**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**«Национальный исследовательский**

**Нижегородский государственный университет им. Н.И. Лобачевского»**

Радиофизический факультет

Кафедра безопасности информационных систем

Специальность «Информационная безопасность телекоммуникационных систем»

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Применение алгоритмов последовательного сглаживания траекторий летательных аппаратов в угломерных суммарно-дальномерных системах пассивного радиотехнического контроля

Научный руководитель Карельский И. Н.

Студент 4-го курса Тройняков В. Д.

Нижний Новгород, 2019

СОДЕРЖАНИЕ

Введение ........................................................................................................... 3

Теоретическая часть ...................................................................... …………..5

Угломерный суммарно-дальномерный метод……………………………....5

Калмановская фильтрация .........................................................................…..6

Ошибки местоопределения…………………………………………………...8

Основные подходы в определении точности позиционных способов координатометрии ИРИ……………………………………………………...13

Постановка задачи………………………………………................................18

Результаты моделирования…………………………………………………..18

Заключение……………………………………………………………………22

Список литературы…………………….………...……………………………...23

Примечания………………………………………………………………………24

ВВЕДЕНИЕ

Современным системам контроля воздушного пространства военного на­значения, базирующимся на применении методов и средств активной радиоло­кации, присущ ряд принципиальных недостатков. К таким недостаткам можно отнести, например, существенное снижение эффективности работы РЛС в ус­ловиях радиоэлектронного подавления, низкая живучесть РЛС при применении противником противорадиолокационного самонаводящегося на излучение оружия, недопустимо малые дальности обнаружения малоразмерных летатель­ных аппаратов (например, БПЛА) и аппаратов, изготовленных по технологии «Стелс» и др.

В связи с изложенным, требуется поиск дополнительных информацион­ных каналов, осуществляющих своевременное обнаружение летательных аппа­ратов (ЛА) и определение их текущих координат с показателями качества близ­кими к к показателям радиолокационной системы. Таким дополнительным ин­формационным каналом может стать пассивный радиотехнический канал - комплекс радиотехнического контроля (КРТК) источников радиоизлучений (ИРИ), устанавливаемых на борту ЛА и решающих задачи радиолокации, ра­диосвязи, радионавигации, радиоэлектронной борьбы, опознавания по принци­пу «свой-чужой» и пр. Прием и соответствующая обработка пассивным радио­каналом сигналов указанных источников позволяет определять не только тип и местоположение ИРИ, но и динамику движения ЛА, а следовательно вести его трассовое сопровождение и выдачу оперативного целеуказания заинтересован­ным потребителям (авиации, зенитным средствам и пр.).

Требования к КРТК по качеству выдаваемой потребителям координатной информации можно, в общем случае, задавать исходя из требований, предъяв­ляемых к обзорным РЛС системы контроля. Например, если проанализировать требования, предъявляемые к современным РЛС ПВО, то они могут составлять по дальности обнаружения до нескольких сотен километров, по темпу выдачи информации от 1 до 20 секунд, по точности определения координат от нескольких метров до единиц километров.

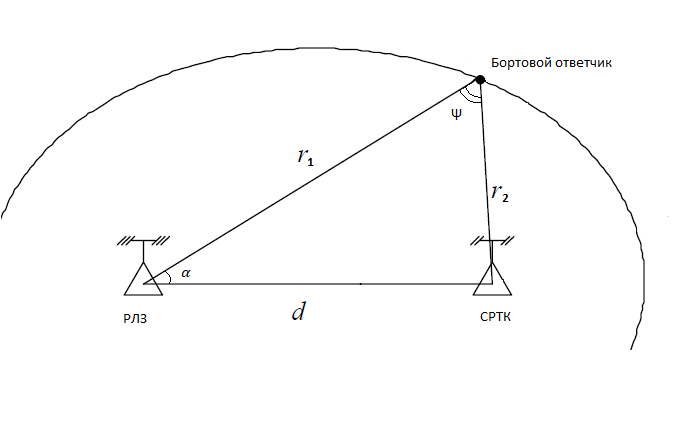
Анализ характеристик разностно-дальномерного КРТК показывает, что его приемно-анализирующая аппаратура обеспечивает обнаружение большин­ства бортовых источников, работающих в диапазоне 850-18000 МГц, на даль­ностях до 450 километров. При наличии в составе КРТК современных вычислительных средств, реализующих суммарно-дальномерное определение место­положения ЛА, и при наличии высокоскоростных линий связи между боковы­ми и центральным постами, нет принципиальных ограничений по периоду выдачи координатной информации о ЛА, не превышающему нескольких секунд. Ошибки определения прямоугольных координат КРТК определяются местопо­ложением источника в рабочей зоне системы. При достаточном соотношении сигнал/шум на входе приемных устройств среднеквадратические круговые ошибки измерения координат могут достигать 2-х километров у оси симмет­рии рабочей зоны (на удалении ~ 100км от центрального поста.) и до десяти и более километров по краям зоны. Таким образом, для обеспечения потребите­лей качественной информацией о бортовых источниках радиоизлучений (БИ) требуется повышение точности оценки их координат.

Известным способом повышения точности оценок координат движу­щихся объектов является применение теории и алгоритмов нелинейной цифро­вой фильтрации результатов единичных измерений, например, на основе фильтров Калмана. Применение подобных фильтров позволяет не только уменьшить величину текущей ошибки измерений, но оценить скорость и уско­рение движения ЛА, а также решить задачу краткосрочной экстраполяции ме­стоположения ЛА. На настоящий момент времени теория калмановской фильт­рации широко применяется при обработке радиолокационной информации. По­этому можно воспользоваться уже известными алгоритмами и при фильтрации траектории движущегося БИ (ЛА). При этом остаются прежними методики вы­бор модели движения ЛА и задание других условий адекватной траекторной оценки с помощью выбранного фильтра. Фильтры обеспечивают текущее ре­куррентное оценивание координат на к-м шаге (экстраполяцию и сглаживание) с использованием результатов предыдущих оценок на к-1, к-2,... шагах. При этом предполагается независимая оценка по каждой координате при стацио­нарных шумах измерения.

Цель работы: снижение ошибок определения координат воздушных объектов в пассивной угломерной суммарно-дальномерной системе радиотехнического контроля.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

**Угломерный суммарно-дальномерный метод**

 Рис. 1. Угломерный суммарно-дальномерный метод

Пусть в пространстве присутствуют 3 объекта: радиолокационный запросчик, ЛА с бортовым ответчиком и система радиотехнического контроля. Антенна РЛЗ вращается с детерминированной скоростью и отсылает запросный сигнал как в сторону ЛА, так и в сторону СРТК. Бортовой ответчик в ответ на запрос посылает ответный сигнал, который также принимает СРТК. Исходя из знания расстояния между РЛЗ и СРТК, а также разности времени приходов 2-х сигналов можно вычислить сумму расстояний r1 + r2, а значит можно построить первую линию положения – эллипс. А исходя из знания скорости вращения антенны(она вычисляется по разности времени прихода двух запросных сигналов) можно вычислить угол ψ, а значит получить вторую линию положения – пеленг. С помощью композиций этих линий положения можно на плоскости вычислить координаты ЛА.

**Калмановская фильтрация**

Рассмотрим принцип обработки результатов текущих измерений коорди­наты с помощью широко применяемой на практике разновидности классиче­ского фильтра Калмана - полиномиального фильтра второго порядка с эффек­тивной конечной памятью, называемого α-β - фильтром.

