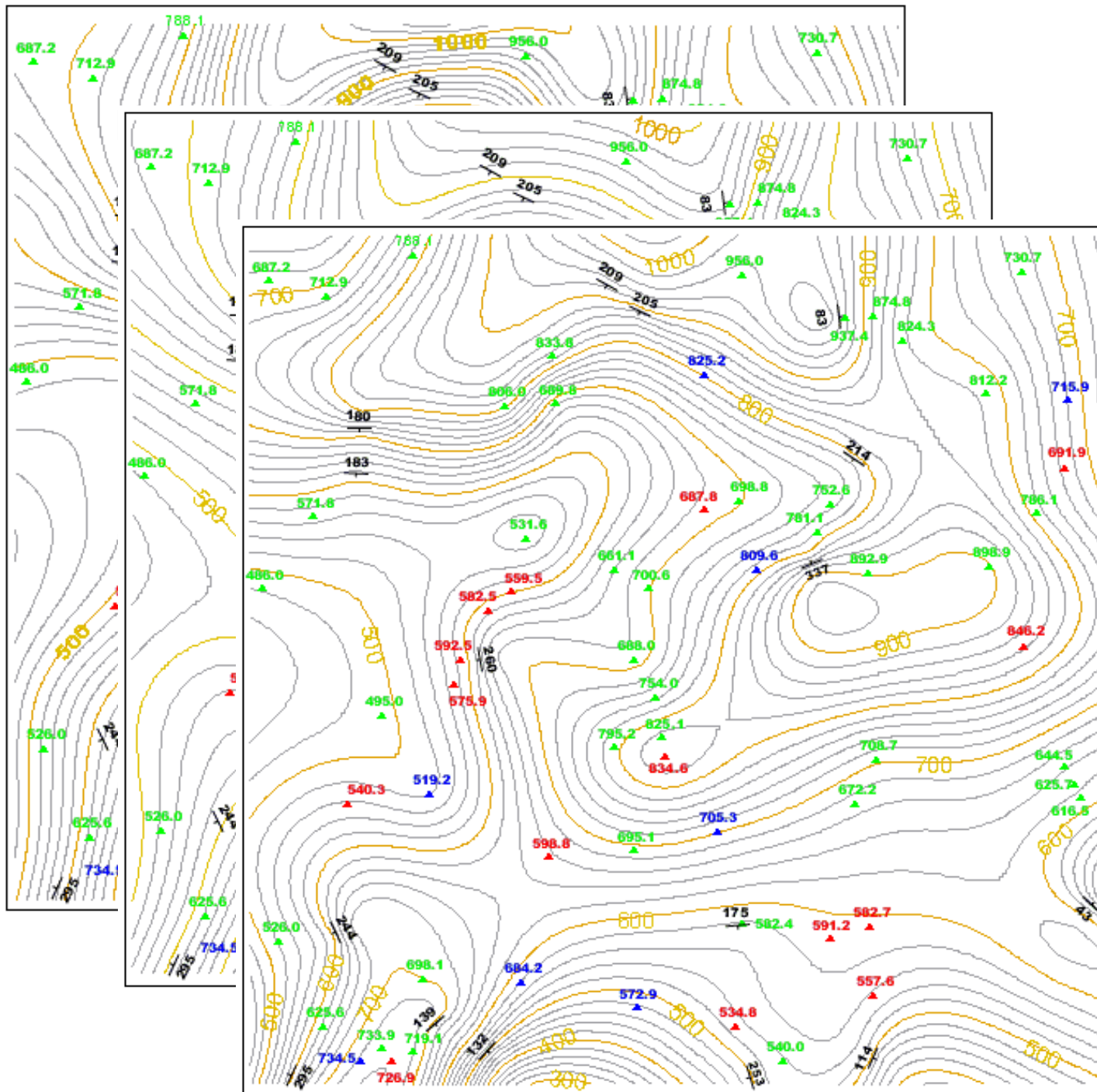
```
n = 50  
x0 = np.linspace(-11, 10, n)  
x0.shape = (n,1)  
x1 = np.ones(n)  
x = x0 * x1
```

```
y0 = np.linspace(-11, 10, n)  
y0.shape = (n,1)  
y1 = np.ones(n)  
y2 = y0 * y1  
y = y2.T
```

```
r = np.sqrt(x*x + y*y)  
z = 1.29 * r
```

```
ax.plot_wireframe(x, y, z, rstride=1,  
                  cstride=1)  
ax.set_xlabel('X Label')  
ax.set_ylabel('Y Label')  
ax.set_zlabel('Z Label')
```

```
plt.show()
```



地层产状、标高约束条件下的3D双曲面插值方法研究

### 3 Demeter电磁卫星管理系统简介

- 项目背景

国家科技支撑项目《天地一体化地震监测、预测研究》的需要，利用地面台网监测系统 + 电磁卫星监测系统，探测地震前兆信息。

Demeter电磁卫星每天15轨数据，每半轨约75M数据，15套探测参数类型，每套参数包括物理量5~10不等，以参数类型不同每半轨500~1000个样点数据，理论合计每日约获取2~3GB数据。自2004年发射以来，已经积累数据1.5TB数据（.tar.gz压缩格式）。

