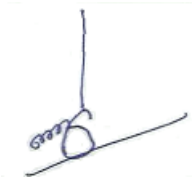
Министерство образования и науки Российской Федерации
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальныйисследовательский университет)

На правах рукописи



ТИН ПХОН ЧЖО

СИСТЕМА УПРАВЛЕНИЯ ПРИОРИТЕТНЫМ ОБСЛУЖИВАНИЕМ ВОЗДУШНЫХ СУДОВ ПРИ ЗАХОДЕ НА ПОСАДКУ И ПАССАЖИРОВ В АЭРОПОРТУ ПОСЛЕ ПРИЛЕТА

Специальность 05.13.01 – системный анализ, управление и обработка информации(информатика, управление и вычислительная техника)

Диссертация на соискание ученой степени
доктора технических наук

Научный консультант: доктор технических наук,
профессорзаслуженный деятель
науки РФ
Лебедев Георгий Николаевич

Москва 2015

Содержание

Введение

Существуют ситуации, когда летящие произвольным курсом самолеты должны попасть на заданную линию пути или в заданный строй. К таким случаям относится, в частности, ситуация внезапного изменения условий посадки на различные взлетно-посадочные полосы (ВПП) по метеорологическим или техническим причинам.

Одна из сложных задач управления комплексом самолетов – внезапная переориентация самолетов, находящихся в зонах ожидания на круге и на посадку на удаленный запасной аэродром. При возникновении такой необходимости большая группа самолетов должна лететь с сохранением безопасных расстояний между ними, а значит реконфигурирована в строй для перелета с последующим перестроением в эшелон ожидания посадки. В зависимости от начального расположения самолетов относительно трассы перелета, их сбор в строй выполняется как сбор на промежуточном круге, так и сбор на трассе, при формировании неупорядоченного их множества, отвечающего условиям полета в заданном направлении при ограничении фактических и прогнозируемых расстояний между ними.

Таким образом, естественно указать в текущий момент времени очередность или приоритет в обслуживании каждого судна и последовательно вводить ихв заданных эшелон, проверяя при этом возможность соблюдения гарантированной безопасности полета [1].

В данной работе этот подход предложено реализовать путем вычисления динамических приоритетов в виде некоторых количественных оценок, учитывающихудаленность воздушного судна(ВС) от заданной трассы, ожидаемую его близость к судам, движущимися уже в эшелоне, а также зависимость от оставшегося запаса топлива. При этом, если очередной приоритет мал, то это означает существование такого риска несоблюдения безопасности совместного движения в эшелоне, при котором происходит

отказ от попытки введения судна в эшелон или строй, и дается команда ухода на повторный круг.

Задача автоматизации управления оперативным планированием прилета на аэродромы Внуково, Домодедово, Шереметьево, а также организации оптимального и эффективного процесса выпуска воздушных судов с этих трех аэродромов уже сегодня является актуальной. Без решения данной задачи невозможно достичь количественных и качественных показателей, заложенных в Программе развития гражданской авиации в РФ. Основная проблема заключается в несовершенстве структуры воздушного пространства Московского узлового диспетчерского района (Московского аэроузла) (рис 1.1), которая может существенно меняться с изменением хотя бы одного посадочного курса, которых всего 8 (по два на каждую ВПП). Для каждого посадочного курса необходимо сформировать собственную, оптимальную по ряду критериев структуру воздушного пространства. Только в этом случаевозможно эффективное разведение потоков прилета и вылета, исключение непредсказуемого так называемого в гражданской авиации «векторения», что позволит существенно увеличить интенсивность воздушного движения без ущерба безопасности полетов[1-4,65-67].

Так, для конфигурации посадочных курсов 194°, 316°, 065° структура маршрутов вылета и прилета в Московском аэроузле представлена на рис 1.2. Схемы прилета в данном варианте имеют по одному стандартному маршруту (STAR) с каждого из 4-х направлений (Юг,Запад,Север,Восток) по каждому аэродрому и содержат в себе элементы «тонкого» регулирования потока типа «тромбон», стандартные схемы вылета (SID) разведены с соответствующими схемами прилета по высотам и географическому асположению[6].

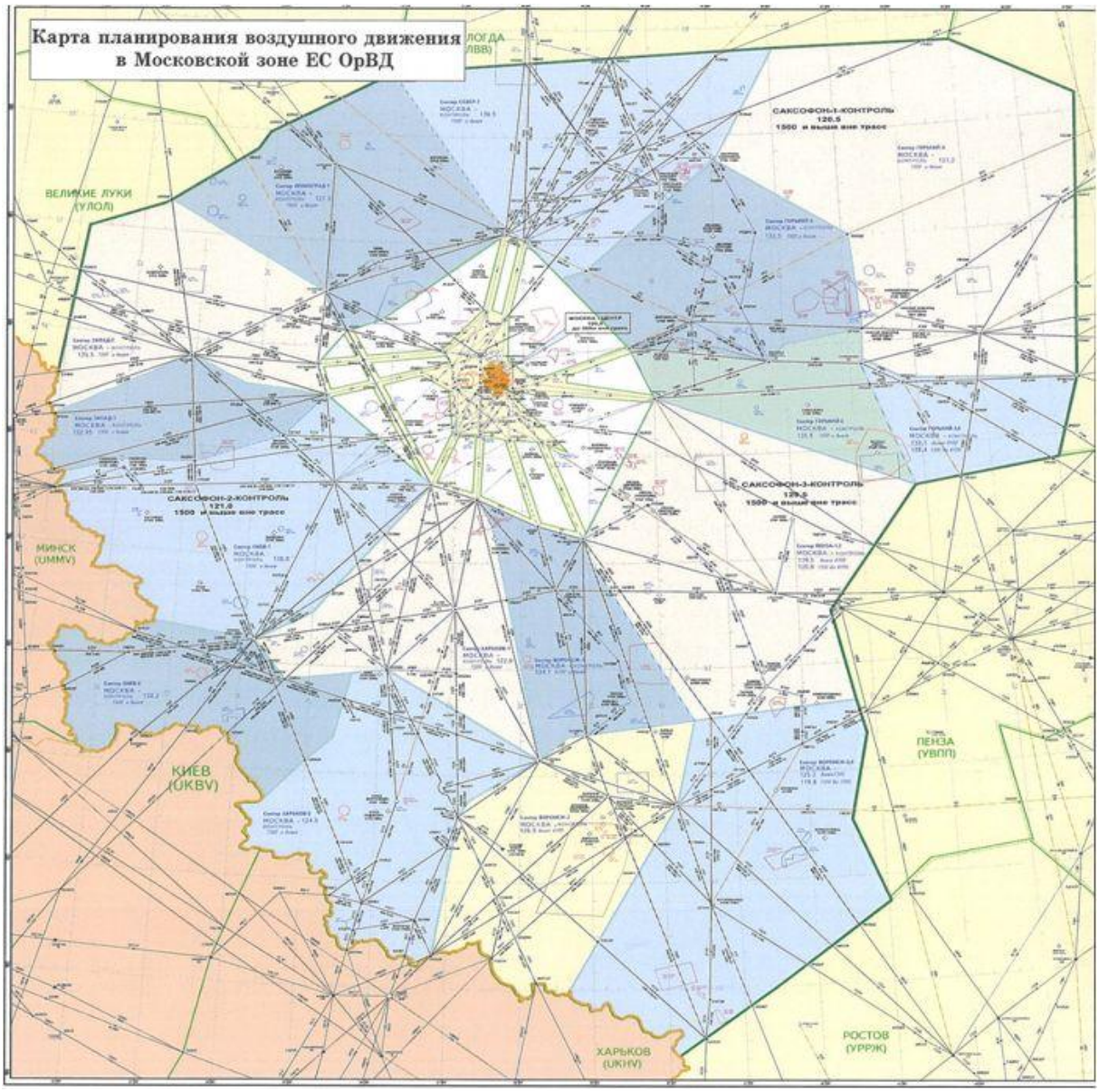


Рис 1.1.Московский аэроузел, включающий аэродромы Внуково, Домодедово, Шереметьево

Практическое решение данной задачи требует учета многочисленных факторов внешней среды, основным из которых являются погодные условия. Так, изменение или неустойчивое направление ветра на взлетно-посадочной полосе может привести к перемене посадочного курса хотя-бы одного из трех аэродромов, что вносит существенные изменения структуры стандартных маршрутов, показанных на рис .1. 2.

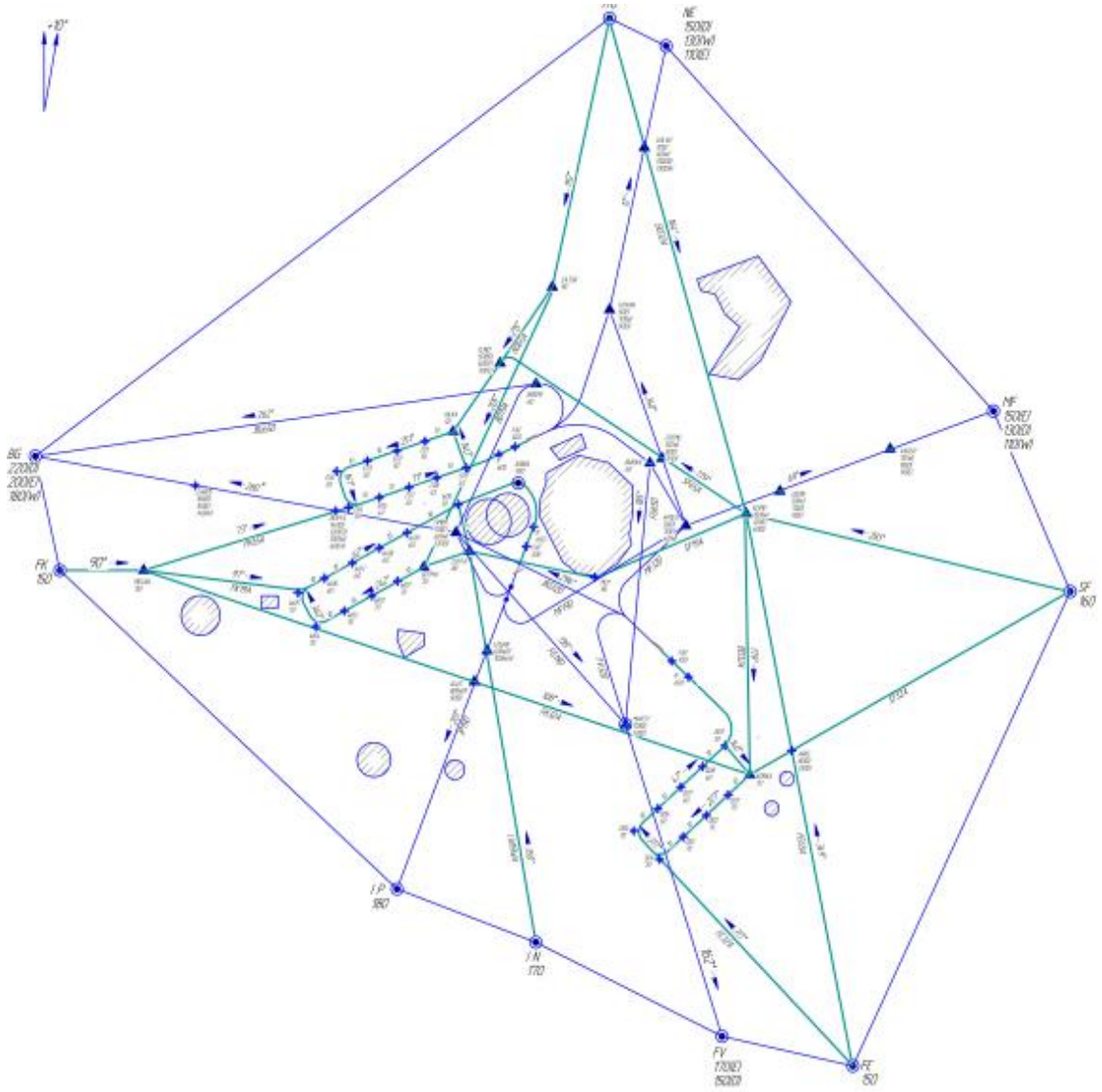


рис 1.2. Организация прилета-вылета для конфигурации посадочных курсов 194/316/065 (А)

На рис1.3 представлена структура маршрутов вылета и прилета для конфигурации посадочных курсов 14°, 316°, 245°. Как и в предыдущем случае, схемы прилета и вылета содержат по одному стандартному маршруту с четырех направлений с аналогичными элементами «тонкого» регулирования потока и бесконфликтными траекториями. Однако, структура на рис 1.3 существенно отличается от предыдущей структуры, изображенной на рис 1.2.Подобные изменения структуры стандартных маршрутов вылета-прилета вследствие изменения конфигурации посадочных курсов довольно часто встречается на практике.

В условиях автоматизированного оперативного планирования потока движения воздушных судов возникает необходимость изменения заданного ранее условия движения.

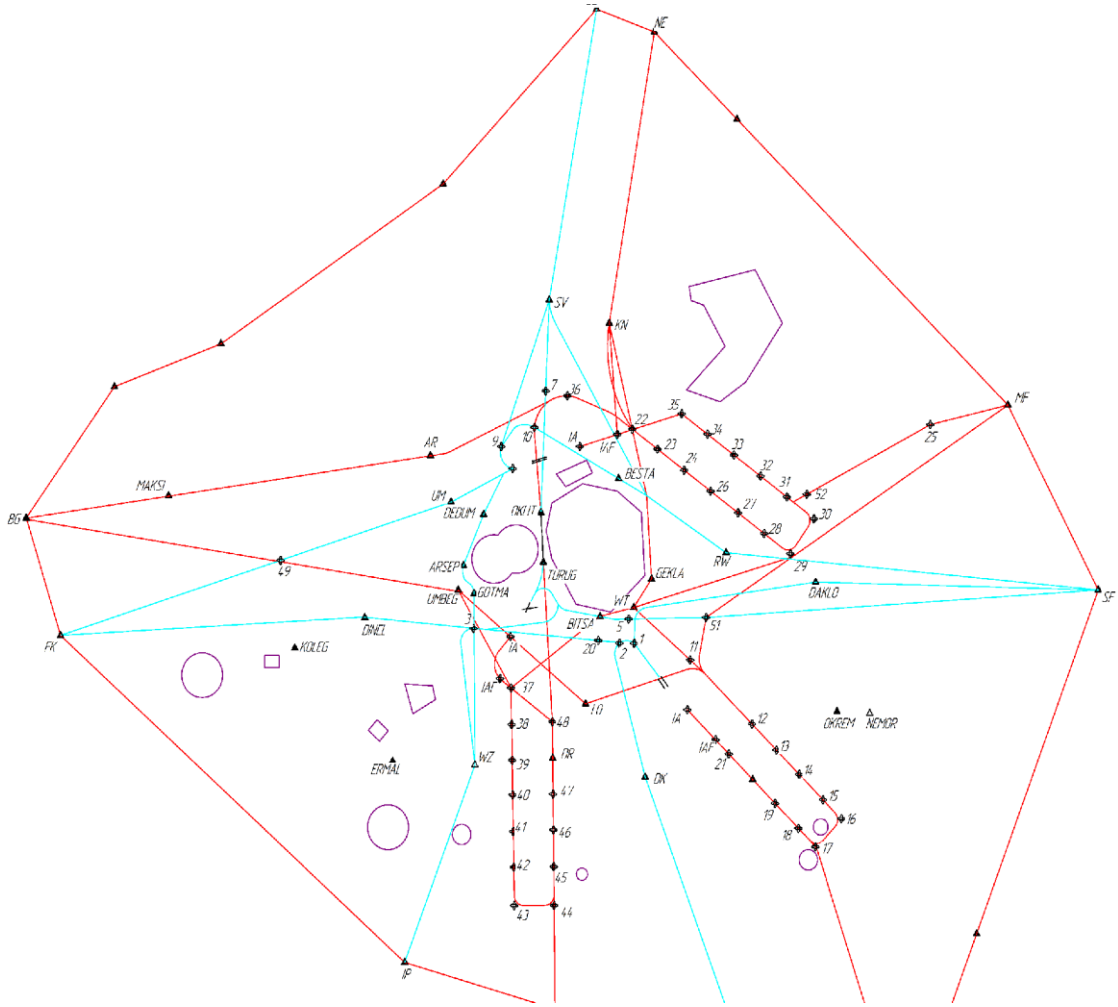


рис 1.3. Организация прилета-вылета для конфигурации посадочных курсов 14/316/245 (B)

Данная задача может интерпретироваться как задача «автоматизации векторения» т.е. задача изменения ранее заданных и отчасти выполненных условий посадки путем задания дополнительных, неформализованных заявок с целью оптимальной адаптации к новым условиям.

Данную процедуру необходимо применять и для аварийных самолётов с малым запасом топлива, с техническими неисправностями, с больным на борту и т.п. Таким воздушным судам нужен быстрый заход на посадку, у них

нет времени навекторение. В экстремальной ситуации необходимы самые удобные условия для посадки авиалайнера, особенно в части направления ветра[3] .

В настоящей работе предлагается решить задачу «автоматизации векторения» путем вычисления динамических приоритетов в виде некоторых количественных оценок, учитывающих удаленность ВС от соответствующей точки РК(fap), ожидаемое его положение в очереди, а также запас топлива. При этом чрезвычайно важным является обеспечение безопасности полета как совместного движения в очереди, который требует управляющих воздействий в виде задержек ВС путем задания дополнительных ограничений по скорости или маневров с изменением курса полета, так и прежде всего создания наиболее благоприятных условий для аварийных самолетов, имеющих малый запас топлива и частично неисправное техническое состояние [3,4]. Задаче оптимального управления и контроля безопасности движения воздушных судов при их приоритетном обслуживании во время прилета посвящена данная диссертационная работа.

С учетом сложности и масштабности поставленной задачи в работе предложено следующее ее поэтапное решение:

- вначале при синтезе оптимального управления устанавливается относительная значимость показателей экономичности полета и его безопасности, исходя из оценки стоимости расхода топлива при попадании на трассу и гораздо большей стоимости возможного ущерба при опасном сближении судов;

- на втором этапе определяется состав посадочных курсов ВПП, на которые при данном ветре разрешается сесть;

- на третьем главном этапе с помощью теории оптимального управления для каждого судна определяется их приоритет захода на посадку

для каждой трассы, после чего формируются списки судов для каждой трассы;

- на завершающем этапе суда, попавшие в трассу, подвергаются оперативному контролю безопасности их попутного движения в эшелоне, а также определяетсяоптимальная длина тромбона для части судов, имеющих минимальный приоритет;

- наконец, после прилета необходимо решить задачу приоритетного обслуживания пассажиров в аэропорту с учетом внеочередного обслуживания пассажиров с аварийного самолета.

Целью диссертационной работы является повышение экономичности и безопасности полетов воздушных судов при их заходе на посадку при внезапном изменении направления ветра.

Объектом исследования является система управления воздушным движением при оперативном планировании прилета на аэродромы Московского аэроузла. Предметом исследования являются методы оптимального управления и принятия альтернативных решений при приоритетном обслуживании судов, имеющих аварийный запас топлива и технические неисправности, а также пассажиров в аэропорту после прилета.

На защиту выносятся следующие научные положения:

1. Единый параметрический критерий оценки экономичности и безопасности управления полетом.
2. Алгоритм назначения динамических приоритетов для каждого воздушного судна при заходе на посадку по любой из заданных трасс.
3. Процедура оперативного контроля безопасности попутного движения воздушных судов в эшелоне.
4. Методика вычисления оптимальной длины очереди воздушных судов в тромбоне по критерию экономичности их захода на посадку.

5. Методика расчета числа каналов бесприоритетного и приоритетного обслуживания пассажиров в аэропорту после прилета.
6. Результаты моделирования на ЭВМ, подтвердившие эффективность предложенного подхода.

Научная новизна полученных результатов состоит в следующем:

1. Предложенный критерий эффективности управления учитывает в линейной свертке как экономичность, так и безопасность полета, при этом весовые коэффициенты значимости обеих показателей найдены с помощью решения обратной задачи линейного программирования при использовании признанных правильными примеров поведения авиадиспетчерской службы.
2. Алгоритм назначения динамических приоритетов воздушных судов является главным научным результатом, полученным на основе теории оптимального управления и учитывающем близость судна к трассе, направление его полета, запас оставшегося топлива и близость с соседними судами при возможном входе в эшелон. Алгоритм позволяет формировать приоритетные списки судов для каждой трассы, и в случае их ограниченного числа - списки очередей в соответствующие тромбоны для каждой трассы, чтобы потом воспользоваться окном и прилететь на тот же аэродром позднее.
3. Процедура оперативного контроля безопасности попутного движения судов в эшелоне сформирована с помощью определения специальной функции риска, вычисленной с помощью используемого в динамическом программировании уравнения Беллмана и, главное, учитывающего не только дистанцию между судами, но и скорости их полета и их изменения.
4. Методика вычисления длины очереди судов в тромбоне использует не только теорию массового обслуживания, но методы параметрической оптимизации, применив критерий минимума общего расхода топлива

всех судов при заходе на посадку. Кроме того, принципиально новым научным результатом является, в отличие от бесприоритетных СМО, получение формул расчета вероятных состояний системы внеочередного приоритетного обслуживания аварийных самолетов, и с помощью этих формул – расчет средней стоимости израсходованного топлива всех судов.

5. Методика расчета необходимого числа каналов приоритетного обслуживания пассажиров в аэропорту учитывает текущую интенсивность прилета самолетов, степень их аварийности и реальную взаимопомощь между каналами при переводе части пассажиров из одного канала в освободившийся, что отличает найденные оценки от идеального случая расчетов с помощью известных формул Эрланга.

Методы исследования. При исследовании поставленных в диссертации задач использовались теория дифференциальных уравнений, теория автоматического регулирования, методы параметрической оптимизации, динамического программирование и принцип максимума Понтрягина из теории оптимального управления, методы теории массового обслуживания. При моделировании динамической системы управления использовался программный пакет MATLAB и C++.

Практическая ценность работы прежде всего состоит в том, что в ней сделана попытка оптимизировать расходы топлива для решения задачи «векторения» воздушных судов из-за погодных условий, но с обязательным соблюдением гарантированной безопасности как с точки зрения учитываемых запасов топлива, так и при соблюдении нужной дистанции между соседними воздушными судами, что подтверждено актами о внедрении. Также предложенный подход был использован при выполнении лабораторных работ по дисциплине «Современные методы теории управления» в рамках магистерской подготовки на кафедре 301 МАИ по учебному направлению «Управление и информационные технологии в

технических системах», а также в тренажерном центре МГТУ ГА при подготовке авиадиспетчеров.

Ценность работы для науки и практики состоит в том, что в ней на основе теории оптимального управления сформирован алгоритм назначения динамических приоритетов воздушных судов, учитывающих совместно их пространственное и техническое состояние, что позволяет рационально распределять их по трассам при заходе на посадку.

Достоверность полученных результатов подтверждена математическим моделированием на ЭВМ системы приоритетного обслуживания судов и пассажиров и использованием при синтезе нужных алгоритмов научно – обоснованных методов параметрической оптимизации, теории массового обслуживания и теории оптимального управления, в первую очередь динамического программирования.

Личный вклад автора состоит в проведении анализа известных систем управления воздушным движением, разработке алгоритмов управления приоритетным обслуживанием судов, формировании методик расчета нужных вероятностных характеристик стоимости и безопасности полетов, личном участии в моделировании на ЭВМ и подготовке основных публикаций.

Глава 1. Анализ функционирования известных систем управления воздушным движением. Общая постановка задачи

1.1 Анализ функционирования известных систем управления воздушным движением

Общеизвестно, что большинство авиакатастроф (36%) происходит на завершающей стадии полета, когда ВС приближается к аэропорту, а летный экипаж управляет заходом на посадку, выравнивает ВС по осевой линии взлетно-посадочной полосы, совершает снижение и саму посадку. Данная стадия полета, при которой совершаются особенно важные процедуры, считается самой сложной и опасной, поэтому требует особенного внимания, концентрации и профессионализма. При посадке крайне важным является совместная работа пилотов, тщательное наблюдение за параметрами, особенно при неблагоприятных погодных условиях, и принятие экипажем правильного решения. Секунды промедления и неспособность членов экипажа договориться может привести к катастрофе. Поэтому им нужна интеллектуальная поддержка в виде технических средств автоматизации принятия решений, вырабатываемых в виде подсказки наземной авиадиспетчерской службы.

Несмотря на то, что самым популярным способом посадки ВС является посадка по приборам (ILS), пилот обязан знать и уметь выполнять все возможные методы – уход на второй круг, визуальный заход на посадку, заход с использованием оборудования всенаправленных дальномерного и азимутального радиомаяков (VOR), заход с использованием приводной радиостанции (NDB), ведь никогда нельзя знать заранее, каким будет установившийся режим при посадке.

Существуют различные способы, повышающие уровень безопасности при заходе на посадку и посадке. Рассмотрим их:

- Визуальный заход осуществляется по ППП (правила полета по приборам), возможен только при хорошей видимости; пилот ориентируется по естественной линии горизонта и обязан видеть аэропорт или самолет, заходящий перед ним на посадку.
- Уход на второй круг – маневр, выполняемый для выравнивания самолета при невозможности безопасного выполнения посадки. Решение об уходе на второй круг принимается пилотом на предпосадочной прямой.
- Посадка по приборам (ILS) – это система наземного оборудования, обеспечивающая четкое управление самолетом при заходе на посадку и посадке, используя комбинации радиосигналов и во многих случаях интенсивную световую матрицу для обеспечения безопасности посадки в сложных метеорологических условиях, таких как низкая предельная высота или ограниченная видимость из-за снега, дождя, тумана.
- Использование приводной радиостанции (NDB) – один из старейших способов, позволяющий пилотам зайти на посадку, полагаясь лишь на летное оборудование, не достигая предписанных минимумов в условиях плохой видимости.
- Использование оборудования всенаправленных дальномерного и азимутального радиомаяков (VOR/DME) – самый интересный и сложный способ захода на посадку, при котором первая часть захода на посадку совершается автоматикой до достижения предписанной высоты, и вторая часть выполняется визуально, управляя самолетом на низкой высоте.
- Поскольку с повышением регулярности полетов экипажам ВС все чаще приходится выполнять заход на посадку в сложных метеоусловиях, принимаются меры по оборудованию аэродромов современными системами посадки. На ВС устанавливают специальное оборудование, позволяющее выполнять полуавтоматический и автоматический заход на посадку[5]. Это требует от летного состава умения выполнять заход на посадку по приборам.

Для поддержания требуемого уровня профессиональной подготовки пилоты систематически проходят тренировки на тренажерах, а также в реальных сложных погодных условиях.

- Посадка ВС на аэродроме производится на ВПП**,** имеющую как правило два направления захода на посадку. Обычно посадку выполняют при встречном и встречно-боковом ветре. При этом для каждого типа ВС боковая составляющая ветра не должна превышать заданного предельного значения. Курс, соответствующий рабочему направлению ВПП, называется посадочным. Заход на посадку выполняют по установленной для данного аэродрома схеме.

Курсо-глиссадная система является наиболее распространенной системой захода на посадку на крупных и оживленных аэродромах. Однако наряду с этим большие скорости полета и требуемая высокая точность выполнения траекторного движения самолета при решении ряда тактических и навигационных задач возможны только при использовании средств автоматического и директорного управления. Прежде всего, вся сложность самолетовождения по заданной траектории в условиях больших скоростей полета вызвана необходимостью восприятия летчиком множества параметров движения самолета, их контроля и принятия логических решений для выработки действий органами управления. В ряде ответственных режимов полета, таких, например, как заход на посадку в сложных метеорологических условиях, при ограниченности времени на принятие решений может произойти изменение заданной траектории полета, потеря координации управления, что нередко приводит к летным происшествиям [3].

Эти факты свидетельствуют о том, что воспринимаемая информация о параметрах полета, представляемая летчику, оказываетсядля него настолько сложной, что он не справляется с пилотированием самолета. Особенно это

проявляется при выполнении групповых действий, когда соседние суда летят в эшелоне слишком близко, либо их маршруты пересекаются. Эти ситуации возникают при появлении вблизи аэродромов большого числа прилетающих самолетов.

В частности, учитывая пиковую загруженность в секторах обслуживающих район Шереметьево, применение маршрутов прибытия и управление полета на их основе является актуальным решением данной проблемы.Основным принципом построения схем прилета и вылета заключается в создании бесконфликтных маршрутов, которые в свою очередь разделены на направления.

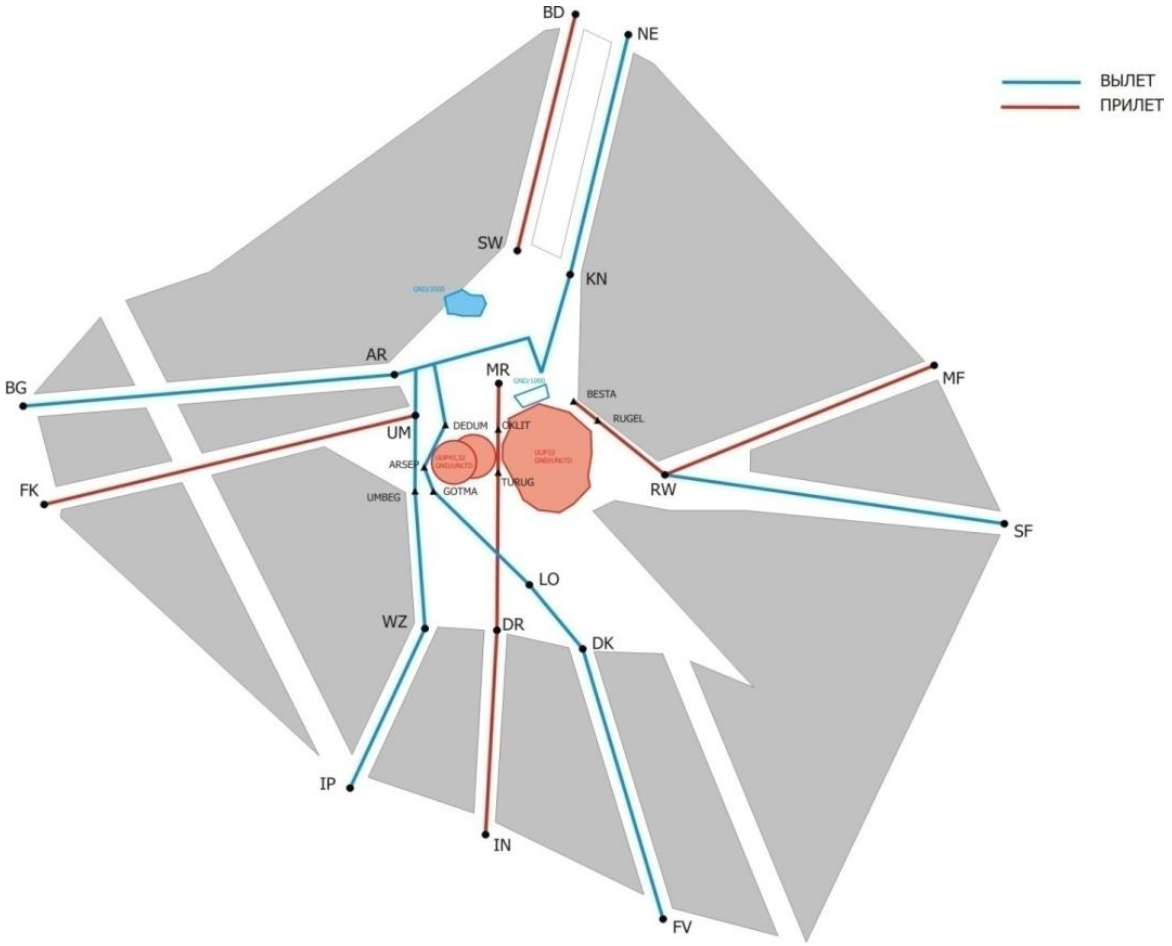


Рис. 1.4. Схема бесконфликтных маршрутов прилета в Московском аэроузле

В процессе проектирования бесконфликтных маршрутов для аэропорта Шереметьево применяются следующие условия:

1. Рабочие ВПП основных аэропортов Московской воздушной зоны

- Шереметьево – 066 градусов
- Домодедово – 316 градусов
- Внуково - 058 градусов

2. Приоритет вылетов и прилетов осуществляется для западного направления.

3. Маршруты разрабатываются для номинальных значений градиентов набора высоты и снижения ВС.

4. Ограничение ВС по путевой скорости, применяемое от земли до эшелона 3000 метров, равно500 км/ч.

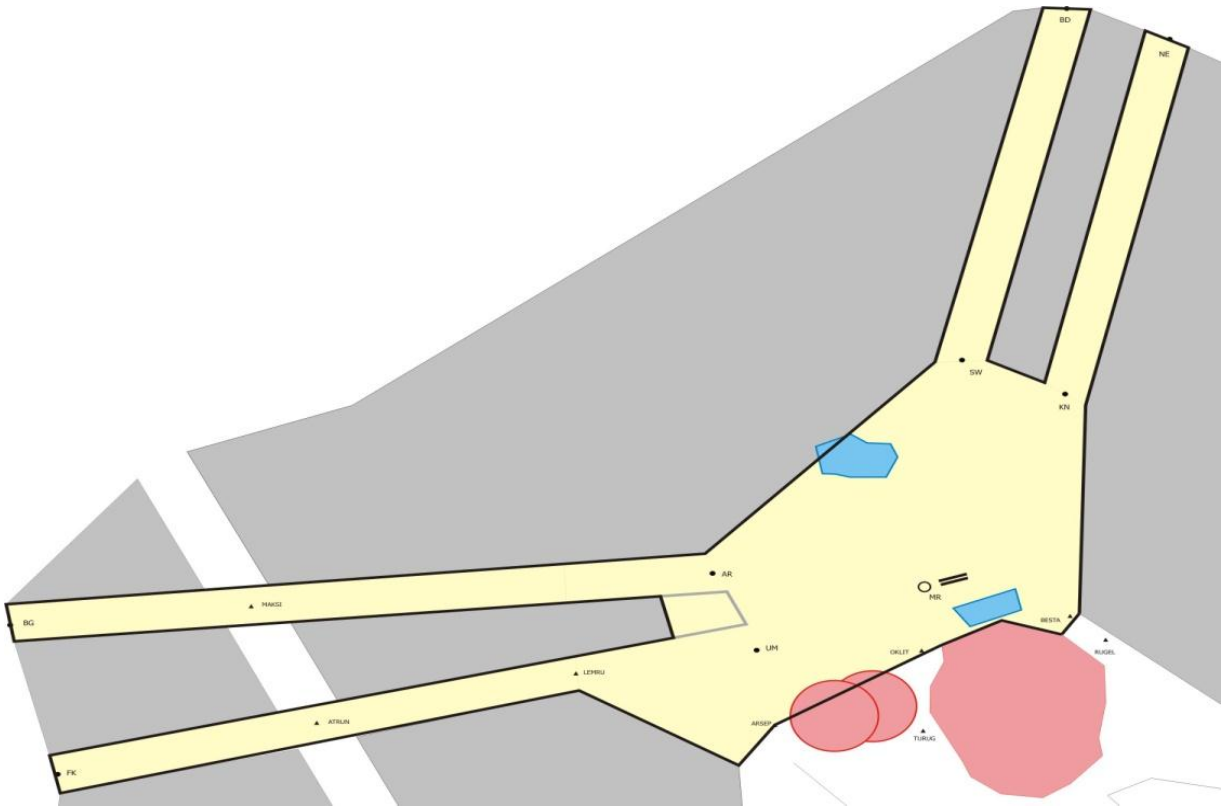


Рис 1.5. Схема приоритетного прилета-вылета для западного направления в аэродроме Шереметьево.