Пусть оценивается координата х и скорость её изменения х, тогда урав­нения для фильтрации и экстраполяции будут иметь вид:

*,*

где - результат текущего измерения координаты х в КРТК;

- экстраполированное значение координаты по результатам преды­дущего измерения;

T0 - период обновления данных от измерителя;

α и β - коэффициенты фильтра, определяющие степень доверия к измерен­ным или экстраполированным данным: если они близки к единице, то резуль­тирующие оценки координаты и скорости в большей степени определяются те­кущими измерениями; если к нулю, - экстраполированными.

Выбор коэффициентов в рассматриваемой модели фильтра имеет прин­ципиальное значение на оптимальность проводимых фильтром оценок. В об­щем случае они зависят от номера отсчета:

, .

С ростом числа наблюдений величина коэффициентов стремиться к нулю, что отражает повышение точности, но при этом фильтр становится нечувствитель­ным к изменению характера движения ЛА, несостоятельным и расходится. Та­кой способ целесообразно применять при малых к. С другой стороны для пре­дотвращения расходимости фильтра желательно применять постоянные коэф­фициенты а и р. Поэтому на основе априорного знания диапазона ошибок из­мерения и относительного постоянства функции изменения траектории на не­котором ограниченном участке движения ЛА выбираются некоторые постоян­ные ненулевые значения рассматриваемых коэффициентов. Этот путь, кроме того, обеспечивает сокращение вычислительных затрат и широко применяется при обработке радиолокационных измерений.

В нашем случае для реализации алгоритма последовательного сглажива­ния результатов измерения с выхода измерителя КРТК необходимо учесть, по крайней мере, два фактора:

1. Период обновления данных о координатах БИ зависит от характери­стик излучения бортового генератора ЛА и для задания некоторого постоянно­го периода обновления результатов измерений в фильтре возможна предварительная регуляризация (дискретизация) промежуточных измерений КРТК на коротком (системном) временном интервале.
2. Ошибки измерения координат в КРТК в значительной степени зависят, как уже отмечалось, от того, на каком участке пространственной зоны действия комплекса находится БИ. Они могут на порядок расходится в различных участ­ках зоны. Это обстоятельство требует дополнительного учета и обоснования выбора адекватных коэффициентов последовательного сглаживания фильтра.

Алгоритм нахождения оптимальных коэффициентов сглаживания должен учитывать минимизацию суммарной дисперсии случайной ошибки (ошибки измерения) и динамической (детерминированной) ошибки, вызванной возмож­ным ускорением ЛА на интервале обновления. Для устойчивой работы фильтра необходимо выполнить условие устойчивости фильтра:

При этом оптимальное соотношение коэффициентов можно получить по формуле:

Выбор коэффициента α определяется решением сложного нелинейного уравнения, поэтому в данной работе он выбирается, исходя из эмпирических суждений.

Чтобы определить зоны, в которых коэффициенты и β будут постоянны, нужно подсчитать ошибку местоопределения для каждой точки траектории ЛА.

**Ошибки местоопределения**

В позиционных способах при измерении координатно-информативных параметров (параметров радиоизлучения) возникают ошибки, величины которых зависят от правильности учета реальных условий распространения радиоволн, отношения сигнал/помеха в тракте обработки, технических параметров радиотехнического устройства координатометрии. Эти ошибки приводят к ошибкам определения параметров положения, а те в свою очередь к ошибкам линий положения и ошибкам определения координат ИРИ. Указанные ошибки могут быть систематическими и случайными.

Систематические ошибкивызывают устойчивое смещение измерений от истинного значения. Сами измерения при этом могут иметь очень малый разброс, но само значение измерения оказывается неверным независимо от точности отсчета. Систематические ошибки в принципе могут быть устранены калибровкой измерителя, т. е. предварительным определением систематических ошибок, и последующим внесением в отсчеты соответствующих поправок. Однако причины ошибок столь многочисленны и разнообразны и часто настолько трудно поддаются учету, что на практике выполнение такой калибровки становится невозможным, а внесение всех многочисленных поправок значительно затруднило использование измерителя. Часть систематических ошибок, не устраненных и не учитываемых в виде поправок, входит в случайные ошибки.

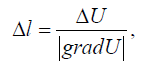
Случайные ошибкиприводят к колебаниям измерения во времени, причем частота и амплитуда этих колебаний не остаются постоянными. При проведении большого количества измерений, проведенных за большой промежуток времени, средняя случайная ошибка стремится к нулю, так как отклонения в одну и другую сторону, как правило, равновероятны. Поэтому основным способом борьбы со случайными ошибками является статистическая обработка измерений.

Наличие ошибок измерения координатно-информативных параметров при-водит к тому, что измеренная геометрическая величина в общем случае будет отличаться от истинной, вследствие чего, найденная линия положения не будет совпадать с истинной.

Расстояние между истинной и найденной линиями положения называется ошибкой линии положения. Для определения зависимости ошибки линии Δ*l* положения от ошибки геометрической величины Δ*U* (параметра положения) используют теорию скалярного поля. Семейство линий положения можно рассматривать как линии уровня плоского скалярного поля геометрической величины *U*. Из теории поля известно, что

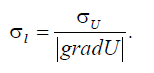


Заменяя дифференциалы конечными приращениями, получаем следующую формулу для расчета ошибки линии положения



Из этого следует, что ошибка линии положений прямо пропорциональна ошибке геометрической величины. Поэтому закон распределения ошибок линий положений будет таким же, как и у ошибок измерения геометрической величины.

В большинстве случаев закон распределения ошибок геометрической вели-чины является нормальным законом распределения, который характеризуется средней квадратической ошибкой . Поэтому ошибка линии положения также вполне характеризуется средней квадратической ошибкой линии положения



Коэффициент пропорциональности между и зависит от вида линии положения и взаимного расположения искомой точки и точек, в которых находятся измерители. Рассмотрим особенности вывода ошибок линий положения для различных способов координатометрии.

Геометрические аспекты определения ошибки линии положения при угломерном способе КМ представлены на рис. 2.

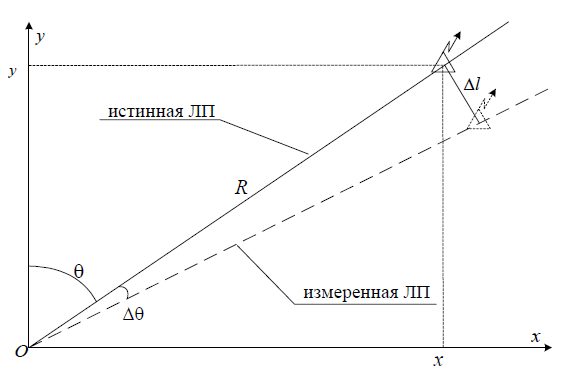
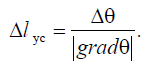
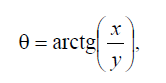


Рис. 2. Ошибка линии положения при угломерном способе КМ

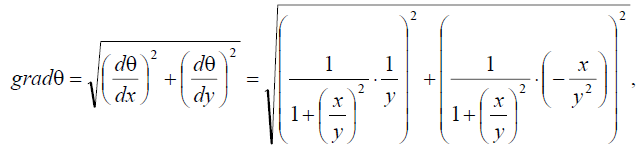
Согласно предыдущему выражению, для угломерного способа КМ можно записать



Выразив азимут θ через декартовы координаты *x* и *y*



и представив θ*grad* в виде совокупности частных производных



после соответствующих преобразований получим



Следовательно, выражение для ошибки линии положения угломерной системы примет вид



Переходя к средней квадратической ошибке получим



Анализ этого выражения показывает, что ошибка линии положения при угломерном способе КМ зависит не только от ошибки пеленгования, но и от расстояния между ИРИ и измерителем, возрастая с увеличении этого расстояния.

