# 通过接口输入数据(实现方式待定)

1. Azimuth;//方位角(rad) Elevation;//仰角(rad) Range;//距离(km)
2. Longitude;//经度(°) Latitude;//纬度(°) Height;//高(m) ReferenceHeight;//基准面高度(m)

结构体

typedef struct

{

vector<double> Param;

double Longitude;//经度(°)

double Latitude;//纬度(°)

double Height;//高(m)

double ReferenceHeight;//基准面高度(m)

}SRadarCoordinate; //站坐标

typedef struct

{

vector<double> t;

vector<double> Azimuth;//方位角(rad)

vector<double>Elevation;//仰角(rad)

vector<double> Range;//距离(km)

}SOriginalDatas; //原始数据

# 雷达数据处理

## 球坐标转换为直角坐标

其中为仰角，为方位角

## 确定弹道平面

对雷达数据的x、y进行线性拟合，确定弹道平面

## 计算由雷达数据确定的2D弹道段

令，得到由雷达数据确定的2D弹道段的一系列离散点。

# 弹道方程符合

以雷达数据确定的2D弹道的起点作为外弹道计算的初始位置，设置一个假定的(如何选取？)初速*v*、射向*θ*和弹道系数*c*进行外弹道计算，计算得到的外弹道轨迹为。计算结果与雷达数据确定的2D弹道对比，不断修正(如何修正？)这三个参数直至满足精度要求。

雷达数据确定的2D弹道

计算得到的2D弹道

## 符合方法

1. 以雷达获得的数据的最后一个点与第一个点间的平均速度为外弹道计算的初始速度并令弹道系数c=1计算得到第一条弹道方程。因为阻力的存在，故此弹道位于实际弹道“下方”，如下图所示。

雷达数据确定的2D弹道

计算得到的2D弹道

1. 显然计算得到的弹道在与雷达相同的时间段内的平均速度小于。将计算初速增大重新进行外弹道计算，得到更为接近雷达测得的弹道。重复上述过程直到计算得到的弹道与雷达捕获的最后一个点数据符合。此时计算得到的弹道与雷达测得的弹道可能有如下图所示的两种关系。

雷达数据确定的2D弹道

计算得到的2D弹道

根据两者的位置关系调整弹道系数c，每次调整弹道系数后需要重复步骤2，直至计算弹道满足精度要求。

判断依据：对于雷达获得的若干个2D弹道段上的离散点，分别在计算得到的外弹道轨迹上取与之相同的时刻，若所有时刻均有误差，则认为满足精度要求，可以进行下一步计算。

# 炮位、落点等的计算

根据3中得到的条件进行全弹道计算，终点为落点；~~取相反的速度和弹道系数c~~计算步长取相反数计算的终点为炮位。经过坐标变换后即可获得炮位、落点相在雷达坐标系下的坐标。