На рис 1.5 изображены новые границы сектора Шереметьево. Как видно на рисунке, между ОПРС Ивановское и ОПРС Бужарово граница сектора

сдвинута в сторону запада. Это было предусмотрено для будущего размещения схемы Тромбон для курса посадки 066 градусов.

При создании маршрутов вылета (рис 1.6) ВС из аэропорта Шереметьево с курсом взлета учитывается поток прилетающих ВС со всех направлений, а также влияние шума на прилегающие жилые территории.

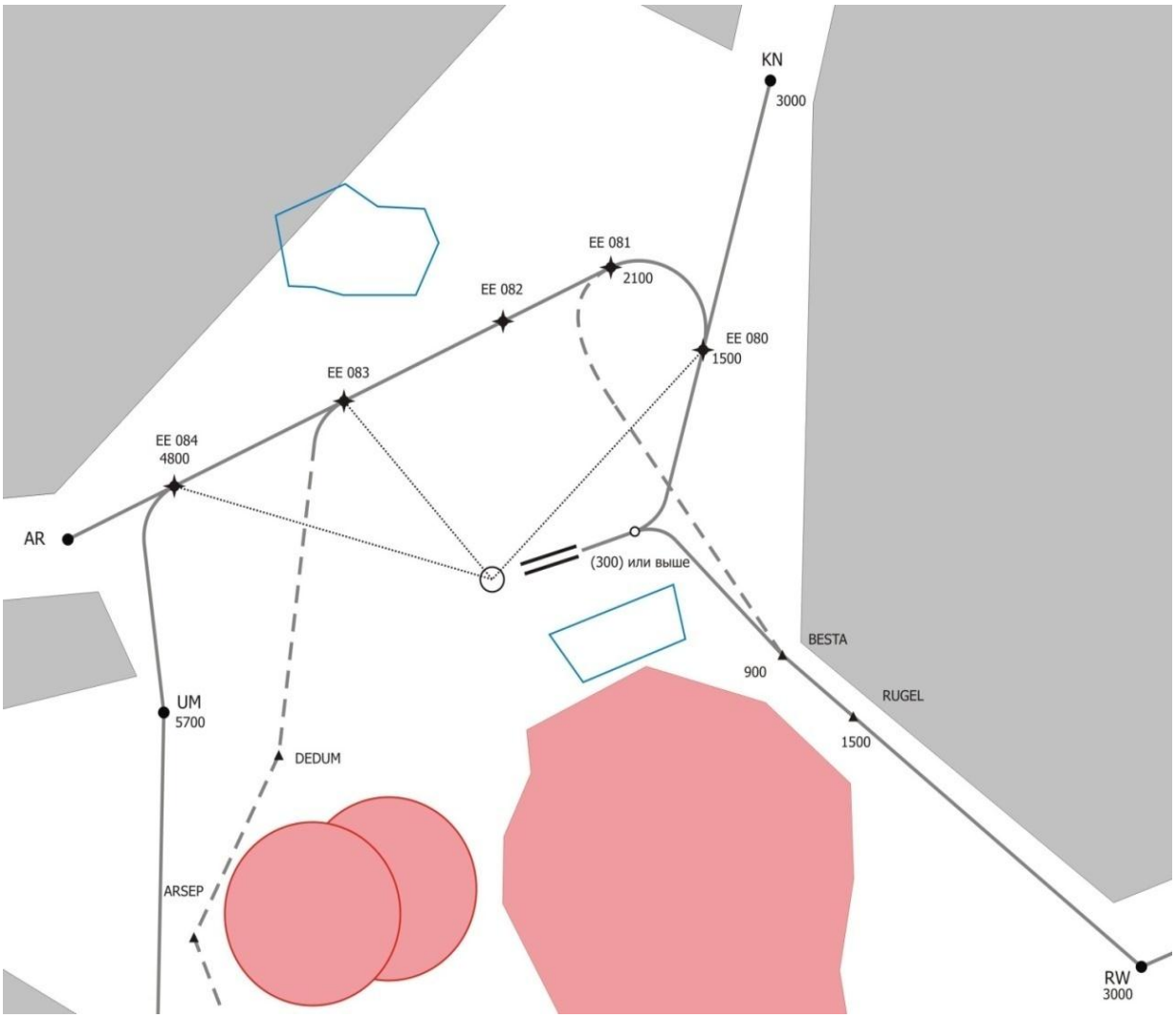


Рис 1.6. Стандартные маршруты прилета-вылета из аэродрома Шереметьево

Существуют стандартные маршруты для прилета, позволяющие в применить упомянутый метод формирования очереди на посадку — это схемы типа «Тромбон» (рис. 1.7)

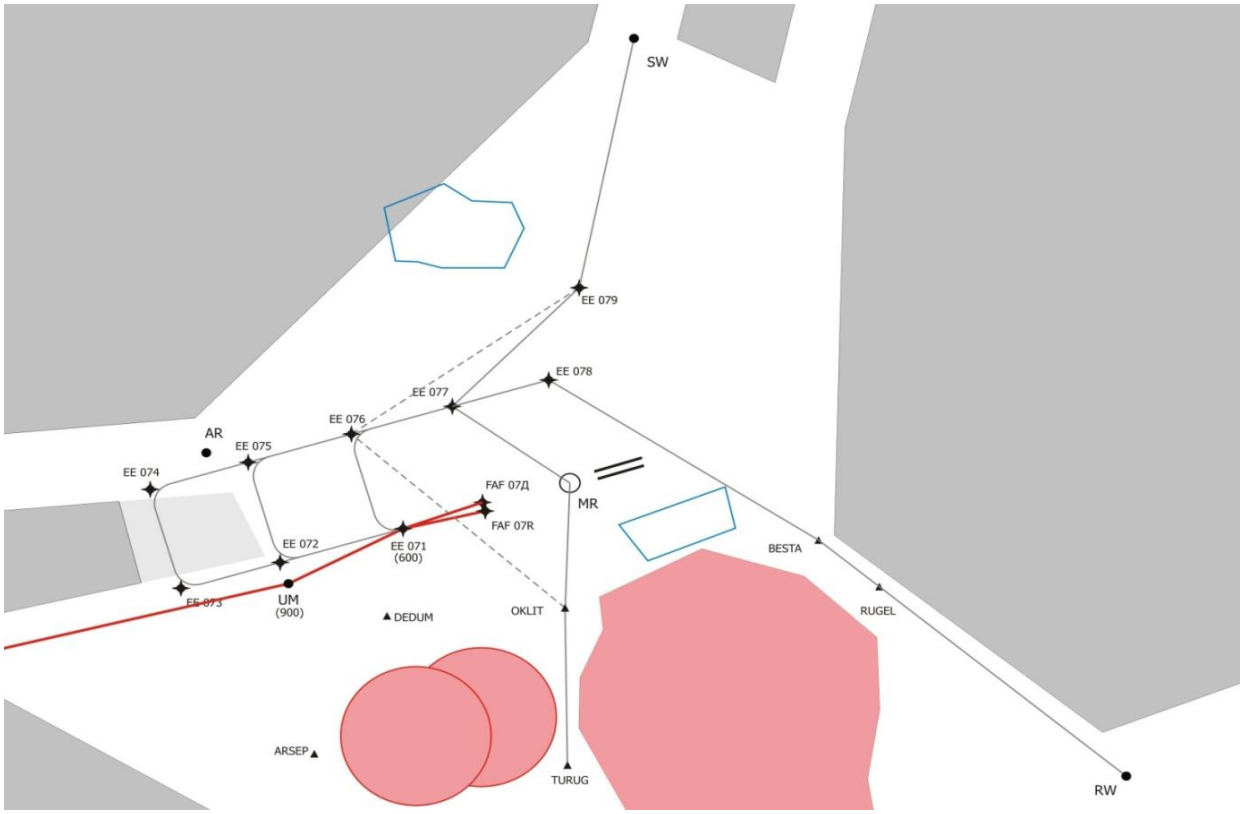


Рис 1.7. Схема «Тромбон» для аэропорта Шереметьево

На данном рисунке изображена схема «Тромбон» для аэропорта Шереметьево с курсом посадки 066 градусов. На нем видна концепция создания основного потока прилетающих ВС с западного направления. Расстояния между точками схемы Тромбон определяют интервалы создаваемые на участках между точками EE071 – FAF07L и EE071 – FAF07R [2].

Критерием начала выполнения «захода» на посадку со снижением до высоты H является момент расхождения конфликтующих воздушных судов относительно основного направления потока, что достаточно просто делается диспетчером «вручную» или автоматизировано. На рис 1.8. изображен вариант конфигурации схемы Тромбон для увеличения интервала захода на посадку.

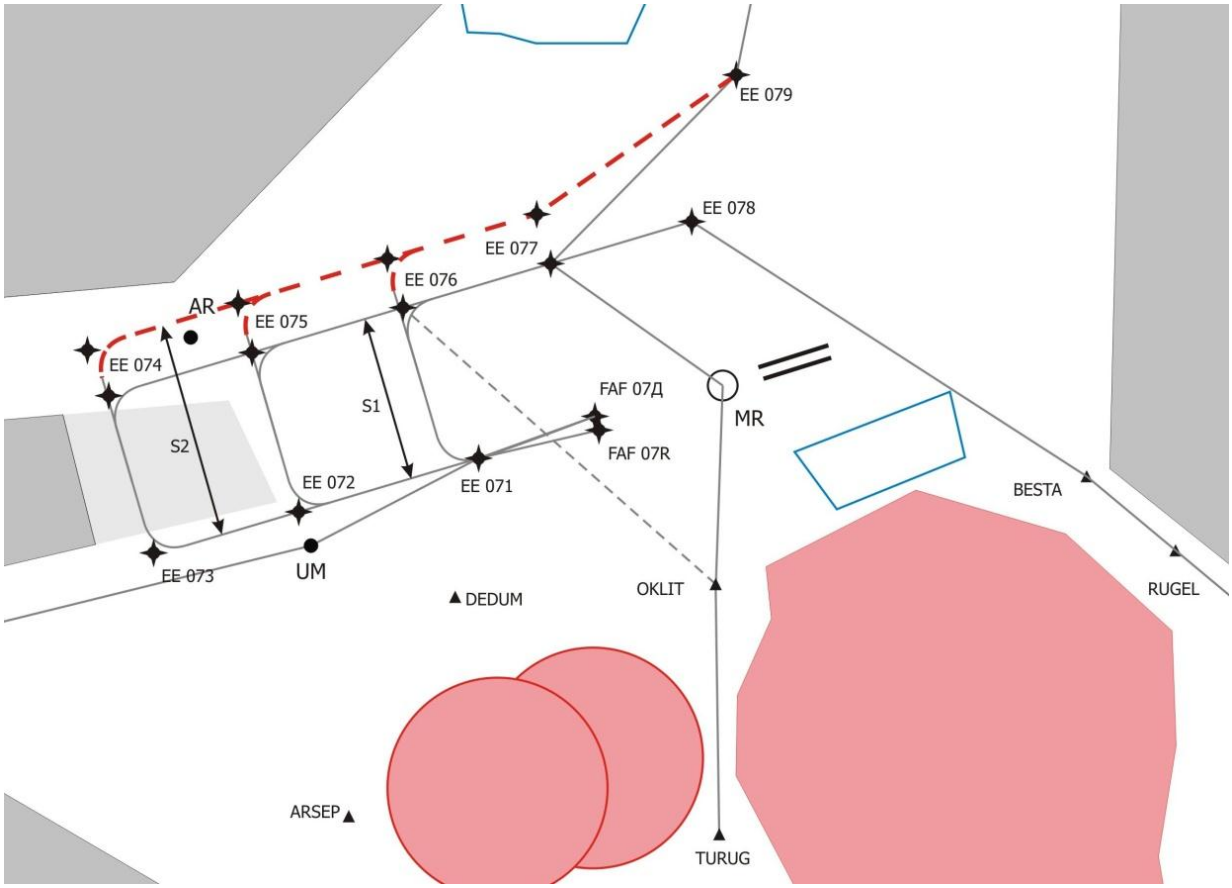


Рис. 1.8. Конфигурация схемы «Тромбон» с разными интервалами захода на посадку воздушных судов

На данном рисунке видны две конфигурации схемы тромбон. Интервалами конечного этапа захода на посадку являются расстояния между точками, находящимися на траверзе (S1=10 км и S2=12 км).

Вторая конфигурация является актуальным решением в сложных метео условиях, где требуются повышенные интервалы между заходящими на посадку ВС. На рис 1.9. представлена ситуация, когда с восточного, северного и южного направления три ВС прилетают одновременно. Бесконфликтность данной ситуации возможна в случае использования схемы тромбон на трех высотах:

- для каждого направления выделена своя высота полета по этой схеме.

-все снижения с высоты полета происходят после выхода на курс посадки, что позволят эффективно выстраивать второстепенные потоки относительно основного.

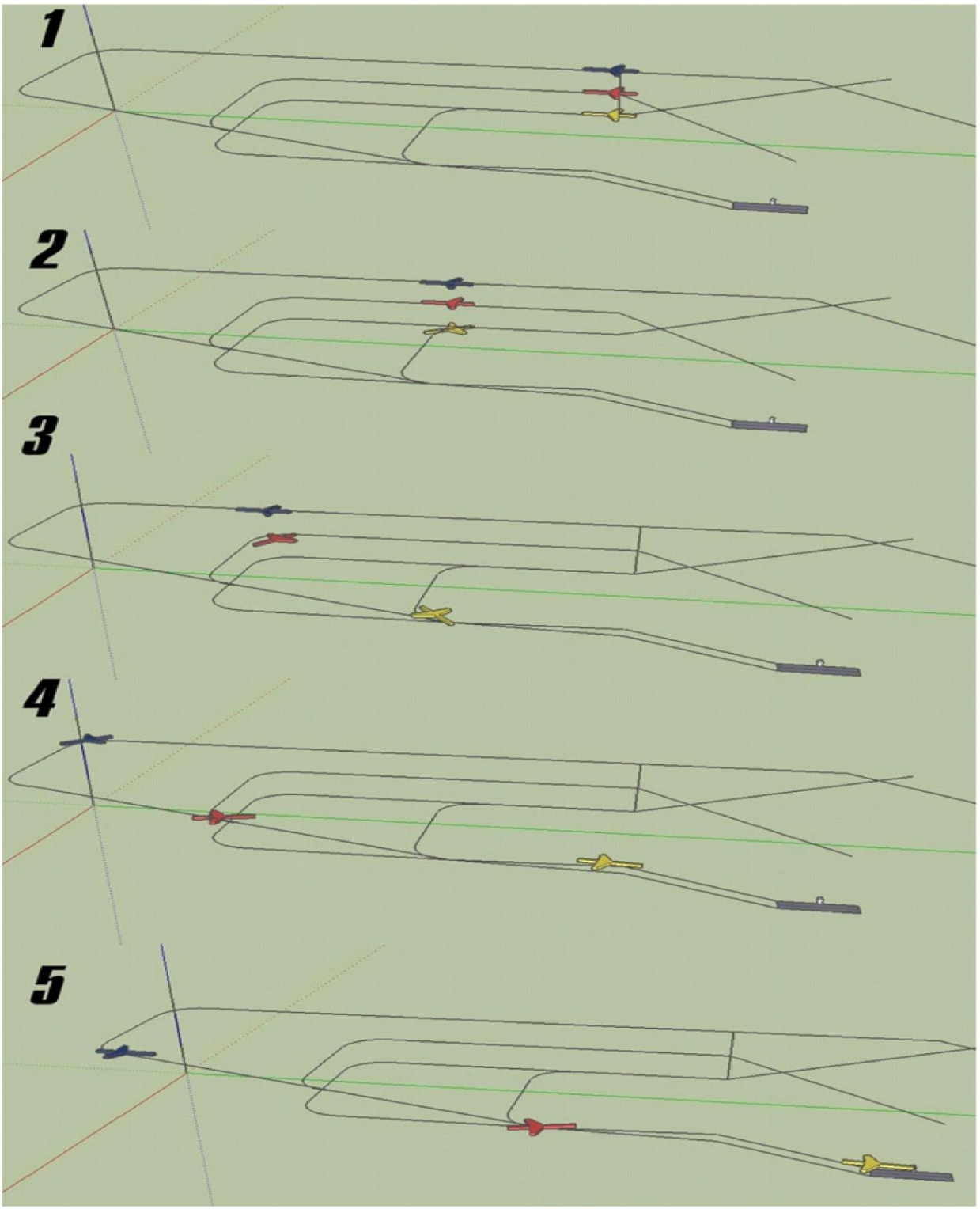


Рис 1.9. Схемы «тромбон» при использовании захода на посадку на различных высотах

Таким образом, известные системы управления воздушным движением предусматривают распределение ВС как по различным трассам, так и в ряде случаев – по соответствующим тромбонам. Решение этой задачи должно отвечать условия безопасности полета ВС и суммарной экономичности, когда минимизируется общий расход топлива у всех ВС при заходе на посадку[7].

1.2 Общая постановка задачи

Сформируем общую постановку решаемой в диссертации задачипри следующих допущениях:

1. Для каждой ВПП заданы исходные углы Ψ_{oj}, j=1...N. Общее число полос равно N. По ним определяются курсы посадки Ψ_{ki}, i = 1…2N (на каждую полосу можно зайти с двух сторон, см. рис.1.10).

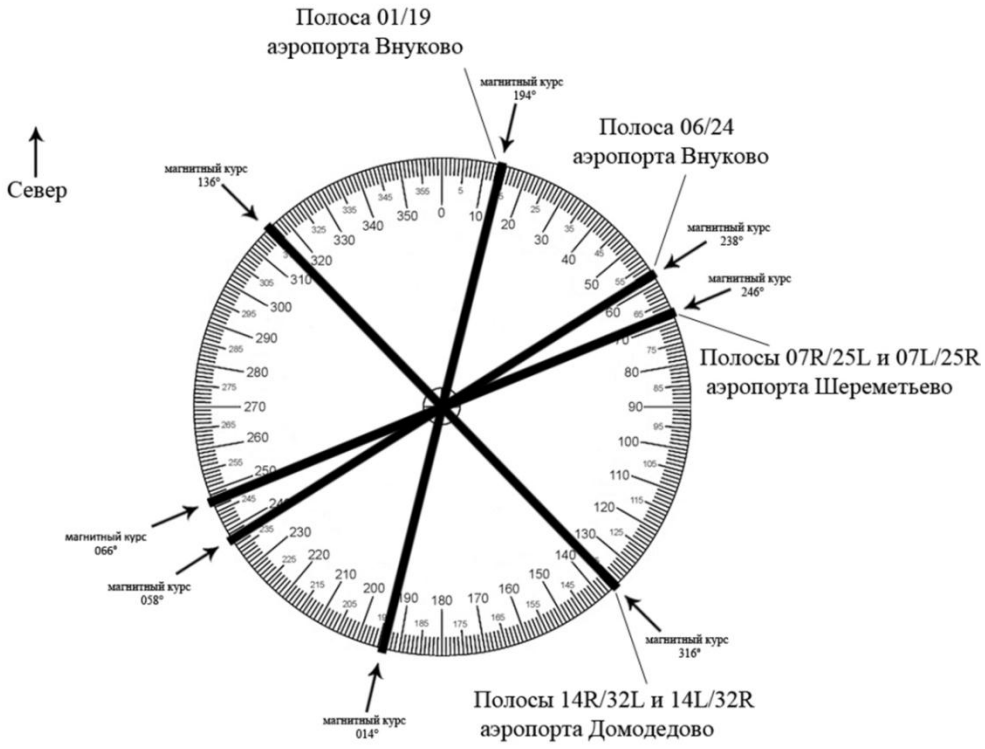


Рис.1.10. Упрощенное представление расположения полос Московского аэроузла.

2. Задан текущий курсовой угол Ψ_W ветра, который может поменять своё значение. В зависимости от его направления для каждой полосы определяется один из двух посадочных курсов.

3. Для каждой из N ВПП рассматривается задача введения на трассу воздушных судов, при их безопасном заходе на посадку, как это показано на рис1.11. При этом анализируется только горизонтальный полет на заданной постоянной высоте.

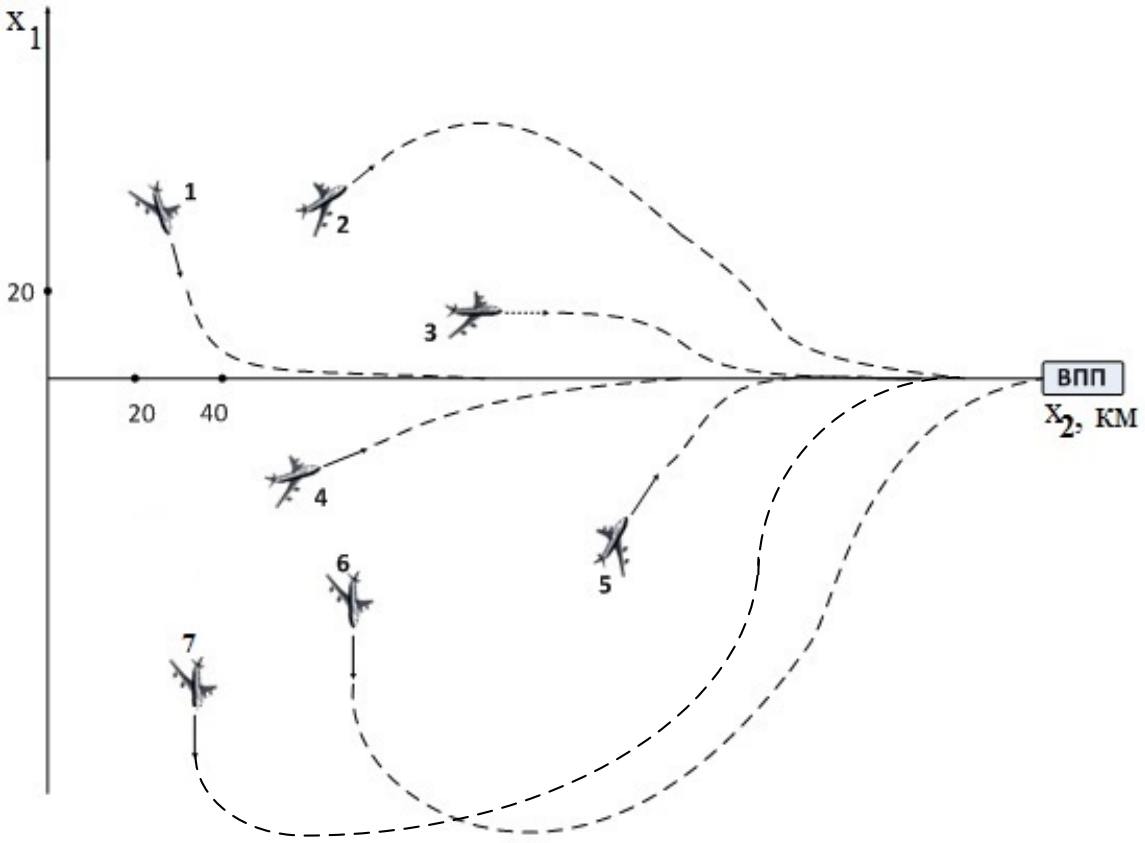


Рис.1.11. Картина выведения воздушных судов на заданную линию попутного движения

4. Каждое судно (ЛА) характеризуется в текущий момент времени вектором состояния, характеризуемым координатами:

x_1 – кратчайшим расстоянием от ЛА до указанной линии пути;

x_2 – минимальным расстоянием до ближайшего судна в эшелоне, уже находящегося на заданной линии пути;

x_3 – курсовым углом, отсчитываемым по отношению к заданному курсу линии пути;

x_4 – потраченным запасом топлива на дополнительное маневрирование.

5. В качестве постоянных параметров принимаются, как известные скорость полета V, максимальное допустимое боковое ускорение a при разворотах, минимальная дистанция r безопасного движения ЛА в эшелонеи запас топлива ΔV, отведенный на маневрирование и определяющий оставшийся на последующие действия запас топлива как $(\Delta V - x_4)$.

6.Принимаемое окончательное решение относится к одной из двухальтернатив *(j=1,2)*.

При *j=1* принимается решение о введении ЛА в воздушный эшелон, если соответствующий ему риск невелик.

При *j=2* дается команда об уходе ЛА на повторный круг или в «тромбон», если существует угроза возникновения аварийной ситуации в воздухе из-за опасного сближения судов, т.е. если на трассе нехватает места для безопасного движения судов, то часть из них направляется в очередь этой трассы, называемой “тромбон”, чтобы потом прилететь на тот же аэродром при первой возможности.

7. Каждая из координат x_i текущего состояния ЛА меняется в соответствии с дифференциальными уравнениями движения, описывающими динамику полета. При этом для простоты каждой координате x_i соответствует одно дифференциальное уравнение. Эти дифференциальные уравнения имеют следующий вид:

Для координаты x_1 принято

$$\dot{x}_1 = \begin{cases} -\dfrac{2x_1}{T_1+T_2} \ при\ j=1 \\ -V \ при\ j=2 \end{cases} \qquad (1.1)$$

Формула (1.1) показывает, что при попадании на линию пути воздушное судно апериодически постепенно стремится обеспечить попасть на указанную трассу при этом постоянные времени (T_1+T_2) апериодического процесса есть время T_2 попадания ЛА на саму линию пути плюс время T_1ускоренного движения по линии пути до точки, имеющей безопасное расстояние r до соседнего $ЛА_0$(см. рис.1.11)

Для координаты x_2 принято

$$\dot{x}_2 = \begin{cases} \dfrac{-x_2}{2T_2} \ при\ j=1 \\ \dfrac{-x_2}{2T_0} \ при\ j=2 \end{cases} \qquad (1.2)$$

Где $T_0 > T_2$ – время движения ЛА на повторном круге.

Динамику изменения курса при входе на заданную линию пути можно описать дифференциальным уравнением, аналогичным (1.2).

$$\dot{x}_3 = \begin{cases} \dfrac{x_3}{2T_2} \ при\ j=1 \\ \dfrac{-x_3}{2T_0} \ при\ j=2 \end{cases} \qquad (1.3)$$

Расход топлива для обеспечения полета должен определяться с учетом того, что на самой линии пути изменение дистанции между летящими ЛА осуществится на форсированном режиме тяги двигателя, при этом расход увеличится в (1+λ)раз, а «скорость догона» одного ЛА по отношению к соседнему ЛА будет лишь $V\lambda$. Поэтому в первом приближении можно записать.

$$\dot{x}_4 = \begin{cases} \dfrac{w_0(1+\lambda)T_1}{T_0} при\ j=1 \\ w_0 \qquad при\ j=2 \end{cases} \qquad (1.4)$$

где w_0 – заданная скорость расхода топлива в обычном режиме работы двигателя, в частности при уходе на повторный круг λ=0,2.

8. Одним из наиболее важных допущений является выбор интегрального критерия оптимальности управления воздушным движением, который должен в свертке оценивать экономичность полета. В данной работе в качестве такого критерия принят минимум интегрального функционала, который учитывает штрафные нежелательные отклонения x_1 - от линии пути, x_3 - от курса ВПП, дистанцию x_2 - между соседними ЛА на самой трассе и величину - x_4 потраченного запаса топлива, что в целом позволяет предложить следующую модель критерия:

;
$$I = \int_{t_0}^{t_k} f_0(\bar{x})dt \to \min$$

$$f_o(j) = \begin{cases} \frac{m_1 x_1^{\,2}}{r^2} + \frac{m_2 (r - x_2)^2}{r^2} + m_3 x_3^2 - \frac{m_4 x_4}{\Delta V} \; npu \;\; j = 1 \\ l - m_4 (\frac{x_4}{\Delta V} + \frac{x_4^2}{\Delta V^2}) npu \;\; j = 2 \end{cases} \qquad (1.5)$$

где m_1–коэффициент штрафа за линейное отклонение от трассы,m_2–коэффициент штрафа за близость ВС к соседним судам в эшелоне, m_3–коэффициент штрафа за отклонение по курсу от заданного курса посадки,m_4–коэффициент, отвечающий за безопасность достижения самолетом указанного места посадки, l– штраф за длину пути при уходе на повторный круг.

Поясним формулу (1.5).При$j=1$, т.е. при попадании ЛА на линию пути, в каждый момент времени штрафуются квадрат отклонения x_1^2 от линии пути, квадраты отклонений x_2^2 и x_3^2 - по росстоянию между судами и курсу, и

относительный расход топлива $\frac{x_4}{\Delta V}$. Чем меньше три первыхслагаемых, тем лучше, и тем быстрее ЛА войдет в эшелон с малым запасом топлива.

Нужно сразу заметить, что правильность назначения самих весовых коэффициентов m_1, m_2, m_3,m_4 всегда вызывало дискуссию в теории и практике оптимального управления. В данной работе было принято пойти по пути неизменного достижения заданныхгарантированных дистанций между ЛА безопасного движения, поэтому фактически штрафуется время, а значит израсходованное топливо для достижения нужной полетной ситуации, что очень важно. Значения коэффициентовm_2 иm_4определяют безопасность полета, m_1и m_3 - длину пути при входе в эшелон, и соотношение между ними важно доопределить предварительно до решения задачи оптимального синтеза.

Требуется:

– определить посадочные курсы ВПП с учетом направления ветра;

– распределить воздушные суда между трассами, для чего необходимо сформировать таблицу приоритетов для всех ЛА. На основании этой таблицыполучить списки ЛА для каждой полосы;

– определить первоочередность захода на посадку и приземления дсудов ля каждой трассы, а также списки судов, направляемых в свой “тромбон” или на запасной аэродром.

Таким образом, в данной работе **согласно общей постановке задачи предполагается найти ее решение с помощью теории оптимального управления.**

1.3 Выводы по главе 1

На основании проведенных в данной главе исследований можно сделать следующие выводы:

1. Анализ существующих систем управления воздушным движением при заходе на посадку показал, что главными требованиями являются безопасность и экономичность полета.
2. Наиболее остро проблема обеспечения названных критериев возникает при возникновении непредвиденных ситуаций, к которым относится аварийное техническое состояние воздушных судов, малый запас топлива и внезапное изменение ветра, что приводит к необходимости переназначения посадочных курсов большого количества идущих на посадку воздушных судов.
3. При увеличении потока прилетающих воздушных судов необходимо их распределение не только по различным трассам, но и в ряде случаев - по различным тромбонам.
4. Сформулированная общая постановка задачи преследует цель обеспечения максимальной экономичности полетов при управлении заходом на посадку и гарантированном уровне безопасности движения группы воздушных судов.
5. Формулы (1.1 – 1.4) были получены с помощью вычисления постоянных времениT_0, T_1 и T_2 рассчитанных из условия максимального быстродействия системы оптимального управления воздушным судном.

$$T_0 = \frac{2\pi R}{V} = \frac{2\pi V}{a}; \qquad T_1 = \frac{r - x_1 + \frac{x_2}{4V}\cdot\sqrt{ax_2}}{\lambda V}; \qquad T_2 = 2\sqrt{\frac{x_2}{a}}$$

Если допустить, что суда начинают движение к трассе на расстояниях, со измеримых с заданной дистанцией r безопасного расстояния между судами, то можно получить следующие приближенные оценки:

$$T_1 = 90 \text{ сек}, \; T_2 = 180 \text{ сек}, T_0 = 600 \text{ сек}$$

Полученные процессы оптимального приближения судна к заданной линии пути были заменены экспоненциональными с помощью дифференциальных уравнений первого порядка. Это можно объяснить с помощью рис 1.12.

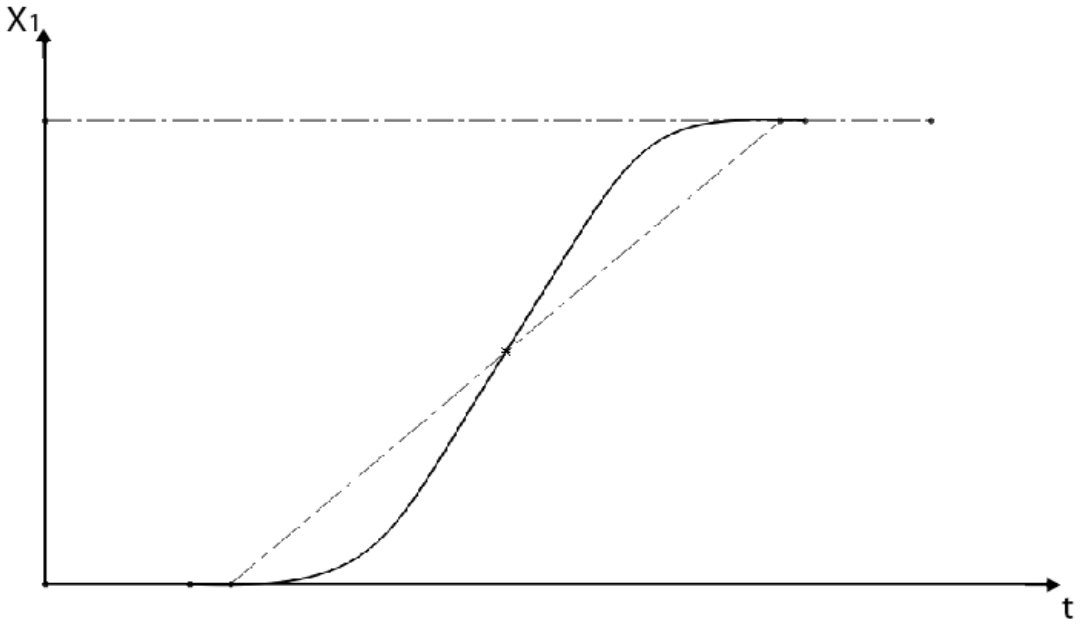


Рис 1.12 . График аппросикмации процесса приближения ЛА к трассе апериодическим процессом

При сложном боковом движении самолет разворачивается по крену и начинает двигаться в сторону трассы. При этом его курс и путевая скорость начинают медленно разворачиваться в направлении сближения с этой трассой за малое время.