Аналогичным образом можно определить ошибку линии положения для дальномерного способа КМ (рис. 3).

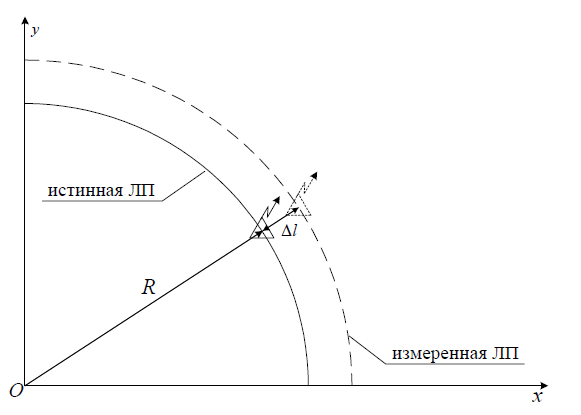


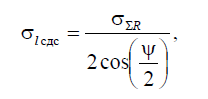
Рис. 3. Ошибка линии положения при дальномерном способе КМ

Выражение для средней квадратической ошибки линии положения при дальномерном способе КМ будет иметь вид



Анализ этого выражения показывает, что при дальномерном способе КМ ошибка ЛП зависит только от ошибки измерения дальности и не зависит от топологии системы координатометрии.

При суммарно-дальномерном способе КМ ошибка ЛП примет вид



где – средняя квадратическая ошибка измерения суммы расстояний, ψ – угол под которым видна суммарно-дальномерная база из точки расположения ИРИ (рис. 4).

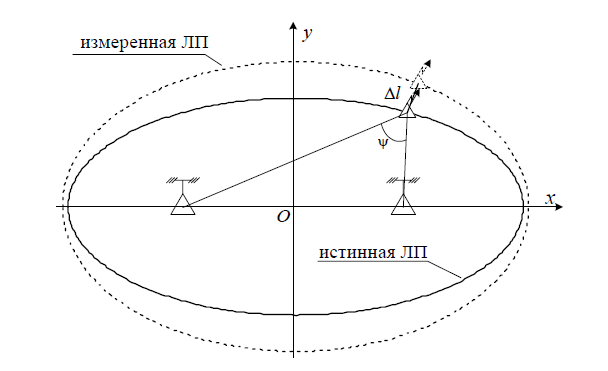


Рис. 4. Ошибка линии положения при суммарно-дальномерном способе КМ

В суммарно-дальномерном способе, ошибка ЛП зависит от дальности и направления на ИРИ, но при этом носит другой характер – при уменьшении угла ψ ошибка также уменьшается.

Обобщив полученные выводы по ошибкам линий положения можно заключить следующее.

1. Для угломерного способа ошибка линии положения , при неизменной ошибке пеленга увеличивается с увеличением расстояния *R* до ИРИ.

2. Для дальномерного способа ошибка линии положения , равна дальномерной ошибке , и, следовательно, зависит от расстояния, только в том случае, если такую зависимость имеет дальномерная ошибка.

4. Для суммарно-дальномерного способа целесообразно уменьшать расстояние между фиксированными точками, так как наименьшая ошибка линии положения будет при ψ = 0o, т. е. при расстоянии между измерителями равном нулю.

5. Коэффициент пропорциональности между ошибкой линии положения и ошибкой измеряемой геометрической величины для угломерных и дальномерных систем не зависит от направления.

**Основные подходы в определении точности позиционных способов координатометрии ИРИ**

При реализации позиционных методов местоположение ИРИ определяется как точка пересечения двух или более линий положения. Погрешности, возникающие при определении каждой линии положения, приводят к тому, что вычисленное положение объекта отличается от истинного.

Если предположить, что объект находится на больших расстояниях от точек наблюдения, и что погрешности определения линий положения много меньше этих расстояний, тогда семейство линий положения, соответствующее различным значениям измеряемых параметров положения около местоположения ИРИ, можно заменить отрезками параллельных прямых независимо от формы линий положения.

Допустим, что истинное положение ИРИ находится в точке *М* пересечения линий положения *AB* и *CD*, соответствующих истинным значениям измеряемых параметров положения (рис. 5). Прямые *A*'*B*' и *C*'*D*' определяют линии положения с учетом погрешностей измерения, *М*' - вычисленное с погрешностями местоположение ИРИ.

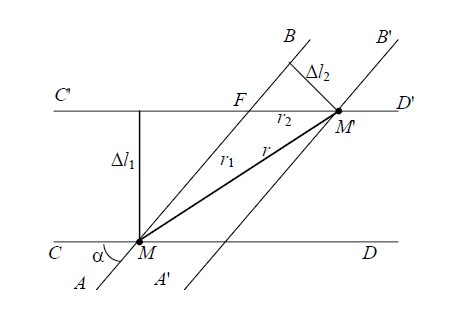
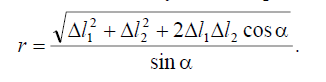


Рис. 5. Обобщенный подход к выводу ошибки определения координат

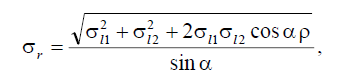
Линии положения *АВ* и *СD* пересекаются под углом α, который называется углом засечки. Точки *М* и *М*' смещены относительно друг друга на расстояние *r*, которое называется погрешностью местоопределения. Величины и представляют погрешности определения линий положения *АВ* и *СD*. Обозначив и , получими . Тогда из треугольника *МM'F* согласно теореме косинусов находим, что



Следовательно погрешность определения местоположения ИРИ для единичного измерения будет рассчитываться следующим образом

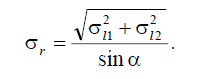


Т.к. погрешности и представляют собой случайные величины, то и погрешность определения местоположения ИРИ случайна. При условии α=const получим выражение для среднеквадратического значения погрешности МО



где ρ – коэффициент взаимной корреляции погрешностей определения линий положения *АВ* и *СD*; и – среднеквадратические значения погрешностей определения линий положения.

При независимых измерениях линий положения *AB* и *CD*, что, как правило, встречается на практике, ρ = 0, тогда выражение для расчета погрешности ОМП значительно упрощается

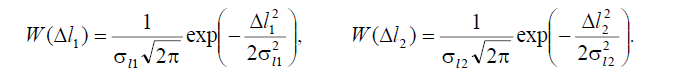


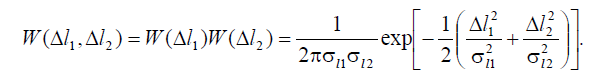
Анализ предыдущих 2-х формул показывает, что среднеквадратическое значение погрешности определения местоположения ИРИ зависит от среднеквадратических значений погрешностей измерения линий положения *AB* и *CD*, а также от угла, под которым пересекаются эти линии и их взаимной корреляционной функции. Максимальная точность при заданных и достигается в том случае, когда линии положения пересекаются под углом 90o.

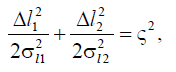
Заметим, что погрешность *r* определения местоположения объекта не распределена по нормальному закону даже тогда, когда ошибки и представляют собой нормальные случайные величины. В ряде случаев при расчетах приближенная оценка погрешностей определения местоположения ИРИ на основе среднеквадратического значения погрешности МО является недостаточной. При этом используются более полные статистические характеристики погрешностей системы МО и, в частности, рассматривается эллипс рассеяния.

Оценим вероятность того, что расчетное значение местоположения объекта находится в некоторой области, окружающей его истинное местоположение.

