飞行器的多基雷达定位

摘 要

本文讨论的是飞行器的多基雷达定位的问题，由于在多基雷达定位过程中，测距精度误差的存在往往导致目标定位精度的不准确，为了解决这个问题，尽可能提高飞行器的多基雷达的定位精度，最好的措施就是在考虑测量误差对目标定位的影响下，设计一种快速算法确定飞行器位置。本文提供了定位算法的一般表达式，并确定算法的精度，并且就测距精度对目标定位的影响给出了严密、合理的分析。

对于问题1，在三个以上地面雷达的基础上确定一个快速算法来确定这个飞行物的空间坐标的同时，并确定所设计算法的精度，我们通过最小残差平方和得到算法模型-无约束的非线性规划模型,并采用中规模搜索法找到最优解。其中，对于模型求解的初始值设定我们采用蒙特卡罗法，避免求得局部最优解。另外，我们还通过进行雷达仿真弥补了纯理论分析的不足。

对于问题2，对于数据的反向辨识，我们认为计算多组雷达站数据，从而得到多组飞行器位置，并通过对比其差异判断是否测量误差过大。可以达到反向判别数据的可靠性的目的。

本文建立了一个模型快速判定飞行器位置，考虑了测距精度对目标定位的影响，最终为控制雷达定位精度提供了可行的、有效的定位算法及建议，进而很大程度上提高了定位精度。

关键词：多基雷达 定位算法 定位精度

1. 问题重述

天空中出现一可疑的飞行物， 我们可以利用多基雷达对其进行精确定位。假设现在有三个以上的地面雷达发现了这个飞行物，每个雷达坐标Ri(xi,yi,zi) 都是已知的。已经获取了每个雷达到飞行物的直线距离ri，需要快速确定飞行物的空间坐标 S(x,y,z)，以便跟踪识别。

由于每个雷达在测量自身坐标和飞行物到各雷达的距离都存在测量误差，这给精确定位带来了困难。设距离误差服从正态分布 N (0,t ) ，坐标误差服从正态分布 N (0,r )。

在这个假设下解决以下问题：

1、在实际情况中，往往使用更多雷达进行精确定位，设计一种快速定位算法。计算飞行物坐标；

2、给出控制雷达定位精度的建议；

3、分析并比较距离误差和坐标误差对定位精度的影响；

4、分析数据反向辨识的可行度

二、问题分析

1、在实际情况中，往往用更多雷达进行精确定位，设计一种定位算法：

通过最小残差平方和，得到非线性规划模型。

2、给出控制雷达定位精度的建议：

解决问题1后，可确定定位飞行物方程组，又结合误差满足正态分布，对方程利用全微分法变形即可。

3、分析并比较距离误差和坐标误差对定位精度的影响；

4、分析数据反向辨识的可行度：

三、模型假设

1、假定雷达运行正常，记录准确，存在误差但不存在错误。

2、距离误差与坐标误差相互独立；

3、认为雷达本身坐标Ri(xi,yi,zi)准确无误差，不考虑距离与坐标之外可能存在的误差。

四、符号说明

Ri 雷达自身坐标

ri 是每个雷达到飞行物的直线距离

S (x,y,z)是飞行物的空间坐标

N (0,t )是距离误差服从的正态分布

N (0,r ) 是坐标误差服从的正态分布

五、模型建立与求解

5.1非线性规划模型

首先，我们给出一个物体坐标残差：

我们要精确定位的标准是残差的大小达到最小。

具体要求：使残差的平方和为最小: ，则该问题可以转化为非线性规划：

可见，该模型为无约束的非线性规划模型。

5.2雷达定位精度

5.2.1当雷达选定数目n=3时

采用对测量方程中各变量进行全微分的方法推导定位误差协方差的计算公式。

其中：

5.2.2 控制雷达定位精度

（1）对题中所提及的雷达测量距离误差和坐标误差而言， 雷达测量距离误差更能影响定位精度， 这就要求在雷达测量数据过程中，尤其要严格控制雷达测量距离误差。

（2） 由模型四的分析可知，雷达定位精度很大程度上依赖于选取的数据，所以在定量计算前需要对测量数据进行筛选、慎重分析，尽量排除偶然误差数据。另外，还要注意选取的数据应该适量， 避免不必要的计算量。