地震前兆信息探测是一个数据挖掘的处理过程，每次获得新数据后都要结合历史数据进行特定的数据关联分析、时间序列分析等等，为此要求建立高效的数据库存储结构、数据检索结构以及数据访问方式。

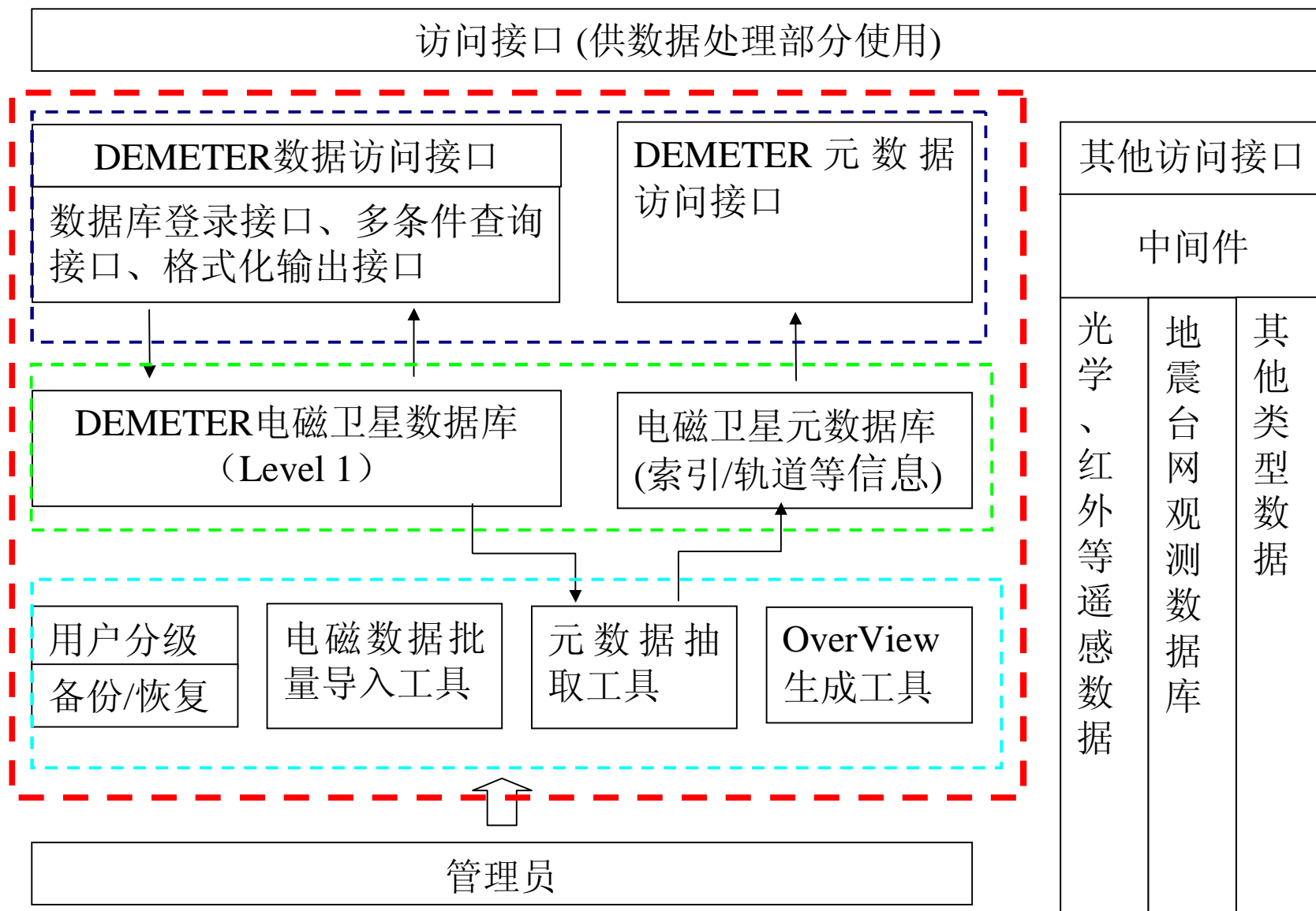
## 硬件

- 操作系统: Windows XP/ Windows2000
- CPU: Core 2 Duo 2.53GHz或更高
- 内存: 1G以上
- 硬盘空间: 500M可用空间