Предположим, что случайные погрешности и определения каждой линии положения независимы и подчиняются нормальному закону распределения. Тогда плотности вероятности имеют вид

 Совместная плотность вероятностей погрешностей и при этом будет определятся следующим образом

 Приравняв показатель степени выражения к постоянному числу, получим уравнение линии, на которой плотность вероятностей W( ), характеризующая погрешность определения местоположения ИРИ, одинакова, т.е.



где ς – некоторая постоянная величина.

Видно, что линия постоянной плотности вероятностей W( ) представляет эллипс в косоугольной системе координат, оси которой совпадают с нормалями к линиям положения, а начало – с истинным положением ИРИ. Эллипс, определяемый соотношением последним, называют **эллипсом погрешностей**, или **эллипсом рассеяния**.

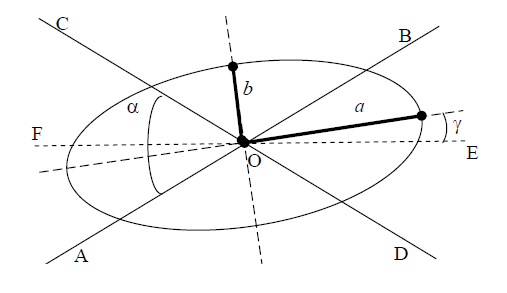
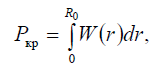


Рис. 6. Эллипс рассеяния

Однако выбор в качестве фигуры ошибок эллипса не всегда представляется возможным. Часто используют в качестве характеристики ошибки местоопределения расстояние *R* между центром эллипса рассеяния и засечкой, т. е. выбирают в качестве фигуры ошибок круг с радиусом R0. Вероятность нахождения ИРИ в пределах круга радиусом R0 определяется по формуле



где *W(r)* – распределение плотности вероятностей случайной ошибки *r*.

Площадь круга Sкр = π, ограничивающего область с вероятностью местонахождения засечки Pкр, больше площади минимального эллипса Sэ = π*ab*, дающего ту же вероятность Pэ = Pкр = P (Sкр > Sэ). Однако при углах засечки 30o < α < 150o это отличие площадей не слишком велико, поэтому при решении большинства практических задач радиус окружности R0 можно найти по приближенной формуле



Например, при Pкр = 0,63 имеем радиус окружности R0 = Rкр = , который называется среднеквадратической круговой ошибкой местоопределения.

ПОСТАНОВКА ЗАДАЧИ

Опираясь на теорию, необходимо было написать программу, которая:

1. моделирует траекторию движения ЛА;
2. вычисляет для каждой точки в пространстве и, в частности, для этой траектории, ошибку определения местоположения;
3. в соответствии с этими ошибками, зашумляет истинную траекторию движения ЛА, имитируя координаты, полученные с помощью СРТК;
4. пропускает эти координаты через α-β фильтр;
5. в результате, получает отфильтрованные траектории движения.

Соответственно, каждый из пунктов можно рассматривать, как отдельную задачу.

РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ

1. Была смоделирована траектория движения ЛА(см. рис. 8)
2. Реализована возможность построить окружность ошибок для каждой точки пространства, внутри которой объект находится с вероятностью 95%. Результаты приведены на рисунке ниже. Видно, что ошибка растет по мере удаления от СРТК и РЛЗ.

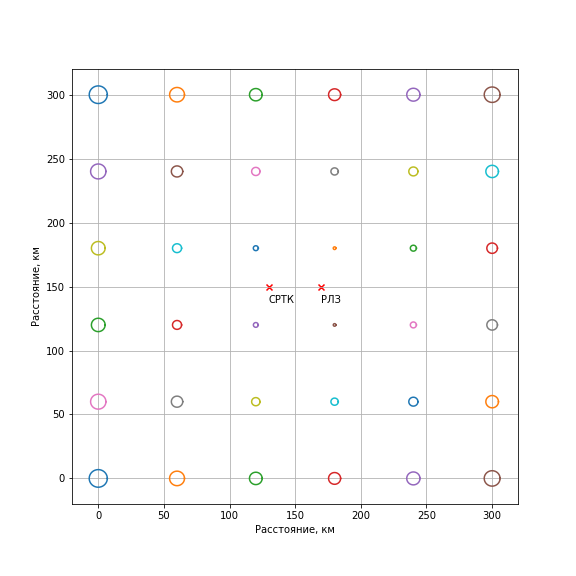


Рис. 7 Поле ошибок.

1. Для каждой точки исходной траектории вычислена ошибка измерения координат, и в соответствии с этими ошибками были сымитированы координаты, полученные с помощью СРТК. Результаты приведены на рисунке 8.

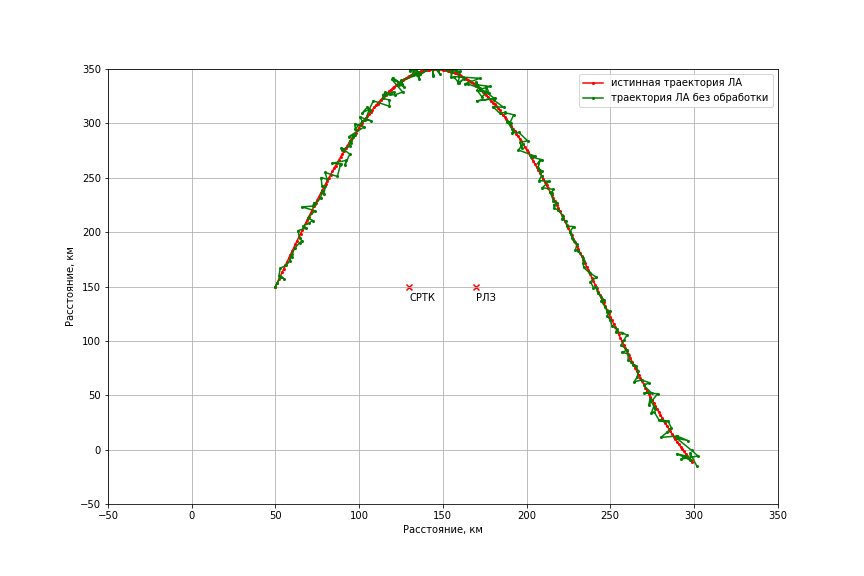


Рис.8 Траектории движения ЛА

1. Для использования α-β фильтра нужно выбрать единственные параметры, которыми можно управлять – коэффициенты α и β. Коэффициенты фильтра точно вычисляются с помощью нелинейного уравнения, решение которого заняло бы большое количество ресурсов, поэтому в данной работе они выбирается методом научного подбора. Так как подбирать коэффициенты для каждой точки сложно, было решено пойти другими путями. Первый из них – задать постоянные коэффициенты для всего пространства, второй – разбить пространство на определенное количество зон и для каждой из них задать свои коэффициенты. Далее приведены коэффициенты, вычисленные опытным путем и наилучшим образом снижающие ошибку.

Коэффициенты для 3-х зон:

* 0 – 75 км: α=0.37, β= 0.08;
* 75 – 150 км: α=0.16, β= 0.01;
* >150 км: α=0.11, β= 0.006.

Коэффициенты, не меняющиеся в пространстве:

* α=0.25, β= 0.035.