（3） 针对不同测量误差，可以采用不同的处理方法，进行雷达误差矫正，最大程度程度上降低其影响。

（4）本题仅限于纯距离雷达定位，如果可能可以多采取几类数据，如角度数据， 综合起来进行定位， 更能提高定位精度。

（5） 在多基雷达空中目标定位中，地面站址误差和测量距离误差对空中目标位置矢量定位精度的影响，往往难以得到估计。而通过建模与计算，可得到一系列有规律性的结论，且可为今后实际应用提供理论依据。同时，进行雷达仿真技术为好的数学模型提供了更有效的事实依据。

5.3测距精度对定位精度的影响分析

由上述分析可见，雷达测量距离误差对飞行物定位有影响， 误差协方差阵愈大，定位误差愈大。为方便分析对飞行物定位精度的影响，我们对误差进行讨论。

根据假设，雷达距离误差服从正态分布。

5.4 模型求解

初始飞行器位置及两组雷达站坐标值为:

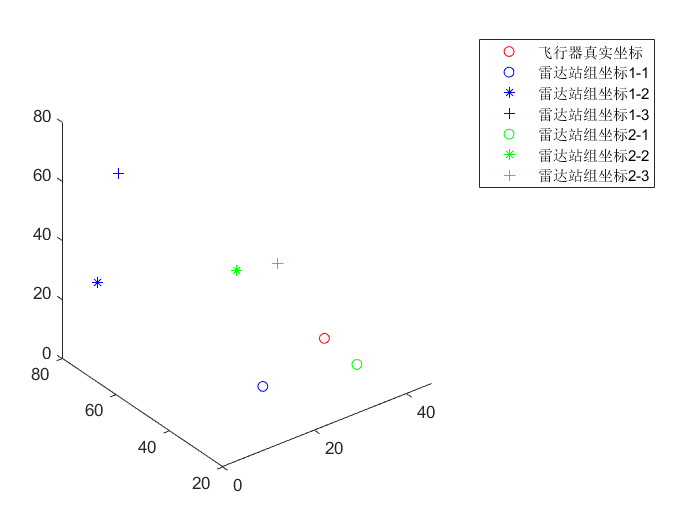


图5-1 初始飞行器位置及两组雷达站坐标值

两组雷达站坐标及距飞行器距离加上5%信噪比高斯白噪声的测量数据:

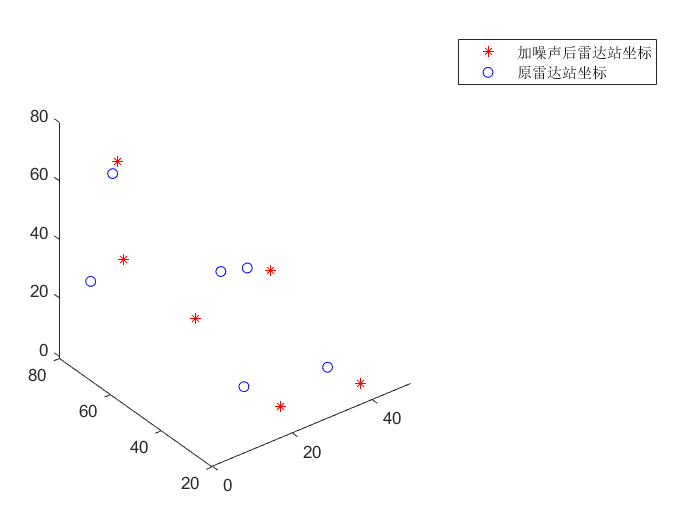


图5-2 雷达坐标加了5%信噪比高斯白噪声的测量数据

最后求得两组飞行器坐标与真实距离的对比，可以看出两组飞行器坐标差值较大，精度较低。

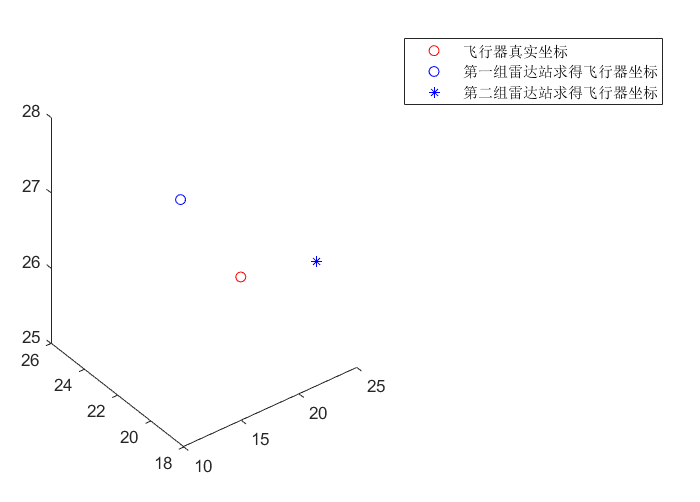


图5-3 两组飞行器坐标与真实距离的对比

六、模型的评价和推广

1. 模型的优点：

1.1将距飞行器距离及雷达站坐标加入正太分布的噪声，可以模拟真实情况下获得的数据；

1.2采用无约束非线性规划模型,通过不断迭代寻找靠近（减小或者增大）目标值，从而找到最优解；

1.3该模型整和了多个测量数据，具有很强的可信度，且操作简便，不失为一种便于实际操作的定位算法。

1. 模型的缺点：

在对无约束的非线性规划模型进行分析时，采用中规模搜索法

缺点：

2.1对学习率关系很大，在极小点附近容易产生显著的锯齿现象，收敛十分缓慢；

2.2对于小扰动是不稳定的，且梯度下降法只是一种局部性质，也就是说从局部看目标函数的值下降得最快，但从总体看可能走了许多弯路（因为每相邻两个方向都是正交的）；

2.3可能得到局部最小值，而不是全局最小值。

综上，该模型可以推广到各类纯距离定位计算，如卫星定位等。

七、参考文献

[1]白菊蓉 ,毛永毅 三基地雷达最佳空间定位算法，安邮电学院，2007年 10月

[2]陈建春，丁鹭飞，多基地雷达最佳定位算法，西安电子科技大学，1001-2400( 1999) 04-0397-04

[3] 郭徽东,章新华,王为颂,叶琼龙, 基于最佳线性无偏估计的双基地雷达定位方法 ( 大连舰艇学院 信号与信息技术研究中心, 辽宁 大连 116018)

附录

Matlab.mat

1.r(计算得雷达站距飞行物距离)

23.5664436487562

56.6130898440458

63.4552017174321

17.8660659116662

28.9509542055180

59.8815169266244

2.radio(6次雷达站坐标数据)

18.4500000000000 36.7240000000000 5.63450000000000

6.75630000000000 78.5340000000000 22.4860000000000

3.76200000000000 65.2340000000000 69.2350000000000

33.4650000000000 27.4250000000000 9.45460000000000

12.5640000000000 36.5454000000000 48.4565000000000

45.4654400000000 77.7800000000000 5.47480000000000

3.r\_noise(加高斯噪声后雷达站距飞行物距离)

26.8708601494567

64.0512344928894

60.9422450938461

15.1872382868725

24.4272028687894

68.6897417206571

4.radio\_noise(加高斯噪声后6次雷达站坐标数据)

17.8488203325881 33.1420995461404 0.610661006690357

11.2789667724915 78.7598366925222 22.1874288805017

-0.143206312028481 63.4166710974184 74.7295253674410

35.0809147333084 20.6782626809867 10.6578532449820

6.96381671880606 34.5448780552003 44.7415766573674

39.6676269422113 76.6294482007818 1.46321121539515

%% 最小残差平方和损失函数(第一组雷达站数值计算)

function f=position\_radio(x)

load matlab.mat

error\_1=r\_noise(1)-sqrt((x(1)-radio\_noise(1,1))^2+(x(2)-radio\_noise(1,2))^2+(x(3)-radio\_noise(1,3))^2);

error\_2=r\_noise(2)-sqrt((x(1)-radio\_noise(2,1))^2+(x(2)-radio\_noise(2,2))^2+(x(3)-radio\_noise(2,3))^2);

error\_3=r\_noise(3)-sqrt((x(1)-radio\_noise(3,1))^2+(x(2)-radio\_noise(3,2))^2+(x(3)-radio\_noise(3,3))^2);

f=error\_1^2+error\_2^2+error\_3^2;

end

%% 最小残差平方和损失函数2(第二组雷达站数值计算)

function f=position\_radio2(x)

load matlab.mat

error\_1=r\_noise(4)-sqrt((x(1)-radio\_noise(4,1))^2+(x(2)-radio\_noise(4,2))^2+(x(3)-radio\_noise(4,3))^2);

error\_2=r\_noise(5)-sqrt((x(1)-radio\_noise(5,1))^2+(x(2)-radio\_noise(5,2))^2+(x(3)-radio\_noise(5,3))^2);