Потом будет **большая боковая скорость сближения,** когда самолет развернулся и начинает лететь по прямой линии. В этом случае большая боковая скорость займет**значительное время**. В конце движения боковая скорость постепенно стремится к нулю, что соответствует апериодическому процессу. Предложенное упрощение позволяет существенно снизить трудоемкость последующего вычисления динамических приоритетов воздушных судов.

6. Сформулированная постановка задачи оптимального управления группой ВС обладает общностью и пригодна для произвольного числа

трасс, общего количества ВС и их текущего положения в воздушном простренстве.

Глава 2. Анализ известных методов параметрической оптимизации, теорииоптимального управления и теории массового обслуживания

2.1 Анализ известных методов параметрической оптимизации

Аналитические методыпараметрической оптимизации подразумевают, что целевая функция известна в виде любой алгебраической формы $Z = f(\overline{x_n})$, где x_i - выбираемыепеременные $i = 1,...,n$, т.е. такой числовой характеристики, меньшему или большему значению которой соответствует наилучшая система[8,12, 74, 75].

В ряде задач параметрического синтеза на переменные x_i накладываются различные ограничения[69,79]. Классическая задача оптимизации предусматривает учет алгебраических равенств:

$$\varphi_j(x_1,...,x_n) = 0,\ j = 1,....,m\ ;\ m < n.$$

Другим типичным случаем является введение ограничений типа неравенств, причем таких, которые исключают возможность существования пустого допустимого множества *X*:

$$g_l(x_1,...,x_n) \geq 0, l = 1,....,\rho$$

Формулируется задача математического программирования: минимизировать веществennозначную целевую функциюZ*n*-мерного векторного аргумента $\overline{x} = (x_1,....,x_n)$, компоненты которого удовлетворяют системе уравнений $\varphi_j(\overline{x}) = 0$, набору неравенств $g_l(\overline{x}) \geq 0$, а также ограничены сверху и снизу, т.е. $x_{i\min} \leq x_i \leq x_{i\max}$. Здесь уравнения $\varphi_j(\overline{x}) = 0$ называются ограничениями в виде равенств, неравенства $g_l(\overline{x}) \geq 0$ - ограничениями в виде неравенств.

Задача стандартно записывается так:

$$\begin{aligned} &z = F(\bar{x}) \to \min; \\ &\varphi_j(\bar{x}) = 0; \;\; j = 1,...,m; \\ &g_l(\bar{x}) \geq 0; \;\; l = 1,...,\rho; \\ &x_{i\min} \leq x_i \leq x_{i\max}; \;\; i = 1,....,n. \end{aligned} \tag{2.1}$$

Эта задача называется задачей оптимизации с ограничениями или задачей условной оптимизации [8] . Если $m = \rho = 0$ и $x_{i\max} = -x_{i\min} = \infty$,то задача называется задачей без ограничений или задачей безусловной оптимизации.

Для дифференцируемых функций при поиске экстремума внутри заданной области (исключая граничные точки) необходимым условием отыскания локального минимума функции $F(x_1,....,x_n)$ нескольких переменных является нахождение стационарных точек, в которых обращаются в нуль частные производные:

$$\frac{\partial F(\bar{x}_n^{\,*})}{\partial x_i} = 0, \;\; i = 1,....,n \tag{2.2}$$

Соотношения (2.2) открывают путь к решению задачи отыскания безусловного минимума – действительно, имеется *n* уравнений относительно *n*искомых переменных x_i .

Рассмотрим теперь случай отыскания условного минимума при наличии ограничений типа равенств

$$\begin{aligned} &z = F(\bar{x}_n) \to \min; \\ &\varphi_j(\bar{x}_n) = 0; \; j = 1,...,m; \;\; m < k. \end{aligned}$$

Классический способ решения данной задачи состоит в том, что дополнительные равенства используются для исключения из рассмотрения *m* переменных, при этом целевая функция *F* становится зависимой от (*n*-*m*) неисключенных переменных, на которые не наложено теперь никаких

ограничений. Однако если равенства имеют сложный вид, то этот способ представляет значительные трудности.

В связи с этим на практике применяется так называемый метод неопределенных множителей, использующий функцию Лагранжа, если функция $F(\bar{x}_n)$ и $\varphi_j(\bar{x}_n)$ дифференцируемы. Сущность метода состоит в том, что вводится в рассмотрение вектор $\lambda = (\lambda_1, ..., \lambda_n)$ и составляется функция Лагранжа Lот $(n+m)$ переменных:

$$L(\bar{x}_n, \bar{\lambda}_n) = F(\bar{x}_n) + \sum_{j=1}^{m} \lambda_j \varphi_j(\bar{x}_n) \tag{2.3}$$

Необходимым условием экстремума этой функции являются

$$\frac{\partial L}{\partial x_i} = 0; \quad i = 1, ..., n; \quad \frac{\partial L}{\partial \lambda_j} = 0; \quad j = 1, ..., m$$

или имеет $(n+m)$ алгебраических равенств

$$\begin{aligned} &\frac{\partial F}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial \varphi_j}{\partial x_i} = 0; \quad i = 1, ..., n; \\ &\varphi_j(\bar{x}_n) = 0; \quad j = 1, ..., m. \end{aligned} \tag{2.4}$$

В случае максимизации целевой функции Zфункция Лагранжа имеет вид

$$L(\bar{x}_n, \bar{\lambda}_n) = F(\bar{x}_n) - \sum_{j=1}^{m} \lambda_j \varphi_j(\bar{x}_n)$$

Таким образом, задача на условный минимум целевой функции сводится к задаче определения стационарных точек функции Лагранжа[19, 74].

В задаче с ограничениями типа неравенств

$$\begin{aligned} &F(\bar{x}_n) \to \min \\ &g_l(\bar{x}_n) \geq 0, l = 1, ..., \rho, \end{aligned}$$

если функции $F(\bar{x}_n)$ и $g_l(\bar{x}_n)$ дифференцируемы, то функция Лагранжа от $(n+\rho)$ переменных составляется с помощью вектора неопределенных множителей

$$L(\bar{x}_n, \overline{\mu_\rho}) = F(\bar{x}_n) - \sum_{l=1}^{\rho} \mu_l g_l(\bar{x}_n)$$

и условия экстремума можно записать в виде $(n+\rho)$ равенств:

$$\begin{aligned}
&\frac{\partial F}{\partial x_i} - \sum_{l=1}^{\rho} \mu_l \frac{\partial g_l(\bar{x}_n)}{\partial x_i} = 0, i = 1, \ldots\ldots, n; \\
&\mu_l g_l(\bar{x}_n) = 0, l = 1, \ldots., \rho; \\
&g_l(\bar{x}_n) \geq 0, l = 1, \ldots., \rho; \\
&\mu_l \geq 0, l = 1, \ldots., \rho.
\end{aligned} \tag{2.5}$$

Объединение перечисленных задач отыскания условного экстремума дает следующую формулировку. Необходимо решить задачу оптимизации с учетом ограничений типа равенств и неравенств. Тогда, составив функцию Лагранжа от $(n+m+\rho)$ переменных в виде

$$L(\bar{x}_n, \bar{\lambda}_m, \bar{\mu}_\rho) = F(\bar{x}_n) + \sum_{j=1}^{m} \lambda_j \varphi_j(\bar{x}_n) - \sum_{l=1}^{\rho} \mu_l g_l(\bar{x}_n),$$

Необходимые условия экстремума записывают следующим образом:

$$\begin{aligned}
&\frac{\partial L(\bar{x}_n)}{\partial x_i} = 0, i = 1, \ldots, n; \\
&\varphi_j(\bar{x}_n) = 0, \; j = 1, \ldots, m; \\
&\mu_l g_l(\bar{x}_n) = 0, l = 1, \ldots., \rho; \\
&g_l(\bar{x}_n) \geq 0, l = 1, \ldots., \rho; \\
&\mu_l \geq 0, l = 1, \ldots., \rho.
\end{aligned} \tag{2.6}$$

Воспользовавшись $(n+m+\rho)$ равенствами и 2 ρ неравенствами, можно в ряде случаев довести задачу до конца. Однако, хотя аналитические методы позволяют изучить качественную сторону поведения оптимальных систем,

все же это возможно лишь при малом числе ограничений, особенно если они имеют нелинейный вид.

Следует также обратить внимание на то обстоятельство, что при использовании метода Лагранжа, наряду с переменными параметрами x_i и минимизируемой функцией $z = F(\bar{x}_n)$, впервые появились сопутствующие параметры λ_j и μ_j, позволяющие учесть имеющиеся ограничения.

Численные методы безусловной одномерной оптимизации,в которой необходимонайти минимум некоторой скалярной целевой функций F(x) при отсутствии ограничений на скалярную переменную x:

$$z = F(x) \to \min$$

относятся к наиболее простым из оптимизационных задач. Тем не менее анализ задач такого типа представляется необходимым главным образом потому, что одномерные методы оптимизации часто используются для решения подзадач, возникающих при реализации численных методов решения многомерных задач оптимизации[9,10].

Среди численных методов безусловной оптимизации наибольшее распространение получили методы исключения интервалов и методы полиномиальной аппроксимации. Методы исключения предназначены для нахождения оптимума функции одной переменной внутри заданного интервала при последовательном исключении подынтегралов, не содержащих точку оптимума. Основная идея, положенная в основу работы методов аппроксимации, связана с возможностью локального описания гладкой целевой функции полиномом и последующего его использования для оценки значения точки оптимума.

Стратегия поиска оптимума в методах исключения интервалов основана на сравнении одних только значений целевой функции *F(x)*в различных пробных точках. В данном случае к исследуемой целевой

функции предъявляется единственное требование: она должна быть унимодальной(рис 2.1.)

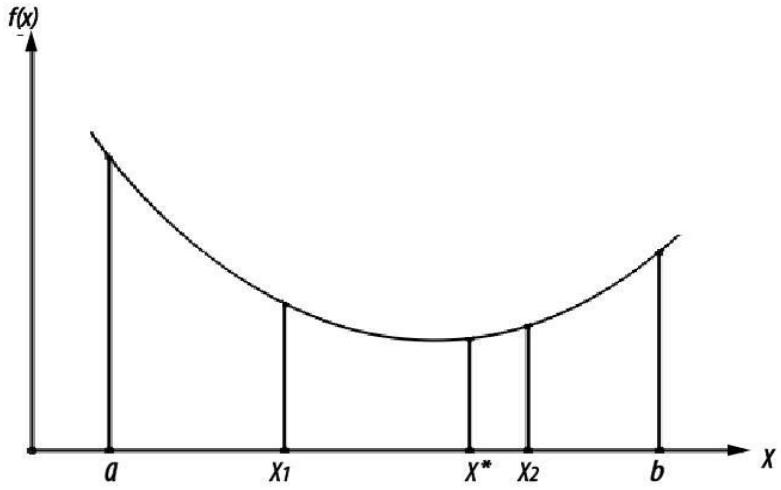


Рис.2.1. Правило исключения интервалов $(F(x_1) \rangle F(x_2))$

В методах исключения процесс поиска можно разделить на два этапа:

- этап установления границ интервала, содержащего оптимум;

- этап уменьшения интервала, на котором исходный интервал последовательно уменьшается до получения интервала заданной длины.

Метод деления интервалов пополам (трехточечный поиск на равных интервалах) позволяет исключать на каждой итерации поиска точно половину интервала. Пробные точки в данном случае должны быть в интервале поиска.

Метод золотого сечения позволяет исключать на каждой итерации поиска часть интервала, меньшую половины, но только при одном вычислении значения оптимизируемой функции *F(x).*

Численные методы безусловной многомерной оптимизации,в которых необходимо найти минимум некоторой скалярной целевой функции $F(\bar{x})$ при отсутствии ограничений на n – мерный вектор переменных $\bar{x}$:

$$Z = F(\bar{x}_n) \to \min$$

можно разделить в зависимости от типа используемой при организации поиска информации на ряд классов:

- методы прямого поиска (методы нулевого порядка), основанные на вычислении только значений целевой функции;

- градиентные методы, в которых используются точные значения первых производных целевой функции;

В методах прямого поиска для построения стратегии поиска требуются только значения исследуемой функции $F(\bar{x})$.

Наиболее простым методом, в котором реализована процедура последовательного перебора направлений поиска, является метод координатного спуска. В этом методе множество направлений поиска совпадает множеством координатных направлений в пространстве неизвестных координат. С помощью методов одномерной оптимизации вдоль каждого из координатных направлений последовательно проводится минимизация целевой функции. При минимизации несложных целевых функций(например, обладающих свойством сферической симметрии) координатный спуск позволяет достигнуть точки оптимума.

При использовании координатного спуска для исследования целевой функций с более сложной топологией, линии уровня которых сильно искривлены и растянуты, поиск оптимума может отказаться неэффективным.В связи с этим возникает необходимость рассмотрения методов безусловной оптимизации, основанных на использовании градиента целевой функции.

В основе логических схем практически всех градиентных методов лежит итерационная процедура, описываемая следующей формулой:

$$\bar{x}^{(k+1)} = \bar{x}^{(k)} + \alpha^{(k)} \bar{s}^{(k)} \qquad (2.7)$$

где $\bar{x}^{(k)}$, $\bar{x}^{(k+1)}$ - текущее и вычисляемое приближение к точке оптимума x^*; $\alpha^{(k)}$ - параметр, характеризующих длину очередного шага;

$\bar{s}^{(k)} = \bar{s}\,(\bar{x}^{(k)})$ - вектор, задающий направление поиска в *n*-мерном пространстве неизвестных.

Градиентный метод наискорейшего спуска основан на определении на каждом шаге поиска направления наибольшего локального уменьшения целевой функции. Этот метод часто называют методом Коши, **который также требует большого числа шагов вычислений при многомерной оптимизации, хотя он выгоднее метода координатного спуска**.

2.2 Линейноепрограммирование

Несмотря на различие содержательных ситуаций, анализируемых при решений технических, экономических и другого рода задач, вычислительная процедура поиска условного экстремума в линейной постановке имеет много общего[13-18,20]. Это позволяет воспользоваться некоторой стандартной формой линейных оптимизационных моделей, для которых:

-значения всех переменных неотрицательны;

-целевая функция является линейной формой и подлежит максимизации или минимизации;

- все ограничения записываются в виде линейных равенств с неотрицательной правой частью.

В исходной постановке задача линейного программирования может выглядеть следующим образом: ищется максимум линейной формы от неотрицательный переменных [73, 76, 77]

$$z = F(\bar{x}_n) = C_1 x_1 + C_2 x_2 + ... + C_n x_n = \sum_{i=1}^{n} C_i x_i \to \max$$

при наличии ограничений как в виде равенств

$$\varphi_j(\bar{x}_n) = a_{j1}x_1 + ... + a_{jn}x_n - b_j = 0; b_j \geq 0;\ j = 1,...,m; m\langle n\ ,$$

так и, главным образом, в виде неравенств при условии, что их число, как правило, превышает число переменных, т.е.

$$g_l(\bar{x}_n) = d_{l1}x_1 + ... + d_{l\,\mathrm{n}}x_n - d_{l0} \geq 0; l = 1,..., \rho; \rho\rangle n$$

Можно убедиться, что неравенства всегда можно свести к строгим равенствам, если ввести дополнительные неотрицательные или, как их называют, искусственные переменные: $x_{n+1}, x_{n+2}, ..., x_{n+\rho}$.Так,например, ограничение

$$d_{l1}x_1 + ... + d_{l\,\mathrm{n}}x_n - d_{l0} \geq 0$$

Можно свести к равенству, вычитая неотрицательную переменную $x_{l+n} \geq 0$:

$$d_{l1}x_1 + ... + d_{l\,\mathrm{n}}x_n - x_{n+l} - d_{l0} = 0$$

Таким образом, без снижения общности задачу линейного программирования с учетом введенных искусственных переменных можно записать в стандартной форме:

$$\begin{aligned}
&z = \sum_{i=1}^{n} C_i x_i \to \max;\\
&a_{11}x_1 + ... + a_{1n}x_n = b_1;\\
&.....\\
&a_{m1}x_1 + ... + a_{mn}x_n = b_m;\\
&b_j \geq 0;\ j = 1,..., m;\\
&x_i \geq 0; i = 1,..., n; n\rangle m
\end{aligned} \qquad (2.8)$$

или в матричной форме:

$$\begin{aligned}
&[C, \bar{x}] \to \max;\\
&A\bar{x} = \bar{b};\\
&\bar{b} \geq 0; \bar{x} \geq 0,
\end{aligned}$$

где $\overline{C}$ и $\overline{x}$ - n мерные векторы; $\overline{b}$ - m- мерный вектор; A – матрица размером $m \times n$.

Существо вычислительной схемы симплекс-метода состоит в реализации упорядоченного просмотра крайних точек, при котором, начиная с некоторой исходной вершины, осуществляется последовательный переход к соседней вершине в оптимальном направлении, пока не будет достигнут максимум. Это позволяет исключить из рассмотрения значительное число базисных решений, заведомо не обращающих в максимум целевую функцию.

Выбор каждой последующей вершины определяется следующими правилами:

- каждая последующая вершина должна быть смежной с предыдущей (например, на рис 2.2 это точки *B*, *C* и т.п. по алфавиту);

-обратный переход не должен производиться;

-в каждой вершине *n*-*m*переменные (включая искусственные) должны быть нулевыми;

-соседние, смежные вершины отличаются в группах базисных и нулевых переменных только одной переменной.

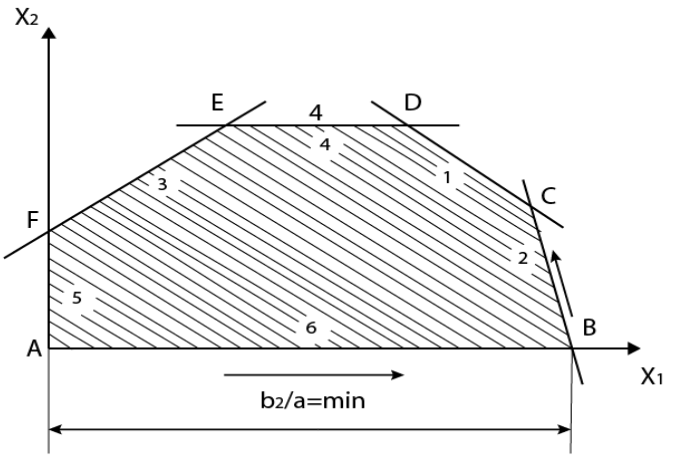


Рис.2.2. Графическая интерпретация исключения переменной при движении к соседней вершине многогранника

По существу необходимо определить итерационную процедуру преобразований при движении по контуру многогранника. Для упрощения этих преобразований удобно ввести специальную форму записи уравнений (2.8) в виде таблицы, содержащей коэффициенты при переменных, Пример такой записи для на этапе выбора начального базиса $x_1 = x_2 = 0$ представлен в табл. 2.1., а сам многогранник – на рис. 2.2[15].

Табл. 1.4. Симплекс-таблица в начале расчета

Базисные переменные	z	y_1	y_2	S_1	S_2	S_3	S_4	Решение
z	1	***-3***	-2	0	0	0	0	0
S_1	0	***1***	2	1	0	0	0	6
S_2	***0***	***2***	***1***	***0***	***1***	***0***	***0***	***8***
S_3	0	***-1***	1	0	0	1	0	1
S_4	0	***0***	1	0	0	0	1	2

В таблице 2.1 в первой строке указаны обозначения целевой функции и всех переменных, как базисных, так и нулевых, эта строка неизменна. Вторая строка содержит информацию о целевой функции, причем второй ее элемент равен единице, последний содержит результат максимизации Z на данном шаге (в частности, в начале расчета при $x_1 = x_2 = 0$ решение z=0), а остальные элементы содержат коэффициенты C_i целевой функции, взятые с обратным знаком. Заметим, что в литературе имеются разного вида описания симплекс-таблиц [5, 7, 17] но это не принципиально.

Следующие строки содержат информацию об ограничениях типа равенств, причем последние элементы строк соответствуют свободным членам b_j . В первых элементах этих строк, образующих первый столбец, записывается группа ненулевых, базисных переменных на данном шаге

расчета (поэтому начальный базис содержит ненулевые переменные $S_1 \div S_4$), а в остальных элементах записываются коэффициенты a_{ji} при переменных, включая ненулевые коэффициенты для функции *Z*.

Согласно условию оптимальности, вводимая переменная есть нулевая переменная, которая должна иметь в строке максимизируемой целевой функции в симплекс-таблице наибольший отрицательный коэффициент (положительный в задаче минимизации). Если все коэффициенты неотрицательны (неположительные), полученное решение является оптимальным, что означает конец поиска. Столбец симплекс-таблицы, ассоциированный с вводимой переменной, называется ведущим столбцом.По условию допустимости исключаемой переменной является базисная переменная, для которой минимально отношение свободного члена b_j в первой части ограничения типа равенства к положительному коэффициенту a_{ji} ведущего столбца.Строку, соответствующую исключаемой переменой, называют ведущей строкой. В итоге вычислительная процедура симплекс-метода состоит из следующих операций. Прежде всего определяется начальное допустимое базисное решение, затем проводится ряд циклических итерационных расчетов, каждый из которых состоит их трех шагов.

Шаг 1. Выбирается вводимая переменная из числа нулевых, обеспечивающая улучшенное значение целевой функции, т.е выбирается ведущий столбец. Если такой переменной нет, то это означает выход из цикла расчетов и конец решения задачи.

Шаг 2. Выбирается исключаемая переменная из числа базисных, которая должна быть обнулена, т.е выбирается ведущая строка.

Шаг 3. Выбирается преобразование симплекс-таблицы, имеющей новый состав нулевых и базисных переменных. Преобразование проводится методом Гаусса-Жордана с помощью двух процедур:

- формирование новой ведущей *j*- й строки по формуле: базисная переменная этой строки заменяется вводимой переменной, а остальные элементы новой ведущей строки равны соответствующим элементам этой строки, поделенным на генеральный коэффициент a_{ji};

- формирование остальных строк таблицы, в том числе и строки целевой функции по формуле: элементы новой *l*- й строки равны элементам этой строки минус соответствующие элементам новой ведущей строки, умноженные на элемент a_{li} ведущего столбца этой строки.

После выполнения преобразований симплекс-таблицы проводится новая итерация расчетов путем возвращения к первому шагу, в результате чего осуществляется переход к смежной вершине многогранника. Как показывает опыт оптимизации систем с помощью **метода линейного программирования, он является наименее трудоемким при большом числе ограничений. Поэтому этот метод выбран в качестве основного при уточнении критерия экономичности и безопасности полета, необходимого для решения задачи обслуживания самолетов при их заходе на посадку.**

2.3 Принцип максимума Понтрягина

2.3.1 Постановка задачи оптимального управления. Принцип максимума Понтрягина

Принцип максимума определяет необходимые условия оптимальности управления в динамических системах. Он распространен на случаи, когда на сигналы управления накладываются ограничения. Дадим более удобную формулировку оптимального управления на основе принципа максимума[21, 22].

Пусть движение объекта описывается системой нелинейных дифференциальных уравнений:

$$\frac{dx_i}{dt} = f_i(x_1, x_2, \ldots, x_n, u_1, u_2, \ldots, u_r) \qquad (2.9)$$

$$i = 1, 2, \ldots, n$$

или в векторной форме:

$$\frac{d\bar{x}}{dt} = f(\bar{x}, \bar{u})$$

Здесь:

$\bar{x}(t)$ - n- мерный вектор состояния объекта

$\bar{u}(t)$ - r- мерный вектор управляющих воздействий

$f(\bar{x}, \bar{u})$ - функция правой части уравнения (2.9)

Полагаем, что вектор управления принимает значения из некоторой замкнутой области U_r-мерного пространства управлений. Положим, что функции $f_i(x,u)$ непрерывны по всем аргументам и имеют непрерывные производные по переменным состояния x_i . Назовем допустимыми управлениями те управления u_r , которые являются кусочно-непрерывными функциями времени и принимают значения из допустимого множества *U*.

Основная задача оптимального управления формулируется следующим образом: среди всех допустимых управлений, приводящих изображающую точку в фазовом пространстве *X* из начального положения x^0 в конечное

x^1 ,если эти управления существуют, нужно найти такие управления, для которых функционал:

$$J(x,u) = \int_0^t f_0(x,u)dt \qquad (2.10)$$

достигает минимума.

Введем в рассмотрение вспомогательные переменные $\psi_1,\psi_2,...,\psi_n$, которые удовлетворяют следующей системе уравнений:

$$\frac{d\psi_i}{dt} = -\sum_{j=1}^{n} \frac{\partial f_j(\bar{x},u)}{\partial x_i}\psi_j$$

$$i = 0,1,2,...,n \qquad (2.11)$$

Система (2.11) называется сопряженной по отношению к системе уравнений (2.9). Введем, в рассмотрение функцию *H*, называемую гамильтонианом

$$H(\bar{\psi},\bar{x},u) = \sum_{i=0}^{n}\psi_i f_i(\bar{x},u) = (\bar{\psi},\ f(\bar{x},u)) \qquad (2.12)$$

Здесь:

$$\bar{\psi} = (\psi_0,\psi_1,\psi_2,...,\psi_n)$$

Тогда системы (2.9) и (2.11) запишутся следующим образом:

$$\frac{d\psi_i}{dt} = \frac{\partial H}{\partial x_i}\ \ i = 0,1,2,...,n \qquad (2.13)$$

Тогда принцип максимума Понтрягина формулируется так “Для того, чтобы допустимое управление было оптимальным, необходимо и достаточно, чтобы гамильтониан системы был максимален”.

2.3.2 Задача об оптимальном быстродействии в линейных системах

Одним из важных случаев применения принципа максимума являетсязадача оптимального быстродействия. Рассматривается случай, когда управляемый объект описывается системой линейных дифференциальных уравнений.

$$\frac{d\bar{y}}{dt} = A(t)\bar{y} + B(t)u \qquad (2.14)$$

Здесь $A(t)=\{a_{ij}\}$ - квадратная матрица размерности $n\times n; y(t), B(t)$ - n - мерный вектор, $u(t)$ - скалярная функция. Управление $u(t)$ ограничено условием

$$|u(t)| \le \delta \tag{2.15}$$

В этом случае доказано, что оптимальное управление $u(t)$ является релейным иудовлетворяет равенству[24, 25]

$$u(t) = sign(\bar{\psi}(t), \quad B(t)) \tag{2.16}$$

Существует также полезная для технических приложений теорема Фельдбаума А. А. [23], согласно которой общее число K переключений в управлении линейным устойчивым динамическим объектом n-го порядка удовлетворяет условию

$$K < n \tag{2.17}$$

Это позволяет путем моделирования экспериментально подобрать моменты времени переключения, если число K невелико.

Однако общим **недостатком принципа максимума является то, что управление является программной функцией времени, т.е. оно реализуется в виде разомкнутой системы**, а это приводит к существенным ошибкам при наличии внешних воздействий.

2.4 Динамическое программирование

Метод динамического программирования, предложенный Р. Беллманам, основан на том, что оптимальное поведение рассматривается как функция состояния системы, описываемого с помощью значений фазовых координат $x_i(t)$ в текущий момент времени t. При этом выбор управления на отдельном шаге производится с учетом не только данного шага, но и всего процесса в целом на всех последующих шагах.

Исходя из этого, Беллманом был сформулирован принцип оптимальности: “Каковы бы ни были начальное состояние и начальное управление, последующие управления должны быть оптимальными относительно состояния, являющегося результатом применения первого управления.” Принцип оптимальности можно также сформулировать следующим образом: оптимальное поведение не зависит от предыстории системы, а определяется только начальным (к данному моменту времени) условием и конечной целью, а текущее управление должно выбираться с учетом последствий в будущем.

Принцип оптимальности справедлив как для дискретных и непрерывных детерминированных, так и для стохастических процессов управления, благодаря чему динамическое программирование широко применяется в ряде технических задач.

Дискретная форма динамического программированиябыланайдена вначале при решении одномерной задачи, когда управляемый автономный

автономный одномерный объект описывается в дискретной форме [72]

$$x_{l+1} = x_l + \varphi(x_l, u_l), l = 0,1,....k, \qquad (2.18)$$

либо в дифференциальной форме

$$\dot{x} = f(x,u),$$

которой соответствует разностное уравнение

$$x_{l+1} = x_l + f(x_l, u_l)\Delta t, \qquad (2.19)$$

где u — ограниченное в общем случае управление, т.е.

$u_{\min} \le u \le u_{\max}$; Δt — Дискрет времени, равный $\frac{1}{k}(t_k - t_0)$.

При заданном начальном состояний $x(t_0)$ объекта и свободном правом конце необходимо за фиксированное время $(t_k - t_0)$ обеспечить минимум заданного функционала

$$J = \int_{t_0}^{t_k} f_0(x,u)dt \approx \sum_{l=0}^{k} f_0(x_l,u_l)\Delta t$$

или в виде аддитивной целевой функции

$$J = \sum_{l=0}^{k} F(x_l,u_l) \to \min_{u_l, l=0,\ldots,k}, \qquad (2.20)$$

Таким образом, *J*есть функция (*к* + 1) выбираемых переменных u_l , для нахождения которых было выведено функциональное уравнение Беллмана [26],

$$S(x_l,t_l) = \min_{u_l}\{F_1(x_l,u_l) + S[x_l + \varphi(x_l,u_l), t_l + \Delta t]\}$$

где, *S* – функция Беллмана, определяющая минимальное значение функционала в зависимости от текущего значения координаты x_l.

Развивая этот же подход применительно к многомерному неавтономному объекту, можно получить общее функциональное уравнение Беллмана в дискретной форме:

$$S[\bar{x}(t_l), t_l] = \min_{u_r(t_l)}\{F[\bar{x}(t_l), u_r(t_l)] + S[\bar{x}(t_{l+1}, \bar{u}_r), t_{l+1}]\} \qquad (2.21)$$

Пошаговый выбор управления с помощью уравнения (2.21) удобен для расчетов на ЭВМ. В этом случае численное решение осуществляют с правого конца задачи. Рассмотренным методом решаются задачи, когда на правом конце часть фазовых координат закреплена[68, 71, 78].

Нужно отметить, что несмотря на определенную утомительность рассмотренной вычислительной процедуры, метод динамического программирования экономит время расчета, требуя, правда, значительного объема памяти ЭВМ. Однако с увеличением размерности задачи дискретизация резко увеличивает число вариантов расчета запоминаемых результатов, что известно как «проклятие размерности» и требует другого подхода к применению динамического программирования**в непрерывной форме**, поскольку принцип оптимальности Беллмана дает достаточно общее

условие, которое можно применять и для непрерывных систем управления[11,27, 28, 42,43].

Рассмотрим следующий предельный случай, когда дискрет времени Δt бесконечно мал, т.е. $\Delta t \to 0$. Обратимся к функциональному уравнению Беллмана (2.21), заменив в нем дискретный момент времени t_l на текущее время. Тогда можно получить выражение

$$S(x,t) = \min_{u(t)} \{ f_0(x,u)\Delta t + S[\bar{x} + \bar{f}(x,u)\Delta t; t + \Delta t]\} \qquad (2.22)$$

При этом функция S во втором слагаемом правой части уравнения также имеет бесконечно малые приращения. Тогда можно разложить функцию $S(x + f\Delta t, t + \Delta t)$*в* ряд Тейлора в точке (x,t) и, пренебрегая членами второго порядка малости, получить известное уравнение Беллмана в частных производных

$$-\frac{\partial S(\bar{x}_n, t)}{\partial t} = \max_{\bar{u}_r(t)} \left\{ f_0(\bar{x}_n, \bar{u}_r, t) + \sum_{i=1}^{n} \frac{\partial S}{\partial x_i} f_i(\bar{x}_n, \bar{u}_r, t) \right\} \qquad (2.23)$$

Очень важно подчеркнуть, что уравнение Беллмана (2.23) является нелинейным дифференциальным уравнением в частных производных, поскольку в нем присутствует операция минимизации, и его общее аналитическое решение не существует.

Поясним смысл слагаемых, входящих в правую часть уравнения (2.23). Первое слагаемое f_0 характеризует скорость потери на текущем шаге, второе слагаемое в виде суммы членов оценивает последствия от принятого решения в будущем. Причем каждый член учитывает изменение текущего состояния по координате x_i , возникающее за счет управления $\bar{u}_r(t)$, с помощью производной $\dot{x}_i = f_i(\bar{x}, \bar{u}, t)$, которая умножается на свой весовой коэффициент $\frac{\partial S}{\partial x_i}$ Таким образом, производные $\frac{\partial S}{\partial x_i}$ есть своего рода

«коэффициенты чувствительности» оставшегося значения минимизируемого функционала к изменениям текущих значений фазовых координат x_i.

Необходимо отметить, что динамическое программирование позволяет найти оптимальное управление как функцию текущего состояния системы, что практически важно.

Также очень важно подчеркнуть, что правая часть уравнения Беллмана (2.23) по своему определению есть функция текущего риска, чем можно воспользоваться при контроле безопасности управляемого попутного движения воздушных судов, что и сделано в данной диссертационной работе. Аналогичный прием был использован при встречном и поперечном движении транспортных средств[29, 31, 38, 40].