1. Получены отфильтрованные траектории движения ЛА

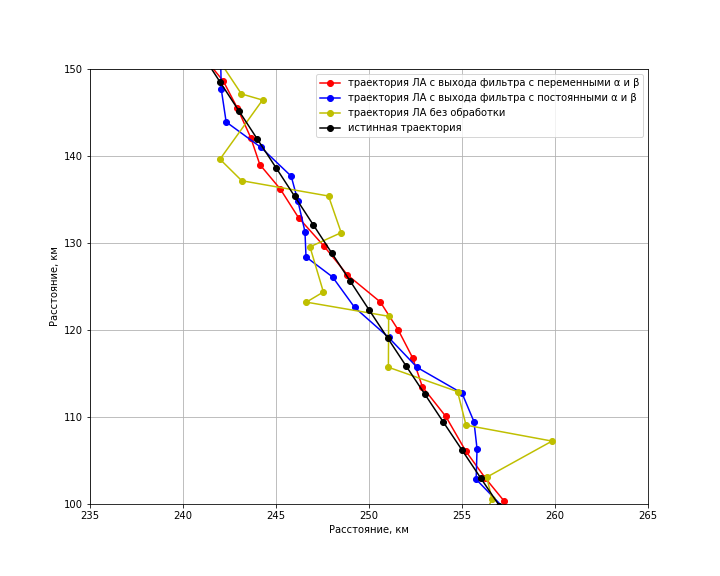


Рис.9 Фрагменты траекторий ЛА

Заметно, что траектории с выхода фильтра намного лучше приближают истинную траекторию движения объекта. Также можно заметить, что у точек с выхода фильтра с постоянными коэффициентами разброс больше, чем у точек с выхода фильтра с переменными коэффициентами.

Чтобы наглядней продемонстрировать результат работы был построен график среднеквадратичного отклонения от исходной траектории в зависимости от номера точки(график построен для точек на траекторий показанных выше). Всплеск вначале объясняется настройкой фильтра. Расчет СКО проводился по ансамблю из 100 опытов.

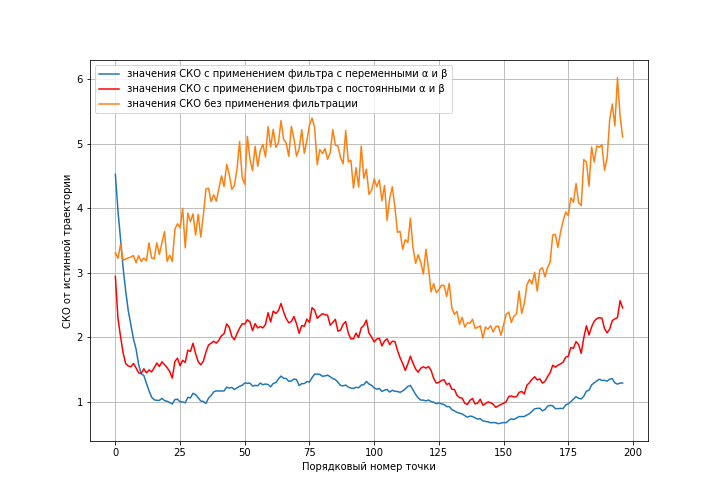


Рис. 10 Зависимость СКО от номера точки

ЗАКЛЮЧЕНИЕ

Применение алгоритмов последовательного сглаживания позволяет улучшить точность определения координат ЛА. Благодаря оптимальному выбору коэффициентов α и β, снижение уровня ошибок происходит на больших и на малых дальностях контроля. В среднем, точность определения местоположения на больших дальностях улучшилась на 76%, а на малых – на 67%, что является хорошим результатом в достижении главной цели работы и означает то, что потребителю будут выдаваться более точное целеуказание.

СПИСОК ЛИТЕРАТУРЫ

1. Дворников С. В. «Теоретические основы координатометрии источников радиоизлучения», Санкт-Петербург, 2006 г.
2. Панасюк Ю. Н.Обработка радиолокационной информации в радиотехнических системах : учебное пособие / Ю. Н. Панасюк, А. П. Пудовкин. – Тамбов : Изд-во ФГБОУ ВПО «ТГТУ», 2016.
3. Коновалов А. А. Основы траекторной обработки радиолокационной информации: в 2 ч. СПб: Изд-во СПбГЭТУ «ЛЭТИ» , 2014.

ПРИМЕЧАНИЯ

Ниже приведен код программы, выполняющей все расчеты

**import** **numpy** **as** **np**

**import** **math**

**import** **matplotlib.pyplot** **as** **plt**

**from** **matplotlib** **import** ticker, cm

**from** **matplotlib** **import** patches

**from** **scipy.spatial.distance** **import** euclidean

**import** **warnings**

warnings.simplefilter('ignore')

In [ ]:

*# создем систему координат и две позиции РЛС*

scale = 300

alpha0 = 0

base = 40

num\_point = 300

X = np.linspace(0, scale, scale\*10)

Y = np.linspace(0, scale, scale\*10)

center\_base\_x = 150

center\_base\_y = 150

RS1\_x, RS1\_y = center\_base\_x - base/2, 0 + center\_base\_y

RS2\_x, RS2\_y = center\_base\_x + base/2, 0 + center\_base\_y

In [ ]:

**def** ellipse\_dist(x0, y0, summa, base, num\_point):

a = summa/2

c = base/2

b = math.sqrt((a\*\*2 - c\*\*2))

tetta = np.linspace(0, 2\*math.pi, num\_point)

x = a \* np.cos(tetta) + x0

y = b \* np.sin(tetta) + y0

**return** x, y

**def** ellips\_err(x0, y0, a, b, num\_point):

tetta = np.linspace(0, 2\*math.pi, num\_point)

x = a \* np.cos(tetta) + x0

y = b \* np.sin(tetta) + y0

**return** x, y

**def** circle\_err(x0, y0, num\_point, sigma1, sigma2, x\_ellip, y\_ellip, x\_pel, y\_pel, P\_e):

pi = math.pi

\_ksi = ksi(P\_e)

alpha = get\_alpha(x\_ellip, y\_ellip, x0, y0, num\_point, x\_pel, y\_pel)

**if** (alpha > pi/2):

alpha = pi - alpha

sigma = sigma\_full(sigma1, sigma2, alpha)

tetta = np.linspace(0, 2\*pi, num\_point)

R\_ck = \_ksi \* sigma

x = R\_ck \* np.cos(tetta) + x0

y = R\_ck \* np.sin(tetta) + y0

**return** x, y, R\_ck

**def** peleng(x0, y0, alpha, alpha0, scale):

tetta = (alpha - alpha0) \* math.pi/180

x = np.linspace(0, scale, scale\*10)

y = math.tan(tetta) \* x

**return** x+x0, y+y0

**def** peleng(x0, y0, x\_tar, y\_tar, scale):

vec\_x = x\_tar-x0

vec\_y = y\_tar-y0

tetta = math.atan(vec\_y/vec\_x)

**if**(vec\_x<0):

x = np.linspace(0, -scale, scale\*10)

**else**:

x = np.linspace(0, scale, scale\*10)

y = math.tan(tetta) \* x

**return** x+x0, y+y0

**def** psi(coor\_point, coor\_RS1, coor\_RS2):

a = euclidean(coor\_point, coor\_RS1)

b = euclidean(coor\_point, coor\_RS2)

c = euclidean(coor\_RS1, coor\_RS2)

psi = math.acos((a\*\*2+b\*\*2-c\*\*2)/(2\*a\*b))