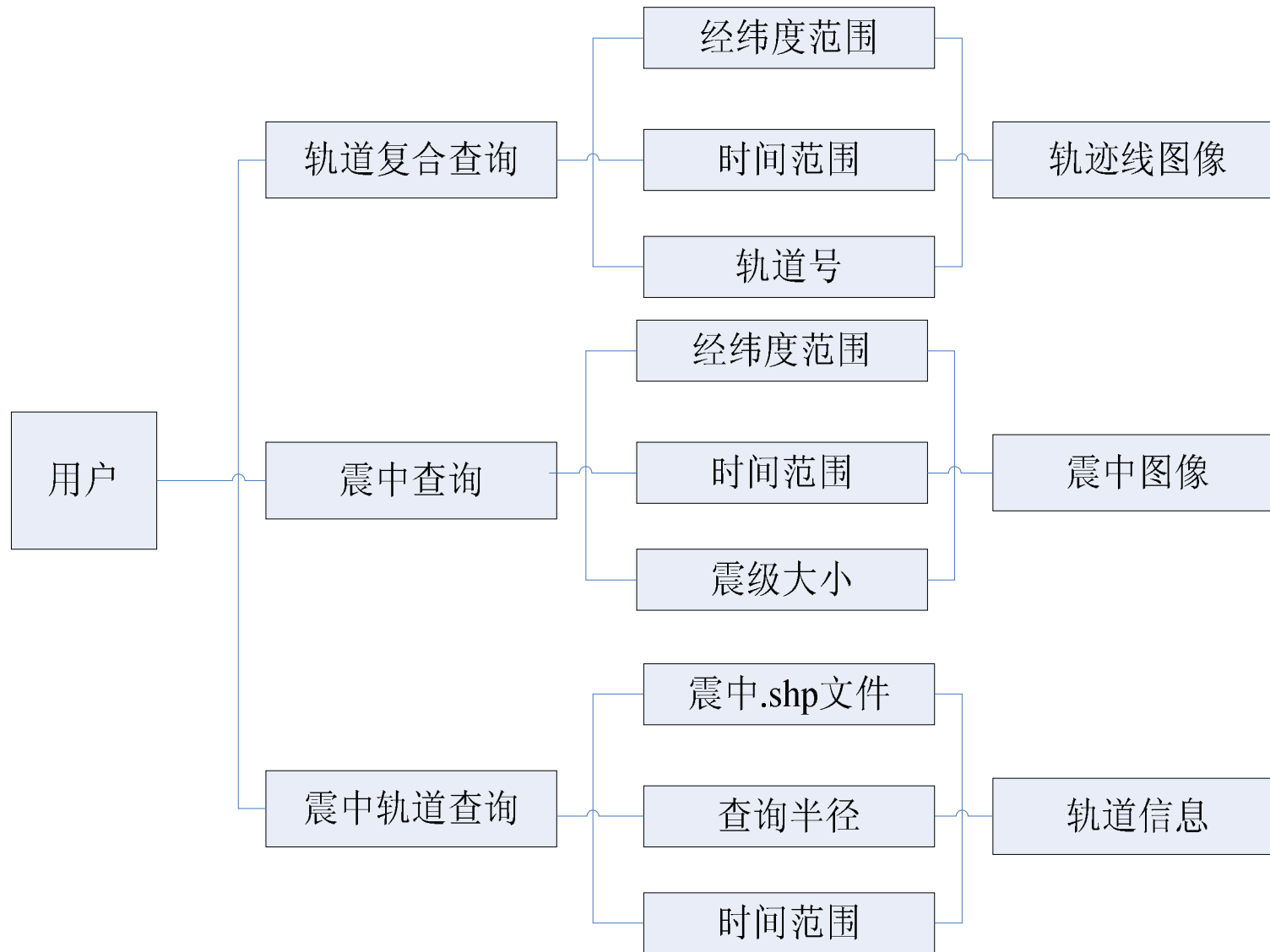
## 软件包

- 数据库Oracle 10g / PostgreSQL
- GUI界面PyQt
- 电磁数据显示matplotlib
- 地图显示QGIS
- 集成语言Python

# 系统结构



APID	表名	描述
1129	DMT_N1_ULFe_WF	超低频范围内3个电场探针的波形
1130	DMT_N1_ELFfe_WF	极低频范围内3个电场探测组件的波形
1131	DMT_N1_VLFe_WF	特低频范围内一个电场探测组件的波形
1132	DMT_N1_VLFe_SP	特低频范围内一个电场探测组件的频谱
1133	DMT_N1_HFe_WF	高频范围内一个电场探测组件的波形
1134	DMT_N1_HFe_SP	高频范围内一个电场探测组件的频谱
1135	DMT_N1_ELFb_WF	极低频范围内三个磁场探测组件的波形
1136	DMT_N1_VLFb_WF	特低频范围内一个磁场探测组件的波形
1137	DMT_N1_VLFb_SP	特低频范围内一个磁场探测组件的频谱
1138	DMT_N1_RNF	神经网络的探测结果
1139-1140	DMT_N1_IAP	IAP试验数据
1141	DMT_N1_IDP_FLUX	IDP试验数据
1142	DMT_N1_IDP_EC	IDP试验数据
1143-1144	DMT_N1_ISL	ISL试验数据
地震事件表	DMT_N1_SEISMIC_EVENTS	记录地震时的信息
单位量纲字典	DMT_N1_UNIT	记录数据中的单位量纲
坐标系统字典	DMT_N1_COORD_SYS	记录坐标系统
采样信息字典表	DMT_N1_SAMPLE	采样频率，采样数据数和持续时间表