2.5 Аналитическоеконструирование оптимальных регуляторов (АКОР)

Одним из направлений в решении задачи синтеза замкнутой системы оптимального управления является разработанный А.М. Летовым подход, названый аналитическим конструированием оптимальных регуляторов [27], когда алгоритм управляющего устройства замкнутой системы находится аналитически в соответствии с определенным функционалом качества, соответствующим квадратическому критерию вида

$$J = 0.5\bar{x}^T(t_k)M\bar{x}(t_k) + 0.5\int_{t_0}^{t_k}\left[\bar{x}(t)P(t)\bar{x}^T(t) + u^T(t)R(t)\bar{u}(t)\right]dt, \qquad (2.24)$$

Минимизация функционала (2.24) соответствует задаче, когда важно удерживать около нуля все компоненты вектора состояния. Первое слагаемое характеризует терминальную ошибку в конечный момент, второе слагаемое внутри интеграла преследует цель обеспечить малость ошибки при удерживании системы в заданном положении. Последнее слагаемое представляет «штраф за большие управления» и оценивает затрачиваемую на управление энергию.Соответственно положительно полуопределенные

матрицы *M, P* и положительно определенная матрица *R*выбираются с учетом значимости указанных факторов, преимущественно с ненулевыми диагональными элементами. При этом, рассматривается линейный объект, описываемый уравнениями

$$\dot{\bar{x}} = A\bar{x} + B\bar{u} \tag{2.25}$$

где на управление $\bar{u}$ никаких прямых ограничений не наложено. Для получения решения в замкнутой форме воспользуемся методом динамического программирования. С учетом терминального члена функцией Беллмана S является

$$S(\bar{x},t) = \min_{\bar{u}} \left\{ 0.5\bar{x}^T(t_k) M \bar{x}(t_k) + \int_t^{t_k} f_0(\bar{x},\bar{u})dt \right\}$$

которая при $t = t_k$ не равна нулю.

С учетом (2.24) и (2.25) уравнение Беллмана имеет вид

$$-\frac{\partial S(\bar{x},t)}{\partial t} = \min_{\bar{u}} \left\{ 0.5\bar{x}^T MP(t)x + 0.5\bar{u}^T R(t)\bar{u} + \frac{\partial S}{\partial \bar{x}}[A(t)\bar{x} + B(t)\bar{u}] \right\} \tag{2.26}$$

При отсутствии ограничений на оптимальное управление вычислим производную от выражения в фигурных скобках и, приравняв ее нулю, получим

$$\bar{u}^T R + \frac{\partial S}{\partial \bar{x}} B(t) = 0$$

Поскольку матрица *R* положительно определена, можно найти, во-первых, оптимальное управление

$$\bar{u}(t) = -R^{-1}B^T(t)\left[\frac{\partial S(\bar{x},t)}{\partial \bar{x}}\right]^T \tag{2.27}$$

и, во-вторых, записать уравнение Беллмана без операции минимизации:

$$-\frac{\partial S}{\partial t} = 0.5\bar{x}^T P(t)\bar{x} - 0.5\frac{\partial S}{\partial \bar{x}} B(t)R^{-1}(t)B^T\left[\frac{\partial S}{\partial \bar{x}}\right]^T + \frac{\partial S}{\partial \bar{x}} A(t)\bar{x} \tag{2.28}$$

Уравнение (2.28) можно решить при условии $S(\bar{x},t_k)=0.5\bar{x}^T M\bar{x}$. Можно показать [31], что уравнение (2.28) имеет точное аналитическое решение, которое представляет собой квадратичную форму

$$S(\bar{x},t)=0.5\bar{x}^T K(t)\bar{x}$$

где *K*(t) — симметричная нестационарная матрица с искомыми элементами. Доказано, что элементы этой матрицы подчиняются системе линейных неоднородных дифференциальных уравнений с граничным условием $K(t_k)=M$:

$$\dot{K}=-KA(t)-A^T(t)K+KB(t)R^{-1}(t)B^T(t)K-P(t) \quad (2.29)$$

Уравнение (2.29) называется матричным уравнение Риккати, решение которого обычно находят численно на ЭВМ до начала работы системы. Оптимальному управлению соответствует в общем случае **линейный закон управления** с переменным коэффициентом передачи

$$\bar{u}(\bar{x})=-R^{-1}KB^T(t)\bar{x} \quad (2.30)$$

В работах Калмана доказывается, что для стационарных объектов, т.е. при постоянных матрицахА, В, К и Р, решение уравнения Риккати для установившегося состояния есть постояннаяматрица*K*, соответствующая алгебраическому уравнению

$$KA+A^TK-KBR^{-1}B^TK+P=0 \quad (2.31)$$

В этом случае оптимальная замкнутая система является стационарной

$$\dot{\bar{x}}=(A-BR^{-1}B^TK)\bar{x} \quad (2.32)$$

и асимптотически устойчивой при $t\to\infty$.

Главной полезной особенностью метода АКОР является то, что закон управления может быть найден алгебраическим путем в квадратурах.

2.6 Оценка возможности применения теории массового обслуживания

Практически в любой своей деятельности человек сталкивается с системами массового обслуживания (СМО) и с процессами, происходящими в них [41,52-56].

Общая блок-схема разомкнутой, однофазной СМО может быть изображена в следующем виде (рис. 2.3):

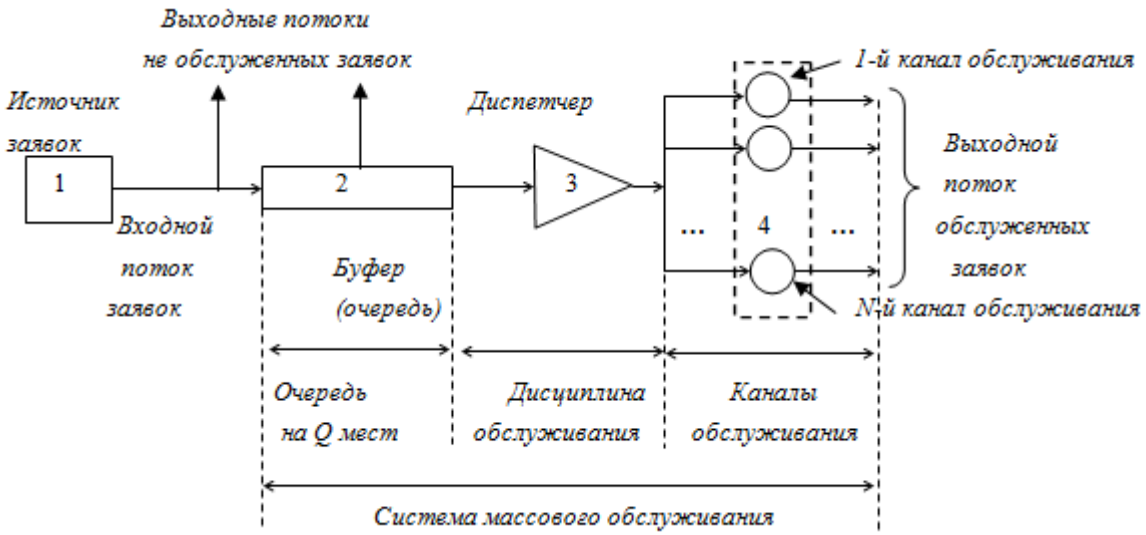


Рис. 2.3. Блок-схема однофазной, разомкнутой СМО

Из источника заявок (блок 1) поток заявок попадает либо в очередь (блок 2), либо через диспетчера (блок 3) на канал обслуживания (блок 4). Распределение заявок по свободным каналам осуществляется по правилам дисциплины обслуживания с помощью некоторого «устройства», названного здесь диспетчер 3. Обычно в схеме блок 3 не указывается, а дисциплина обслуживания задается в описании СМО. Если в момент прихода заявка застает все каналы занятыми, то она встает в очередь в буферное устройство 2 ограниченной емкости, ожидая там своего обслуживания. При этом, заявка может находиться в очереди только ограниченное время, превышение которого заставит покинуть систему. Эти заявки образуют выходящий поток необслуженных заявок. Состояние рассматриваемой системы может быть и таким, что заняты все места в буфере и заняты все каналы, тогда заявка еще

до буфера покидает систему необслуженной, т.е. происходит отказ в обслуживании[70, 81, 82].

На практике существует много различных типов СМО. В частности согласно общепринятой классификации по форме Кендала [54], если определять систему обслуживания пассажиров в аэропорту при подлете n самолетов, подчиняющихся пуассоновскому закону, то это можно определить формулой

$$(M/M/K): (FiFO / n\text{-}k / \alpha)$$

Это можно сформулировать например так: задана СМО с пуассоновскими входным потоком пассажиров и временем обслуживания *K*каналами аэропорта по правилу “первым пришел первым обслужен”, с очередью не более (*n*-*k*),при неограниченной емкости источника прилета самолетов.

В СМО без отказов количество мест в очереди и время ожидания неограниченны. Входной поток представляет собой случайную последовательность во времени определенных дискретных событий.

Поток событий называется *ординарным,* если вероятность $P_{>1}(t,\Delta t)$ того, что на малый промежуток времени Δt , примыкающий к моменту времени t, попадет больше одного события, пренебрежимо мала по сравнению с вероятностью $P_1(t,\Delta t)$ того, что на тот же интервал времени попадет ровно одно событие:

$$P_1(t,\Delta t) >> P_{>1}(t,\Delta t).$$

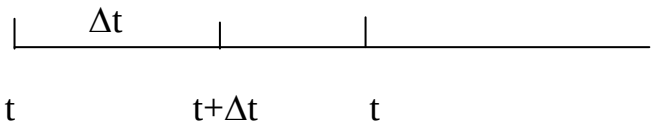


Рис. 2.4. К определению интенсивности потока событий

В данной работе потоки прилетающих воздушных судов и пассажиров в аэропорту считаются ординарными.

Для стационарного потока его интенсивность не зависит от времени и представляет собой постоянную величину, равную среднему числу событий, наступающих в единицу времени:

$$\lambda(t)= \lambda = \text{const.}$$

Тип входного потока определяется видом распределения случайных интервалов τ_i. Случайные интервалы времени τ_i между наступлениями событий в потоке могут подчиняться различным законам распределения, например, равномерному или нормальному. Однако в подавляющем большинстве работ по теории массового обслуживания рассматривается, как правило, пуассоновский, т.е. простейший поток.

Для пуассоновского потока вероятность поступления в промежуток времени t ровно k требований задается формулой Пуассона

$$P_k(t) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad (2.33)$$

где, $\lambda>0$ – плотность потока (или параметр потока).

Число каналов обслуживания является одной из основных характеристик СМО. Каналом обслуживания называется вся совокупность технических устройств, обеспечивающих обслуживание одного требования или заявки.

Работа каждого канала характеризуется тем временем, которое затрачивается на обслуживание одной заявки. Время обслуживания является важнейшей характеристикой каждого канала (прибора, аппарата, линии) обслуживания системы и определяет ее пропускную способность. В общем случае это время является случайным, и в теории СМО оно, как правило, имеет экспоненциальное распределение с параметром μ.

От выбранного числа каналов прежде всего зависят вероятности состояний системы p_k, которые определяются как вероятности P_k (k=1,….,n) того, что в n-канальной СМО находится ровно k=0, 1, …n требований (или, что то же: обслуживанием заявок занято ровно k=0, 1, …,n каналов), а также вероятности p_{k+w} того, что в очереди ждет своего обслуживания w заявок (w=1,…5)

$$P_0 , P_1, \ldots, P_k, \ldots, P_n.$$

Проще говоря, P_k - это вероятности того, что:

«в системе нет ни одной заявки» = P_0, т.е. система пуста;

«в системе одна заявка» = P_1;

«в системе две заявки» = P_2;

…

«в системе k - заявок» = P_k;

…

«в системе n – заявок, т.е. все каналы заняты, но очереди нет» = P_n+0

«в систем n – заявок и w заявок» – в очереди = P_{n+w}

«в системе n заявок обслуживается, и длина очереди достигла максимума S», что соответствует вероятности отказа в обслуживании вновь поступившей заявки = P_{n+S}.

Пользуясь общими правилами составления уравнении перехода для вероятностей состояния можно получить

$$\begin{aligned}
\frac{dp_0}{dt} &= -\lambda p_0 + \mu p_1 \\
\frac{dp_1}{dt} &= -(\lambda + \mu) p_1 + \lambda p_0 + 2\mu p \\
&\cdots \\
\frac{dp_k}{dt} &= (k+1)\mu p_{k+1} \frac{dp_{k+1}}{dt}
\end{aligned} \tag{2.34}$$

Эти уравнения называются уравнениями Эрланга.

В результате их решения можно получить формулы:

Для СМО с отказами:

$$P_i = \frac{\rho^i P_0}{i!}; \quad i = 1,..n \tag{2.35}$$

Для СМО с ожиданием в очереди:

$$\begin{aligned}
P_{n+\omega} &= \frac{\rho^{\omega} P_n}{\prod_{m=1}^{\omega} (n + m\beta)}; \quad \omega = 1,..S \\
P_0 + \sum_{i=1}^{n} P_i + \sum_{\omega=1}^{S} P_{n+\omega} &= 1
\end{aligned} \tag{2.36}$$

Нужно отметить, что **методы теории массового обслуживания весьма полезны, для решения поставленной в работе задачи, т. к они учитываютвероятностный характер процессов прилета вылета воздушных судов.**

Однако нужно подчеркнуть, что указанные формулы справедливы для бесприоритетного обслуживания. Вопрос расчета приоритетной СМО является одной из задач, решаемой в данной диссертационной работе.

Подводя общий итогпроведенному анализу известных методов, можно сказать, что наиболее перспективным является метод динамического программирования для синтеза детерминированного управления полетом, а для анализа случайных процессов – теория массового обслуживания.

2.7 Выводы по главе 2

На основании проведенных в данной главе исследований можно следующие выводы:

1. При решении задач выбора посадочных курсов в Московском аэроузле, определения длины очередей самолетов при входе в воздушные эшелоны и числа каналов обслуживания пассажиров в аэродрому необходимо использовать методы параметрической оптимизации.
2. При решении задач управления безопасным попутным движением целесообразно применить методы теории оптимального управления и динамического программирования.
3. Для получения законов управления полетом в квадратурах полезно использовать метод аналитического конструирования оптимальных регуляторов.
4. При рассмотрении самолетов с аварийным состояниям и их приоритетном обслуживании необходимо кроме теории оптимальных систем использовать методы теории массового обслуживания.

Глава 3. Формирование единого критерия безопасности и экономичности полета при заходе напосадку

3.1 Представление критерия качества воздушного движения в линейнойформеи сущность обратной задачи линейного программирования

В ряде важных технических задач бывает так, что на параметры, определяющие эффективность системы, наложены известные ограничения, которые в первом приближении могут быть представлены в виде линейных неравенств, но сам критерий лицу, принимающему решение (ЛПР), неизвестен.

Данная глава посвящена решению обратной задачи линейного программирования с целью установления весовых коэффициентов значимости экономичности и безопасности движения судов в эшелоне захода на посадку[9].

Задача обеспечения максимальной безопасности и экономичности захода самолетов на посадку требует в общем виде знания критерия оптимальности в аналитической форме, чтобы принимать нужные решения о следовании в эшелоне одного самолета за другим на определенной дистанции, не тратив при этом лишнее топливо на маневрирование[32, 33, 36, 37].

Однако, как это бывает и в других задачах, диспетчер знает, ***как надо действовать*** в конкретном случае, но математическая модель критерия ему ***неизвестна***. В одной ситуации он принимает альтернативные решения на сокращение, либо увеличение дистанции между самолетами за счет бокового маневра или увеличения тяги, в других случаях при возрастании риска воздушного движения, диспетчер отказывается от вхождения одного из самолетов в эшелон и дает команду его ухода на повторный круг. Поэтому возникает целесообразность воссоздания критерия по отдельным примерам

оптимального поведения, чтобы затем его использовать в общем случае.Этот критерий можно представить в виде линейной свертки от параметров X_i. назовем их в дальнейшем переменными

$$Z = \sum_{i=1}^{n} C_i X_i \quad : \tag{3.1}$$

Если сами переменные по характеру своего поведения неотрицательны, а выбираемые ЛПР решения альтернативны, то процесс выбора наиболее рационального решения можно трактовать как нахождение одной из вершин выпуклого многогранника, показанного на рис 3.1, что можно было бы сделать с помощью известной прямой задачи линейного программирования. Рис 3.1 иллюстрирует решение следующей частной задачи (пример 1)

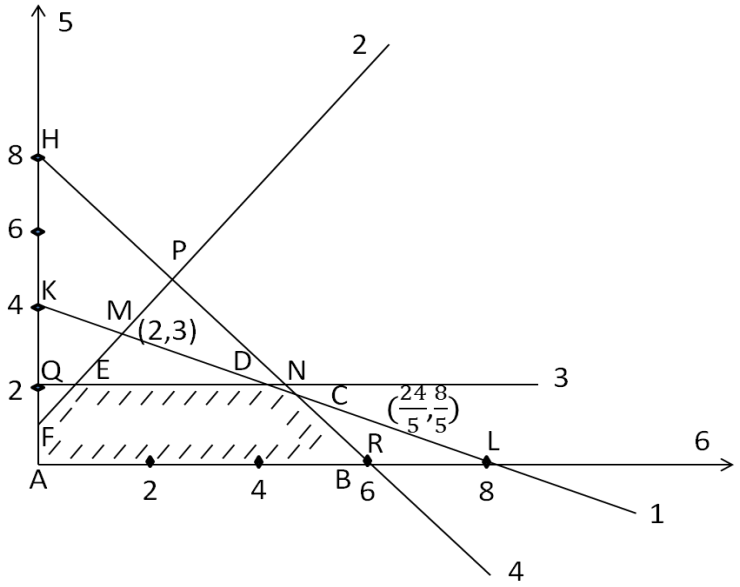


Рис.3.1. Выпуклый многогранник для примера решения задачи с двумя переменными

$$\begin{aligned} &Z = C_1 X_1 + C_2 X_2 \to \max \\ &X_1 + 2X_2 \le 8 \\ &X_2 - X_1 \le 1 \\ &X_2 \le 2 \\ &4X_1 + 3X_2 \le 24 \\ &X_1 \ge 0; X_2 \ge 0 \end{aligned} \tag{3.2}$$

Однако для тех ситуаций, когда весовые коэффициенты C_i не заданы, но известны результаты рационального поведения ЛПР в виде указанных им ответов(или вершин), можно попытаться решить обратную задачу линейного программирования при следующих условиях[29]:

– задана группа линейных неравенств в виде ограничений на множество переменных $X_i : i = 1.....n$;

- известна принятая за лучшее решение вершина многогранника с координатами

$X_i(0); i = 1....n$

- в качестве параметрического критерия оптимальности принята линейная модель (3.1);

- требуется по возможности более точно идентифицировать коэффициенты критерия C_i.

Данную постановку можно расширить, если считать, что в практически реальных условиях состав и параметры ограничений могут меняться известным образом, и для множества этих ситуаций имеется множество известных решений прямой задачи, выбранных ЛПР, в силу чего можно осуществить множество попыток идентификации линейного критерия, который всегда остается неизменным. Это позволит в перспективе провести дополнительное уточнение. В данной работе более подробно рассмотрен случай однократного принятия решения в виде одной вершины, с помощью чего нужно воссоздать параметры линейной свертки.

Нужно заметить, что и в этом случае на выбранном пути имеется ряд трудностей. Во-первых, при известных координатах выбранной вершины $X_i(0)$ нам неизвестны координаты $X_i(l)$ других вершин, и в первую очередь – соседних, в которых в функция критерия Z по утверждению ЛПР явно

меньше, чем в выбранной, что позволило бы записать желанную систему линейных неравенств.

$$\sum_{i=1}^{n} C_i.X_i(0) \rangle \sum_{l=1}^{n} C_l.X_l(l)\ ; l=1\ldots n \qquad (3.3)$$

Известно только, что совокупность этих вершин относится к выпуклому многограннику,находящемуся внутри n-мерного параллелепипеда и вне призмы, как это показано пунктором на рис 3.2 для n=3, что будет использовано ниже при оценкеточности предложенного подхода.

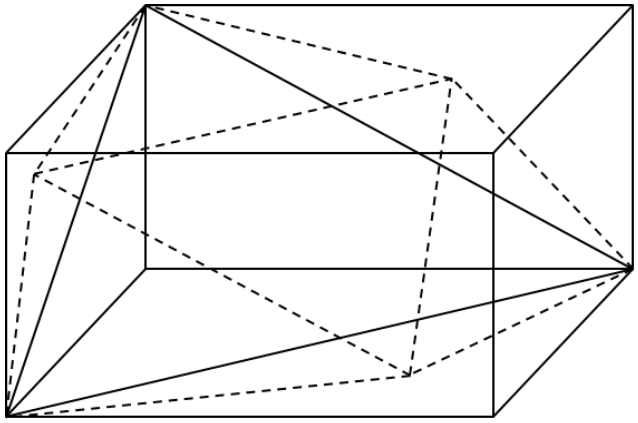


Рис.3.2. Выпуклый многогранник допустимых решений при n=3

При решении прямой задачи нам также известен факт, что чем ближе промежуточная вершина, тем больше значение критерия Z, а его приращения как правило постепенно уменьшаются, но это неважно – ясно одно, что в соседних к выбранной ЛПР, эта разница минимальна, поэтому наибольшее уточнение в идентификации дадут неравенства(3.3), записанные для смежных вершин, координаты которых нам неизвестны.

Поэтому нет необходимости искать координаты всех вершин многогранника, тем более что эта трудоемкая операция связана с выбором из общего числа решений системы неравенств, превращенных в равенство с помощью искусственных переменных - только тех, которых относятся к

выпуклому многограннику. В частности, для примера1 формально это 15 точек пересечения прямых линий, а интересующих нас вершин 6 – *A, B, C, D, E, F*.

С учетом высказываемых соображений главный замысел предлагаемого подхода таков-

- зная координаты $X_i(0)$ выбранной вершины, определим для нее ту группу неравенств, которая превращается в равенства, (в трактовке известного симплекс-метода решения прямой задачи соответствующие им искусственные переменные S_l будут равны нулю).
- с помощью найденной группы равенств найдем координаты соседних, смежных вершин, с помощью которых можно получить искомую группу условий (3.3) для идентификации критерия *Z*.

Однако для того, чтобы решить эту задачу, необходимо выполнить дополнительные операции, для пояснения которых вначале повторим ход решения прямой задачи на примере1 воспользовавшись симплекс-методом, и после уяснения особенностей этого решения, поясним последовательность предложенных операций в удобном для изложения порядке.

3.2 Пример 1. Решение прямой задачи линейного программирования

Рассмотрим решение примера 1 при известныхкоэффициентах $C_1 = 1/9$, $C_2 = 1$, воспользовавшись также известным способом записи симплекс-таблицы, изложенным, в частности в [15], если ввести искусственные переменные $S_1,....,S_4$ в неравенства(3.2). Тогда получим исходную симплекс-таблицу 3.1.

Табл. 3.1. Исходные данные для прямой задачи

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	$-\frac{1}{9}$	**-1**	0	0	0	0	0
S_1	1	**2**	1	0	0	0	8
S_2	**-1**	**1**	**0**	**1**	**0**	**0**	**1**
S_3	0	**1**	0	0	1	0	2
S_4	4	**3**	0	0	0	1	24

Согласно правилам симплекс-метода вначале выберем ведущим столбец 3, имеющий наибольший отрицательный коэффициент, и ведущую строку 4, у которой отношение правой части уравнения к “своему ” элементу ведущего столбца, равному 1, минимально и равно 1.Затем после соответствующего пересчета других строк путем вычитания из них ведущей строки с нужным коэффициентом (операции вычитания и нормализации нам пригодятся для обратной задачи) получаем новую таблицу 3.2. В правом столбце этой таблицы появились координаты вершины $F(X_1 = 0 : X_2 = 1)$, находящиеся на оси X_2.

Табл.3.2. Полученные данные для вершины $F(X_1 = 0 : X_2 = 1)$

	$\boldsymbol{X_1}$	X_2	S_1	S_2	S_3	S_4	ρ
Z	$-\frac{10}{9}$	*0*	0	1	0	0	1
S_1	***3***	*0*	1	-2	0	0	6
X_2	***-1***	*1*	*0*	*1*	*0*	*0*	*1*
$\boldsymbol{S_3}$	***1***	***0***	***0***	***-1***	***1***	***0***	***1***
S_4	***7***	*0*	0	-3	0	1	21

Полученный результат движения по контуру выпуклого многогранника по часовой стрелке, что видно на рис 3.1, позволяет повторить расчеты, выбирая ведущим столбец 2, а ведущей строкой-строку 5, что позволяет вновь повторить операции и получить таблицу 3 для вершины *E*. Заметим, что в правом столбце впервые появились ненулевые значения $X_1 = 1 : X_2 = 2$, являющиеся координатами этой вершины, находящейся на пересечении прямых линий 2 и 3.

Табл.3.3. Полученные данные для вершины $E(X_1 = 1, X_2 = 2)$

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	0	0	0	$-\frac{1}{9}$	$\frac{10}{9}$	0	$\frac{19}{9}$
S_1	0	0	1	***1***	-3	0	3
X_2	0	1	0	***0***	1	0	2
$\boldsymbol{X_1}$	1	***0***	***0***	***-1***	***1***	***0***	***1***
S_4	0	0	0	***4***	-7	1	14

В строке целевой функции остался один отрицательный коэффициент ($-1/9$), что определяет ведущим столбец 5, а строка 3, став ведущей, определяет переход в следующую вершину *D* с координатами $X_1 = 4 : X_2 = 2$, появляющимся в последнем столбце таблицы 3.4. Это соответствует точке пересечения линий 1 и 3 и концу решения прямой задачи.

Обратим особое внимание в полученной таблице на те элементы, которые нам известны в обратной задаче – это всего лишь значения $X_1 = 4$ и $X_2 = 2$, не зная пока, из каких неравенств, ставивших равенствами, они получились.Получение остальной части данных потребует дополнительных расчетов и, как показано ниже, все элементы за исключением строки целевой

функции, будут восстановлены абсолютно точно, а коэффициенты C_1 и C_2 будут ограничены найденной группой неравенств при указанной в качестве оптимальной вершины *D*. Поэтому, допуская возможность нужного восстановления табл.3.4, сделаем главную попытку решения обратной задачи – найдем координаты смежных свершиной *D* соседних вершин, если в нашем распоряжении имеется таблица 3.4, в которой строка целевой функции отсутствует.

Таблица 3.4. Полученные данные для выбранной оптимальной вершины *D* $(X_1 = 4 : X_2 = 2)$

	X_1	X_2	S_1	$\boldsymbol{S_2}$	S_3	S_4	ρ
Z	0	0	$\frac{1}{9}$	***0***	$\frac{7}{9}$	0	$\frac{22}{9}$
$\boldsymbol{S_2}$	***0***	***0***	***1***	***1***	***-3***	***0***	***3***
X_2	0	1	0	***0***	1	0	2
X_1	1	0	1	***0***	-2	0	4
S_4	0	0	-4	***0***	5	1	2

3.3 Процедура определения координат ближайших вершин при заданном оптимальном решении прямой задачи

Прежде чем приступить к необходимым рассуждением, немного упростим талблица 3.4, убрав неизвестную пока строку целевой функции и обозначив нулевые переменные S_1 и S_3 в вершине *D* через специальные координаты y_1 и y_2. Тогда получив матрицу II, показанную в виде таблица 3.5, проведем следующие рассуждения.

Таблица 3.5. Матрица II для нахождения соседних вершин

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
S_2	0	0	1	1	-3	0	3
X_2	0	1	0	0	1	0	2
X_1	1	0	1	0	-2	0	4
S_4	0	0	-4	0	5	1	2

Сравним таблицы 3.4 и 3.5, относящихся к соседним вершинам *E*и *D*. Видно, что для перехода из вершины *E* к вершин *D* одна из нулевых переменных S_2 определяла ведущий столбец,(чтобы указать направление движения по контуру многогранника), с помощью которого находилась ведущая строка по известному правилу прямого симплекс-метода, и затем эта переменная исключалась из нулевых, в которые попадала переменная S_1. Тем самым определялась соседняя вершина *D* на пересечении прямых линий 1 и 3, что соответствует вновь полученным нулевым переменным S_1 и S_3.

Таким образом, замена одной из нулевых переменных на новую описанным способом обеспечивает попадание в ближайшую вершину. Значит, если взять в матрице IIлюбую из переменных y_i, то можно найти один из ведущих столбцов, а затем и ведущую строку и координаты соседней вершины.

Возьмем столбец 4для y_1 и представим его ведущим. Для него ведущей является строка 2, т.к у нее неотрицательное отношение $3/1$ минимально. Тогда вместо S_2 ненулевой переменной станет y_1, а после пересчета матрицы IIполучим таблицу 3.6, в которой в правом столбце возникнут нужныекоординаты соседней вершины $E(X_1 = 1, X_2 = 2)$ на прямой линии 3, показанной на рис.1.

Таблица 3.6. Полученные данные для соседней вершины $E(X_1=1, X_2=2)$

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
Y_1	0	0	1	1	-3	0	3
X_2	0	1	0	0	1	0	2
X_1	1	0	0	1	1	0	1
S_4	0	0	0	4	-7	1	14

Теперь повторим те же действия с матрицей II, взяв в качестве ведущего другой столбец y_2 – для которого ведущей является строка 5 с минимальным неотрицательным отношением $2/5$. После соответствующего пересчета ведущей строки и других строк по известным правилам прямого симплекс-метода получим таблицу 3.7, в которой в правом столбце возникнут нужные координаты другой соседней вершины $C(X_1=4,4, X_2=1,6)$.

Таблица 3.7. Полученные данные для соседней вершины $C(X_1=4,4, X_2=1,6)$

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
S_2	0	0	$-\frac{7}{5}$	1	0	$\frac{3}{5}$	$\frac{21}{5}$
X_2	0	1	$\frac{4}{5}$	0	0	$-\frac{1}{5}$	$\frac{8}{5}$
X_1	1	0	$-\frac{3}{5}$	0	0	$\frac{2}{5}$	$\frac{22}{5}$
Y_2	0	0	$-\frac{4}{5}$	0	1	$\frac{1}{5}$	$\frac{2}{5}$

Показанный расчет позволяет сформулировать общую группу правил определения координат соседних вершин:

- сформировать матрицу IIбез строки целевой функции, обозначив нулевые переменные для указанной оптимальной вершины через $y_i(i=1.....n)$.

-с помощью каждой переменной y_i поочередно определить ведущий столбец, затем для него известным симплекс-методом- ведущую строку, с помощью которой пересчитать все строки матрицыII и получить *n*таблиц с указанными в правом столбце координатами $X_i(l)$ $(l = 1.....n)$ соседних вершин.

Полученный в примере ответ открывает путь составления группы неравенств (3.3), уточняющих возможные интервалы искомых коэффициентов C_1 и C_2 критерия эффективности

$$C_i X_1(0) + C_2 X_2(0) \rangle C_i X_1(1) + C_2 X_2(1)$$

$$C_i X_1(0) + C_2 X_2(0) \rangle C_i X_1(2) + C_2 X_2(2)$$

Оптимальная вершина *D* имеет координаты - $X_1(0) = 4$; $X_2(0) = 2$; вершина *E* - координаты $X_1(1) = 1$; $X_2(1) = 2$; вершина *C*- координаты $X_1(2) = 4,4$; $X_2(2) = 1,6$.Тогда получим,

$$C_1.4 + C_2.2 \rangle C_1.1 + C_2.2 \text{ или } 3C_1 \rangle 0$$

$$C_1.4 + 2C_2 \rangle C_1.4,4 + 1,6C_2 \text{ или } 0,4C_1 \langle 0,4C_2$$

$$\text{что в итоге означает } 0 \langle C_1 \langle C_2 \qquad (3.4)$$

Поэтому, если принять $C_2 = 1$, условие (3.4) позволяет для коэффициента C_1 взять любое положительное число, меньшее единицы, например в середине интервала: $C_1 = 0,5$. Непосредственные повторные расчеты прямой задачи показывают, что полученный ответ, как и прежде, соответствует вершине *D*, что обнадеживает, хотя при неотрицательности весовых коэффициентов C_i допуск (3.4) на C_1 получился в данном примере односторонним.

Теперь **можно снова вернуться к вопросу получения данных таблицы 3.4 без использования строки целевой функции,** учитывая при ее восстановлении только факт принадлежности выбранной вершины как лучшего решения.

3.4 Формирование матрицы данных для выбранной оптимальной вершины без использования строки целевой функции

Чтобы решить эту частную задачу, обратимся к исходной таблице 3.1, для которой начало координат $X_1 = X_2 = 0$ является первой опорной вершиной, с чего начинается план обхода многогранника. В прямой задаче этой ситуации, во-первых, нулевыми являются исходные переменные X_1 и X_2, и своими обозначениями они уже отличаются от ненулевых переменных S_j. В назначенной ЛПР оптимальной вершине нулевыми переменными могут быть как координаты X_i, так и переменные S_j. Поэтому стоит их, как было сказано выше, выделить для наглядности в первой строке и в левом столбце начальной таблицы 3.1 особыми координатами y_l, которые для вершины D соответствуютискусственнымпеременным S_1 и S_3, т.е. пусть $S_1 = y_1$, $S_3 = y_2$.

Во-вторых, при переходе из вершины A в соседнюю вершину осуществляется замена одной из имеющихся ненулевых переменных (например, S_2 из общего из множества, указанного в левом столбце таблица1). Эта замена осуществляется известным в прямом симплекс-методе способом назначения ведущего столбца (с помощью строки целевой функции), ведущей строки и пересчета всех строк таблицы в новую матрицу. В нашем случае отсутствия целевой функции в качестве назначаемого столбца можно взять первым любой, соответствующий одной из координат X_i, а в качестве ведущей строки-туиз принадлежащих к переменным y_j строку, у которой неотрицательное отношение $\frac{b_j}{a_{ij}}$ минимально.

В-третьих, оптимальная вершина преимущественно не является ближайшей к началу координат, и поэтому при n=2 отличается не одной, а двумя ненулевыми переменными- вместо X_1 и X_2 ими должны стать переменные y_1 и y_2, как это показано в таблице 3.8.

Таблица 3.8. Матрица **I** исходных данных с указанием нулевых переменных в назначенной оптимальной вершине

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
Y_1	1	2	1	0	0	0	8
S_2	-1	+1	0	1	0	0	1
Y_2	0	1	0	0	1	0	2
S_4	4	3	0	0	0	1	24

В связи с этим **предлагается перенестиначало преобразований из вершины *A* сразу в оптимальную вершину *D* путем двойного пересчета таблицы 3.1.** При этом считается, что нам задана симплекс-таблица в виде тех же строк, но строка целевой функции пустая, так как весовые коэффициенты критерия не назначены. Однако имеются координаты лучшей вершины многогранника. Если известно, что оптимальная вершина находится в какой-то точке, то исскуственные переменные в этой точке соответствуют двум равенствам и равны нулю. Значит, нам просто даны еще столбцысимплекс-таблицы, в которых исскуственные переменные равно нулю.