**return** psi

**def** sigma\_tau(sigma\_tau0, R):

**return** sigma\_tau0 \* R

**def** sigma\_summ(sigma\_delta\_tau, C):

**return** sigma\_delta\_tau \* C

**def** sigma\_pel(sigma\_delta\_phi, R):

**return** sigma\_delta\_phi \* R

**def** sigma\_ellip(sigma\_delta\_sum, angle\_psi):

**return** sigma\_delta\_sum/(2\*math.cos(angle\_psi))

**def** get\_tangle\_ellip(x\_ellip, y\_ellip, x\_tar, y\_tar, num\_point):

diff1, diff2 = np.diff([x\_ellip, y\_ellip])

error = 2\*math.pi/num\_point \* 100

num = -1

**for** i **in** range(len(x\_ellip)-1):

**if**(abs(x\_ellip[i]-x\_tar)<error **and** abs(y\_ellip[i]-y\_tar)<error):

num = i

**break**

**if** num >= 0:

**return** [diff1[num], diff2[num]]

**else**:

**return** 0, 0

**def** get\_tangle\_peleng(x\_pel, y\_pel):

**return** [(x\_pel[10]-x\_pel[0]), (y\_pel[10]-y\_pel[0])]

**def** get\_alpha(x\_ellip, y\_ellip, x\_tar, y\_tar, num\_point, x\_pel, y\_pel):

tan\_el = get\_tangle\_ellip(x\_ellip, y\_ellip, x\_tar, y\_tar, num\_point)

tan\_pel = get\_tangle\_peleng(x\_pel, y\_pel)

alpha = math.acos(np.dot(tan\_el, tan\_pel)/(np.linalg.norm(tan\_el) \* np.linalg.norm(tan\_pel)))

**if** alpha>math.pi/2:

alpha = math.pi-alpha

**return** alpha

**def** ksi(P):

**return** math.sqrt(-math.log1p(-P))

**def** gamma(sigma1, sigma2, alpha):

gamma = math.atan((sigma1\*\*2 - sigma2\*\*2)/(sigma1\*\*2 + sigma2\*\*2)\*math.tan(alpha))/2

**return** gamma

**def** half\_axis\_ellip\_error(angle\_ksi, sigma1, sigma2, alpha):

a = 2 \* angle\_ksi \* sigma1 \* sigma2 / math.sqrt(sigma1\*\*2 + sigma2\*\*2 - \

math.sqrt((sigma1\*\*2 + sigma2\*\*2)\*\*2 - \

4\*sigma1\*\*2 \* sigma2\*\*2 \* math.sin(alpha)))

b = 2 \* angle\_ksi \* sigma1 \* sigma2 / math.sqrt(sigma1\*\*2 + sigma2\*\*2 + \

math.sqrt((sigma1\*\*2 + sigma2\*\*2)\*\*2 - \

4\*sigma1\*\*2 \* sigma2\*\*2 \* math.sin(alpha)))

**return** a, b

**def** get\_error(x0, y0, RS1\_x, RS1\_y, RS2\_x, RS2\_y, sigma\_delta\_tau0, sigma\_delta\_phi):

R1 = euclidean([x0, y0], [RS1\_x, RS1\_y])

R2 = euclidean([x0, y0], [RS2\_x, RS2\_y])

R\_sum = R1+R2

sigma\_delta\_tau = sigma\_tau(sigma\_delta\_tau0, R\_sum)

sigma\_delta\_sum = sigma\_summ(sigma\_delta\_tau, C)

angle\_psi = psi([x0, y0], [RS1\_x, RS1\_y], [RS2\_x, RS2\_y])

sigma\_delta\_ellip = sigma\_ellip(sigma\_delta\_sum, angle\_psi)

sigma\_delta\_pel = sigma\_pel(sigma\_delta\_phi, R2)

**return** sigma\_delta\_ellip, sigma\_delta\_pel, R\_sum

**def** sigma\_full(sigma1, sigma2, alpha):

**return** math.sqrt(sigma1\*\*2 + sigma2\*\*2)/math.sin(alpha)

**def** get\_err(x0, y0, num\_point, sigma1, sigma2, x\_ellip, y\_ellip, x\_pel, y\_pel, P\_e):

pi = math.pi

alpha = get\_alpha(x\_ellip, y\_ellip, x0, y0, num\_point, x\_pel, y\_pel)

**if** (alpha > pi/2):

alpha = pi - alpha

sigma = sigma\_full(sigma1, sigma2, alpha)

**return** sigma

In [ ]:

*# зададим начальные ошибки*

sigma\_delta\_tau0 = 0.000000005

sigma\_delta\_phi = 1. \* math.pi/180

P\_e = 0.95

C = 300000

In [ ]:

*# построим эллипс ошибок для точки*

x\_test, y\_test = 200, 200

R1 = euclidean([x\_test, y\_test], [RS1\_x, RS1\_y])

R2 = euclidean([x\_test, y\_test], [RS2\_x, RS2\_y])

R\_sum = R1+R2

sigma\_delta\_tau = sigma\_tau(sigma\_delta\_tau0, R\_sum)

sigma\_delta\_sum = sigma\_summ(sigma\_delta\_tau, C)

angle\_psi = psi([x\_test, y\_test], [RS1\_x, RS1\_y], [RS2\_x, RS2\_y])

sigma\_delta\_ellip = sigma\_ellip(sigma\_delta\_sum, angle\_psi)

sigma\_delta\_pel = sigma\_pel(sigma\_delta\_phi, R2)

ellip = ellipse\_dist(center\_base\_x, center\_base\_y, R\_sum, base, num\_point)

pelng = peleng(RS2\_x, RS2\_y, x\_test, y\_test, scale)

circl\_err = circle\_err(x\_test, y\_test, num\_point, sigma\_delta\_ellip, sigma\_delta\_pel,

ellip[0], ellip[1], pelng[0], pelng[1], P\_e)

tan\_pel = get\_tangle\_peleng(pelng[0], pelng[1])

i = [1,0]

phi = math.acos(np.dot(tan\_pel, i)/(np.linalg.norm(i) \* np.linalg.norm(tan\_pel)))

f, ax = plt.subplots(figsize=(8,8))

plt.scatter([RS1\_x, RS2\_x], [RS1\_y, RS2\_y], c='red', marker='x')

plt.plot(circl\_err[0], circl\_err[1])

plt.plot(ellip[0], ellip[1])

plt.plot(pelng[0], pelng[1])

plt.plot([x\_test, x\_test+10\*get\_tangle\_ellip(ellip[0], ellip[1], x\_test, y\_test, num\_point)[0]],

[y\_test, y\_test+10\*get\_tangle\_ellip(ellip[0], ellip[1], x\_test, y\_test, num\_point)[1]], c='red')

plt.plot([x\_test, x\_test+10\*get\_tangle\_peleng(pelng[0], pelng[1])[0]],

[y\_test, y\_test+10\*get\_tangle\_peleng(pelng[0], pelng[1])[1]], c='red')

plt.ylim([0,scale])

plt.xlim([0,scale])

plt.grid()

plt.show()

In [ ]:

x\_line = np.linspace(0, num\_point, 6)

y\_line = np.linspace(0, num\_point, 6)

x\_test, y\_test = np.meshgrid(x\_line, y\_line)

f, ax = plt.subplots(figsize=(8,8))

plt.scatter([RS1\_x, RS2\_x], [RS1\_y, RS2\_y], c='red', marker='x')

ax.text(RS1\_x, RS1\_y-13, 'СРТК')

ax.text(RS2\_x, RS2\_y-13, 'РЛЗ')

plt.ylim([-20,scale+20])

plt.xlim([-20,scale+20])

plt.grid()

plt.xlabel('Расстояние, км')

plt.ylabel('Расстояние, км')