error\_3=r\_noise(6)-sqrt((x(1)-radio\_noise(6,1))^2+(x(2)-radio\_noise(6,2))^2+(x(3)-radio\_noise(6,3))^2);

f=error\_1^2+error\_2^2+error\_3^2;

end

%% 飞行器的多基雷达定位

% 首先准备两组雷达站准确值及飞行器坐标,计算得到距飞行器真实距离

% 再将这几组雷达真实坐标及其距飞行器距离加入高斯白噪声来模拟真实情况下的测量数据

% 并以此建立无约束非线性规划模型来求解飞行物坐标,并与飞行物真实坐标进行比较

% 其中在求解规划模型里，采用了LargeScale寻优函数

clear;clc

format long g

load matlab.mat

s=[25,25,25] %预设飞行器真实坐标

r(1)=sqrt((radio(1,1)-s(1))^2+(radio(1,2)-s(2))^2+(radio(1,3)-s(3))^2);

r(2)=sqrt((radio(2,1)-s(1))^2+(radio(2,2)-s(2))^2+(radio(2,3)-s(3))^2);

r(3)=sqrt((radio(3,1)-s(1))^2+(radio(3,2)-s(2))^2+(radio(3,3)-s(3))^2);

disp('求得第一组雷达站距飞行器真实距离为：'); disp(r(1:3,:))

r(4)=sqrt((radio(4,1)-s(1))^2+(radio(4,2)-s(2))^2+(radio(4,3)-s(3))^2);

r(5)=sqrt((radio(5,1)-s(1))^2+(radio(5,2)-s(2))^2+(radio(5,3)-s(3))^2);

r(6)=sqrt((radio(6,1)-s(1))^2+(radio(6,2)-s(2))^2+(radio(6,3)-s(3))^2);

disp('求得第二组雷达站距飞行器真实距离为：'); disp(r(4:6,:))

plot3(s(1),s(2),s(3),'ro',radio(1,1),radio(1,2),radio(1,3),'bo',radio(2,1),radio(2,2),radio(2,3),'b\*',radio(3,1),radio(3,2),radio(3,3),'b+',radio(4,1),radio(4,2),radio(4,3),'go',radio(5,1),radio(5,2),radio(5,3),'g\*',radio(6,1),radio(6,2),radio(6,3),'g+')

legend('飞行器真实坐标','雷达站组坐标1-1','雷达站组坐标1-2','雷达站组坐标1-3','雷达站组坐标2-1','雷达站组坐标2-2','雷达站组坐标2-3')

%% 加了5%信噪比高斯白噪声的测量数据

figure(2)

r\_noise=awgn(r,10\*log10(0.05))

radio\_noise=awgn(radio,10\*log10(0.05))

plot3(radio\_noise(:,1),radio\_noise(:,2),radio\_noise(:,3),'r\*',radio(:,1),radio(:,2),radio(:,3),'bo')

legend('加噪声后雷达站坐标','原雷达站坐标')

% 使用蒙特卡罗的方法来找初始值(推荐）

n=10000000; %生成的随机数组数

x1=unifrnd(-100,100,n,1); % 生成在[-100,100]之间均匀分布的随机数组成的n行1列的向量构成x1

x2=unifrnd(-100,100,n,1); % 生成在[-100,100]之间均匀分布的随机数组成的n行1列的向量构成x2

x3=unifrnd(-100,100,n,1); % 生成在[-100,100]之间均匀分布的随机数组成的n行1列的向量构成x3

fmin=+inf; % 初始化函数f的最小值为正无穷（后续只要找到一个比它小的我们就对其更新）

for i=1:n

x = [x1(i), x2(i),x3(i)]; %构造x向量

% 判断是否满足条件

error\_1=r\_noise(1)-sqrt((x(1)-radio\_noise(1,1))^2+(x(2)-radio\_noise(1,2))^2+(x(3)-radio\_noise(1,3))^2);

error\_2=r\_noise(2)-sqrt((x(1)-radio\_noise(2,1))^2+(x(2)-radio\_noise(2,2))^2+(x(3)-radio\_noise(2,3))^2);

error\_3=r\_noise(3)-sqrt((x(1)-radio\_noise(3,1))^2+(x(2)-radio\_noise(3,2))^2+(x(3)-radio\_noise(3,3))^2);

result=error\_1^2+error\_2^2+error\_3^2;