速度： >1MB/秒 （压缩格式tar.gz情况）  
90GB/天

2004-2009数据入库：

DEMETER LEVEL1数据 / 辅助数据

1.5TB, 17天

结果格式（新增）：

.tar.gz文件原样还原

地震局要求格式

ASCII方式输出

## #批量入库工具 – python cx\_Oracle

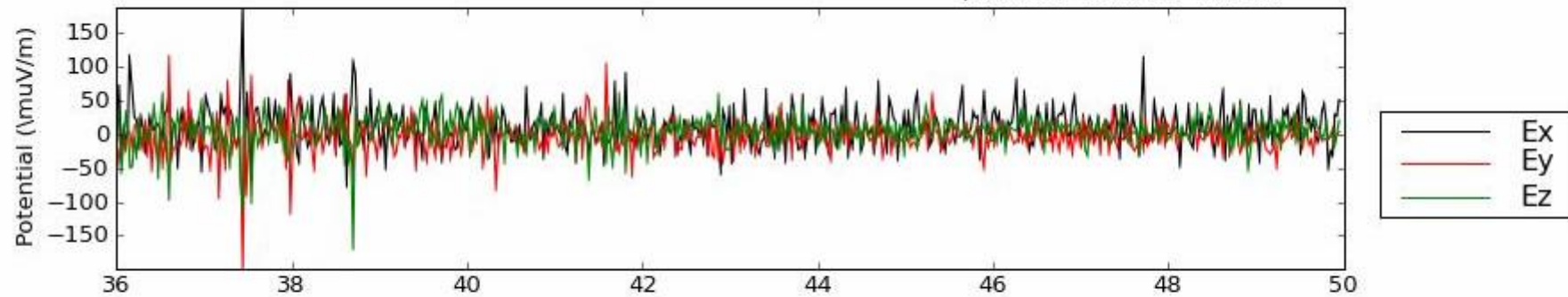


# #OVERVIEW图像生成工具 - MATPLOTLIB

根据指定的轨道及物理量，生成OVERVIEW图像。

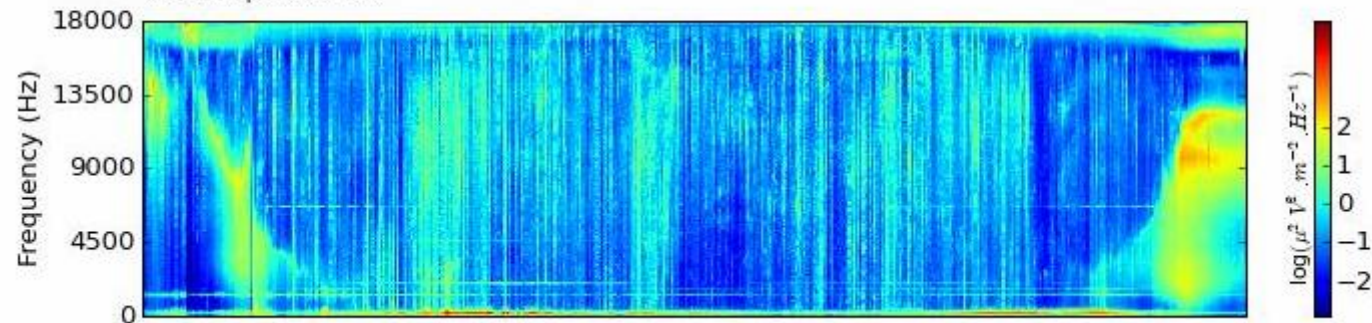
ELF ELECTRIC WAVEFORM

$\mu$ C: 1.7 soft: 3.1 cal: 3.0

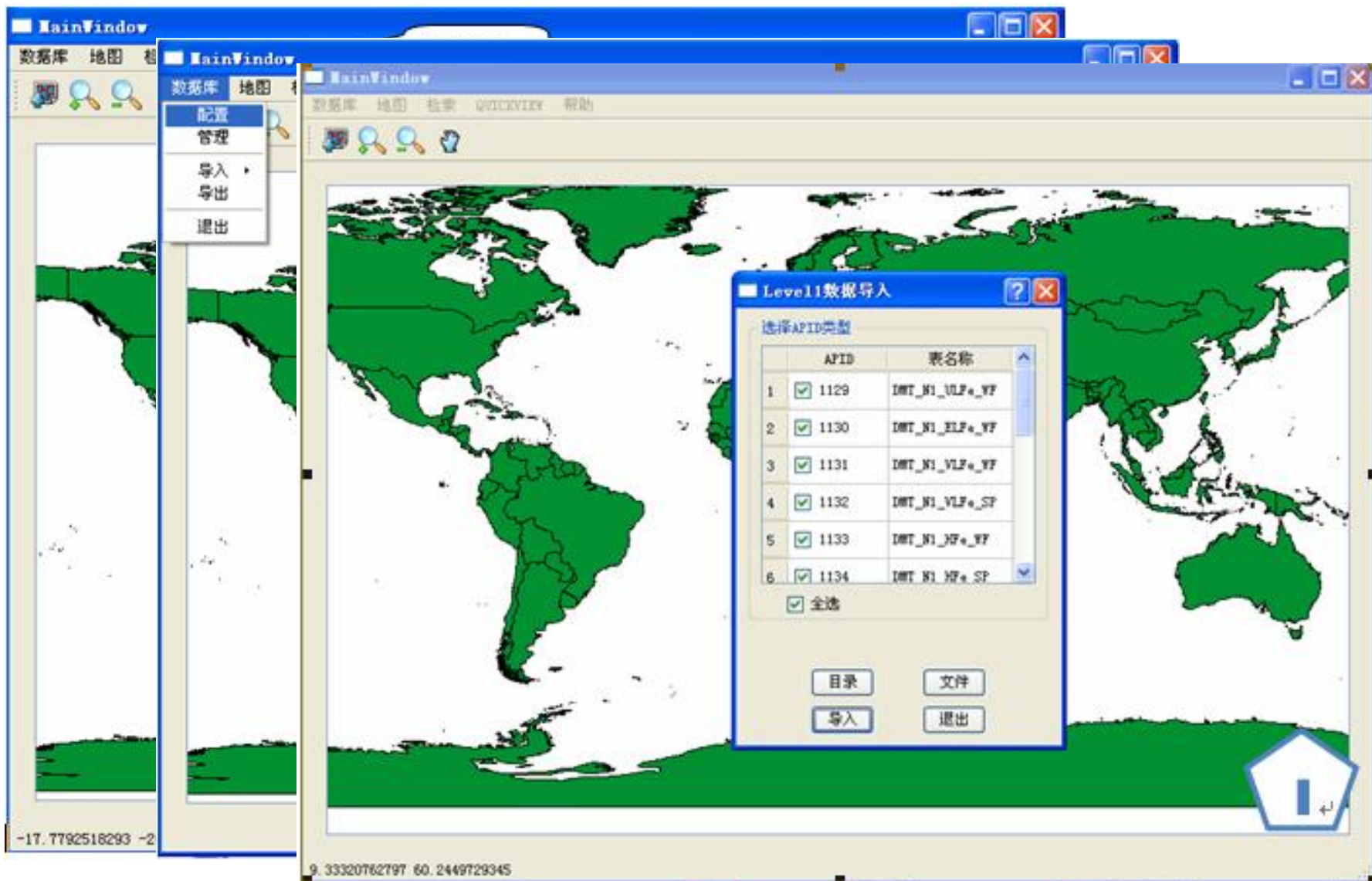


UT/LT	2007-10-13 10:05:52	2007-10-13 10:04:55	2007-10-13 10:03:56	2007-10-13 10:02:58	2007-10-13 10:02:01
Lat.	36.00	39.50	43.00	46.50	50.00
Long.	7.21	8.24	9.36	10.58	11.94