**for** i **in** range(len(x\_line)):

**for** j **in** range(len(y\_line)):

x0, y0 = x\_test[i,j], y\_test[i,j]

sigma1, sigma2, R\_sum = get\_error(x0, y0, RS1\_x, RS1\_y, RS2\_x, RS2\_y, sigma\_delta\_tau0, sigma\_delta\_phi)

ellip = ellipse\_dist(center\_base\_x, center\_base\_y, R\_sum, base, num\_point)

pelng = peleng(RS2\_x, RS2\_y, x0, y0, scale)

circl\_err = circle\_err(x0, y0, num\_point, sigma1, sigma2,

ellip[0], ellip[1], pelng[0], pelng[1], P\_e)

plt.plot(circl\_err[0], circl\_err[1])

*# print(circl\_err[2])*

plt.savefig('C:**\\**Users**\\**Владимир**\\**Desktop**\\**по фк и мп**\\**поле ошибок.png')

plt.show()

In [ ]:

**def** extrapolate(speed, curr, T0):

a = np.array([[1, T0], [0, 1]])

b = np.array([curr, speed])

**return** a @ b

**def** estimate(curr, extra, alpha):

**return** extra + alpha \* (curr - extra)

**def** speed(curr, last, beta, T0, last\_speed=0):

**return** last\_speed + beta / T0 \* (curr-last)

**def** coeff\_beta(alpha):

**return** alpha\*\*2/(2-alpha)

In [ ]:

X = np.linspace(-100, 400, 200)

XX1, XX2 = np.meshgrid(X, X)

Z = np.sqrt((XX1 - center\_base\_x)\*\*2 + (XX2 - center\_base\_y)\*\*2) //75

In [ ]:

R\_zone = np.array([0, 112.5, 187.5, 262.5])

alpha\_zone = np.array([0.4, 0.13, 0.11, 0.06])

beta\_zone = coeff\_beta(alpha\_zone)

a = 0.32

b = coeff\_beta(a)

T0 = 0.02

Num\_point = 200

t = np.linspace(0, 1.3 \* math.pi, Num\_point)

x = np.arange(-150, 150, 1.5)

y = 200 \* np.sin(t)

x1 = x + center\_base\_x + 30

y1 = y + center\_base\_y

error\_mass\_noise = np.array([])

mass\_extrapolate\_coor = np.zeros((1, Num\_point, 2))

mass\_extrapolate\_coor1 = np.zeros((1, Num\_point, 2))

mass\_noise\_coor = np.zeros((1, Num\_point, 2))

**for** j **in** range(100):

last\_coor = np.array([x1[0]+np.random.normal(0, 4), y1[0]+np.random.normal(0, 4)])

curr\_coor = np.array([x1[1]+np.random.normal(0, 4), y1[1]+np.random.normal(0, 4)])

R = euclidean([x1[1], y1[1]], [center\_base\_x, center\_base\_y])

alpha = alpha\_zone[(abs(R\_zone - R)).argmin()]

beta = beta\_zone[(abs(R\_zone - R)).argmin()]

speed\_coor = np.array([speed(curr\_coor, last\_coor, beta, T0)])

speed\_coor1 = np.array([speed(curr\_coor, last\_coor, b, T0)])

coor = np.array([last\_coor, curr\_coor])

noise\_coor = np.array([last\_coor, curr\_coor])

coor1 = np.array([last\_coor, curr\_coor])

**for** i **in** range(2, len(x1)):

x0, y0 = x1[i], y1[i]

sigma1, sigma2, R\_sum = get\_error(x0, y0, RS1\_x, RS1\_y, RS2\_x, RS2\_y, sigma\_delta\_tau0, sigma\_delta\_phi)

ellip = ellipse\_dist(center\_base\_x, center\_base\_y, R\_sum, base, num\_point)

pelng = peleng(RS2\_x, RS2\_y, x0, y0, scale)

err = get\_err(x0, y0, num\_point, sigma1, sigma2,

ellip[0], ellip[1], pelng[0], pelng[1], P\_e)

x0 +=np.random.normal(0, err)

y0 +=np.random.normal(0, err)

R = euclidean([x0, y0], [center\_base\_x, center\_base\_y])

alpha = alpha\_zone[(abs(R\_zone - R)).argmin()]

beta = beta\_zone[(abs(R\_zone - R)).argmin()]

extra = extrapolate(speed\_coor[-1], coor[-1], T0)

extra1 = extrapolate(speed\_coor1[-1], coor1[-1], T0)

last\_coor = curr\_coor

curr\_coor = np.array([x0, y0])

speed\_coor = np.append(speed\_coor, [speed(curr\_coor, last\_coor, beta, T0, speed\_coor[-1])], axis=0)

speed\_coor1 = np.append(speed\_coor1, [speed(curr\_coor, last\_coor, b, T0, speed\_coor[-1])], axis=0)

coor = np.append(coor, [estimate(curr\_coor, extra[0], alpha)], axis=0)

coor1 = np.append(coor1, [estimate(curr\_coor, extra1[0], a)], axis=0)

noise\_coor = np.append(noise\_coor, [[x0, y0]], axis=0)

**if** j == 1:

error\_mass\_noise = np.append(error\_mass\_noise, err)

mass\_extrapolate\_coor = np.append(mass\_extrapolate\_coor, [coor], axis=0)

mass\_extrapolate\_coor1 = np.append(mass\_extrapolate\_coor1, [coor1], axis=0)

mass\_noise\_coor = np.append(mass\_noise\_coor, [noise\_coor], axis=0)

mass\_extrapolate\_coor\_1 = np.zeros\_like(mass\_extrapolate\_coor)

mass\_extrapolate\_coor1\_1 = np.zeros\_like(mass\_extrapolate\_coor1)

mass\_noise\_coor\_1 = np.zeros\_like(mass\_noise\_coor)

In [ ]:

**for** i **in** range(mass\_extrapolate\_coor.shape[1]):

mass\_extrapolate\_coor\_1[1:, i] = (mass\_extrapolate\_coor[1:, i] - np.array([x1[i], y1[i]]))

mass\_extrapolate\_coor1\_1[1:, i] = (mass\_extrapolate\_coor1[1:, i] - np.array([x1[i], y1[i]]))

mass\_noise\_coor\_1[1:, i] = (mass\_noise\_coor[1:, i] - np.array([x1[i], y1[i]]))

error\_extrapolate\_mass = np.array([np.sqrt(mass\_extrapolate\_coor\_1[1:, i, 0].std()\*\*2 +\

mass\_extrapolate\_coor\_1[1:, i, 1].std()\*\*2) **for** i **in** range(mass\_extrapolate\_coor.shape[1])])

error\_extrapolate\_mass1 = np.array([np.sqrt(mass\_extrapolate\_coor1\_1[1:, i, 0].std()\*\*2 +\

mass\_extrapolate\_coor1\_1[1:, i, 1].std()\*\*2) **for** i **in** range(mass\_extrapolate\_coor.shape[1])])

error\_noise\_mass = np.array([np.sqrt(mass\_noise\_coor\_1[1:, i, 0].std()\*\*2 +\

mass\_noise\_coor\_1[1:, i, 1].std()\*\*2) **for** i **in** range(mass\_extrapolate\_coor.shape[1])])

In [ ]:

plt.plot(error\_extrapolate\_mass)

plt.plot(error\_extrapolate\_mass1, color='r')

plt.plot(error\_noise\_mass)

plt.show()