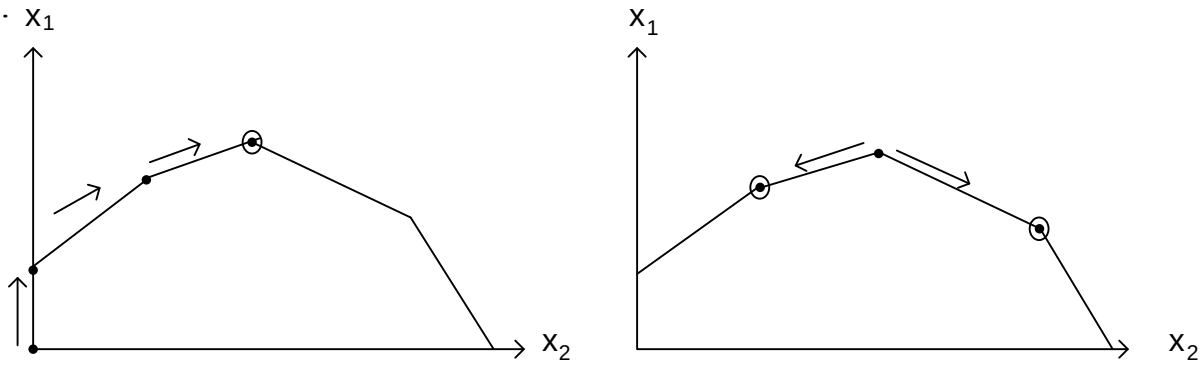


Рис 3.3. Сравнение действий при поиске ближайшихсоседних вершин в прямом и предложенном обратном симплекс-методе

Разница в известном и предложенном методах состоит в следующем :

При решении прямой задачи поиск лучшей вершины начинается с начала координат при движении по контуру многогранника. При решении обратной задачи необходимо начать движение с лучшей вершины, а не с начала координат, а затем, как и в прямом методе, нужно искать ближайшие соседние вершины.

Обратный метод - зная оптимальную вершину, смотрим все ближайшие вершины, рядом находящиеся, получаем координаты соседних вершин, как показано на рис 3.3. А дальше уже можно составить необходимые неравенства.

Отличие состоит в том, что движение при поиске начинается не сначала координат, а с оптимальной вершины. Во-вторых, это движение происходит во всех возможных направлениях движения. Таким образом, обратная задача будет решена, если:

1 – есть правило переноса расчета из начала координат в оптимальную вершину;

2 – есть правило выбора **всех** сближающих соседних вершин;

3 – пишутся неравенства для всех соседних вершин согласно критерию(3.1), куда входят все неизвестные коэффициенты,в частности C_1 и C_2. Получается система неравенств или условия двухстороннего допуска.

Иными словами, предлагается новый “опорный план” с началом в вершине *D*.С этой целью исходную таблицу 3.1 преобразуем в матрицу **I** этого опорного плана, которая не содержит строки целевой функции, а искусственные переменные S_1 и S_3 в левом столбце и правой строке заменены на нулевые переменные y_1 и y_2, играющие роль системы отсчета координат вместо X_1 и X_2.

Нужно сразу заметить, что в самом начале решения нам известны лишь координаты $X_1(0)=4$ и $X_2(0)=2$ вершины D без указания того, какие неравенства для этой вершины превращаются в равенства, или какие переменные становятся нулевыми. Но это нетрудно сделать предварительно, подставив значения $X_i(0)$ в имеющиеся ограничения (3.2). В частности, в данном примере из условий (3.2) получим при $X_1=4:\ X_2=2$

$$1.\ S_1=8-X_1-2X_2=0$$

$$2.\ S_2=1+X_1-X_2=3$$

$$3.\ S_3=2-X_2=0$$

$$4.\ S_4=24-4X_1-3X_2=2$$

Видно, что нулевыми переменными в вершине D стали S_1 и S_3, которые переобозначены через y_1 и y_2. В общем случае в имеющиеся линейные неравенства

$$\sum_{i=1}^{n} a_{ij}x_i \le b_j;\ j=1...k \qquad (3.5)$$

вводятся искусственные переменные S_j, чтобы получить равенства, решение которых при заданных координатах $X_i(0)$ оптимальной вершины дает ответ, какие из этих переменных нулевые. Число этих переменных $S_l=0(l=1....n)$ равно n, и их нужно обозначить через y_l, если для них выполняются условия

$$S_l=y_l=b_l-\sum_{i=1}^{n} a_{il}x_i(0)=0;l=1,.....n \qquad (3.6)$$

Проведем теперь необходимые расчеты. Возьмем в качестве любой из координат X_i переменную X_1, что определяет ведущим столбец 2. В этом столбце среди строк 2 и 4, принадлежащих переменным y_1 и y_2, ведущей

является строка 2, не требующая дополнительной нормализации. Поэтому вычтем ее из остальных строк и получим новую промежуточную таблицу 3.9.

Таблица 3.9. Таблица данных после замены переменной y_1 на X_1

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
X_1	1	2	1	0	0	0	8
S_2	0	3	1	1	0	0	9
Y_2	0	1	0	0	1	0	2
S_4	0	-5	-4	0	0	1	-8

После обращения к столбцу с X_1 переходим к другой нулевой переменной X_2 - ведущим стал столбец 3, а в нем принадлежащаяк y_2 строка 4 –единственная, имеющаянеотрицательный элемент 1, поэтому она становится ведущей. Это позволяет без нормализации пересчитать остальные строки. Нетрудно убедиться, что после расчетов из таблицы 3.9 формируется матрица **II**, представленная выше таблицей 3.5.

Можно также пояснить геометрический смысл сделанных преобразований с помощью рис 3.1 – переход от данных таблицы 3.8 к таблице 3.9 означает вначале движение от точки *A* в точку *L*(на линию 1), а затем - точку *D*.

Рассмотренный способ пересчета при перехода из начала координат X_i в новую опорную вершину можно теперь обобщить на случай произвольного числа nпеременных X_i в виде **следующих общих правил перехода в заданную оптимальную вершину**[29]:

- в качестве ведущего столбца поочередно используются столбцы с искомыми ненулевыми переменными X_i.

- в назначенном столбце анализируются только те элементы, которые принадлежат строкам с нулевыми переменными y_j (j= 1,…,n) в назначенной вершине.

- из всех строк ведущей является строка, имеющая главным неотрицательный элемент, а отношение b_j / a_j коэффициентов правом столбце к этому элементу минимально;

- после выбора ведущей строки стоящая в левом столбце переменная y_l заменяется на переменную X_i из ведущего столбца;

- ведущая строка нормализуется, чтобы ее главный элемент стал равным1;

- остальные строки матрицы пересчитываются известным в прямом симплекс-методе способом;

- во вновь найденной матрице находится новый ведущий столбец (i=2 …….n) до тех пор, пока небудут учтены все переменные X_i, и процедура выявления новой ведущей строки и пересчета остальных срок повторяется.

- число повторяющихся циклов пересчета равно числу искомых ненулевых переменных, в результате чего будет сформирована матрица **II**.Приведенных вычислительных операций достаточно, чтобы целиком описать процесс решения обратной задачи в нужной последовательности, представленной ниже на рис.3.4 в виде блок-схемы обратного симплекс-метода.

3.5 Общая процедура обратного симплекс-метода решения задачи линейного программирования

Блок-схема состоит из четырех основных блоков. Первый блок определяет необходимые действия при условии, что неравенствам (3.5), определяющим содержание строк таблицы 3.1, соответствует выпуклый

многогранник, находящийся внутри *n*– мерного параллелепипеда, размеры которого будут уточнены ниже. Назначаемая ЛПР заданная вершина с известными координатами $X_i(0)$ принадлежит этому многограннику, для которой значение неизвестной целевой функции *Z* максимально.

Относительно коэффициентов C_i линейной свертки критерия известно лишь то, что все они положительны. Тогда при заданных неравенствах (3.5) и координатах $X_i(0)$ определяются нулевые переменные y_l; $l = 1....n$, и с их помощью формируется исходная матрица **I**, поставив их в соответствующие элементы левого столбца и первой строки.

Второй блок осуществляет перенос условий задачи из начала координат в

заданную оптимальную вершину с помощью последовательногоперебора переменных X_i .В отличие от прямого симплекс-метода при переходе в ближайшую соседнюю вершину, совокупность операций по выбору ведущих столбца и строки и пересчету остальных строк в обратном симплекс-методе выполняется нужное число раз, вместо одного цикла в прямом методе. При этом

в каждом *i* - том цикле вычисления проводятся над преобразованной матрицей **II**, полученной на предыдущем шаге. Сами циклы образуются путем последовательного использования столбцов, принадлежащих указанным в верхней первой строке переменным X_i. При этом необходимо сделать следующее замечание – в ряде редких случаев часть координат X_i $(i = 1....k)$ оптимальной вершины может быть назначена нулевыми самим ЛПР. Например,

если в примере.1 выбрана вершина *F*, то $X_1 = 0$, а для вершины *R*значение $X_2 = 0$. Поэтому эти переменные в циклах вычисления матрицы **II** не участвуют, в связи с чем на рис 3.2 в блоке 3 специально отмечено, что

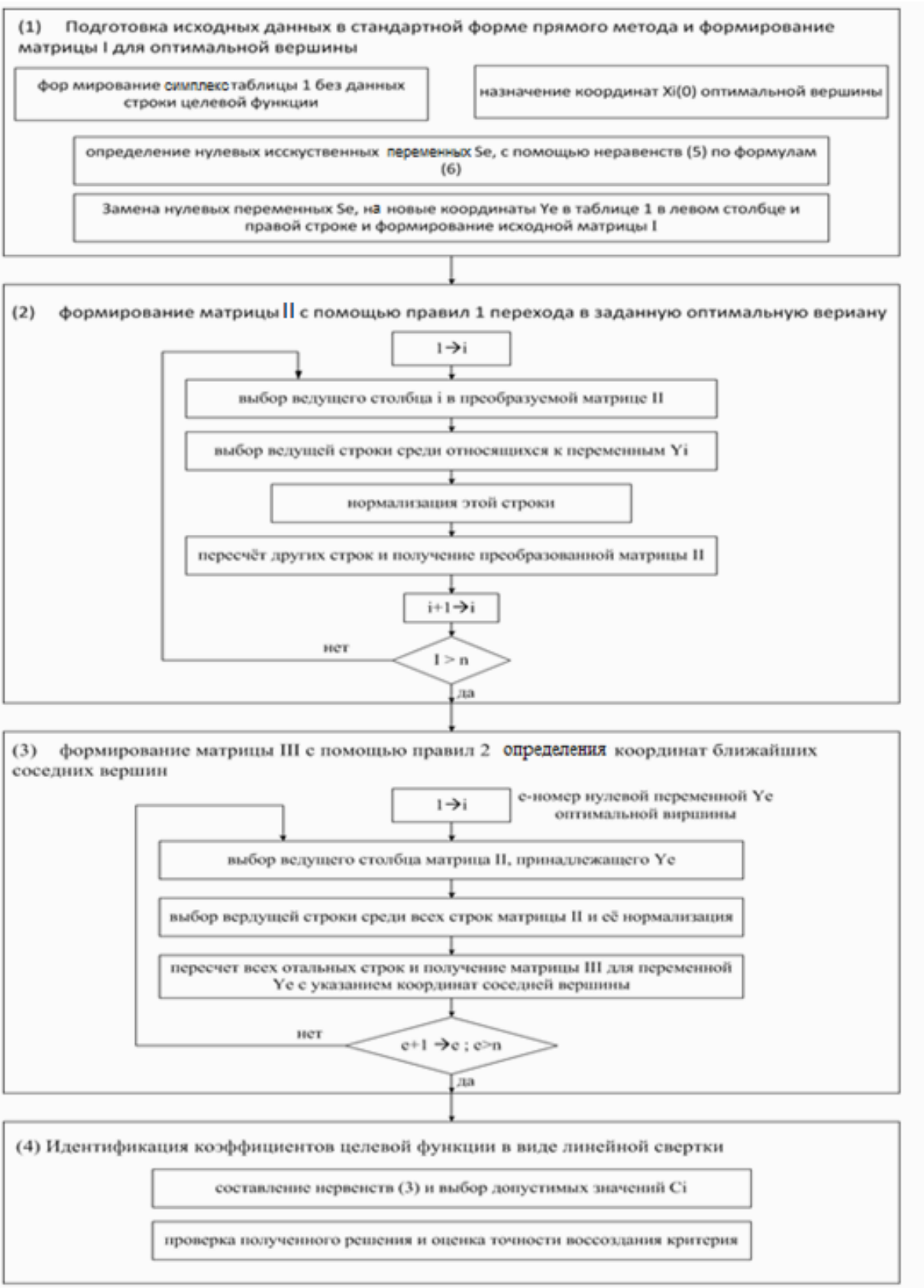


рис 2. Блок-схема вычислительных операций обратного симплекс - метода при неизвестной целевой функции

Рис.3.4 Блок-схема вычислительных операций обратного симплекс-метода при неизвестной целевой функции

i – это номер нулевой переменной, участвующей в расчетах, число циклов будет равно (*n*-*k*), а *k* - число нулевых переменных в оптимальной вершине. Более подробно этот случай будет рассмотрен ниже примере 3.4.

В третьем блоке задающими последовательный перебор являются нулевые переменные в первом столбце. При этом на каждом шаге преобразованиям подвергается одна и та же исходная матрица II. Зато результатом каждого шага является получение “своей” *l* – той матрицы III, в которой в правом столбце указаны координаты *l* – той соседней вершины. Поэтому в конце третьего блока вычислений получается n – матриц III.

Четвертый блок является завершающим. После составления неравенств(3.3), дающих интервальную оценку искомых коэффициентов C_i, можно назначить их точечные значения внутри интервалов и затем для проверки найти решение прямой задачи с восстановленной целевой функцией. Если полученный результат альтернативного выбора совпадает с заданным ЛПР ответом, то идентификацию можно признать успешной, а найденная форма критерия может быть использована для других целей. Например, можно сравнить полученное значение *Z* с некоторым идеальным случаем попадания координат X_i в вершину *n*-мерного параллелепипеда, “противостоящую” началу координат, и в результате сравнения оценить эффективность оптимизации при назначенных ограничениях (3.5) (см. вершину *W* на показанных ниже рис 3.4 и рис 3.5).

Можно использовать также полученную целевую функцию при других неравенствах (3.5) в новых условиях, которые могут меняться, что часто бывает в инженерной практике. Тогда в динамической обстановке удастся решать новую прямую задачу линейного программирования с известным критерием, что упростит необходимые действия. Сравнительный анализ прямого и обратного симплекс-методов и проверка их эффективности проиллюстрированы для примера 1 совокупностсовокупностью таблиц 3.10.

Картина сравнительно анализа показывает, чтовершина, найденная с помощью полученной обратным методом целевой функции, совпала с исходной вершиной *D*, однако значение целевой функции *Z* имеет сильное расхождение, что очевидно из-за неточного интервального оценивания весовых коэффициентов свертки C_i . Поэтому целесообразно проанализировать, от чего эта точность зависит, и понять, есть ли способы ееповышения.

Таблица 3.10. Картина сравнительного анализа прямого и обратного симплекс-методов

Прямой симплекс-метод при $C_1 = \frac{1}{9}$; $C_2 = 1$

(1) Табл 1. Исходные данные для начала координат

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	$-\frac{1}{9}$	-1	0	0	0	0	0
S_1	1	2	1	0	0	0	8
S_2	-1	1	0	1	0	0	1
S_3	0	1	0	0	1	0	2
S_4	4	3	0	0	0	1	24

(2)Табл 2. Данные для вершины *F*

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	$-\frac{10}{9}$	0	0	1	0	0	1
S_1	3	0	1	-2	0	0	6
X_2	-1	1	0	1	0	0	1
S_3	1	0	0	-1	1	0	1
S_4	7	0	0	-3	0	1	21

Обратный симплекс-метод без целевой функции

(1)Матрица I с нулевыми переменными в оптимальной вершине

	X_1	X_2	Y_1	S_2	Y_2	S	ρ
Y_1	1	2	1	0	0	0	8
S_2	-1	1	0	1	0	0	1
Y_2	0	1	0	0	1	0	2
S_4	4	3	0	0	0	1	24

(2)Матрица II для оптимальной вершины с целью нахождения соседних вершин

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
S_2	0	0	1	1	-3	0	3
X_2	0	1	0	0	1	0	2
X_1	1	0	1	0	-2	0	4
S_4	0	0	-4	0	5	1	2

Повторный расчет при полученной целевой функции

(1)Таблица 1. Новые исходные данные

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	$-\frac{1}{2}$	-1	0	0	0	0	0
S_1	1	2	1	0	0	0	8
S_2	-1	1	0	1	0	0	1
S_3	0	1	0	0	1	0	2
S_4	4	3	0	0	0	1	24

(2)Табл 2. Новые данные для вершины *F*

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	-1,5	0	0	1	0	0	1
S_1	3	0	1	-2	0	0	6
X_2	-1	1	0	1	0	0	1
S_3	1	0	0	-1	1	0	1
S_4	7	0	0	-3	0	1	2

(3)Табл 3. Данные для вершины E

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	0	0	0	$-\frac{1}{9}$	$\frac{10}{9}$	0	$\frac{19}{9}$
S_1	0	0	1	1	-3	0	3
X_2	0	1	0	0	1	0	2
X_1	1	0	0	-1	1	0	1
S_4	0	0	0	4	-7	1	14

(3)Табл 3. Матрица III для вершины E

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
Y_1	0	0	1	1	-3	0	3
X_2	0	1	0	0		0	2
X_1	1	0	0	1	1	0	1
S_4	0	0	0	4	-7		4

Матрица III для вершины C

	X_1	X_2	Y_1	S_2	Y_2	S_4	ρ
S_2	0	0	-1,4	1	0	0,6	4,2
X_2	0	1	0,8	0	0	-0,2	1,6
X_1	1	0	-0,6	0	0	0,4	4,4
Y_2	0	0	-0,8	0	1	0,2	0,4

(3)Табл 3. Новые данные для вершины E

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	0	0	0	$-\frac{1}{2}$	$\frac{3}{2}$	0	$\frac{5}{2}$
S_1	0	0	1	1	-3	0	3
X_2	0	1	0	0	1	0	2
X_1	1	0	0	-1	1	0	1
S_4	0	0	0	4	-7	1	14

Прямой симплекс-метод при $C_1 = \frac{1}{9};\ C_2 = 1$

Обратный симплекс-метод без целевой функции

Повторный расчет при полученной целевой функции

(4)Табл 4. Данные для оптимальной вершины D

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	0	0	$\frac{1}{9}$	0	$\frac{7}{9}$	0	$\frac{22}{9}$
S_2	0	0	1	1	-3	0	3
X_2	0	1	0	0	1	0	2
X_1	1	0	1	0	-2	0	4
S_4	0	0	-4	0	5	1	2

(4)Составление неравенств (3)

$$\begin{cases} 4C_1 + 2C_2 > C_1 + 2C_2 \\ 4C_1 + 2C_2 > 4{,}4C_1 + 1{,}6C_2 \end{cases}$$

$$0 < C_1 < C_2$$

(4)Табл 4. Новые данные для оптимальной вершины D

	X_1	X_2	S_1	S_2	S_3	S_4	ρ
Z	0	0	$\frac{1}{2}$	0	0	0	4
S_2	0	0	1	1	-3	0	3
X_2	0	1	0	0	1	0	2
X_1	1	0	1	0	-2	0	4
S_4	0	0	-4	0	5	1	2

(5)Ответ

$X_1 = 4\ ;\ X_2 = 2\ ; Z = 2{,}44$

(5)Ответ

пусть $C_2 = 1$; тогда $C_1 = 0{,}5$

(5)Ответ

$X_1 = 4\ ;\ X_2 = 2\ ; Z = 4$

3.6 Оценка точности решения обратной задачи линейного программирования при одной заданной оптимальной вершине

Анализируя системунеравенств (3.3) для искомых коэффициентов C_i, становится ясным, что наибольшая точность идентификации достигается в случае, если этой системе соответствует группа двухсторонних допусков на каждой коэффициент. Покажем это в частных случаях.

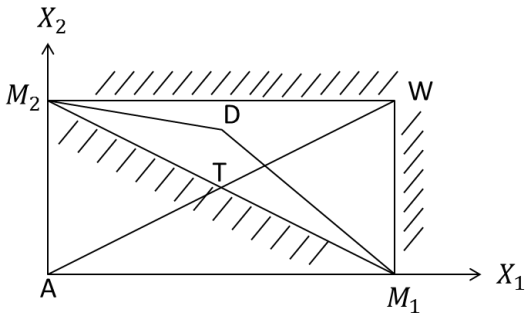


Рис 3.5. Расположение оптимальной и двух соседних вершин в двумерном случае.

На рис3.5 рассмотрен случай при *n*=2. Точки M_1 и M_2 – значения координат прямоугольника, которые надо еще уточнить. Диагональ, соединяющая эти точки – граница, выше которой должен находиться выпуклый многогранник,новнутри выделенного треугольника. Точка *D* - заданная оптимальная вершина, а диагональ, соединяющаяначало координат *A* и “верхнюю грань” *W* (при которой целевая функция *Z* в идеальном случае достигнет абсолютного максимума), будет использоваться как верный признак качества решения обратной задачи.

Чтобы доказать это, представим точку *D* расположенной на этой диагонали между точками *T* и *W*. Тогда, считая соседними точки M_1 и M_2, можно записать неравенства (3.3) в виде

$$C_1X_1(0)+C_2X_2(0)\rangle C_1M_1 \text{ или } C_1[M_1-X_1(0)]\langle C_2X_2(0)$$

$$C_1X_1(0)+C_2X_2(0)<C_2M_2] \text{ или } C_1X_1(0)\rangle C_2[M_2-X_2(0)]$$

Откуда получаем двусторонний допуск на отношение $\frac{C_2}{C_1}$

$$\frac{M_1 - X_1(0)}{X_2(0)} < \frac{C_2}{C_1} < \frac{X_1(0)}{M_2 - X_2(0)} \tag{3.7}$$

Неравенство (3.7) позволяет сразу убедиться, что если переместить точку *D*по диагонали в точку *W*($X_1(0) = M_1, X_2(0) = M_2$,), то получим $0 < C_2 < 1 < \infty$, т.е обратный симплекс-метод нам не даст никакого уточнения. Если же наоборот, переместить точку *D* в точку *T*, то левая и правая границы допуска в (3.7) становятся равными друг другу и достигают значения $\frac{M_1}{M_2}$. Значит, нахождение заданной вершины *D* по отношению к точкам *T*и*W* определяют относительную точность, которую можно оценить с помощью следующего коэффициента относительной эффективности Э обратного симплекс-метода.

$$Э = 2\left[1 - \sqrt{\frac{\sum_{i=1}^{n} X_i^2(0)}{\sum_{i=1}^{n} M_i^2}}\right] \tag{3.8}$$

В частности, для примера 1 максимальное значение координаты X_1 в найденном на рис. 3.1 многограннике равно $M_1 = 6$, для координаты X_2 это значение равно $M_2 = 2$, а координаты оптимальной вершины *D* нам заданы: $X_1(0) = 4;\ X_2(0) = 2$. Поэтому для точки *D* получим $Э = 0{,}57$, а соответственно для точки T значение $Э = 1$, для точки *W* значение $Э = 0$.

Остается не до конца исследованным вопрос – какие значения M_i ребер *n*-мерного параллелепипеда нужно брать при расчетах. В приведенном выше расчете предполагалось, что после воссоздания целевой функции все вершины многогранника с помощью прямого симплекс-метода становятся известными, и затем можно определить максимальные значения переменных X_i, приняв из за M_i.Но это не единственный способ. Можно воспользоваться

максимальными значениями X_i, считая неравенства (3.3) за равенства. В этом случае не потребуется решать прямую задачу целиком, однако полученные значения M_i будут завышены (в примере 1 согласно рис 3.1 значения $M_1 = M_2 = 8$ вместо истинных $M_1 = 6$, $M_2 = 2$.)

Другим способом является расчет ближайших к началу координат вершин, если поочередно принять гипотезу $C_i = 1$; $C_{i \neq j} = 0$; $j = 1....n$ и решать прямую задачу *n*раз, но по одному шагу расчетов в каждой задаче. Тогда оценки M_i будутзанижены (в примере 1 этому способу соответствуют вершины *F*и *R* с координатами $M_1 = 6$; $M_2 = 1$). В данной работе выбран первый способ полного решения прямой задачи, хотя этот вопрос требует дальнейших исследований.

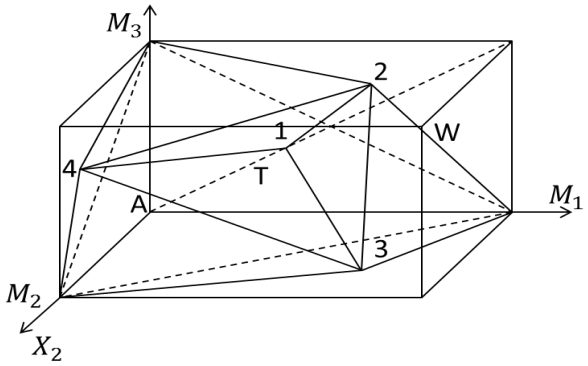


Рис 3.6. Расположение оптимальной и трех соседних вершин при *n*=3

Перейдем к рассмотрению случая *n*=3, проиллюстрированного на рис 3.6. Согласно рисунку, на главной диагонали *AW* точка *T* находится на границе плоскости, имеющей вид показанного пунктиром треугольника с вершинами M_1, M_2, M_3. Если назначенная оптимальной является вершина 1, лежащая на главной диагонали в виде точки *T*, то эффективность обратной задачи совершенно аналогично двухмерному случаю может быть оценена с помощью формулы (3.8). Вместе с тем можно убедиться с помощью расчетов, что показанная на рис 3.6 промежуточная позиция вершины 1 вне треугольной призмы M_1, M_2, M_3, но внутри параллелепипеда позволяет

представить неравенства (3.3) в форме двухстороннего допуска на искомые коэффициенты критерия C_i. В частности, если принять координаты точки 1 $X_0(i) = m_0$, а координаты точки 2,3,4 как $X_l(k-1) = X_l(k) = \max; X_l(k+1) = 0$, но неравенства (3.3) можно свести к следующим неравенствам

$$C_1 + C_2 + C_3 \rangle (C_1 + C_2) d$$

$$C_1 + C_2 + C_3 \rangle (C_2 + C_3) d$$

$$C_1 + C_2 + C_3 \rangle (C_1 + C_3) d$$

или получить оценки, ограниченные сверху и снизу

$$(d-1)\sum_{i \neq j}^{3} C_i \langle C_j \langle (\sum_{i \neq j}^{3} C_i - d \max_{i \neq j} C_i) \frac{1}{d-1} \qquad (3.9)$$

3.7 Примеры использования обратного симплекс-метода в задаче обеспечения безопасных дистанций между самолетами в воздушном эшелоне при заходе на посадку

При заходе на посадку самолеты гражданской авиации должны подлетать к аэродрому в заданном коридоре на безопасных дистанциях друг от друга. Существующие летные нормы определяют минимально допустимую дистанцию в обычных штатных полетных ситуациях. Однако при ухудшении погодных условий, недостаточном запасе топлива на борту и других экстренных случаях наземная авиадисперческая служба принимает особые альтернативные решения по дополнительному маневрированию одного, двух или трех самолетов, исходя как из условий безопасности движения в воздушном эшелоне, так и учитывая экономичность их полета.

Особенно часто усложненная задача принятия решений происходит при входе в коридор дополнительного самолета (например находящегося в аварийной ситуации) между двумя уже летящими в эшелоне самолетами. Примеры выбора авиадиспетчером альтернатив по времени дополнительного маневрирования самолетов известны, однако если спросить у него, каков

единый критерий безопасности и экономичности полета им используется, ответ будет отрицательный. Между тем идентификация целевой функции позволит автоматизировать действие авиадиспетчерской службы и снизить влияние человеческого фактора.

Более подробно эта проблема рассмотрена в[29, 35, 61]. В данной работе считается, что в качестве обучения используется один пример в виде координат одной заданной оптимальной вершины для одной типичной полетной ситуации. В качестве этих координат X_i используются времена Δt_i дополнительных маневров самолетов, на которые накладываются ограничения как по суммарной длительности (что определяет суммарные затраты топлива.)

$$\sum_{i=1}^{n} X_i \le b$$

так и по длительности каждого маневра

$$X_i \le a$$

где, *a*- допустимая разрешенная величина, уменьшающаяся по мере приближения кместупосадки, соответствующему точке принятия окончательного решения о посадке.

Считается также, что принимаемые решения альтернативны – самолет либо не маневрирует, либо маневрирует малое, среднее или максимально допустимое время.

Пример 2. Пусть взаимодействуют два самолета, т.е *n*=2, при следующих условиях

$$Z = C_1 X_1 + C_2 X_2 \to \max;$$

$$X_1 \le a; X_2 \le a \qquad (3.10)$$

$$X_1 \ge 0; X_2 \ge 0$$

$$X_1 + X_2 \le b \ \text{где} (a < b < 2a)$$

где X_1 и X_2 - выбираемое время дополнительного маневра каждого самолета. Нужно сразу заметить, что в примере 2 и остальных примерах в качестве переменных выступают **времена** Δt_i выполнения маневров судами в эшелоне, но это **не есть параметры безопасности и экономичности полета**. Идентифицированные с помощью обратной задачи параметры C_1 иC_2 косвенно зависят от интересующих нас коэффициентов m_1 иm_2, входящих в единый критерий оптимальности (1.5), и эти коэффициенты нужно доопределить,где, m_1 – коэффициент штрафа, связанный с оценкой экономичности поелта,m_2 – коэффициент, связанный с оценкой безопасности полета.

Пояснения по этому вопросу изложены ниже в параграфе 3.8.

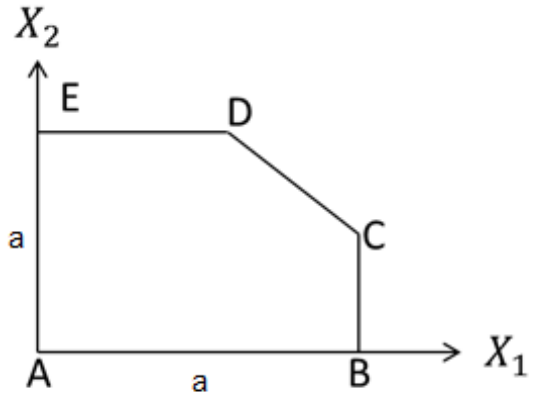


Рис 3.7. Выпуклый многогранник при трех альтернативах полета двух самолетов

Критерий Z подразумевает условие максимума в линейной свертке безопасности и экономичности полета, коэффициенты C_i - неизвестны, выпуклый многогранник представлен на рис 3.7. У каждого самолета есть три альтернативы полета – лететь без маневра, маневрировать в течение малого времени (b-a) и максимально возможного времениa. В качестве примера правильныхдействий диспетчера возьмем одну из вершин – D или

C. В частности,вершина *D* имеет координаты $X_1(0) = b - a; X_2(0) = a$.Тогда матрицы **I** и **II** имеют вид таблиц 3.11.

Таблица 3.11. МатрицыI и II для оптимальной и соседних вершин в примере 2

Матрица**I**

	X_1	X_2	S_1	Y_1	Y_2	ρ
S_1	1	0	1	0	0	a
Y_1	0	1	0	1	0	a
Y_2	1	1	0	0	1	b

матрица **II**

	X_1	X_2	S_1	Y_1	Y_2	ρ
S_1	0	0	1	1	-1	2a-b
Y_2	0	1	0	1	0	a
Y_1	1	0	0	-1	1	b-a

Нетрудно убедиться, что после вычисления двух матриц **III** с помощью обратного симплекс-метода соседними будут вершины $C(X_1(1) = a, X_2(1) = b - a)$ и E $(X_1(2) = 0; X_2(2) = a)$.Это позволяет составить неравенства (3.3) в следующем виде

$$\begin{cases}(b-a)C_1 + aC_2 \rangle aC_1 + (b-a)C_2 \\ (b-a)C_1 + aC_2 \rangle C_2 a\end{cases} \text{ или } C_2 \rangle C_1 \rangle \mathbf{O} \qquad (3.11)$$

Видно, что двухсторонний допуск имеется, но точность решения обратной задачи низкая. Нужно также подчеркнуть, что вершины *E* и *B* в качестве примеров для идентификации никакого уточнения не дают.

Пример 3. Пусть взаимодействуют три самолета – один входящий в эшелон, один впереди и один сзадилетящий (*n*=3) при следующих условиях

$$Z = C_1X_1 + C_2X_2 + C_3X_3 \to \max$$

$$0 \le X_i \le a; i = 1, \ldots, 3 \qquad (3.12)$$

$$X_1 + X_2 + X_3 \le b(b < 3a)$$

Выпуклый многогранник представлен для этого случая на рис. 3.8.

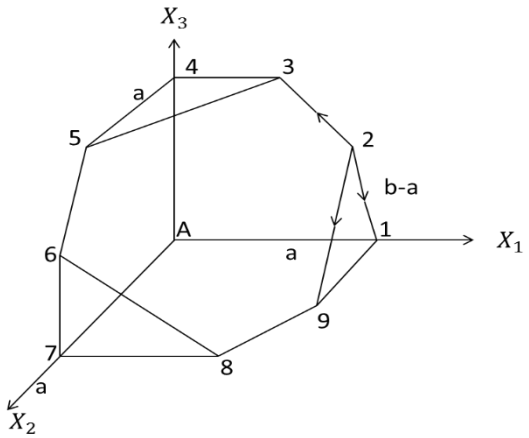


Рис 3.8. Выпуклый многогранник при трех альтернативах полета трех самолётов

Из рисунка видно, что кроме начала координат A имеется 9 вершин, из них вершины 1, 4, 7 – недающие уточнения, а остальные могут быть примерами для обратной задачи.

В частности, для вершины 2 с координатами $X_1(0)=a; X_2(0)=0; X_3(0)=b-a$ исходная матрица **I** имеет вид таблицы 3.12.

Таблица 3.12. Матрица I для оптимальной вершины в примере 2

	X_1	Y_3	X_1	Y_1	S_2	S_3	Y_2	ρ
Y_1	1	0	0	1	0	0	0	a
S_2	0	1	0	0	1	0	0	a
S_3	0	0	1	0	0	1	0	a
Y_2	1	1	1	0	0	0	1	b

Первые две координаты и y_2 появились из-за того, что равны нулю переменные S_1 и S_4 , т.к в точке 2 имеют место равенства $X_1=a; X_2=0; X_1+X_3=b$. Третьей нулевой переменной в точке 2 является

координата X_2, поэтому обозначим ее через y_3. Сформируем матриц II, используя правила 1 с помощью лишь двух столбцов, относящихся к переменными X_1 и X_3. Тогда получим результат в правом столбце этой матрицы $X_1 = a; X_2 = 0; X_3 = b - a$, что соответствует координатам оптимальной вершины в точке 2.

