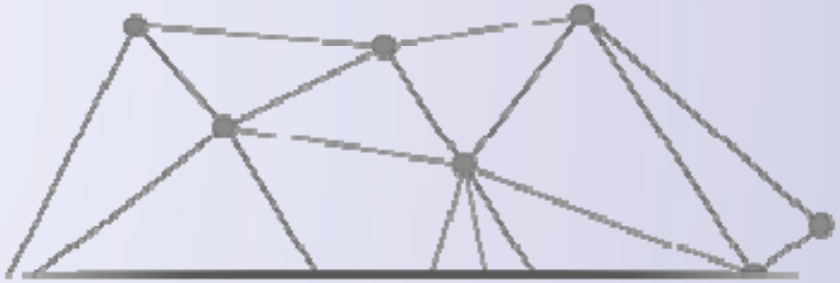
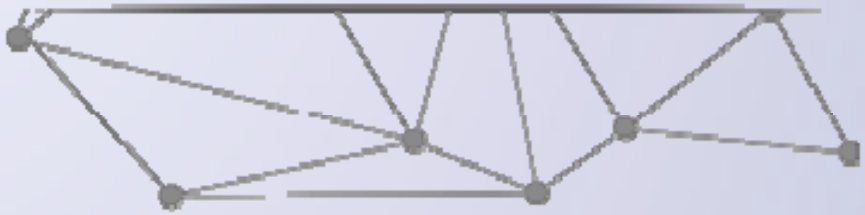
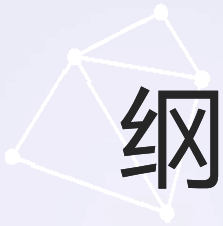


Numpy库应用实例



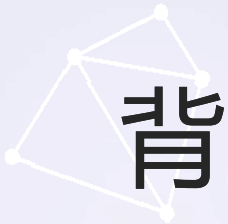
礼 欣

北京理工大学



纲要

- 背景介绍
- GPS定位的基本原理与建模
- GPS定位计算的程序实现
- 小结



背景介绍

- 定位系统
- GPS全球定位系统(Global Positioning System)
- 以GPS系统为例介绍卫星定位的计算方法

python 语言程序设计

GPS定位的基本原理

- GPS定位的基本原理是根据高速运动卫星的瞬间位置作为已知的起算数据，采用空间距离-后方交会的方法，确定待测点的位置。
  - 假设t时刻在地面待测点上安置GPS接收机，可以测定GPS信号到达接收机的时间△t，再加上接收机所接收到的卫星星历等其它数据，就可以确定一个方程组求对位置信息 进行求解。

GPS定位的基本原理

- 假设地球上一个点R，同时收到6颗卫星（S1,S2,...,S6）发射的信号，假设接受信息如下表所示。其中x，y表示卫星的经纬度，z表示卫星的高度。由于上述6个卫星和地球在高速运动，从卫星发出的位置信息以光速传输到GPS接收端需要一定的时间。假设(x,y,z,t)表示R当前的位置，t是R的相对时间，卫星S1（发出信号时刻）到（当前接收时刻)满足以下关系(其中c是光速)。
- 该公式表示以(x, y, z,t)为参数的(欧式空间距离)与信号传输距离相等。
- (x-3)^2 + (y-2)^2 + (z-3)^2 = [(10010.00692286 – t)*c]^2 ,

对于卫星S1,S2,...,S6，满足方程组

$$\begin{cases}(x-3)^2+(y-2)^2+(z-3)^2-[(10010.00692286-t)*c]^2=0\\(x-1)^2+(y-3)^2+(z-1)^2-[(10013.34256381-t)*c]^2=0\\(x-5)^2+(y-7)^2+(z-4)^2-[(10016.67820476-t)*c]^2=0\\(x-1)^2+(y-7)^2+(z-3)^2-[(10020.01384571-t)*c]^2=0\\(x-7)^2+(y-6)^2+(z-7)^2-[(10023.34948666-t)*c]^2=0\\(x-1)^2+(y-4)^2+(z-9)^2-[(10030.02076857-t)*c]^2=0\end{cases}$$

其中，光速为常数c=0.299792458km/us，上述方程组是非线性的，但很容易将所有二次项都消去（每个公式减去第一个公式），从而得到：

$$\begin{pmatrix}4&-4&-12&3.59751\\0&-2&-16&2.99792\\8&6&-10&2.39834\\0&6&-12&1.79875\\12&4&-4&1.19917\end{pmatrix}\begin{pmatrix}x\\y\\z\\t\end{pmatrix}=\begin{pmatrix}35971.1\\29957.2\\24031.4\\17993.5\\12059.7\end{pmatrix}\qquad(2)$$