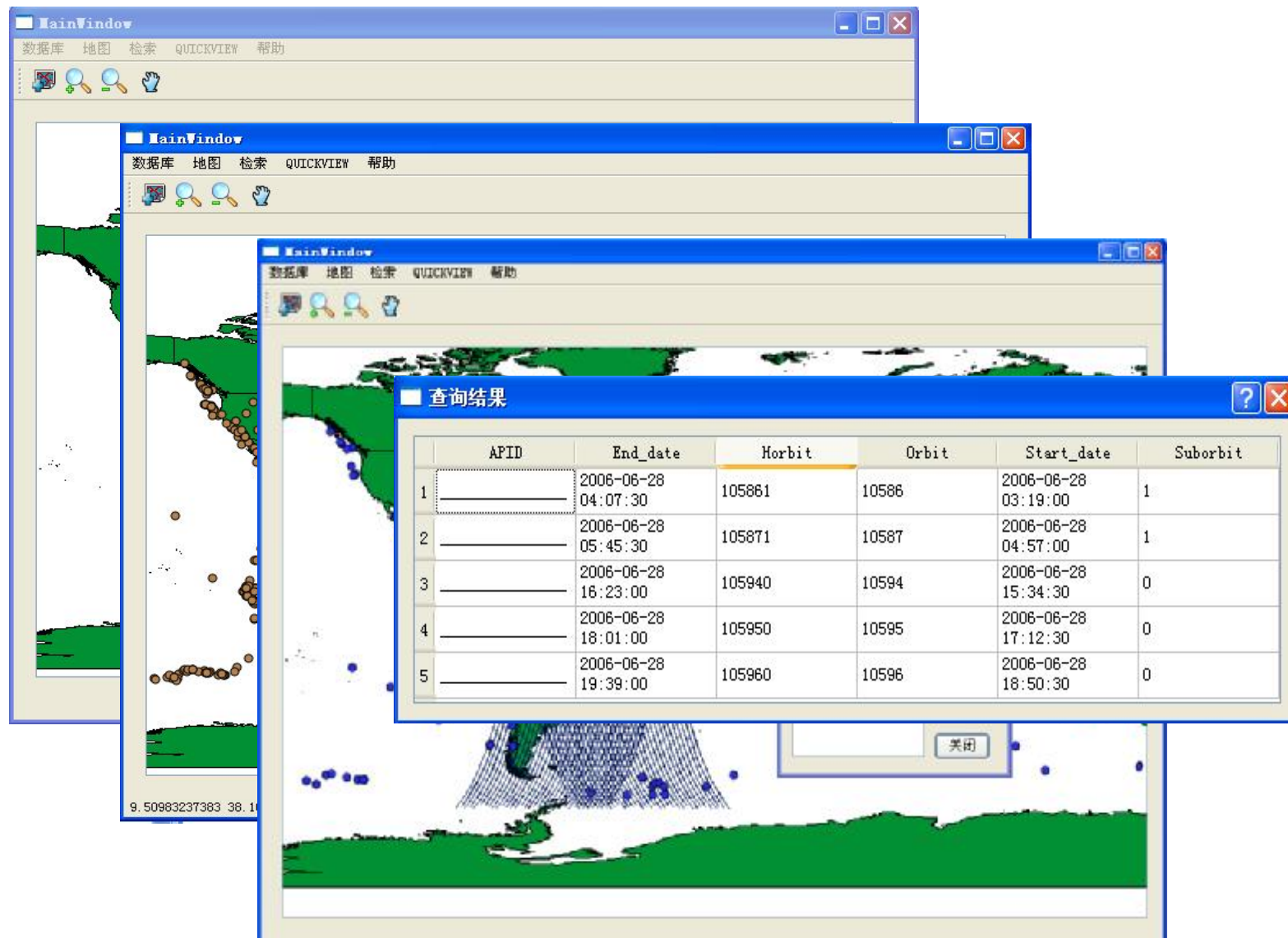
ICE VLF spectrum E12



# #Qt + QGIS



# #加载图层

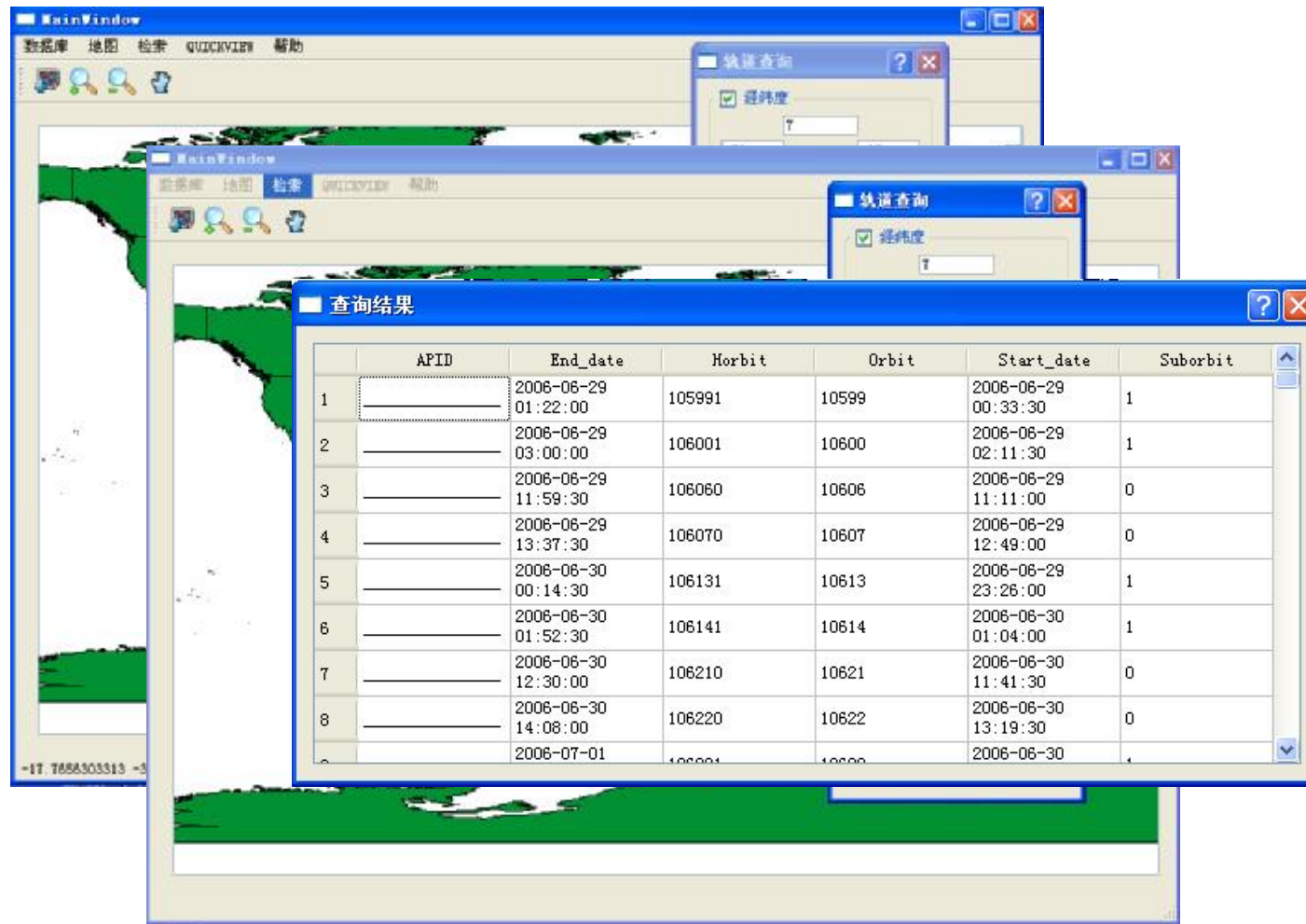


The screenshot shows a GIS application interface with multiple overlapping windows. The main window displays a map with a green landmass and a blue data overlay. A dialog box titled '查询结果' (Query Results) is open, showing a table of data points. The table has columns for APID, End\_date, Horbit, Orbit, Start\_date, and Suborbit. The data points are listed as follows:

	APID	End_date	Horbit	Orbit	Start_date	Suborbit
1		2006-06-28 04:07:30	105861	10586	2006-06-28 03:19:00	1
2		2006-06-28 05:45:30	105871	10587	2006-06-28 04:57:00	1
3		2006-06-28 16:23:00	105940	10594	2006-06-28 15:34:30	0
4		2006-06-28 18:01:00	105950	10595	2006-06-28 17:12:30	0
5		2006-06-28 19:39:00	105960	10596	2006-06-28 18:50:30	0

The dialog box also includes a '关闭' (Close) button at the bottom right. The background map shows a green landmass with a blue data overlay consisting of a grid of points and lines. The application windows have a menu bar with '数据库' (Database), '地图' (Map), '检索' (Search), 'QUICKVIEW', and '帮助' (Help). The status bar at the bottom left of the main window shows the coordinates '9.50983237383 38.1'.