Не вдаваясь в подробности расчетов матриц **III**, нетрудно убедиться, что соседними вершинами являются ближайшие точки 1, 3, 9 если признать, что найденная координата $y_3 = b - a$ есть переменная X_2. Составление неравенств (3.3) в примере 3 приводит к трехступенчатому допуску.

$$0 < C_2 < C_3 < C_1 \tag{3.13}$$

Однако и в этом случае точность идентификации недостаточна, поэтому рассмотрим следующий пример.

Пример 4. Пусть при взаимодействии двух самолетов каждый из них может изменять дистанцию между ними, осуществляя дополнительный маневр тремя способами – с малым, средним и максимальным временем при следующих условиях

$$\begin{aligned} & C_1 X_1 + C_2 X_2 \to \max \\ & X_1 \le a; X_2 \le a \\ & X_1 + X_2 \le b \left(a < b < 2a \right) \\ & X_1 + kX_2 \le d \left(a < d < b; k < 1, d > kb \right) \\ & X_2 + kX_1 \le d \end{aligned} \tag{3.14}$$

Выпуклый многогранник для этого случая показан на рис 3.9и содержит кроме начала координат *A* еще 6 вершин.

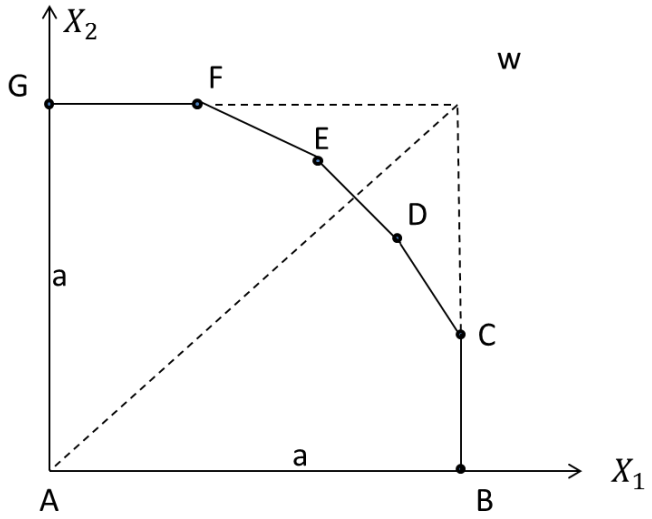


Рис 3.9. Выпуклый многогранник при трех-альтернативном маневре самолетов

Участки *GF*и *BC* соответствуют максимальному ограничению по длительности маневраодного самолета, участок *ED* учитывает условие суммарной экономичностиманевров двух самолетов. Участки *EF* и *CD* указывают, что чем большее время маневрирует один самолет, тем меньшее время приходится маневрировать другому, что оправдано в процессе их “взаимопомощи”.

Пусть задана вершина *D* в качестве оптимальной с известными координатами $X_1(0) = \frac{d - kb}{1 - k}$; $X_2(0) = \frac{b - d}{1 - k}$. При превращении части неравенств (3.14) в равенства можно получить две нулевые переменные y_1 и y_2, соответствующие прямым линиям *ED* и *DC*. Тогда матрицы **I** и **II** имеют вид таблиц 3.13.

Нетрудно видеть, что в последнем правом столбце матрицы **II** в двух нижних строках указаны заданные координаты оптимальной вершины *D*.

Таблицы 3.13. Матрицы для оптимальной и соседних вершин в примере 4.

матрица **I**

	X_1	X_2	S_1	S_2	S_3	Y_1	Y_2	ρ
S_1	1	0	1	0	0	0	0	a
S_2	0	1	0	1	0	0	0	a
S_3	k	1	0	0	1	0	0	d
Y_1	1	k	0	0	0	1	0	d
Y_2	1	1	0	0	0	0	1	b

матрица **II**

	X_1	X_2	S_1	S_2	S_3	Y_1	Y_2	ρ
S_1	0	0	1	0	0	$\frac{-1}{1-k}$	$\frac{k}{1-k}$	$\frac{a+kb-d}{1-k}$
S_2	0	0	0	1	0	$\frac{1}{1-k}$	$\frac{-1}{1-k}$	$\frac{a+d-b}{1-k}$
S_3	0	0	0	1	1	1	$\frac{1}{k}$	2d-b(1+k)
X_1	1	0	0	0	0	$\frac{1}{1-k}$	$\frac{k}{1-k}$	$\frac{d-kb}{1-k}$
Y_2	0	1	0	0	0	$\frac{-1}{1-k}$	$\frac{1}{1-k}$	$\frac{b-d}{1-k}$

Далее, вычисляя матрицы **III** для нулевых переменных y_1 и y_2, можно убедиться, что соседние вершины C и E в выпуклом многограннике имеют координаты – в точке C - это $X_1(1)=a; X_2(1)=\frac{d-a}{k}$, в точке E – это $X_1(2)=\frac{b-d}{1-k}; X_2(1)=\frac{d-kb}{1-k}$.

Поэтому можно составить неравенства (3.3) для рассматриваемого примера 4.

$$\begin{cases} \frac{d-kb}{1-k}C_1+\frac{b-d}{1-k}C_2 \rangle aC_1+\frac{d-a}{k}C_2 \\ \frac{d-kb}{1-k}C_1+\frac{b-d}{1-k}C_2 \rangle \frac{b-d}{1-k}C_1+\frac{d-kb}{1-k}C_2 \end{cases} \qquad (3.15)$$

После ряда упрощений первое и второе неравенства сводятся к условиям

$$C_1 \rangle C_2$$

$$C_1 < \frac{C_2}{k}$$

или в виде одного неравенства получим двухсторонний допуск

$$k < \frac{C_2}{C_1} < 1 \qquad (3.16)$$

Получив интервальную оценку коэффициента C_2при C_1=1 и k=0,7, можно затем уточнить коэффициенты m_1, m_2,m_3 , m_4значимости в интегральном критерии (1.5),если учесть конкретную зависимость коэффициентов C_1 и C_2 от условий безопасности и экономичности полета двух самолетов, а сами значения коэффициентов принять равными C_1=1, C_2=0,8.

В заключение примера 4 оценим по формуле (3.8) эффективность решения обратной задачи. В частности, при *d=1,3 f;b=1,5 a;k=0,4a*;длина отрезка *AD* равна *0,9a*, длина диагонали *AW* равна *1,41a*. Поэтому коэффициент эффективности $Э \cong 0,7$, что соответствует успешной попытке решения обратной задачи.

3.8 Определение коэффициентов относительной важности безопасности и экономичности полета в объединенном параметрическом критерии при использовании результатов решения обратной задачи линейного программирования.

3.8.1 Постановка задачи идентификации коэффициентов критерия

После того как получены оценки коэффициентов C_i в результате решения обратной задачи линейного программирования, возникает целесообразность воссоздания критерия по отдельным примерам оптимального поведения, чтобы затем его использовать в общем случае, заменив действия диспетчера на решение прямой задачи линейного программирования в каждойновой ситуации. Альтернативный характер принимаемых решений наводит на мысль о том, что в рамках учитываемых ограничений диспетчер выбирает одну из вершин многогранника, а процедура оптимизации напоминает задачу линейного программирования.

В данной работе считается заданной не только модель критерия в виде линейной свертки выбираемых времен дополнительного маневра каждого самолета, но и когда весовые коэффициенты этой свертки также линейно зависят от неизмеряемых параметров и малого числа других измеряемых параметров модели, которые нужно идентифицировать.

Рассмотрим решение задачи при следующих условиях [61]

1. Считается, что группа самолетов при заходе на посадку осуществляет горизонтальный полет, как это показано на рис.3.10. Состав этой группы и очередность их вхождения в эшелон для посадки заранее определены.

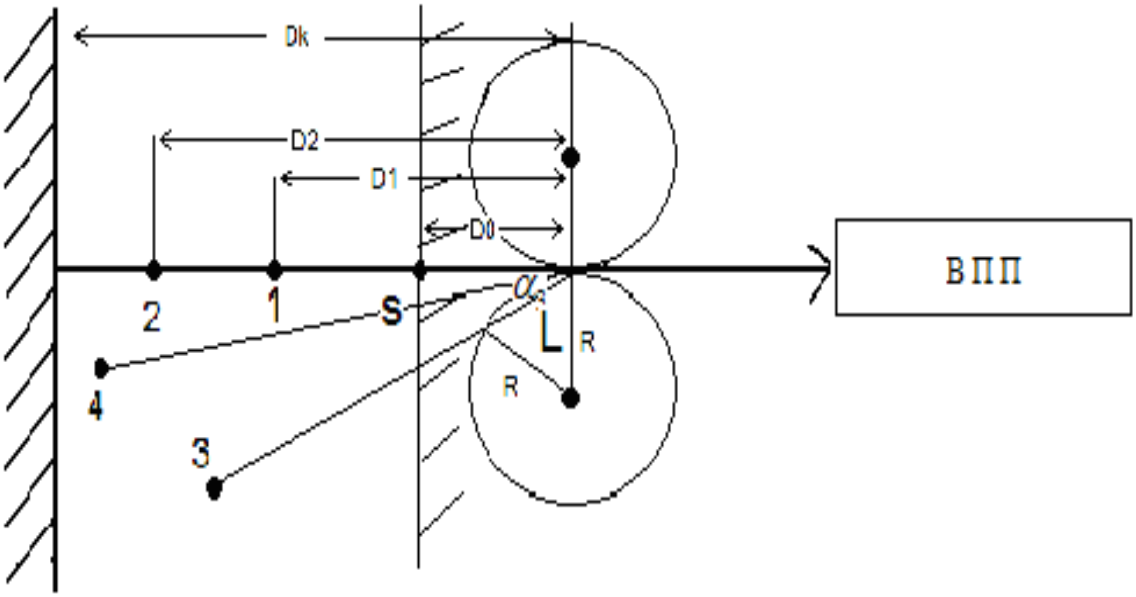


Рис.3.10. Схема расположения группы самолетов в зоне взаимодействия с диспетчером при заходе на посадку.

На рисунке ВПП - взлетно-посадочная полоса, Г- точка вхождения в глиссаду, D_i -дальности самолетов до этой точки, α_I- углы захода на посадку, R - радиус кругла вхождения в глиссаду, S - длина сегмента при полете самолета по этому кругу, l -длина хорды.

2. Принимается, что траектория полета каждого самолета состоит из двух участков - прямолинейного и движения по кругу, что обеспечивает вписывание в глиссаду. Однако если нужно увеличить время полетаi -того самолета к точке Г на величину Δt_i , то самолет вместо прямолинейного движения осуществляет координированный разворот, теряя при этом дополнительно часть топлива. Радиус круга R считается заданным, дальность D_i и угол α_iзахода на посадку каждого самолета, попавшего в зону обслуживания, измеряются и сообщаются диспетчеру вместо с оценкой $Ш_i$ запаса топлива на борту.

3. Диспетчер после получения нужной информации принимает альтернативные решения при обслуживании каждого самолета- либо продолжать прямолинейный полет, либо совершить дополнительный маневр, либо уйти на повторный круг. При этом он учитывает в уме ряд ограничений по возможностям маневрирования, по расходу топлива и, главное, по взаимному расположению самолетов в воздухе, но форма критерия или его параметры ему неизвестны.

4. Считается, что задача параметрической оптимизации решается в классе задач линейного программирования, когда для каждого самолета выбирается время $\Delta t_i \geq 0$ дополнительного маневра по критерию

$$z = \sum_{i=1}^{n} c_i \Delta t_i \rightarrow \max \qquad (3.17)$$

где n -число обслуживаемых самолетов, C_i -подлежащие оценке весовые коэффициенты и зависящие в общем случае от дальностей D_i , углов α_i захода на посадку, коэффициентов лобового сопротивления при боковом маневре, и условий безопасного воздушного движения.

5. КоэффициентыC_i в свою очередь линейно зависят от измеряемых и сообщаемых диспетчеру параметров полета следующим образом

$$c_i = A_{0i} + A_{1i} D_{i-1} + A_{2i} D_i + A_{3i} D_{i+1} + A_{4i} |\alpha_i| + A_{si} Ш_i \qquad (3.18)$$

т.е. для i -того самолета важной является **дистанция** $D_{i+1} - D_i$, $D_i - D_{i-1}$ **до соседей - впереди и сзадилетящего самолета, а также показатель** $Ш_i$ **ограниченного запаса топлива на борту.** Сразу заметим,что **коэффициент** C_2 **относится к впередилетящему самолету, а** C_1 **– к сзадилетящему самолету**.

6. Коэффициенты C_i включают в себя неизмеряемые параметры модели критерия, которые являются общими для всех самолетов и определяют безопасность и экономичность полета с учетом условной стоимости аварийных ситуаций и стоимости топлива. Именно они подлежат идентификации. Считается, что число их невелико и не превышает числа самолетов.

7. На выбираемое дополнительное время Δt_i бокового маневра накладывается ряд ограничений. Схема ограничений в одном из частных случаев примера 2 для двух самолетов представлена на рис. 3.11.

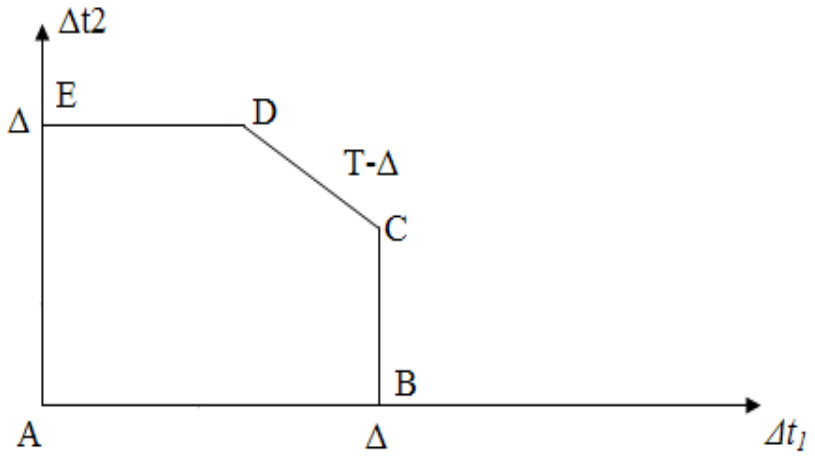


Рис 3.11. Область допустимых решений по маневрированию двух самолетов в виде многогранника.

Первым является ограничение на время полета при максимальном значении перегрузки при боковом маневре, что соответствует условию

$$\Delta t_i \le \Delta \qquad (3.19)$$

Вторым ограничением является условие, что в полете все самолеты не могут одновременно совершать максимальный по перегрузке дополнительный маневр (это опасно)

$$\sum_{i=1}^{n} \Delta t_i \le T, \quad где T < n\Delta \qquad (3.20)$$

Третьим возможным ограничением является условие, что не все самолеты летят одновременно прямолинейно и с постоянной скоростью (хотя это необязательно)

$$\sum_{i=1}^{n} \Delta t_i \ge t_0, \quad где\ t_0 > 0 \qquad (3.21)$$

8. Исходя из перечисленных допущений, требуется :

-задаться структурной (или математической моделью) параметрического критерия, который в явном виде зависит от искомых неизмеряемых параметров безопасности и экономичности полета;

- по данным примеров принятия альтернативных решений провести идентификацию искомых параметров;

-обобщить полученные результаты и получить общее правило решения прямой задачи на основе полученного критерия, что позволит “заменить” диспетчера или помочь ему в других полетных ситуациях.

3.8.2 Оценка безопасности полета в эшелоне при заходе на посадку сучетом дистанции между соседними самолетами

Сформулируем математическую модель критерия поэтапно, вначале оценив безопасность полетов для двух самолетов в предположении, что углы их захода на посадку одинаковы (или равны нулю, как это показано на рис 3.10 для *i=1,2*), и они летят к точке вхождения в глиссаду в одном эшелоне.

С этой целью оценим вероятность P_i возникновения аварийной ситуации в воздухе у двух самолетов, летящих без совершения дополнительного маневра, с помощью экспоненциальной модели, или приближенно-линейной моделью

$$P_i = P_0 e^{-\frac{(D_i - D_{i-1})}{r}} \approx P_0 \left(1 - \frac{\Delta D_i}{r}\right) \tag{3.22}$$

где $\Delta D_i, P_0 \, и \, r$ - искомые параметры, подлежащие идентификации, r - характеризует безопасную дистанцию, а P_0 - максимальную вероятность столкновения при D_i , которую можно принять за единицу.

Если i-ый самолет для увеличения дистанции D_i совершает дополнительный маневр некоторое время Δt_i , летя не по прямой, а по окружности с некоторой ограниченной боковой перегрузкой (а значит, и $\Delta t_i \le \Delta$), то снижение ΔP_i опасности, которое нужно максимизировать, можно оценить так

$$\Delta P_i = P_0 e^{-\Delta D_i / r} - P_0 e^{(-\Delta D_i + V \Delta t_i)/r} \cong P_0 \frac{V}{r} \Delta t_i$$

Если рассмотреть более общую ситуацию, когда кроме двух обслуживаемых самолетов, на границах зоны действия диспетчера есть еще два самолета с дальностями D_0 и D_k, то при суммировании снижения риска опасных ситуаций можно получить линейную свертку

$$\begin{aligned} -\Delta P_i / P_0 &= (e^{-(\Delta D_1 + V\Delta t_1)/r} - e^{-\Delta D_1/r}) + (e^{-(\Delta D_2 + V\Delta t_2 - V\Delta t_1)/r} - -e^{-\Delta D_2/r}) + (e^{-(\Delta D_3 - V\Delta t_2)/r} - e^{-\Delta D_3/r}) \approx \\ &\frac{V}{r^2}\left[(2D_1 - D_2 - D_0)\Delta t_1 + (2D_2 - D_3 - D_1)\Delta t_2\right] \end{aligned} \tag{3.23}$$

Нужно отметить, что знаки сомножителей при Δt_1, и Δt_2 могут быть как положительными, так и отрицательными[61].

3.8.3 Объединение оценок безопасности и экономичности полета в едином параметрическом критерии

Формула (3.23) показывает, что снижение риска опасных ситуаций для каждого самолета определяется следующим образом.

$$\Delta P_i \approx \frac{V}{r^2}(D_{i-1} + D_{i+1} - 2D_i)$$

Оценим теперь дополнительные потери топлива в случае совершения бокового маневра. Известно, что при развороте по крену и ненулевом курсовом угле лобовое сопротивление самолета растет, а потери топлива $Ш_i$ в первом приближении пропорциональны времени совершения этого маневра

$$Ш_i = K\,\Delta t_i$$

где, δ - коэффициент, зависящий от стоимости топлива, типа самолета и других условий и подлежащий идентификации, но он весьма мал по сравнению с фактором безопасности полета.

Объединим оба показателя в максимизируемую целевую функцию критерия Z в виде линейной свертки

$$Z_i = S'_{LA}\Delta P_i - S_T K\Delta t_i = S_{LA}(\Delta P_i - \delta\Delta t_i) = S_{LA}\Delta t_i[-\delta + \frac{V}{r^2}(D_{i-1} + D_{i+1} - 2D_i)] = S_{LA}\Delta t_i.C_i \qquad (3.24)$$

Тогда:

$$Z = \max \sum_{i=1}^{n} C_i \Delta t_i \qquad (3.25)$$

где Z – максимизируемое снижение стоимости обслуживания самолетов в воздухе с учетом экономичности и безопасности полета, S_{LA} - стоимость аварийной ситуации, S_T - стоимость единицы расхода топлива, а коэффициент $\delta = K\frac{S_T}{S_{LA}}$ - подлежащий идентификациипараметр

относительной стоимости топлива по сравнению со стоимостью аварийной ситуации в воздухе, который равен отношение $\frac{m_1}{m_2}$.

Объединяя формулы (3.18) и (3.25) , представим максимизируемую функцию Z в виде (3.17). Тогда весовые коэффициенты c_i равны

$$C_1 = \delta + V/r^2\, D_0 - 2V/r^2\, D_1 + V/r^2\, D_2 = A_{01} + A_{11}D_0 + A_{21}D_1 + A_{31}D_2 \qquad (3.26)$$

$$C_2 = \delta + V/r^2\, D_1 - 2V/r^2\, D_2 + V/r^2\, D_3 = A_{02} + A_{12}D_2 + A_{22}D_2 + A_{32}D_3$$

Таким образом, формулы (3.19 – 3.26) образуют классическую задачу линейного программирования

$$\begin{aligned} &C_1\Delta t_1 + C_2\Delta t_2 \to \max \\ &0 \le \Delta t_i \le \Delta,\ i = 1,2 \\ &t_0 \le \Delta t_1 + \Delta t_2 \le T \end{aligned}$$

что соответствует выпуклому многограннику на рис. 3.11. Ясно, что решение задачи принадлежит одной из вершин *A*, *B*, *C*, *D* , *E* в зависимости от конкретных условий, а ответом диспетчера может быть одна из альтернатив, указанных ниже.

$$\begin{aligned} &j = 1 \quad \Delta t_1 = \Delta; \quad \Delta t_2 = T - \Delta - \textit{вершина } C \\ &j = 2 \quad \Delta t_1 = T - \Delta; \quad \Delta t_2 = \Delta - \textit{вершина } D \\ &j = 3 \quad \Delta t_1 = 0; \quad \Delta t_2 = \Delta - \textit{вершина } E \\ &j = 4 \quad \Delta t_1 = 0; \quad \Delta t_2 = 0 - \textit{вершина } A \\ &j = 5 \quad \Delta t_1 = \Delta; \quad \Delta t_2 = 0 - \textit{вершина } B \end{aligned} \qquad (3.27)$$

Естественно, что выбор ответа базируется на учете “в уме” диспетчера коэффициентов C_1 и C_2 , но он их не вычисляет. Наша задача состоит в определении их с определенной точностью[61].

3.8.4 Оценка неизмеряемых параметров критерия с помощью решения обратной задачи линейного программирования

В формулах оценки коэффициентов C_i линейной свертки диспетчеру известны параметры V, D_{i-1}, D_{i+1}, D_i, r для каждого полетного случая, а параметр δ подлежит дополнительную определению. Если имеются оценки коэффициентов C_i с некоторой точностью, то тогда можно найти параметр δ, где величина δ по смыслу есть интересующее нас отношение $\frac{m_1}{m_2}$.

Пусть дальности, определяющие расположение трех самолетов на линии пути, удовлетворяют ограничениям.

$$D_0<D_1<D_2 \; ; D_1<D_2<D_3$$

Это позволяет объединить два уравнения (3.26) в одно с неизвестным δ при известном параметре r. В частности, разделив одно уравнение на другое, которые представлены ниже:

$$C_1 = \delta - \frac{\nu}{r^2}\left(2D_1 - D_2 - D_0\right) = 1$$

$$C_2 = \delta - \frac{V}{r^2}\left(2D_2 - D_1 - D_3\right) = 0{,}8$$

получим следующую формулу для вычисления $\delta = \frac{m_1}{m_2}$:

$$\delta = \frac{VC_2}{r^2(C_1 - C_2)}\left[\frac{C_1}{C_2}\left(2D_2 - D_1 - D_3\right) - \left(2D_1 - D_2 - D_0\right)\right] \qquad (3.28)$$

Приведём следующий пример расчета. Пусть

$C_1 = 1,\ C_2 = 0{,}8,\ V = 100 м/\text{сек};\ r = 6000,\ D_1 - D_0 = 4000,\ D_2 - D_1 = 10000;$
$D_3 - D_2 = 4000.$

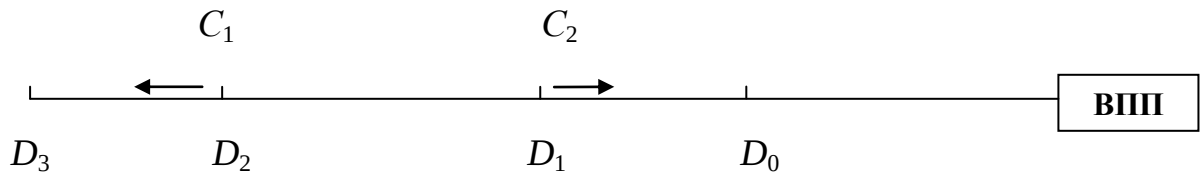


Рис . 3.12. Пример движения трех воздушных судов в эшелоне

Тогда величина δ равна 0,025. Эта оценка есть результат использования двух примеров действия диспетчера - для самолета 1, имеющего параметр его расположения на линии пути D_1, и самолета 2, имеющего параметр D_2 (рис. 3.12). При увеличении числа примеров точность оценки $\frac{m_1}{m_2}$ будет возрастать, хотя сам характер относительной важности коэффициентов критерия не изменится – получилось, что значимость коэффициента m_2 безопасности полета в $10 \div 100$ раз выше значимости коэффициента m_1 его экономичности.

3.9 Выводы по главе 3

На основании проведенных в данной главе исследований можно сделать следующие выводы:

1. Предложен обратный симплекс-метод идентификации неизвестной целевой функции критерия в виде линейной свертки, если задан результат решения при известных ограничениях на выбираемые переменные в виде линейных неравенств.
2. Предложенный подход к решению обратной задачи линейного программирования принципиально отличается от двойственной задачи [15], в которой весовые коэффициенты критерия в виде линейной свертки считаются известными.
3. Воссоздание целевой функции с помощью полученных интервальных оценок позволяет использовать ее при решении прямой задачи в новых условиях-либо в автоматическом режиме, либо в роли подсказчика, снижая тем самым влияние человеческого фактора.
4. Точность решения обратной задачи зависит от того, какие вершины многогранника и сколько их становится известнымив результате рациональных альтернативных действий ЛПР, что может быть использовано для идентификации. В любом случае уточнение весовых коэффициентов единого критерия безопасности и экономичности захода на посадку позволяет перейти к прямой задаче оптимизации.

Глава 4. Автоматизированный выбор посадочных курсов в Московском аэроузле при изменении направлении ветра

В данной главе рассматривается задача определения посадочных курсов для четырех ВПП Московского аэроузла, когда необходимо выбрать 4 из 8 возможностей, из которых одно из направлений имеет встречный ветер. При этом в работе рассмотрено два случая расположения судов в воздушном пространстве.

В первом случае считается, что воздушные суда летятпо заранее намеченных трассам, и при подлете к Москве на определенном расстоянии с радиусом круга порядка 200-300 км принимается решение – продолжать полет по прежнему маршруту или перелететь на другую трассу из-за изменения ветра. Этот случай рассмотрен в данной главе.

Во втором, рассмотренном в главах 5-6 случае считается, что суда могут подлетать к Москве, занимая произвольное положение в воздушном пространстве, и в зависимости от него впервые выбирается одна из возможных трасс. В обеих случаях необходимо определить посадочные курсы с учетом метеорологических условий и технического состоянии ВПП[37].

В данной главе формулируется и решается важная практическая задача перераспределения воздушных судов при их заходе на посадку на разные трассы Московского аэроузла в случае внезапного изменения ветра.

При заходе самолётов в Московский аэроузел диспетчер дает команду экипажам самолётов на снижение и на векторение по радиомаякам. Самолёт, пролетая по радиомаякам и слушая команды диспетчера, заходит на посадочную прямую или трассу, которая совпадает с магнитным курсом полосы. Приоритетный выбор самой полосы зависит от ряда факторов, включая кроме направления ветра учет экономичности перелета на другую

трассу при ограниченном запасе топлива на борту, и осуществляется с помощью особого алгоритма, рассмотренного ниже.

Для аварийного самолёта очень много может значить направление ветра. В экстремальной ситуации необходимы самые удобные условия для посадки авиалайнера, особенно в части направления ветра.

4.1 Алгоритм выбора посадочных курсов ВПП

На рис 4.1. представлена упрощенная схема расположения полос аэропортов московского аэроузла, а задача выбора посадочных курсов решается при следующей постановке задачи.

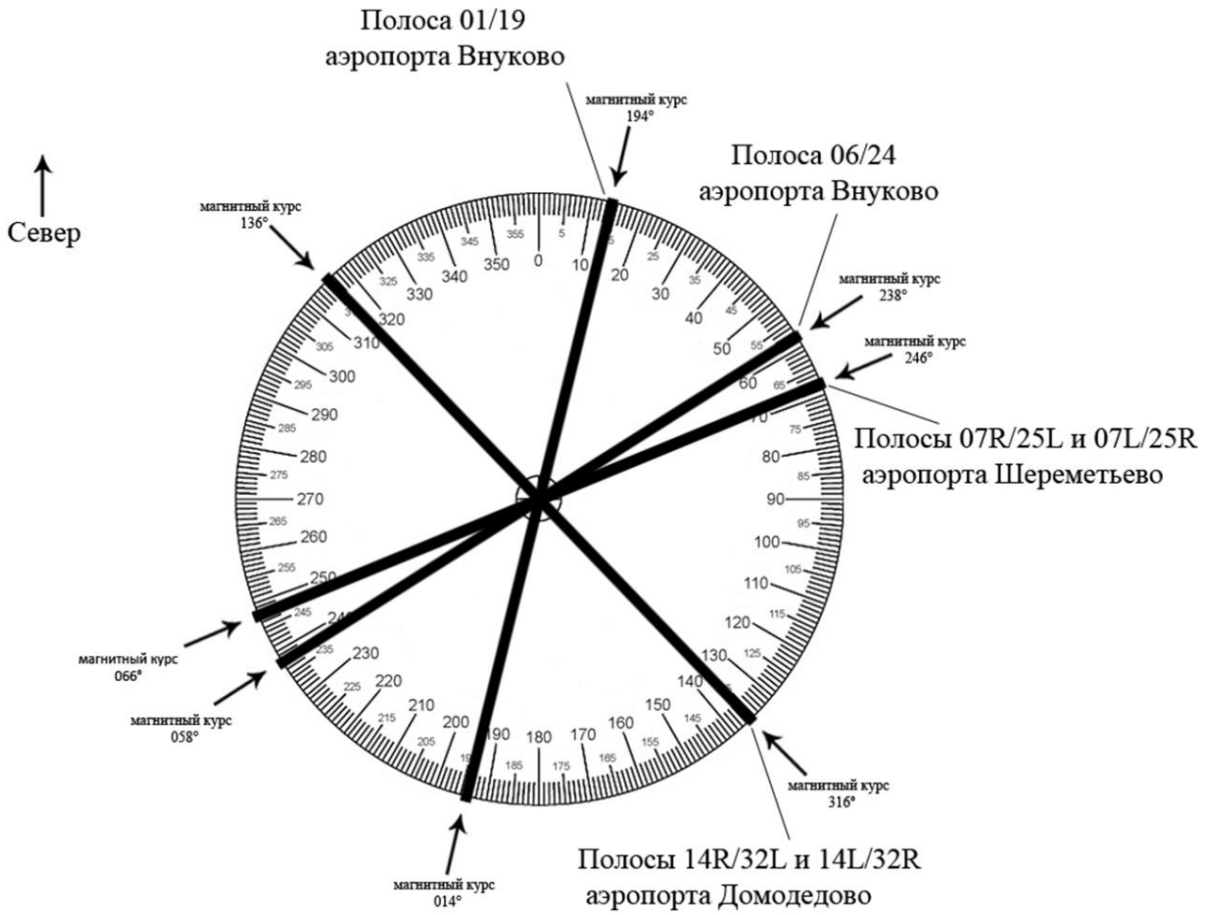


Рис 4.1. Упрощенное представление расположения полос Московского аэроузла

Постановка задачи

Рассматривается задача введения воздушных судов на одну из нескольких взлетно-посадочных полосах (ВПП) на разных аэродромах:

1. Для каждой ВПП заданы исходные углы Ψ_{oj}, j=1...N. Общее число полос равно N. По ним определяются курсы посадки $\Psi_{\kappa i}$, i= 1…2N. Общее число посадочных курсов равно 2N (на каждую полосу можно зайти с двух сторон, см. рис. 4.1).

2. Задан текущий курсовой угол Ψ_W ветра, который может поменять своё значение. В зависимости от его направления для каждой полосы определяется один из двух посадочных курсов.

3. Каждое воздушное судно уже летит по одной из 8 заданных трасс или приблизилось к границе заданного круга, показанного на рис 4.1 и предназначенного для перелета на другие трассы. При этом анализируется только горизонтальный полет на заданной постоянной высоте.

4. В качестве постоянных параметров принимаются, как известные скорость полета V, максимальное допустимое боковое ускорение a при разворотах, минимальная дистанция r безопасного движения ЛА в эшелоне.

5. Если на трассе не хватает места для безопасного движения судов, то часть из них направляется в очередь этой трассы, называемой “тромбон”, чтобы потом прилететь на тот же аэродром при первой возможности.

Требуется:

- определить посадочные курсы ВПП с учетом направления ветра;
- априорно распределить воздушные суда между ВПП, если они летят позаранее известным трассам, и сформировать таблицу приоритетов для всех ЛА.
- определить первоочередность захода на посадку и приземления судов для каждой трассы.

Предложенный алгоритм определения посадочных курсов ВПП состоит их двух блоков. В первом блокеисходными данными в первой задаче являются N углов ВПП – Ψ_{oj} (сторона для ВПП задается любая). Поэтому сначала определяются $2N$ курсов Ψ_{ki} возможного подлета самолетов по радиомаякам. Блок-схема алгоритма представлена на рис 4.3.