In [ ]:

f, ax = plt.subplots(figsize=(10, 7))

ax.set\_xlabel('Порядковый номер точки')

ax.set\_ylabel('СКО от истинной траектории')

ax.plot(error\_extrapolate\_mass[3:], label=('значения СКО с применением фильтра c переменными ' +

unicodedata.lookup("GREEK SMALL LETTER ALPHA") +

' и ' + unicodedata.lookup("GREEK SMALL LETTER BETA")))

plt.plot(error\_extrapolate\_mass1[3:], color='r', label=('значения СКО с применением фильтра c постоянными ' +

unicodedata.lookup("GREEK SMALL LETTER ALPHA") +

' и ' + unicodedata.lookup("GREEK SMALL LETTER BETA")))

ax.grid()

ax.plot(error\_noise\_mass[3:], label='значения СКО без применения фильтрации')

ax.legend()

plt.savefig('C:**\\**Users**\\**Владимир**\\**Desktop**\\**по фк и мп**\\**СКО.png')

plt.show()

In [ ]:

(error\_extrapolate\_mass[25:112].sum() + error\_extrapolate\_mass[187:200].sum())/(error\_noise\_mass[25:112].sum() + error\_noise\_mass[187:200].sum())

In [ ]:

T0 = 0.02

R\_zone = np.array([0, 112.5, 187.5, 262.5])

alpha\_zone = np.array([0.37, 0.16, 0.11, 0.06])

beta\_zone = coeff\_beta(alpha\_zone)

a = 0.33

b = coeff\_beta(a)

fig, ax = plt.subplots(figsize=(12, 8))

*# ax.contourf(XX1, XX2, Z, cmap='CMRmap\_r')*

t = np.linspace(0, 1.3 \* math.pi, 250)

x = np.arange(-100, 150, 1)

y = 200 \* np.sin(t)

x1 = x + center\_base\_x

y1 = y + center\_base\_y

last\_coor = np.array([x1[0], y1[0]])

curr\_coor = np.array([x1[1], y1[1]])

R = euclidean([x1[1], y1[1]], [center\_base\_x, center\_base\_y])

alpha = alpha\_zone[(abs(R\_zone - R)).argmin()]

beta = beta\_zone[(abs(R\_zone - R)).argmin()]

speed\_coor = np.array([speed(curr\_coor, last\_coor, beta, T0)])

speed\_coor1 = np.array([speed(curr\_coor, last\_coor, b, T0)])

coor = np.array([last\_coor, curr\_coor])

noise\_coor = np.array([last\_coor, curr\_coor])

coor1 = np.array([last\_coor, curr\_coor])

**for** i **in** range(2, len(x1)):

x0, y0 = x1[i], y1[i]

sigma1, sigma2, R\_sum = get\_error(x0, y0, RS1\_x, RS1\_y, RS2\_x, RS2\_y, sigma\_delta\_tau0, sigma\_delta\_phi)

ellip = ellipse\_dist(center\_base\_x, center\_base\_y, R\_sum, base, num\_point)

pelng = peleng(RS2\_x, RS2\_y, x0, y0, scale)

circl\_err = circle\_err(x0, y0, num\_point, sigma1, sigma2,

ellip[0], ellip[1], pelng[0], pelng[1], P\_e)

err = get\_err(x0, y0, num\_point, sigma1, sigma2,

ellip[0], ellip[1], pelng[0], pelng[1], P\_e)

*# print(err)*

x0 +=np.random.normal(0, err)

y0 +=np.random.normal(0, err)

R = euclidean([x0, y0], [center\_base\_x, center\_base\_y])

alpha = alpha\_zone[(abs(R\_zone - R)).argmin()]

beta = beta\_zone[(abs(R\_zone - R)).argmin()]

extra = extrapolate(speed\_coor[-1], coor[-1], T0)

extra1 = extrapolate(speed\_coor1[-1], coor1[-1], T0)

last\_coor = curr\_coor

curr\_coor = np.array([x0, y0])

speed\_coor = np.append(speed\_coor, [speed(curr\_coor, last\_coor, beta, T0)], axis=0)

speed\_coor1 = np.append(speed\_coor1, [speed(curr\_coor, last\_coor, b, T0)], axis=0)

coor = np.append(coor, [estimate(curr\_coor, extra[0], alpha)], axis=0)

coor1 = np.append(coor1, [estimate(curr\_coor, extra1[0], a)], axis=0)

noise\_coor = np.append(noise\_coor, [[x0, y0]], axis=0)

*# ax.plot(coor[:, 0], coor[:, 1], color='b', label='траектория ЛА с выхода фильтра')*

*# ax.plot(coor1[:, 0], coor1[:, 1], color='y')*

ax.plot(x1[:], y1[:], color='r', label='истинная траектория ЛА', marker='o', markersize=2)

ax.plot(noise\_coor[:, 0], noise\_coor[:, 1], color='g', label='траектория ЛА без обработки', marker='o', markersize=2)

ax.grid()

ax.scatter([RS1\_x, RS2\_x], [RS1\_y, RS2\_y], c='red', marker='x')

ax.text(RS1\_x, RS1\_y-13, 'СРТК')

ax.text(RS2\_x, RS2\_y-13, 'РЛЗ')

ax.set\_xlabel('Расстояние, км')

ax.set\_ylabel('Расстояние, км')

ax.set\_xlim([-50, 350])

ax.set\_ylim([-50, 350])

ax.legend()

tt = np.linspace(0, 2\*math.pi, 200)

xx = 75 \* np.cos(tt) + center\_base\_x

yy = 75 \* np.sin(tt) + center\_base\_y

xxx = 150 \* np.cos(tt) + center\_base\_x

yyy = 150 \* np.sin(tt) + center\_base\_y

*# xxxx = 150 \* np.cos(tt) + center\_base\_x*

*# yyyy = 150 \* np.sin(tt) + center\_base\_y*

*# ax.plot(xx, yy)*

*# ax.plot(xxx, yyy)*

plt.savefig('C:**\\**Users**\\**Владимир**\\**Desktop**\\**по фк и мп**\\**шум траектория.png')

plt.show()

In [ ]:

coeff\_beta(0.25)

In [ ]:

**import** **unicodedata**

fig, ax = plt.subplots(figsize=(10, 8))

*# ax.contourf(XX1, XX2, Z, cmap='CMRmap\_r')*

ax.plot(coor[:, 0],

coor[:, 1],

color='r',

label=('траектория ЛА с выхода фильтра c переменными ' +

unicodedata.lookup("GREEK SMALL LETTER ALPHA") +

' и ' + unicodedata.lookup("GREEK SMALL LETTER BETA")),

marker='o')

ax.plot(coor1[:, 0],

coor1[:, 1],

color='b',

label=('траектория ЛА с выхода фильтра c постоянными ' +

unicodedata.lookup("GREEK SMALL LETTER ALPHA") +

' и ' + unicodedata.lookup("GREEK SMALL LETTER BETA")),

marker='o')

ax.plot(noise\_coor[:, 0], noise\_coor[:, 1], color='y', label='траектория ЛА без обработки', marker='o')

ax.plot(x1, y1, color='black', label='истинная траектория', marker='o')

ax.grid()

ax.scatter([RS1\_x, RS2\_x], [RS1\_y, RS2\_y], c='red', marker='x')

*# ax.text(RS1\_x, RS1\_y+13, 'РТС 1')*

*# ax.text(RS2\_x, RS2\_y-13, 'РТС 2')*

ax.set\_xlim([235, 265])

ax.set\_ylim([100, 150])

ax.legend()

ax.set\_xlabel('Расстояние, км')

ax.set\_ylabel('Расстояние, км')

plt.savefig('C:**\\**Users**\\**Владимир**\\**Desktop**\\**по фк и мп**\\**детальная траектория.png')

plt.show()