Отчёт о выполнении работы.

Введение.

В радиолокации существует очень много проблем и задач. Одной из них является разрешение целей, а то есть определение одна ли цель обнаружена радиолокатором или две. Как происходит отождествление целей. Оно происходит в три этапа. Первый – привязка координат к единому времени. Второй – отождествление по координатам. Третий – отождествление по скорости. После проведения этих этапов можно сказать две перед нами цели или одна. А знание этого очень важно для принятия решения командным центром.

Сейчас нас интересует второй этап, а именно отождествление по координатам. Как оно происходит? (Здесь кусок теории.) Нас же интересует ковариационная матрица ошибок разности координат. Здесь предлагается оценить характеристики двух подходов. Что выгоднее использовать: вычислять каждый раз ковариационную матрицу в декартовых координатах, или же использовать ковариационную матрицу уже готовую в сферических координатах, но содержащую некоторые погрешности для самого плохого случая?

Проделанная работа.

Для моделирования нашего процесса будем использовать MatLab. Прежде всего смоделируем самый простейший случай. Координаты целей получаются в декартовых координатах и ковариационная матрица постоянна.

function h = JustForStart(R)

% Заведём начальные данные

% Константа С1

C1 = 11.372;

% Истинные координаты целей

l1 = [R 0 0]';

l2 = [R 0 0]';

% Вектора расположения МФР отсносительно ПБУ

g1 = [10 10 0]';

g2 = [10 10 0]';

%Заведём сигма для каждого МФР

SIGMA = 100;

%"Получим" данные о координатах целей от МФР

% нормальное распределение с МО = x1 и СКО = SIGMA

x1 = normrnd(l1,SIGMA);

x2 = normrnd(l2,SIGMA);

% Заведём ковариационную матрицу для каждого МФР

S = SIGMA \* SIGMA \* ones(1,3);

k1 = diag(S);

k2 = k1;

% Вычисления

% Получаем координаты предполагаемых целей относительно ПБУ

Lambda1 = x1 + g1;

Lambda2 = x2 + g2;

% Находим дельта лямбда, ковариационную матрицу ошибок разности координат

DL = Lambda1 - Lambda2;

K = k1 + k2;

% Отождествление по координатам путём сравнения с порогом обобщённого

% расстояния между координатами двух трасс

h = DL'\*inv(K)\*DL

if h < C1

h = 0;

else

h = 1;

end

end

Здесь построена функция зависящая от расстояния X до целей.Y и Z можно менять как нам удобно. Нам нужен этот кусок кода для того чтобы оценить вероятность того что будет ложный результат отождествления. В этом нам поможет следующий код.

clear all;

% Заведём массив для подсчёта количества ошибочного разделения

A = zeros(1,50);

% Делаем двойной цикл

% Внешний для измения R

for i = 1:50

%Этот j раз выполняет функцию

for j = 1:10000

H(i,j) = JustForStart(i\*2000);

% Если получилось что цели разные, то прибавим 1 к элементу массива

if H(i,j) == 1

A(i) = A(i)+1;

end

end

end

% Получим вероятность того что цели разные

A = A/10000;

% Завели массив для расстояния

X = 2000:2000:100000;

% Построили график

plot(X,A);

grid;

title('Вероятность ошибочного разделения целей');

xlabel('Расстояние до цели(ей)');

ylabel('Верояность');

Получены след. результаты. Для сигма по координатам в 10 м.



Для сигма по координатам в 100 м.



Как и следовало ожидать, вероятность ошибочного разделения одной цели находится на уровне 1%.

Но не всё так просто. На самом деле ков. матрица в декартовых координатах конечно же зависит от самих координат. А получают её из ков. матрицы в сферических координатах, которая в свою очередь принимается постоянной и при её расчёте опираются на самые плохие условия работы.

Теперь же попробуем совершить некоторое упрощение, которое нам поможет в дальнейшем и позволит проверить некоторые выкладки. Будем работать в двумерном пространстве в полярных координатах. В них имеем постоянную ковариационную матрицу. Она диагональна. По диагоналям стоит дисперсия R и theta. Чтобы получить ковариационную матрицу в декартовых координатах нужно провести следующую операцию U\*K\*U’. (Здесь оформлю нормально, красивее), где матрица U – матрица частных производных, матрица Якоби, а К – ков. матрица. Проверим так ли это. В этом нам поможет следующий код.

clear all;

% Проверим наши выкладки для двух координат. С ними работать проще и

% нагляднее.

% Заведём начальные данные. В полярных координатах расстояние и угол тэта.

R = 100000;

th = 0.785398;

%th = 0;

% Заведём ковариационную матрицу.

S\_R = 100;

S\_Az = 0.0001;

K = [S\_R 0; 0 S\_Az];

% Совершим преобразование координат

[x,y] = pol2cart(th,R);

% Матрица перехода от полярных к декартовым

U = [x/R y/R;-y/(R^2) x/(R^2)];

% Смотрим как выглядит ковариационная матрица в декартовых координтах.

K1 = U'\*K\*U;

Dl = [1,1];

K3 = inv(K1);

h = Dl\*K3\*Dl'

Сейчас просто мысли по поводу результатов.

При theta = 45 градусам получим красивую матрицу [50 50, 50 50] что вроде логично. (А и вопрос как тогда происходит вычисление в этом случае: обратной для неё ж нет). Но при theta = 135 такую же матрицу уже не получить, получается [50 -50, -50 50]. Хотя матрицы должны быть одинаковыми. Значит нужны модули перед матрцами U? А так же при очевидном случае когда theta = 0, должны получить след. результаты для матрицы [дельта\_р 0, 0 R\*sin(дельта\_тэта)], что в наших числах и довольно малом дельта\_тэта должно давать [100 0, 0 10] получается вместо этого [100 0, 0 10^(-14)]. Плюс при рассчётах числа для сравнения с С1 получаются числа очень большого порядка.