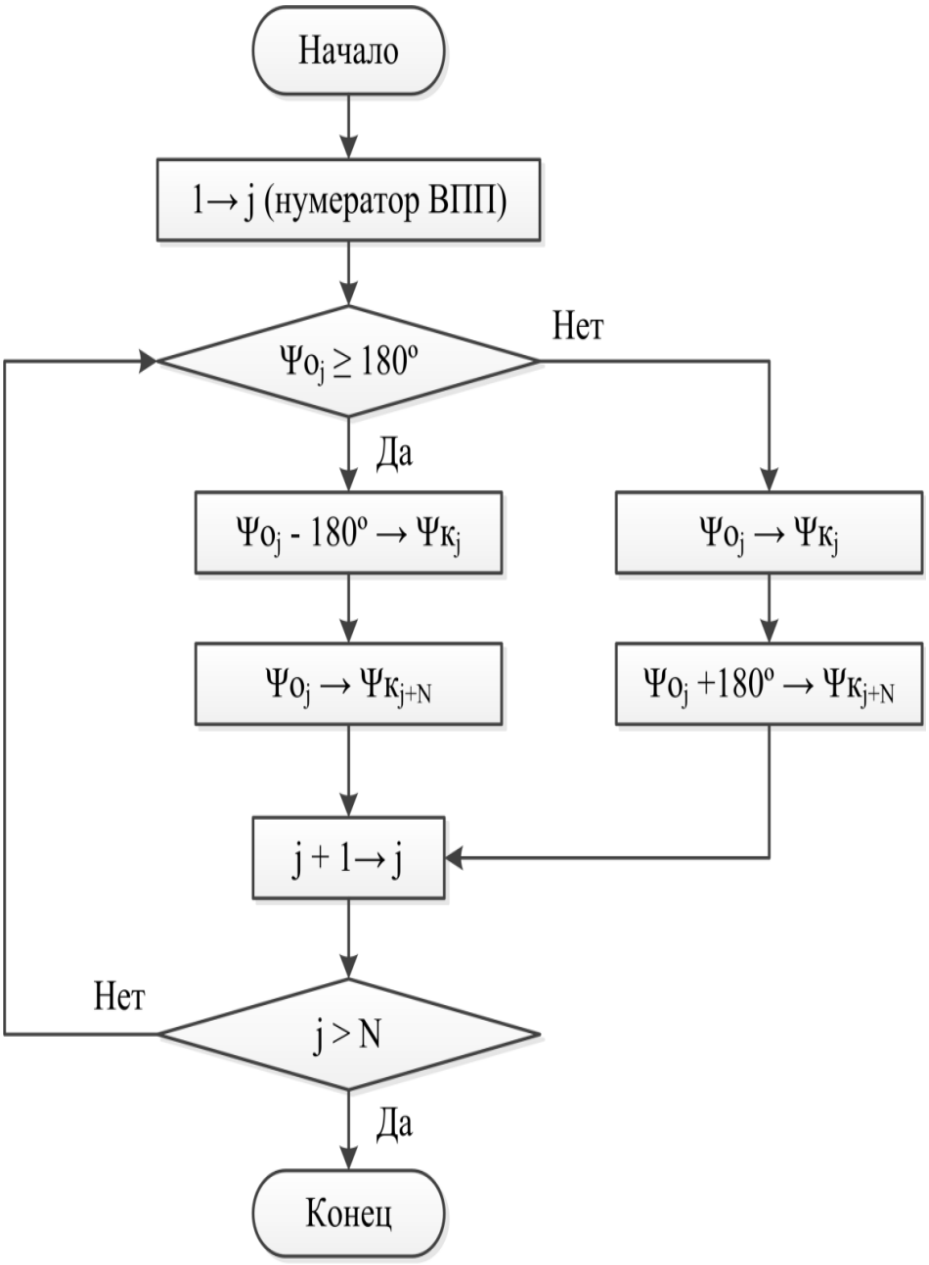


Рис. 4.2. Блок-схема алгоритма записи $2N$ посадочных курсов

Смысл этого блока состоит в том, что при предъявленном первоначальном списке четырех углов ВПП - ψ_{oj} , имеющем произвольный

порядок, формируется упорядоченный список из 8 элементов - $\Psi_{\kappa i}$, расположенныхпо возрастанию. Из алгоритма видно, что курсы $\Psi_{\kappa i}$ записаны по возрастанию, т.е. меньшиекурсы полос имеют номера 1...N, а большие – (N+1)...2N (углы измеряются от 0º до 360º так, как это показано на рис. 4.1).

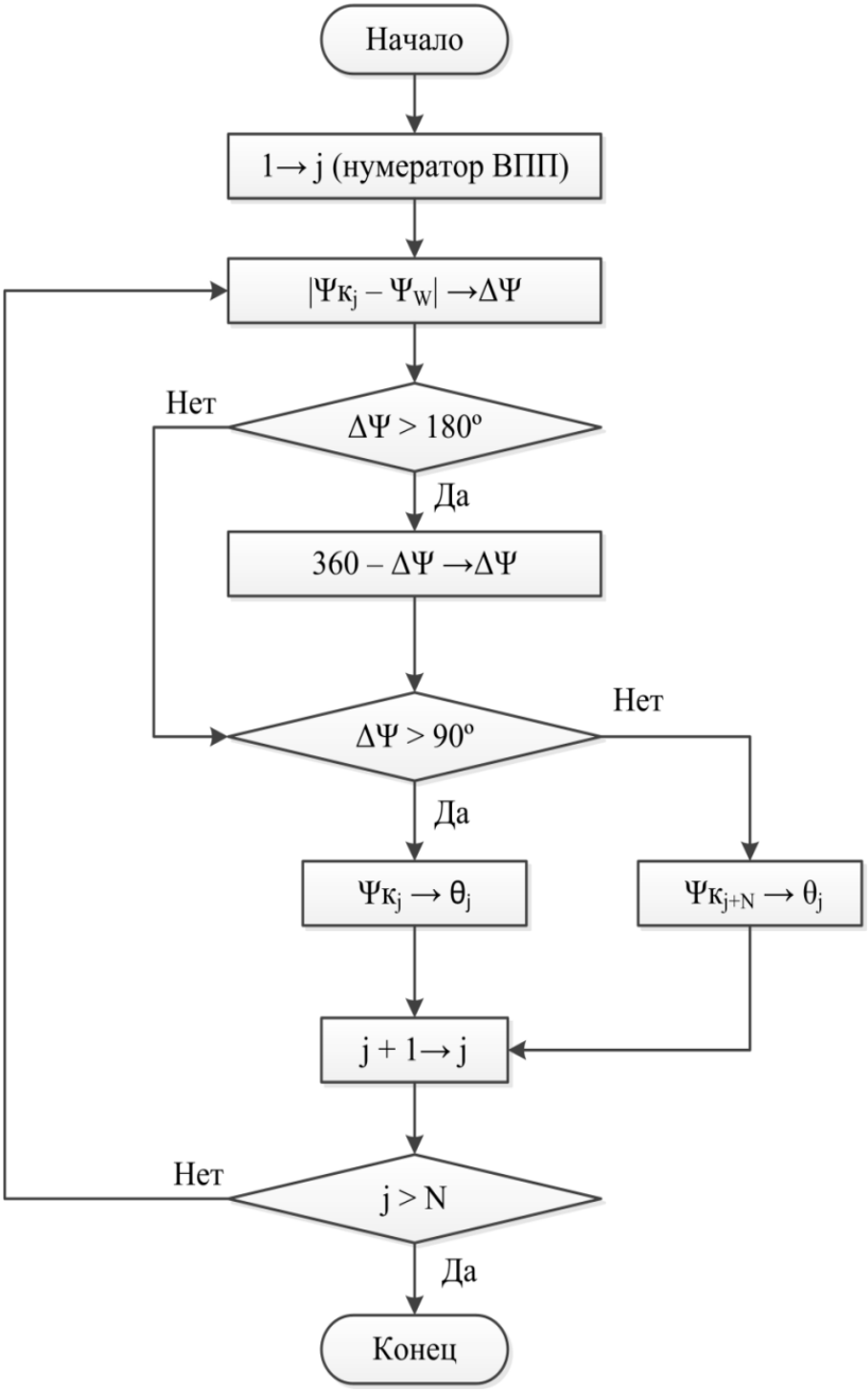


Рис. 4.3. Блок-схема алгоритма определения посадочного курса для каждой ВПП

Дальше в блоке 2 происходит определение одного из двух посадочных курсов для каждой ВПП в зависимости от направления ветра (θ_j). Выбор стороны ВПП для посадки выбирается таким образом, чтобы ветер был встречным, т.е. выполнялась проверка условия $\Delta\psi > 90^\circ$, где , $\Delta\psi = |\psi_k - \psi_w|$ при $\Delta\psi < 180^\circ$. Если же$\Delta\Psi > 180^\circ$, то вносится поправка, а затем опять повторяется проверка условия $\Delta\psi > 90^\circ$. Блок-схема алгоритма представлена на рис 4.3.

Согласно представленной на рис. 4.3 структуре, на выходе этого блока остаются 4 посадочных курса ψ_{kj}, для которых новое направлениеветра имеет встречную составляющую, а остальные 4 курса ВПП бракуются. Пример полученного списка разрешенных курсов посадки показан таблицей 4.1.

Таблица 4.1. Номера разрешенных курсов посадки

K	1	2	3	N
M	1	2	8	7

4.2 Постановка задачи оптимизации захода на посадку на разные аэродромы воздушных судов, подлетающих к Москве только по заданным трассам

1. Рассматривается возможность посадки на одну из нескольких ВПП на разных аэродромах. Для каждой ВПП заданы исходные разрешенные курсы$\Psi_{ко}$(k=1...N). Общее число полос равно N.

2. Заданы фиксированные курсовые углы Ψ_{io}(i=1…8)трасс возможного подлета самолета при движении по радиомаякам. Общее число этих углов равно 2N, чтобы садиться на каждую ВПП с двух сторон, как это показано на рис 4.1.

3. Задан текущий курсовой угол Ψ_w ветра, который может поменять своё направление.

4. Заданы координаты X_i, Z_i (i=1…2N) пересечения маршрутов движения с контуром, окружающим группу аэродромов в аэроузле, с помощью которых возможен предварительный расчет потерь топлива $П_{im}$при вынужденном перелете по контуру самолетов из точки i в одну из подходящих точек m. Считается, что результаты расчета сведены в таблицу 4.2, полученную в виде примера для случая N=4, показанного на рис. 4.1. Эта матрица имеет нули в диагональных элементах, она симметричная и имеет вид волнообразной поверхности, показанной на рис 4.4. Максимум потерь $П_m$ соответствует случаюпосадки на исходно заданную ВПП с противоположного направления.

Таблица 4.2. Расход топлива при перелете с одной трассы iна другую j

	1	2	3	4	5	6	7	8
1	0	$П_1$	$П_2$	$П_3$	$П_{max}$	$П_3$	$П_2$	$П_1$
2	$П_1$	0	$П_1$	$П_2$	$П_3$	$П_{max}$	$П_3$	$П_2$
3	$П_2$	$П_1$	0	$П_1$	$П_2$	$П_3$	$П_{max}$	$П_3$
4	$П_3$	$П_2$	$П_1$	0	$П_1$	$П_2$	$П_3$	$П_{max}$
5	$П_{max}$	$П_3$	$П_2$	$П_1$	0	$П_1$	$П_2$	$П_3$
6	$П_3$	$П_{max}$	$П_3$	$П_2$	$П_1$	0	$П_1$	$П_2$
7	$П_2$	$П_3$	$П_{max}$	$П_3$	$П_2$	$П_1$	0	$П_1$
8	$П_1$	$П_2$	$П_3$	$П_{max}$	$П_3$	$П_2$	$П_1$	0

5. Если самолёты не летят по одному из 2N заданных маршрутов, а их координаты X_i, Z_i и курсовые углы также известны, то эта задача решается ниже в главах 5-6.

6. Запаса топлива на борту самолетов достаточно, чтобы в рамках данной работы не учитывать факт возможного возникновения аварийных ситуаций.

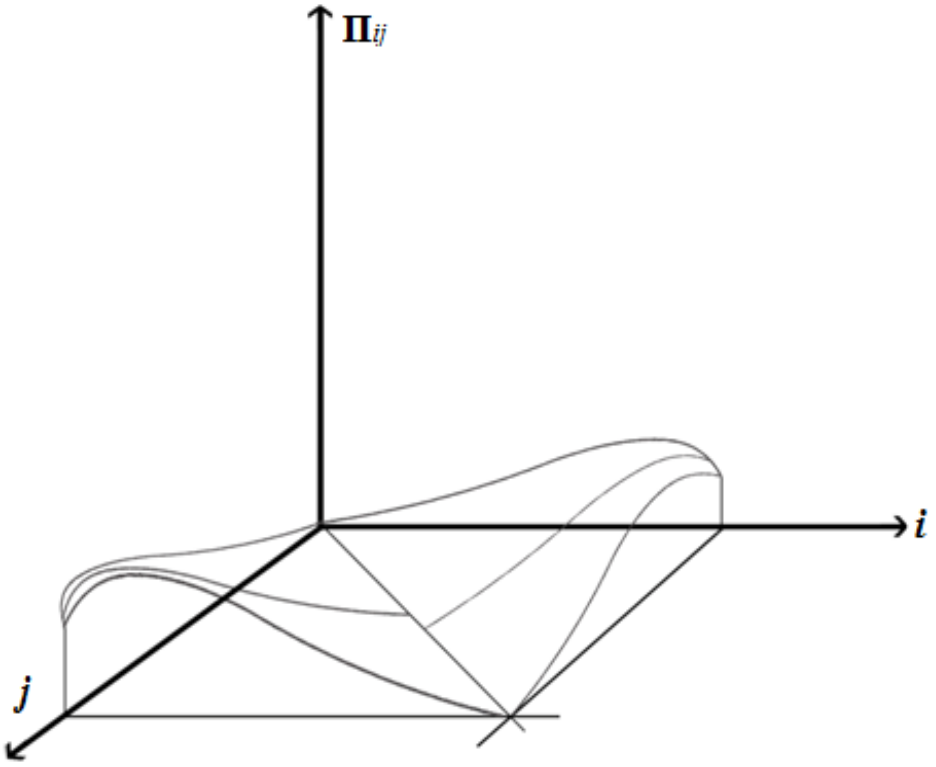


Рис. 4.4. График потерь топлива при перелете по контуру вокруг Москвы с запрещенной трассы на соединение по и против часовой стрелки

7. При перелете из точки *i* в точку *m* альтернатива попадания самолета на «свой аэродром» (когда $m=i\pm N$), имеет предпочтение, хотя и требует максимального расхода топлива. Пусть коэффициент предпочтения *M* задан. Тогда расходы топлива нужно пересчитать - вместо $П_{max}$ следует взять значение $\frac{П_{\max}}{M}$.

Требуется сформировать структуру принятия альтернативных решений по выбору наиболее экономичного варианта посадки группы самолетов, летящих в заданных направлениях.

4.3 Структура принятия альтернативных решений по посадке на группу ВПП самолетов, летящих в заданных направлениях.

Решение поставленной задачи в данной работе предусматривает выполнение в цикле следующих основных операций:

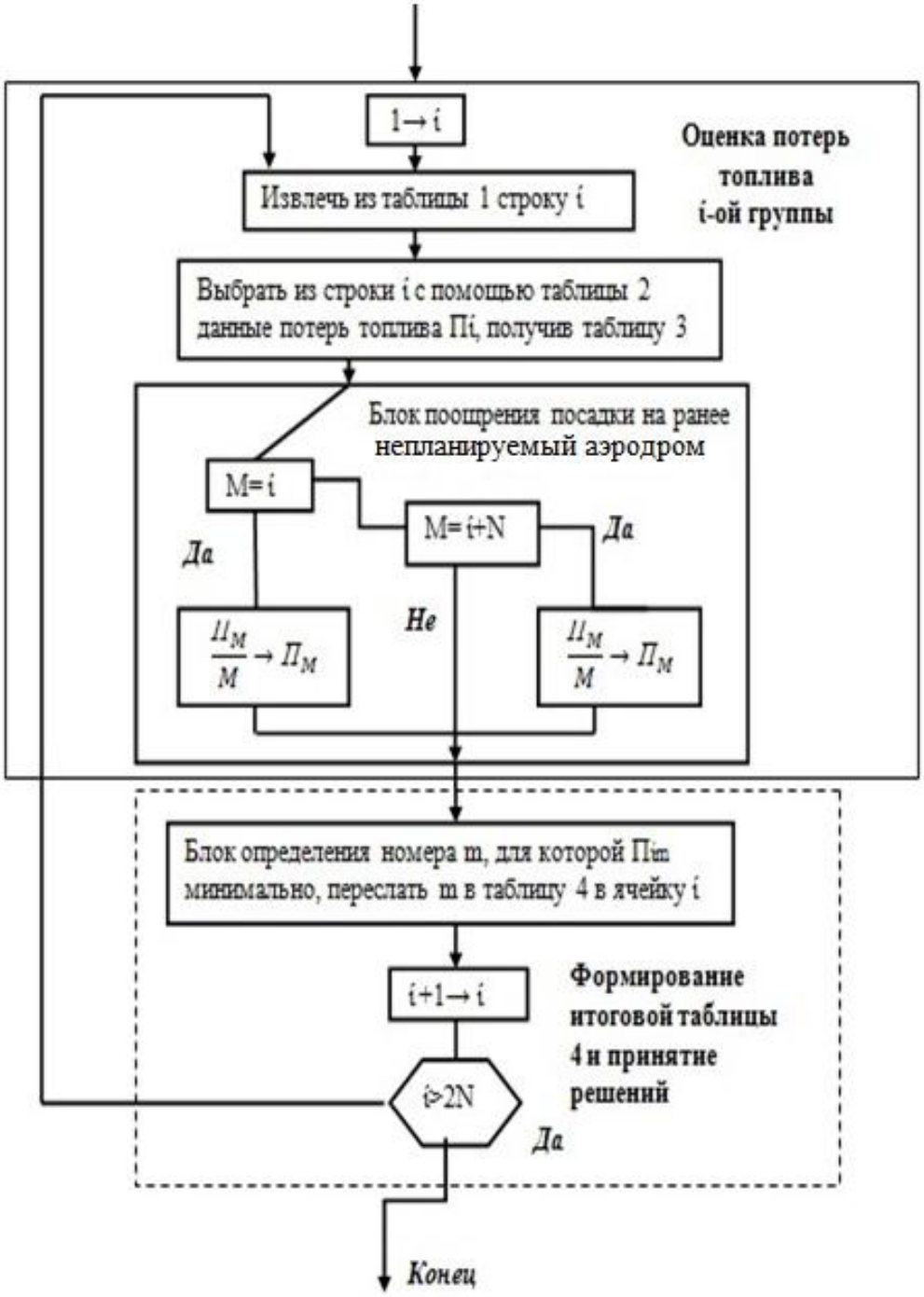


Рис. 4.5. общая блок-схема принятия решений по выбору посадочного курса для воздушных судов, движущихся по заданным трассам**.**

- определение одного из двух посадочных курсов для каждой ВПП в зависимости от направления ветра;

- оценка потерь топлива для каждой из групп самолетов, летящих в заданном направлении, при их посадке на каждую ВПП

-снижение штрафа за потери топлива при посадке на запланированный аэродром

- выбор наиболее экономичного варианта посадки для каждой из групп самолетов. Структура принятий решений представлена на рис. 4.5.

Таблица 4.3. Варианты разращенного перелета для оценки затрат топлива на перелет из трассы*i*на каждую из разрешенных трасс

i	1	2	3	4	5	..	$2N$
M_i	1	2	2	3	3	..	8

Во втором блоке ищетсянаилучший вариант посадки, при котором расходы топлива на перелет минимальны. В итоге формируется таблица 4.4, указывающая для всех воздушных судов, на какую трассу им ориентироваться.

Таблица 4.4. Назначение нужной трассы посадки для каждого судна по критерию минимального расхода топлива.

M	1		2		8	7	
$П_i$	$П_{max}$		$П_3$		$П_2$	$П_3$	
i	1	2	3	4	5	..	2N
M i	1	2	2	3	3	..	8

Работа общей блок-схемы можно более детально пояснить на представленном ниже примере, в котором согласно рис 4.6 нумерация трасс от 1 до 8 принята при движении против часовой стрелки.

Обозначим через T_i номера трасс. Тогда согласно принятому решению, самолеты летящие по трассам 1-4, не меняют своих маршрутов и трассы назначения T_{ni} совпадают с имеющимся

$$T_{H1} = T_1; T_{H2} = T_2; T_{H3} = T_3; T_{H4} = T_4$$

Самолеты с края “разрешенного сектора”, т.е на трассах $T_{н1}$ и $T_{н4}$ начнут принимать самолеты с трасс $T_5 - T_8$ в первую очередь, особенно с трасс T_5 и T_8. Самолеты с трасс T_6 и T_7 выбираются более сложным образом, например самолет с трассы T_6 согласно рис 4.6 может полететь на трассу $T_{н1}$ и $T_{н4}$, какая будет ближе, согласно геометрическому решению задачи в верхней полусфере Москоского региона.

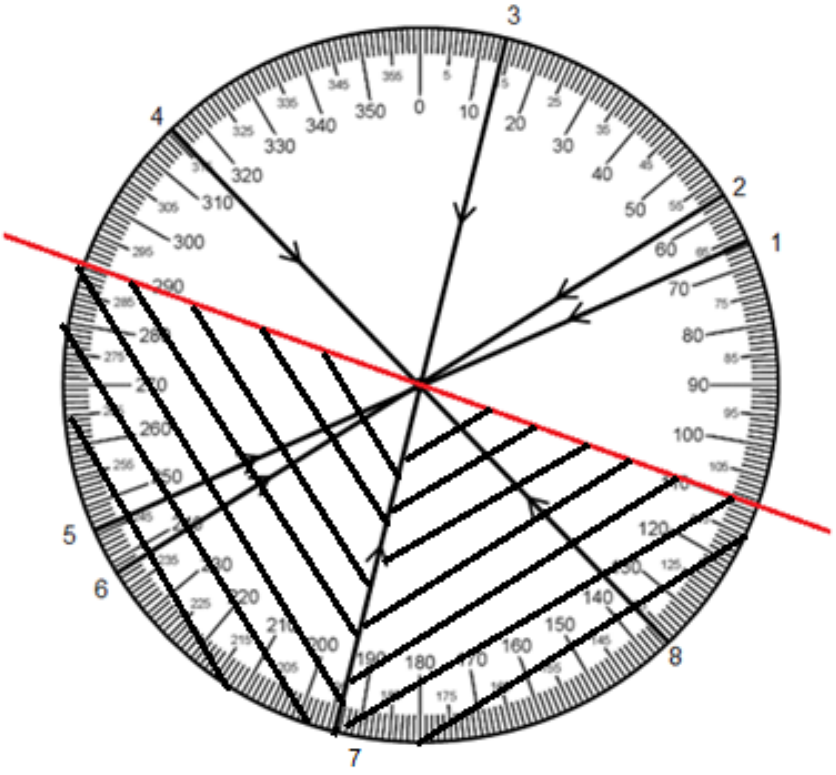


Рис. 4.6. Нумерация трасс движения судов при подлете к Москве в случае, когда ветер определил посадочные курсы от 1-4

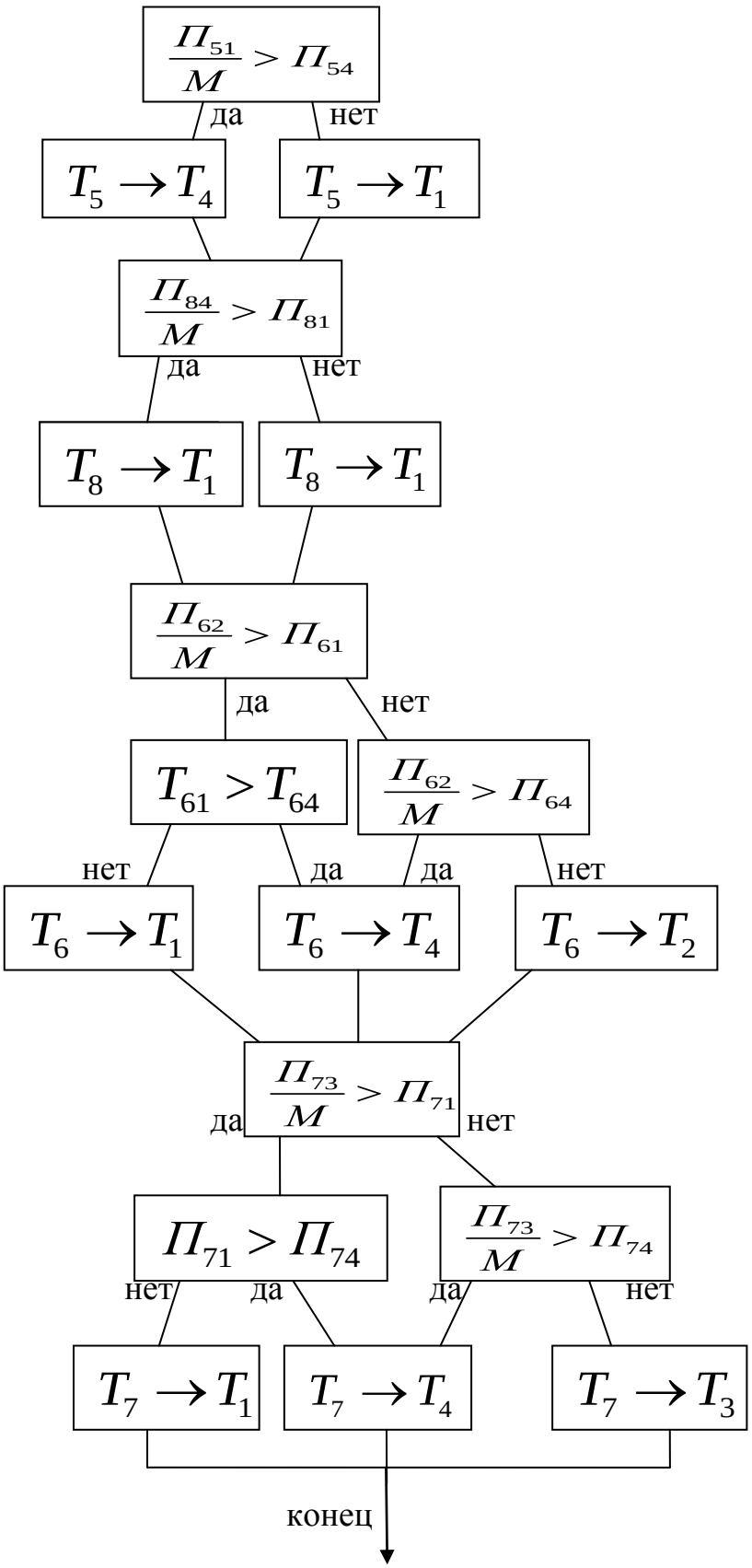


Рис 4.7. Логика выбора трасс назначения для примера, показанного на рис 4.6.

Логика выбора трасс T_{ni} для данного примера может быть проиллюстрирована на рис 4.7. Потери топлива на перелет с трассы (i)на трассу (j) обозначена $П_{ij}$. В этой логике учтена также важная возможность приоритетного выбора «своей трассы» при попадании самолета на заранее намеченный аэродром, если

$$\frac{П_{j+4,j}}{M} < П_{j+4,1}, \frac{П_{j+4,j}}{M} < \frac{П_{j+4,2}}{M}$$

где, $M > 1$ - заданная назначенная величина. Полученные ответы на рис4.7 являются элементам для заполнения итоговой таблицы 4.4.

4.4 Выводы по главе 4

На основании проведенных в данной главе исследований можно сделать следующие выводы:

1. Сформирован алгоритм определения посадочных курсов Московского аэроузла в зависимости от направления ветра.
2. Поставлена и решена задача априорного выбора посадочного курсов для воздушных судов, летящих по заранее назначенным трассам, с учетом имеющегося направления ветра.
3. Задача априорного выбора посадочных курсов для воздушных судов решена по критерию минимального расхода топлива на перелеты с одной трассы на другую. При этом учтено предпочтение попадания самолета на «свой аэродром», хотя при этом требуется максимальный расход топлива в случае посадки на исходно заданную ВПП с противоположной стороны.
4. Воздушные суда, попавшие в заднюю полусферу ВПП, могут быть автоматически переадресована на ближайшие ВПП, на которые разрешено садиться, и согласно логике выбора трасс в нужном списке судов без расчета динамических приоритетов.

Глава 5. Формирование динамических приоритетов посадки самолетов на одну из ВПП по критерию экономичности и безопасности полета

5.1 Подход к решению задачи методом динамического программирования.

При оживленном воздушном движении в процессе прилета приходится принимать ответственные решения по выбору состава судов, попадающих в соответствующий эшелон захода на посадку. Поскольку этот выбор зависит от расположения судов относительно трассы перелета и их текущего курса, оптимальное управление полетом должно быть функцией текущего состояния координат движения судна.

Другим случаемявляется приближение судов с аварийно низким запасом топлива или возможными техническими отказами борта, что требует внеочередного обслуживания при одновременном стремлении уступить им место судами с нормальным состоянием. Это означает, что в состав координат текущего состояния судна, помимо оценки его положения в пространстве должно в*х*одить, по крайней мере, значение оставшейся части топлива, необ*х*одимого на дополнительное маневрирование. Для решения указанной задачи наиболее подходящим является метод динамического программирования, поскольку именно ему соответствует принцип «управления по состоянию». Метод динамического программирования позволяет указать в текущий момент времени очередность или приоритет в обслуживании каждого судна и последовательно вводить их в заданных эшелон, проверяя при этом возможность соблюдения гарантированной безопасности полета.

В данной работе этот подход предложено реализовать путем вычисления динамических приоритетов в виде некоторых количественных оценок, учитывающих удаленность воздушного судна от заданной трассы,

ожидаемую его близость к судам, движущимися уже в эшелоне, а также от оставшегося запаса топлива. При этом, если очередной приоритет мал, то это означает существование такого риска несоблюдения безопасности совместного движения в эшелоне, при котором происходит отказ от попытки введения судна в эшелон, и дается команда ухода на повторный круг.

В данной главе решена задача вычисления динамических приоритетов на вход самолета в один заданный воздушный эшелон.

5.2 Решение с помощью уравнения Беллмана задачи назначения динамических приоритетов при движении судов, летящих параллельным курсом с заданной линией пути.

В данной задачев начале анализируется наиболее распространенный случай полета судов в сторону аэродрома, т.е x_3=0, хотя его положение с заданной линией пути может не совпадать. Особенностью этой задачи является ее принадлежность к задачам альтернативного управления, для которых метод АКОР не пригоден. Поэтому в данной работе используется разработанный в метод рабочей точки[30].В основу этого методарасчета входят процедуры оценки риска в различных ситуациях в окрестности этой точки, после чего искомая функция откликаилифункция Беллмана S апроксимируется степенным полиномом заданного порядка.

В решаемой задаче **непрерывного управления нет**, **а есть две альтернативы**, лететь на трассу или нет. В этом случае правая часть уравнения Беллмана имеет вид двух функций риска F_1 и F_2, как показано на рис 5.1[30].

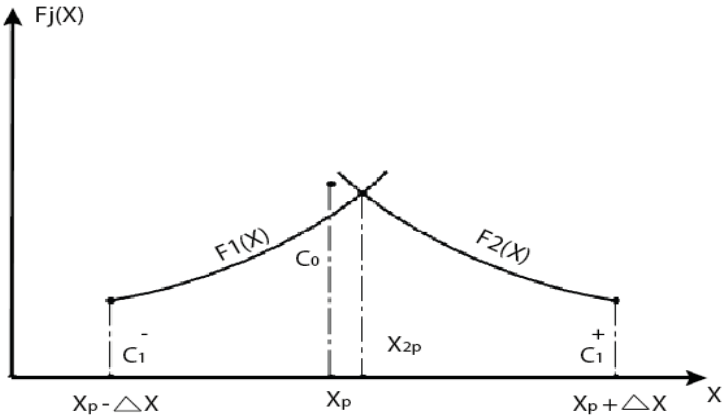


Рис 5.1. Поведение функций риска при двухальтернативном принятии решений

На этом рисунке имеются вычисленные различные значения правой части уравнения Беллмана, которые являются функциями риска, т.е ординатами C_0, C_1^+, C_1^- и. т. д. В [8] доказано, что в установившемся состоянии оптимальному выбору альтернативных решений соответствует условие равенства этих ординат друг другу.

$$C_i = C_{ik} = C_0, \text{ при} i, k=1,\ldots,n \qquad (5.1)$$

Это условие аналогично условию $\frac{\partial S}{\partial t} = 0$ длязадачи непрерывного управления.Число этих ординат должно быть таково, чтобы количество образующихся равенств было равно числу искомых коэффициентов степенного полинома, апроксимирующего функцию Беллмана.

Особую роль играет ордината риска C_0 с такими значениями координат рабочей точки, для которых нельзя указать предпочтение ни одной альтернативе.

Пользуясь этим подходом при решении поставленной задачи, в качестве рабочей точки выбрана ситуация, в которой при нахождении в некотором отдалении от заданной трассы нельзя отдать предпочтение ни одной из альтернатив. Примером такого отдаления является расположение рабочей точки на биссектрисе между трассами, как это показано в главе 9 на рис 9.1 и рис 9.2. Отклонение от этой биссектрисы по боковому пути x_1 или

курсуx_3сразу приводит к явному предпочтению, куда лететь. Выбранные координаты x_{pi}и отклонения Δx_i приведены ниже.

$$x_{p1} = r;\ \Delta x_1 = r;\ x_{p2} = r;\ \Delta x_2 = r;\ x_{p3} = 0;\ \Delta x_3 = \pi/2;\ x_{p4} = 0,4\Delta V;\ \Delta x_4 = 0,4\Delta V$$

Будем вычислять ординаты риска в нужном числе полетных ситуаций и затем, приравнивая их, найдем аппроксимацию функции Беллмана *S*, а значит и текущие функции риска F_j (j = 1,2), определяющие приоритет в принятии решений. Чем больше величина F_1 риска входа в эшелон и чем меньше величина F_2 риска ухода на повторный круг, т.е. чем меньше $\Delta F = F_2 - F_1$, тем меньше шансов на попадание в эшелон. Значит, если взять величину ΔF в качестве приоритета П, то можно проранжировать все воздушные суда и поочередно планировать их введение в эшелон до тех пор, пока условия безопасности не нарушатся[32, 57].