此时，上述等式变成了A*X=B形式，根据线性代数方法，X=A-1*B，即只需对系数矩阵求逆，再乘以常数矩阵便可以得到方程组的解。

GPS定位的问题建模

- 上面给出了GPS的定位原理，如何利用计算机辅助GPS的定位计算呢？
- 以6颗卫星为例，GPS定位计算问题的IPO模式--描述如下：
  - 输入：6颗卫星的欧式坐标和信号时间戳
  - 处理：GPS定位算法
  - S接收设备的地理坐标和当前时间

GPS定位的问题建模

- 假设第i颗卫星的坐标和时间戳表示为(x_i,y_i,z_i,t_i)，结合上述例子，GPS定位算法可以描述为如下公式：

$$\begin{pmatrix} x_1-x_2 & y_1-y_2 & z_1-z_2 & c^2(t_1-t_2) \\ x_1-x_3 & y_1-y_3 & z_1-z_3 & c^2(t_1-t_3) \\ x_1-x_4 & y_1-y_4 & z_1-z_4 & c^2(t_1-t_4) \\ x_1-x_5 & y_1-y_5 & z_1-z_5 & c^2(t_1-t_5) \\ x_1-x_6 & y_1-y_6 & z_1-z_6 & c^2(t_1-t_6) \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} [(t_2^2-t_1^2)c^2-(x_2^2-x_1^2+y_2^2-y_1^2+z_2^2-z_1^2)]/2 \\ [(t_3^2-t_1^2)c^2-(x_3^2-x_1^2+y_3^2-y_1^2+z_3^2-z_1^2)]/2 \\ [(t_4^2-t_1^2)c^2-(x_4^2-x_1^2+y_4^2-y_1^2+z_4^2-z_1^2)]/2 \\ [(t_5^2-t_1^2)c^2-(x_5^2-x_1^2+y_5^2-y_1^2+z_5^2-z_1^2)]/2 \\ [(t_6^2-t_1^2)c^2-(x_6^2-x_1^2+y_6^2-y_1^2+z_6^2-z_1^2)]/2 \end{pmatrix} \quad (3)$$

GPS定位的问题建模

- 我们下面将使用Numpy函数库实现上述矩阵操作。首先预习一下程Numpy.linalg.inv(a)序中用到的函数：

函数	含义
numpy.dot(a,b)	计算矩阵a与矩阵b的点积
Numpy.linalg.inv(a)	求矩阵a的逆矩阵

GPS定位的程序实现

- Python代码如下：
- 其中zeros是NumPy提供的函数用来建立指定维度的数组，我们用zeros生成数组x用来存储接受来自外部输入的六颗卫星坐标，数组a，b用来存放前面算法中的系数矩阵，例程中我们还展示了两种数组的索引方法，最后调用求矩阵逆的函数及点乘操作完成坐标计

GPS定位的程序实现

```
from numpy import  *
i = 1
c = 0.299792458 #光速0.299792458km/μs
x = zeros((6, 4)) #存储6个卫星的(x,y,z,t)参数
while i <= 6:
    print("%s %d" % ("please input (x,y,z,t) of group",i))
    temp = input()
    x[i-1] = temp.split()
    j = 0
    while j < 4:
        x[i-1][j] = float(x[i-1][j])
        j = j + 1
    i = i + 1
a = zeros((4, 4)) #系数矩阵
b = zeros((4, 1)) #常数项
j = 0
while j < 4:
    a[j][0] = 2 * (x[5][0] - x[j][0])
    a[j][1] = 2 * (x[5][1] - x[j][1])
    a[j][2] = 2 * (x[5][2] - x[j][2])
    a[j][3] = 2 * c * c * (x[j][3] - x[5][3])
    b[j][0] = x[5][0] * x[5][0] - x[j][0] * x[j][0] + \
              x[5][1] * x[5][1] - x[j][1] * x[j][1] + \
              x[5][2] * x[5][2] - x[j][2] * x[j][2] + \
              c * c * (x[j][3] * x[j][3] - x[5][3] * x[5][3])
    j = j + 1
a = linalg.inv(a)        #系数矩阵求逆
print(dot(a, b))
```

GPS定位的程序实现

- zeros是用来建立指定维度的数组
- Zeros用来生成数组x用来存储接受来自外部输入的六颗卫星坐标
- 数组a，b用来存放前面算法中的系数矩阵
- 例程中我们还展示了两种数组的索引方法，最后调用求矩阵逆的函数及点乘操作完成坐标计算。

python 语言程序设计

GPS定位的程序实现

- 运行程序后，依次输入6颗卫星的坐标，运算结果如下

```
>>>
please input (x,y,z,t) of group 1
3 2 3 10010.00692286
please input (x,y,z,t) of group 2
1 3 1 10013.34256381
please input (x,y,z,t) of group 3
5 7 4 10016.67820476
please input (x,y,z,t) of group 4
1 7 3 10020.01384571
please input (x,y,z,t) of group 5
7 6 7 10023.34948666
please input (x,y,z,t) of group 6
1 4 9 10030.02076857
[[  5.00000000e+00]
 [  3.00000000e+00]
 [  1.00000000e+00]
 [  1.00000000e+04]]
```