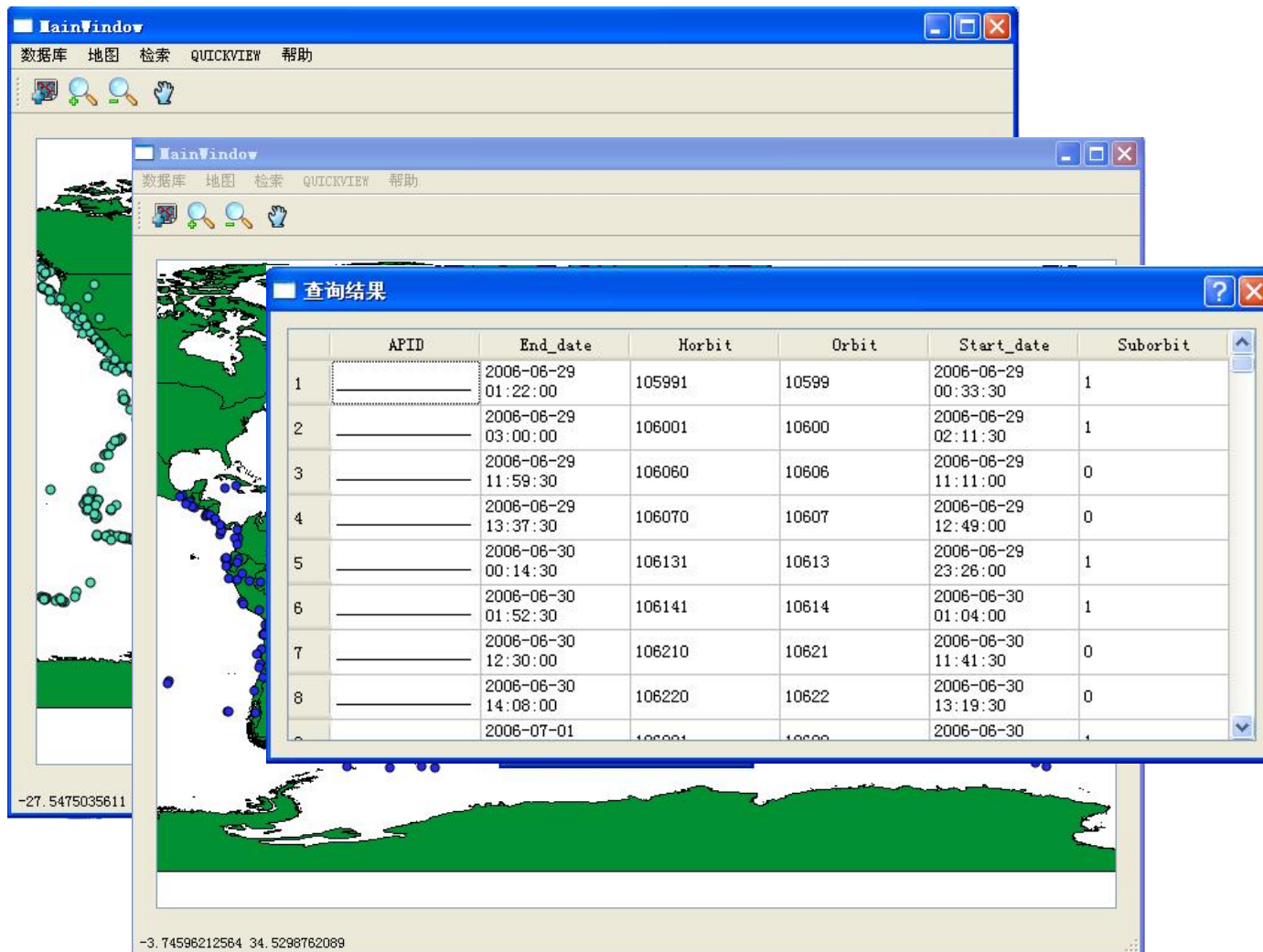
# #区域轨道查询 – cx\_Oracle



The screenshot shows a GIS application interface with multiple windows. The foreground window, titled '查询结果' (Query Results), displays a table with the following data:

	APID	End_date	Horbit	Orbit	Start_date	Suborbit
1		2006-06-29 01:22:00	105991	10599	2006-06-29 00:33:30	1
2		2006-06-29 03:00:00	106001	10600	2006-06-29 02:11:30	1
3		2006-06-29 11:59:30	106060	10606	2006-06-29 11:11:00	0
4		2006-06-29 13:37:30	106070	10607	2006-06-29 12:49:00	0
5		2006-06-30 00:14:30	106131	10613	2006-06-29 23:26:00	1
6		2006-06-30 01:52:30	106141	10614	2006-06-30 01:04:00	1
7		2006-06-30 12:30:00	106210	10621	2006-06-30 11:41:30	0
8		2006-06-30 14:08:00	106220	10622	2006-06-30 13:19:30	0
9		2006-07-01	106221	10622	2006-06-30	1