% 如果满足条件就计算函数值

if result < fmin % 如果这个函数值小于我们之前计算出来的最小值

fmin = result; % 那么就更新这个函数值为新的最小值

x0 = x; % 并且将此时的x1 x2 x3更新新为初始值

end

end

disp('蒙特卡罗选取的初始值为：'); disp(x0)

%% 对寻优函数搜索方式的设定 ，LargeScale指大规模搜索，off表示在规模搜索模式关闭

options = optimset('LargeScale','off');

optimset('Display','off')

[x,fval,exitflag ] = fminunc('position\_radio',x0,options)

disp('无约束非线性规划求得第一组飞行器坐标为：'); disp(x)

[x1,fval,exitflag ] = fminunc('position\_radio2',x0,options)

disp('无约束非线性规划求得第二组飞行器坐标为：'); disp(x1)

figure(3)

plot3(s(1),s(2),s(3),'ro',x(1),x(2),x(3),'bo',x1(1),x1(2),x1(3),'b\*')

legend('飞行器真实坐标','第一组雷达站求得飞行器坐标','第二组雷达站求得飞行器坐标')

>>命令行

s =

25 25 25

求得第一组雷达站距飞行器真实距离为：

23.5664436487562

56.6130898440458

63.4552017174321

求得第二组雷达站距飞行器真实距离为：

17.8660659116662

28.950954205518

59.8815169266244

r\_noise =

19.3749878866668

42.0841238235201

64.2452859811387

20.0557155934694

34.764330806257

59.2471556096422

radio\_noise =

1 至 2 列

24.11201887103 31.1364442574603

12.564491078013 74.9581344321395

4.75226554714526 64.9192195909663

38.9322342392074 23.0642885200125

-0.247309821810184 26.590207810243

49.6206220283552 75.2530082736604

3 列

-0.848210545075035

28.9533406522081

72.8857151796575

3.59191610678374

46.0124391565701

3.97060312626232

蒙特卡罗选取的初始值为：

1 至 2 列

20.1579148828493 34.9291722510665

3 列

17.7273219894795

ans =

包含以下字段的 struct:

Display: 'off'

MaxFunEvals: []

MaxIter: []

TolFun: []

TolX: []

FunValCheck: []

OutputFcn: []

PlotFcns: []

ActiveConstrTol: []

Algorithm: []

AlwaysHonorConstraints: []

DerivativeCheck: []

Diagnostics: []

DiffMaxChange: []

DiffMinChange: []

FinDiffRelStep: []

FinDiffType: []

GoalsExactAchieve: []

GradConstr: []

GradObj: []

HessFcn: []

Hessian: []

HessMult: []

HessPattern: []

HessUpdate: []

InitBarrierParam: []

InitTrustRegionRadius: []

Jacobian: []

JacobMult: []

JacobPattern: []

LargeScale: []

MaxNodes: []

MaxPCGIter: []

MaxProjCGIter: []

MaxSQPIter: []

MaxTime: []

MeritFunction: []

MinAbsMax: []

NoStopIfFlatInfeas: []

ObjectiveLimit: []

PhaseOneTotalScaling: []

Preconditioner: []

PrecondBandWidth: []

RelLineSrchBnd: []

RelLineSrchBndDuration: []

ScaleProblem: []

SubproblemAlgorithm: []

TolCon: []

TolConSQP: []

TolGradCon: []

TolPCG: []

TolProjCG: []

TolProjCGAbs: []

TypicalX: []

UseParallel: []

Local minimum found.

Optimization completed because the size of the gradient is less than

the default value of the optimality tolerance.

<stopping criteria details>

x =

1 至 2 列

12.1275174344157 19.6768123206494

3 列

27.8464680703745

fval =

57.2158546206891

exitflag =

1

无约束非线性规划求得第一组飞行器坐标为：

1 至 2 列

12.1275174344157 19.6768123206494

3 列

27.8464680703745

Local minimum found.

Optimization completed because the size of the gradient is less than

the default value of the optimality tolerance.

<stopping criteria details>

x1 =

1 至 2 列

22.4654557956717 18.6696447364406

3 列

26.4642541325439

fval =

55.7012119377231

exitflag =

1

无约束非线性规划求得第二组飞行器坐标为：

1 至 2 列

22.4654557956717 18.6696447364406

3 列

26.4642541325439