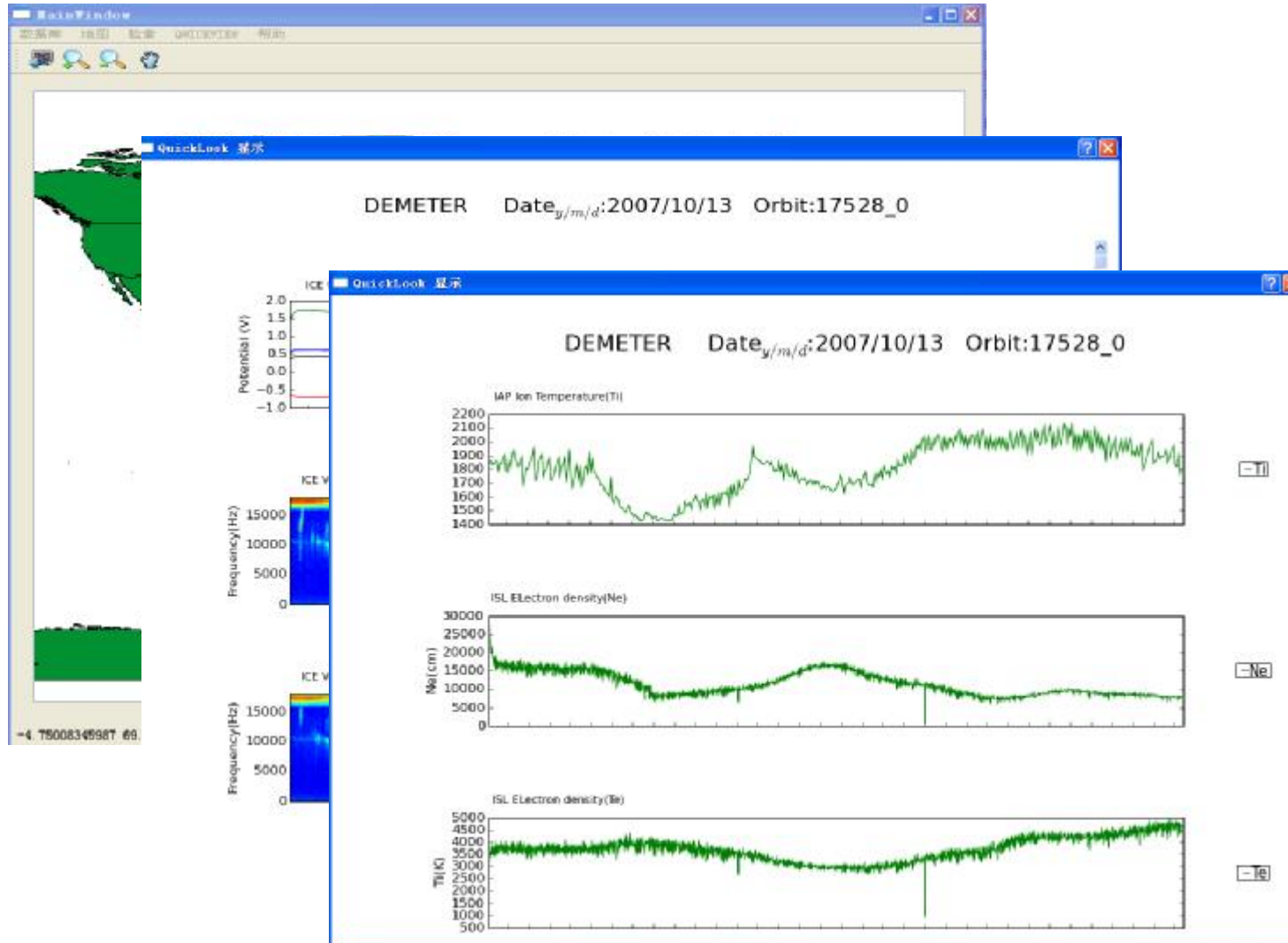
# #震中轨道查询



The screenshot shows a GIS application interface with a map of Japan. A red dot on the map indicates an earthquake epicenter. A '查询结果' (Query Results) window is open, displaying a table with the following data:

	APID	End_date	Horbit	Orbit	Start_date	Suborbit
1		2006-06-29 01:22:00	105991	10599	2006-06-29 00:33:30	1
2		2006-06-29 03:00:00	106001	10600	2006-06-29 02:11:30	1
3		2006-06-29 11:59:30	106060	10606	2006-06-29 11:11:00	0
4		2006-06-29 13:37:30	106070	10607	2006-06-29 12:49:00	0
5		2006-06-30 00:14:30	106131	10613	2006-06-29 23:26:00	1
6		2006-06-30 01:52:30	106141	10614	2006-06-30 01:04:00	1
7		2006-06-30 12:30:00	106210	10621	2006-06-30 11:41:30	0
8		2006-06-30 14:08:00	106220	10622	2006-06-30 13:19:30	0
9		2006-07-01	106230	10623	2006-06-30	0

# #电磁波谱绘制 - Matplotlib



To be a pythoner is easy, but to be pythonic is difficult

谢谢大家