Решение начнем с записи уравнения Беллмана для двух альтернатив j=1,2, пользуясь заданными соотношениями (1.1-1.5) и задавшись следующей аппроксимацией Беллмана *S* в виде степенного полинома [27].

$$S = \beta_1 x_1 + \gamma_1 \frac{x_1^2}{2} + \beta_2 x_2 + \gamma_2 \frac{x_2^2}{2} + \beta_4 x_4 + \gamma_4 \frac{x_2^2}{2} + \psi_{12} x_1 x_2 + \psi_{14} x_1 x_4 + \psi_{24} x_2 x_4 \qquad (5.2)$$

Тогда нужные частные производные будут равны

$$\frac{\partial S}{\partial x_1} = \beta_1 + \gamma_1 x_1 + \psi_{12} x_2 + \psi_{14} x_4;$$
$$\frac{\partial S}{\partial x_2} = \beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{24} x_4;$$
$$\frac{\partial S}{\partial x_4} = \beta_4 + \gamma_4 x_4 + \psi_{14} x_1 + \psi_{24} x_2$$

где $\beta_i, \gamma_i, \psi_{ik}$ - искомые коэффициенты, которые необходимо определить. Представляя эти производные и известные соотношения (1.1-1.4) в условие оптимальности(2.23), можно получить

$$-\frac{\partial S}{\partial t} = \min_{j=1,2} \left\{ F_j(x_1, x_2, x_4) \right\} \tag{5.3}$$

где функции риска приj= 1 иj = 2 равны:

$$\begin{aligned} F_1 &= \frac{m_1 x_1^2}{r^2} + \frac{m_2 (r - x_2)^2}{r^2} + m_3 x_3^2 - \frac{m_4 x_4}{\Delta V} - \frac{x_1}{0.5(T_1 + T_2)} (\beta_1 + \gamma_1 x_1 + \psi_{12} x_2 + \psi_{14} x_4) - \\ &- \frac{x_2}{0.5 T_2} (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{24} x_4) + \frac{w_o}{T_o} \left[T_1 (1 + \lambda) + T_2 \right] (\beta_4 + \gamma_4 x_4 \psi_{14} x_1 + \psi_{24} x_2) \\ F_2 &= l - m_4 \left(\frac{x_4}{\Delta V} + \frac{x_4^2}{\Delta V^2} \right) - V (\beta_1 + \gamma_1 x_1 + \psi_{12} x_2 + \psi_{14} x_4) - \frac{x_2}{0.5 T_o} (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{24} x_4) \\ &+ w_o (\beta_4 + \gamma_4 x_4 \psi_{14} x_1 + \psi_{24} x_2) \end{aligned} \tag{5.4}$$

Теперь можно приступить самому расчету путем вычисления нужных ординат риска, учитывая, что при x_3=0 в степенном полиноме (5.2) имеется 9 искомых коэффициентов. Значит, нужно вычислить 10 ординат риска, которые множно разбить на две группы – в одной группе очевидно решение j=1, в другой –j=2.

Сначала вычислим ординаты C_1^-, C_1^+, C_0, чтобы получить равенство.

$$C_1^- + C_1^+ = 2C_0 \tag{5.5}$$

$$C_1^- = F_1 \begin{pmatrix} x_1 = 0 \\ x_2 = r \\ x_4 = 0,4\Delta V \end{pmatrix} = 0,25 m_2 - 0,4 m_4 + \frac{4 r \beta_2}{3 T_2} + M W_0 \left(\beta_4 + 0,4 \Delta V \gamma_4 \right) \tag{5.6}$$

где $M = \frac{\left[(1 + \lambda) T_1 + T_2 \right]}{T_0}$

$$C_1^+ = F_2 \begin{pmatrix} x_1 = 2r \\ x_2 = r \\ x_4 = 0,4\Delta V \end{pmatrix} = l - 0,4 m_4 - V \left(\beta_1 + 0,4 \Delta V \psi_{14} + 2 r \gamma_1 \right) - \frac{4 r \beta_2}{3 T_2} + W_0 \left(\beta_4 + 0,4 \Delta V \gamma_4 + W_0 \psi_{14} \right) \tag{5.7}$$

$$2C_0 = F_1\left(\bar{x}_p\right) + F_2\left(\bar{x}_p\right) = m_1 + l + 0,25m_2 - 0,56m_4 - \left(\beta_1 + 0,4\Delta V\psi_{14} - r\gamma_1\right)\left[V + \frac{2r}{T_1 + T_2}\right] \quad (5.8)$$
$$+W_0\left(\beta_4 + 0,4\Delta V\gamma_4 + \psi_{14}r\right)\left(1 + M\right)$$

В результате решения равенства (5.5) получим

$$\gamma_1 = -\psi_{14}\frac{W_0}{V} \quad (5.9)$$

Вычислим теперь ординаты риска в полетных ситуациях,в которых очевиден уход на повторный круг, т.е $j = 2$. Этими ординатами являются функции $C_{12}^{+-}, C_1^+, C_4^-, C_2^-$, найденные с помощью F_2. Например, ордината C_1^+ вычисляется по формуле (5.7), а ордината C_4^- равна

$$C_4^- = F_2\begin{pmatrix} x_1 = 0 \\ x_2 = r \\ x_4 = 0 \end{pmatrix} = l - V\left(\beta_1 + \gamma_1 r\right) - \frac{4r\,\beta_2}{3T_0} + w_0\left(\beta_4 + r\psi_{14}\right) \quad (5.10)$$

Приравнивая эти ординаты друг другу и ординате C_o, можно убедиться

$$\psi_{12} = \psi_{24} = 0; \gamma_2 = -\frac{\beta_2}{3x_{2p}}; w_o\gamma_4 = V\psi_{14} \quad (5.11)$$

Теперь рассмотрим полетные ситуации, в которых очевидно решение войти в эшелон, то есть j = 1. Для этого вычислим ординаты риска $C_1^-, C_{14}^{--}, C_{12}^{-+}, C_4^+, C_2^+$ аналогичным выше способом, только пользуясь теперь функций F_1. Например, ордината риска C_{14}^{--} вычисляется так

$$C_{14}^{--} = F_1\begin{pmatrix} x_1 = 0 \\ x_2 = r \\ x_4 = 0 \end{pmatrix} = 0,25m_2 + \frac{4r\,\beta_2}{3T_2} + Mw_0\beta_4 \quad (5.12)$$

Тогда из условия $C_{14}^{--} = C_1^-$ получим

$$\gamma_4 = -\frac{m_4}{W_0 \Delta V} \tag{5.13}$$

Далее вычислим ординату C_4^+

$$C_4^+ = F_1 \begin{pmatrix} x_1 = r \\ x_2 = r \\ x_4 = 0,8\Delta V \end{pmatrix} = m_1 + 0,25m_2 - 0,8m_4 - \frac{2r}{T_1 + T_2}\left(\beta_1 + r\gamma_1 + \psi_{14} 0,8\Delta V\right) + \frac{4r\beta_2}{3T_2} + Mw_0\left(\beta_4 + 0,8\Delta V\gamma_4 + \psi_{14} r\right) \tag{5.14}$$

Приравнивая ординаты C_4^+ и C_1^- друг другу, получим

$$\beta_1 = -\frac{1}{V} \tag{5.15}$$

Действуя аналогичным образом с остальными ординатами, можно вычислить остальные коэффициенты β_2, β_4, ψ_{14}. В частности, коэффициент β_2 можно найти, если приравнять ординаты C_2^- и C_1^+

$$\psi_{14} = -\frac{1}{2rw_0};\ \beta_2 = -\frac{0,75T_0}{r};\ \beta_4 = -\frac{2,8}{w_0} \tag{5.16}$$

Это позволяет в конце концов вычислить функции риска F_1 и F_2 аналитически, если ввести следующие дополнительные безразмерные переменные:

$$y_1 = \frac{x_1}{r}; y_2 = \frac{x_2}{r}; y_4 = \frac{x_4}{\nabla v} \tag{5.17}$$

Тогда получим в общем виде

$$F_1 = 1,4 + \left[\frac{2r}{V\left(T_1 + T_2\right)} - 0,5M\right] y_1 + m_1 y_1^2 + \frac{2T_0}{T_1 + T_2} y_1 y_4 + m_2\left(1 - y_2\right)^2 - \left(2m_2 + m_4\right) y_4 \tag{5.18}$$

$$F_2 = 3,8 + l + 1,5y_2\left(1 - 0,3y_2\right) - 2m_4 y_4 \tag{5.19}$$

Или в численном выражении при $m_1 = 1$, $m_2 = m_4 = 40$, $l = 1$,

$$F_1 = 1,4 + 0,2y_1 + 1,2y_1^2 + 43(1 - y_2)^2 - 120y_4 + 2y_1y_4 \qquad (5.20)$$

$$F_2 = 14 + 1,5y_2 - 110y_4 \qquad (5.21)$$

Эти формулы можно интерпретировать графически, представив на рис5.2 функцию переключения $\Delta F = F_1 - F_2$ в зависимости от трех координат y_1, y_2, y_4:

$$\Delta F \cong 0,2y_1 + 20(1 - y_2) + 10(1 - y_4) - 6$$

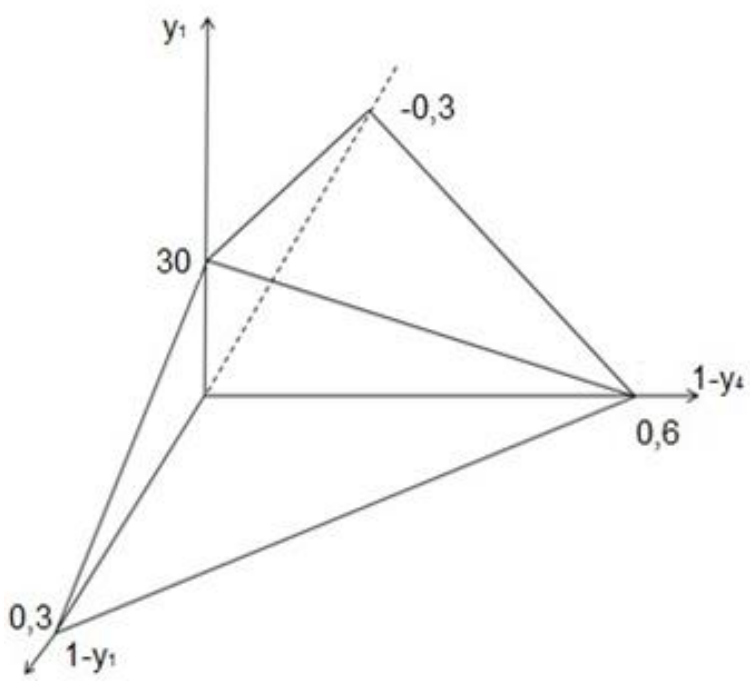


Рис 5.2. Функция переключения альтернатив принятия решений в виде призмы, внутри которой оптимальной является альтернатива о вхождении в воздушный эшелон

Из рисунка видно, что область выбора альтернативы j=1 о входе ЛА в воздушный эшелон в первом приближении имеет вид призмы, т.е. при малых отклонениях и большом расходе топлива лучше придерживаться этой альтернативе. Вне этой призмы принимается решение j=2 об уходе на другую трассу или повторный круг.

5.3 Решение задачи назначения динамических приоритетов при движении судов с произвольным курсом.

Рассмотрим теперь более общий случай при $x_3 \neq 0$ [44]. Тогда, вычисляя третью группу ординат риска $C_3^-, C_{13}^{--}, C_{34}^{--}$ и приравнивая их друг другу, можно убедиться, что $\psi_{13} = \psi_{34} = 0$. Таким образом, осталось доопределить три коэффициента $\beta_3, \gamma_3, \psi_{23}$ функции Беллмана S в общем случае. Для этого, вычислим дополнительно ординаты C_3^+, C_{23}^{+-} , что позволит найти окончательный ответ, если из условия физического смысла решаемой задачи принять $\beta_3 = 0$. Тогда получим $\gamma_3 = -0.73; \psi_{23}^- = 0.2 * 10^{-3}$

Это позволяет с учетом формул (5.20) и (5.21) для представления интересующих нас приоритетов $П_1$ и $П_2$, понимаемых как величины $П_1 = -F_1, П_2 = -F_2$, получить в завершение аналитическую форму возможного ранжирования воздушных судов

$$П_1 = \left[-0,2y_1 + 120y_4 - 2y_1y_4 - 44,4\right] + \left[0.013y_2x_3 - 0.008x_3^2 - 43y_2^2 + 86y_2\right] \quad (5.22)$$

$$П_2 = \left[110y_4 - 14\right] + \left[0.004y_2x_3 - 0.0024x_3^2 - 1.5y_2\right]$$

Здесь $П_1$ - приоритет входа ЛА в воздушный эшелон, $П_2$ - приоритет ухода на повторный круг. По существу эти приоритеты соответствуют количественной оценке затрат топлива при обязательном соблюдении заданных дистанций безопасного движения. Первому слагаемому$П_1$ в квадратных скобках соответствует перевернутая призма на рис5.2, второе слагаемое опускает или поднимает эту призму вниз или вверх.

5.4 Пример расчета динамических приоритетов для воздушных судов, имеющих различные запасы топлива при заходе на посадку по одной трассе

Рассмотрим движение семи воздушных судов, летящих с различными курсами и на разных расстояниях от заданной линии пути, как это показано на рис.5.3.

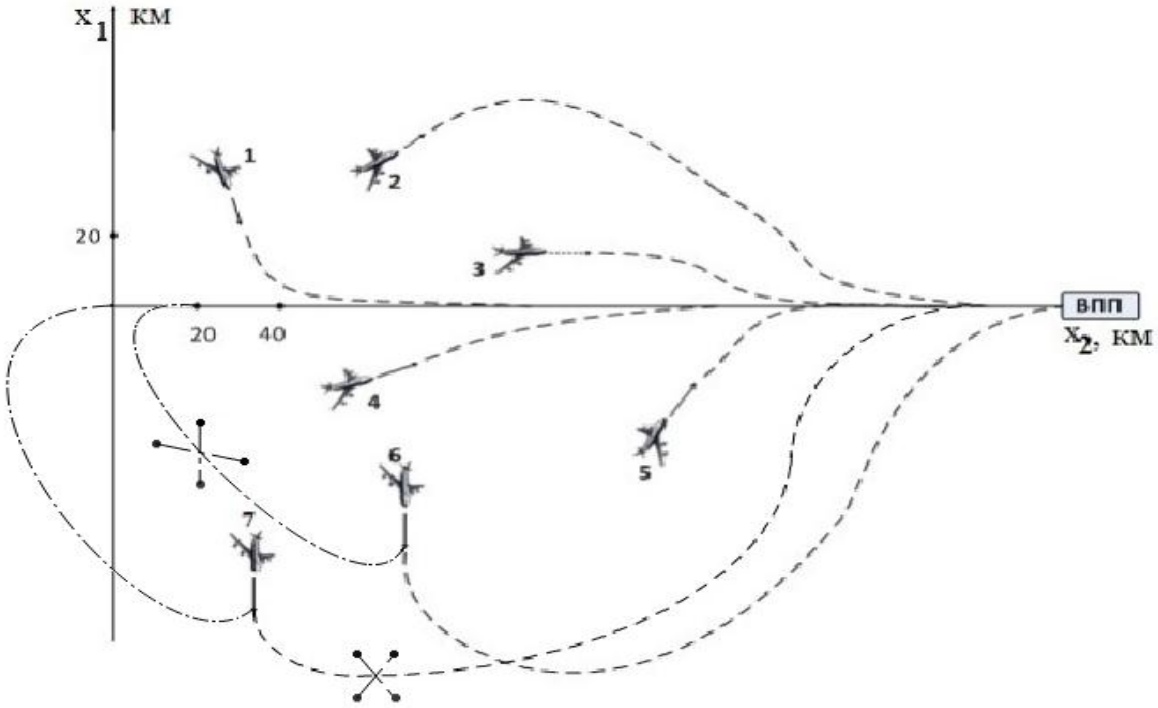


Рис 5.3. Картина движения воздушных судов на заданную линию пути и в тромбон

Таблица 5.1 исходных данных для расчетов содержит на текущий момент времени 21 полетную ситуацию – по 3 варианта запаса топлива для каждого из 7 судов. Координаты x_1 и x_2 даны в километрах, курсовой угол x_3 – в радианах, расход топлива x_4 – в долях от общего запаса ΔV. Вычисления функций риска F_1и F_2приоритетов $П_1$ и $П_2$ проводились по формулам (5.22).

Таблица 5.1. Исходные данные состояния 7 воздушных судов вблизи одной трассы

j	1			2			3			4			5			6			7		
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
x_1	6	6	6	3	3	3	6	6	6	4,5	4,5	4,5	9	9	9	12	12	12	15	15	15
y_1	1	1	1	0,5	0,5	0,5	1	1	1	0,75	0,75	0,75	1,5	1,5	1,5	2	2	2	2,5	2,5	2,5
x_2	6	6	6	6	6	6	2	2	2	4	4	4	4	4	4	2	2	2	4	4	4
$1\text{-}y_2$	0	0	0	0	0	0	0,7	0,7	0,7	0,33	0,33	0,33	0,33	0,33	0,33	0,7	0,7	0,7	0,33	0,33	0,33
x_3	0	0	0	0	0	0	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	0	0	0	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$	$\frac{\pi}{2}$
y_4	0	0,4	0,8	0	0,4	0,8	0	0,4	0,8	0	0,4	0,8	0	0,4	0,8	0	0,4	0,8	0	0,4	0,8
F_1	2,8	54,8	106,8	1,8	53,8	105,8	26,4	78,4	130,4	7	59	111	10,3	62,3	114,3	27,8	77,8	131,8	15,3	67,3	119,3
F_2	2,9	34,9	66,9	2,9	34,9	98,9	1,9	33,9	65,9	2,45	34,5	66,5	2,45	34,5	98,5	15,3	47,4	79,3	2,45	3,45	66,45

Оказалось, что и следовало ожидать, высший приоритет остаться в эшелоне имеют суда 1 и 2. Команде попадания в тромбон или ухода на повторный круг должны подчиняться суда 3, 4, 5, 6, если их запас топлива ($\Delta V - y_4$) велик. Для этой команды нужно выделить в таблице 1 столбец с такими двумя элементами $П_1$ и $П_2$, для которых $П_2 > П_1$.Особый случай относится к воздушному судну 6 в полетной ситуации 18, для которой характерно такое количество потраченного топлива, равное 0.8 ΔV, при котором уход на повторный круг невозможен. Поэтому находящееся в аварийном состоянии судно 6 должно быть введено в воздушный эшелон, что и подтверждено расчетами, т.к. в этом случае $П_1 > П_2$**.** Значит, в этом случае **судно 6 имеет неоспоримой приоритет.**

Поэтому одним из преимуществ данного подходы является ранжирование судов с учетом его ресурсов и технической исправности, что очень важно.

5.5 Задача бесприоритетного обслуживания самолетов при их попадании в тромбон во время захода на посадку

В данном разделе предложена методика расчета вероятных характеристик обслуживания самолетов в тромбоне и оценка максимально допустимого их числа в очереди с использованием критерия максимальной экономичности полетов. При превышении этого числа подлетающий к трассесамолет должен

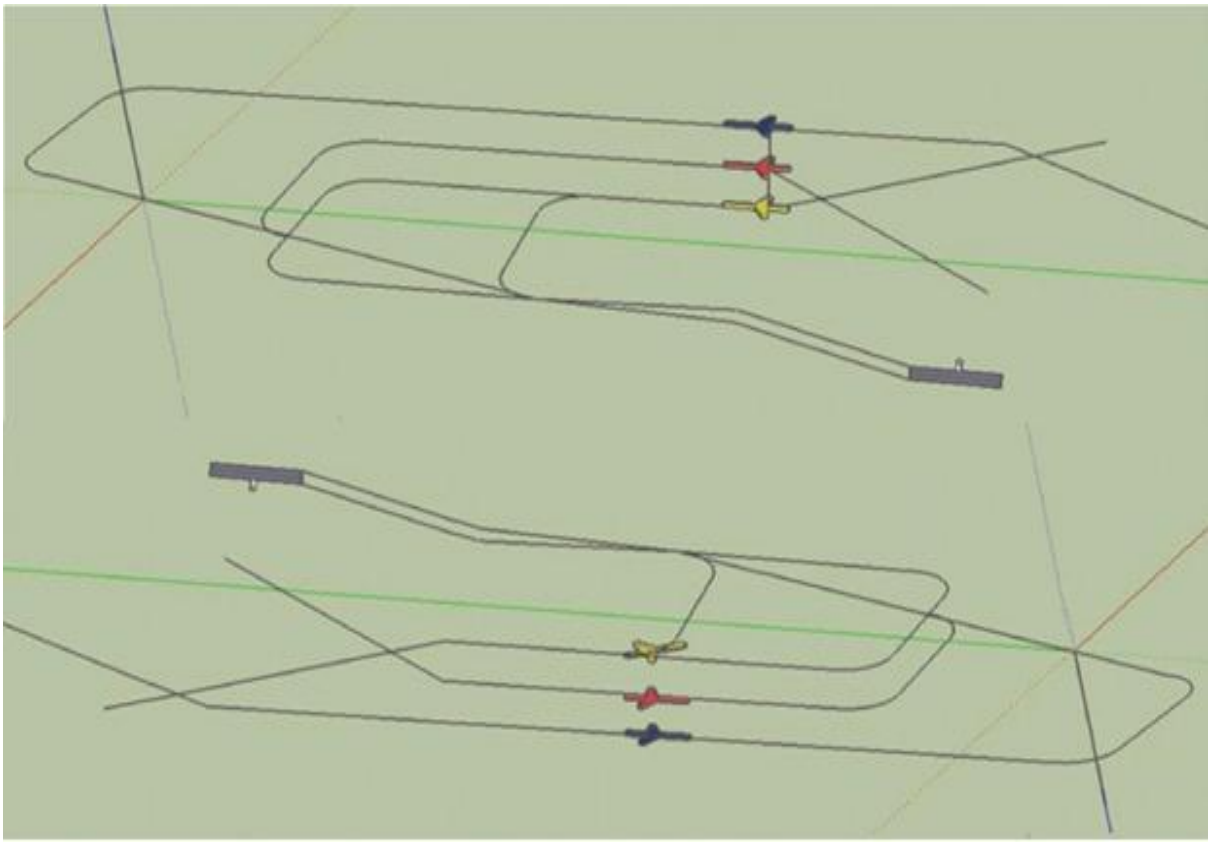


Рис.5.4. Схема захода на посадку по двум линиям пути с помощью тромбонов

пересечь ее и отправиться на другую трассу, и это должно быть оправдано из соображений экономичности полета всей группы самолетов. Ниже рассмотрен случай бесприоритеного обслуживания самолетов при попадании в один из тромбонов, показанных, в частности, на рис 5.4. При этом считается, что аварийные самолеты никогда не будут отправлены в тромбон и сразу будут отправлены на ближайшую трассу[33, 34].

5.6 Случай бесприритеного обслуживания самолетов, попавших в очередь

Вначале разберем многоканальную систему бесприоритетного обслуживания с очередью, также число мест в очереди примем равным максимальным числу самолетов, уже летящих в эшелоне. Пусть на вход СМО поступает поток неприоритетных заявок с интенсивностью λ; интенсивность обслуживания (для одного канала) μ; число мест *n* ;

Состояние системы будем нумеровать по числу заявок, связанных с системой:

X_0 – оба каналы свободны, система свободна от заявок;

$X_{1,0}$ - в системе обслуживается одна заявка;

$X_{2,0}$ - в системе обслуживается 2 заявки;

.
.
.

$X_{n,0}$ – в системе обслуживается *n* заявок и все n каналов заняты;

$X_{n,1}$ – заняты все *n* каналов и одна заявка стоит в очереди;

$X_{n,2}$ – заняты все *n* каналов, и 2 заявки стоят в очереди;

.
.
.

$X_{n,n}$- заняты все *n* каналов и *n* заявок стоят в очереди, т.е. и в очереди мест нет.

Поток заявок вводится с интенсивностью λ, а поток обслуживания - с интенсивностью, равной μ, умноженной на число занятых каналов. При этих условиях можно составить уравнения перехода одних вероятностей в другие в дискретной форме, используя схему перехода в виде стрелок, которая в частности для вероятности простоя P_0имеет вид

$$P_{1,0}(k+1) \longrightarrow \begin{cases} \nearrow & P_{1,0}(k)\left[1-(\lambda_1+\lambda_2+\mu+sv)\Delta t\right] \\ \rightarrow & P_0\left(\mu+(s+1)\right)\Delta t \\ \searrow & P_{1+1,0}\left(\mu(s+1)\right)\Delta t \end{cases} \qquad (5.23)$$

где:

-первое слагаемое описывает состояние, когда ни одна заявка не пришла в очередь и ни одна не обслужилась из нее;

- второе слагаемое описывает, когда в очередь пришла одна простая заявка;

-третье слагаемое описывает состояние, когда из очереди ушла простая заявка, и сразу же пришла одна простая заявка;

Напишем выражения для предельных вероятностей состояний, сразу же обозначая λ/μ = ρ:

$$\left.\begin{aligned} P_0 &= [1+\frac{\rho}{1!}+\frac{\rho^2}{2!}+\frac{\rho^3}{3!}+\frac{\rho^n}{n!}\frac{\rho/n-\rho/n^{m+1}}{1-\rho/n}]^{-1} \\ P_1 &= \frac{\rho}{1!}P_0 \\ P_2 &= \frac{\rho^2}{2!}P_0 P_{n+1} = \frac{\rho^n}{n*n!}P_0 \\ P_{n+2} &= \frac{\rho^{n+2}}{n^2*n!}P_0 \\ P_{n+n} &= \frac{\rho^{n+n}}{n^n*n!}P_0 \end{aligned}\right\} \qquad (5.24)$$

Подставляя заданное нам значение коэффициента загрузки, например равного ρ=0.7 для *n* каналов с длиной очереди *n*, мы получим:

$$P_0 = \left[1+\frac{\rho}{1!}+\frac{\rho^2}{2!}+\frac{\rho^3}{3!}+\frac{\rho^n}{n!}\frac{\rho/n-\rho/n^{m+1}}{1-\rho/n}\right]^{-1}$$

Подставляя P_0 в формулу (5.24), мы можем получить остальные вероятностные состояния n канальной СМО с ожиданием, в том числе интересующую нас вероятность отказа P_{n+n} в обслуживании

$$P_{n+n} = \frac{\rho^{n+n}}{n^n * n!} P_0 \qquad (5.25)$$

Теперь можно построить график логарифмической зависимости вероятности отказа в обслуживании тромбоном от допустимого числа самолетов n. Этой зависимости соответствует график на рис 5.5.

Рис. 5.5. Зависимость вероятности отказа в обслуживании тромбоном от допустимого числа самолетов в очереди.

Формулы (5.24) и (5.25) для бесприоритетного обслуживания известны в общем виде формул Эрланга [54], но они приведены для методического анализа и получения новых формул, приведенных ниже для расчета оптимальной длины очереди самолетов в тромбоне.

5.7 Расчет оптимального числа самолетов в очереди в тромбоне

Как уже было сказано выше, длина очереди самолетов в тромбоне обычно ограничена как по экологическим причинам при его расположении над крупным населенным пунктом, так и по экономическим соображениям. Рассмотрим последнее обстоятельство более подробно при следующей постановке задачи[34, 58].

Пусть при попадании самолетов в очередь, существующую в тромбоне, самолет должен дополнительно пролетать некоторый путь до выхода на трассу, пропорциональный имеющемуся числуk впередилетящих самолетов. Пусть безопасное расстояние l между ними задано и примерно равно 20 км. Это означает, что при скорости 400 км/час эти самолеты будут приземляться друг за другом в аэропорту через время $l/v = 200$ сек, т.е. примерно через три минуты. Этому времени соответствуют лишние затраты топливаR_1, а общие затраты в тромбоне при очереди длиной k_0 будут равны$k_0 R_1$.

Пусть также у самолета есть другая возможность перелетать на соседнюю трассу, минуятромбон, преодолев другой путь, длина которого равна в среднем расстоянию между соседними трассами. При имеющейся конфигурации Московского аэроузла параметром круга с радиусом 100 км получим примерную оценку длины этого пути примерно 80 км, что определяет свои дополнительные затраты R_2 топлива.

$$R_2 = k_0 R_1 \text{, где } \quad k_0 = 4$$

Ставится задача выбора такой оптимальной величины n допустимого числа самолетов в тромбоне, при котором общие затраты топлива R_0 для всех самолетов будут минимальны.

С учетом вероятностного состояния n – канальной системы обслуживания можно записать следующий параметрический критерий

оптимальности

$$R_0 = R_1 \sum_{i=1}^{n} iP_i + R_2 P_n = R_1 \sum_{i=1}^{n-1} iP_i + P_n(n + R_1 + R_2) = R_1 \left[\sum_{i=1}^{n-1} iP_i + P_n(n + k_0) \right] \to \min \quad (5.26)$$

Выражение в квадратных скобках дает возможность выбрать нужное значение n, если для определения вероятностей $P_i, i = 1 \ldots n$ воспользоваться формулами Эрланга для бесприоритетного обслуживания. Так как средняя скорость обслуживания самолетов в тромбоне выше скорости их поступления, т.е. $\rho < 1$, зададимся значением $\rho = 0,7$. Расчеты показали, что при $k = 4$ минимум R_0 обеспечивается при $n = 5$, т.е. расчетная длина тромбона не превышает 100 км и оказалась соизмеримой с дистанциейухода на повторный круг.

5.8 Выводы по главе 5

1. Предложен удобный для расчетов подход аналитической оценки динамических приоритетов воздушных судов при заходе на посадку, позволяющий их ранжировать и последовательно вводить в план попадания на заданную линию пути.
2. Найденные формулы позволяют учесть не только положение судна в пространстве, но и его запасы топлива и техническое состояние в процессе ранжирования.
3. Найденные численные значения коэффициентов в формулах (5.22) могут вызвать возможную критику и потребовать внесения определенных корректив. Дело в том, что процессы оптимального приближения судна к заданной линии пути были заменены экспоненциональными с помощью дифференциальных уравнений первого порядка. Однако эти коррективы принципиально ничего не изменят в главном – с помощью уточненной, но опять-таки компактной аналитической формы (5.22), удобной для расчетов, может быть найдена функция предпочтения между судами для каждой линии пути. Это позволит определить порядок последовательного планирования траекторий входа судов в нужный воздушный эшелон как без учета на начальном этапе возможных пересечений траекторий, так и на заключительном этапе – с учетом этого.
4. Показано, что при бесприоритетном обслуживании самолетов без учета оставшегося на борту топлива, нужная малая вероятность отказа при заходе в заданный эшелон требует использования тромбона значительного размера, который ограничен другими причинами.
5. Показана возможность выбора оптимального числа самолетов в очереди в тромбоне, при котором в среднем обеспечивается максимальная экономичность полетов группы подлетающих к Москве самолетов.

6. Замечание о назначении коэффициентов штрафов m_i . В данной работе реализуется следующий подход. Вначале с учетом результатов главы 3 назначаются коэффициенты экономичности полета $m_1=m_3=1$ и его безопасности $m_2 = m_4 = 40$.

Затем по первым результатам моделирования на ЭВМ они уточняются в зависимости от реальной дистанции между трассами, которые нужно выбрать с тем расчетом, чтобы для аварийных судов с малым запасом топлива штраф за расходы топлива на перелет с биссектрисымежду трассами на одну из них был меньше штрафа за потерю безопасности. В результате оказалось, что значенияm_i лучше назначать следующим образом

$$m_1=1; m_2=200 : m_3=1,\ m_4 = 40$$

Эти коэффициенты взяты за основу новых расчетов в следующей главе, когда исходные формулы функций риска F_1и F_2 имеют вид

$$F_1 = 1,4 + 0,2y_1 + 1,2y_1^2 + 200(1 - y_2)^2 - 425y_4 \qquad (5.27)$$

$$F_2 = 14 + 1,5y_2 - 80y_4$$

Глава 6. Решение задачи распределения воздушных судов при их заходе на посадку

6.1 Алгоритм назначения приоритетов воздушных судов для каждой ВПП Московского аэроузла безучета их близости на трассе

Формулу (5.27) для вычисления функций риска F_1 и F_2 можно использовать по-другому без учета функцииF_2. Если взять величину (- F_1) в качестве приоритета П и просчитать приоритеты на всех ВПП, то можно для каждого воздушного судна проранжировать варианты ВПП, тем самым выбрав наиболее экономичный вариант для посадки. С помощью этого приема можно вычислить относительный динамический приоритет каждого судна. При этом дополнительно формулу (5.27) внесены следующие уточнения. Во-первых, зависимость приоритета от относительного расстояния y_2 между судамив эшелоне взята не в виде степенного полинома, а в виде экспоненциальной функции e^{-3y_2} . Во-вторых, зависимость от отклонения y_3по курсу – симметричная независимо от знака курса, поэтому в формуле используется модуль$|y_3|$, а отклонение y_3 измеряется в радианах. Кроме того, внесено необходимые уточнение в виде слагаемого $0{,}1\,y_3^2$.

Это позволяет с учетом формулы (5.27) получитьв завершение аналитическую форму вычисления приоритета для воздушного судна [2] :

$$П = (-1{,}4 + 0{,}2y_1 + 0{,}6(3 - |y_3 - 3|) + 0{,}1y_3^2 + [-200e^{-3|y_2|} + 250y_4] \qquad (6.1)$$

По существу этот приоритет в первую очередь учитывает оценку затрат топлива при обязательном соблюдении заданных дистанций безопасного движения между судами, т.к. коэффициенты слагаемых, стоящих в фигурных скобках, во много раз больше остальных. В формуле (6.1) есть слагаемые, стоящие в круглых скобках (они отвечают за экономичность полета т.к. y_1 и y_3 характеризуют длину пути при попадании самолета на заданную трассу), и в квадратных скобках (они отвечают за безопасность полета, т.к. показатель

y_4 определяет оставшийся запас топлива, а y_2 определяет дистанцию между самолетами при движении в эшелоне по заданной трассе), что очень важно для оценки угрозы успешного достижения места посадки.

Однако в самом начале расчетов величина y_2 зависящая от расстояния между судами на трассе, заранее неизвестна. Поэтому вначале нужно найти приоритет судов без учета y_2 а затем, расставив суда на заданной линии пути, доопределить y_2 и получив уточнение в приоритете, сформировать окончательные списки ранжирования судов.

6.2 Алгоритм последовательного формирования приоритетных списков судов для каждой трассы

Представленный на рис6.1 алгоритм приоритетного ранжирования воздушных судов по трассам работает следующим образом. Считается, что предварительно суда проранжированы по спискам для всех четырех трасс, а списки для каждой трассы имеют в своем начале (в первых строках) суда с максимальным приоритетом. В своей работе алгоритм организует два цикла – внутренний цикл при изменении *i*(где *i* - номер воздушного судна) имеет цель скорректировать список приоритетов для одной трассы *j*, внешний цикл по *j*повторяет необходимые действия для множества трасс $j = 1,..., N$ [37].

Во внутреннем цикле вначале берутся данные x_i, z_i, ψ_i местоположения *i*-того судна относительно трассы *j*, и вычисляется координата Φ_{oi} попадания судно на заданную линию пути относительно точки начала снижения по глиссаде для*j*-той ВПП. Вычисленная координата Φ_{oij} попадает в промежут-

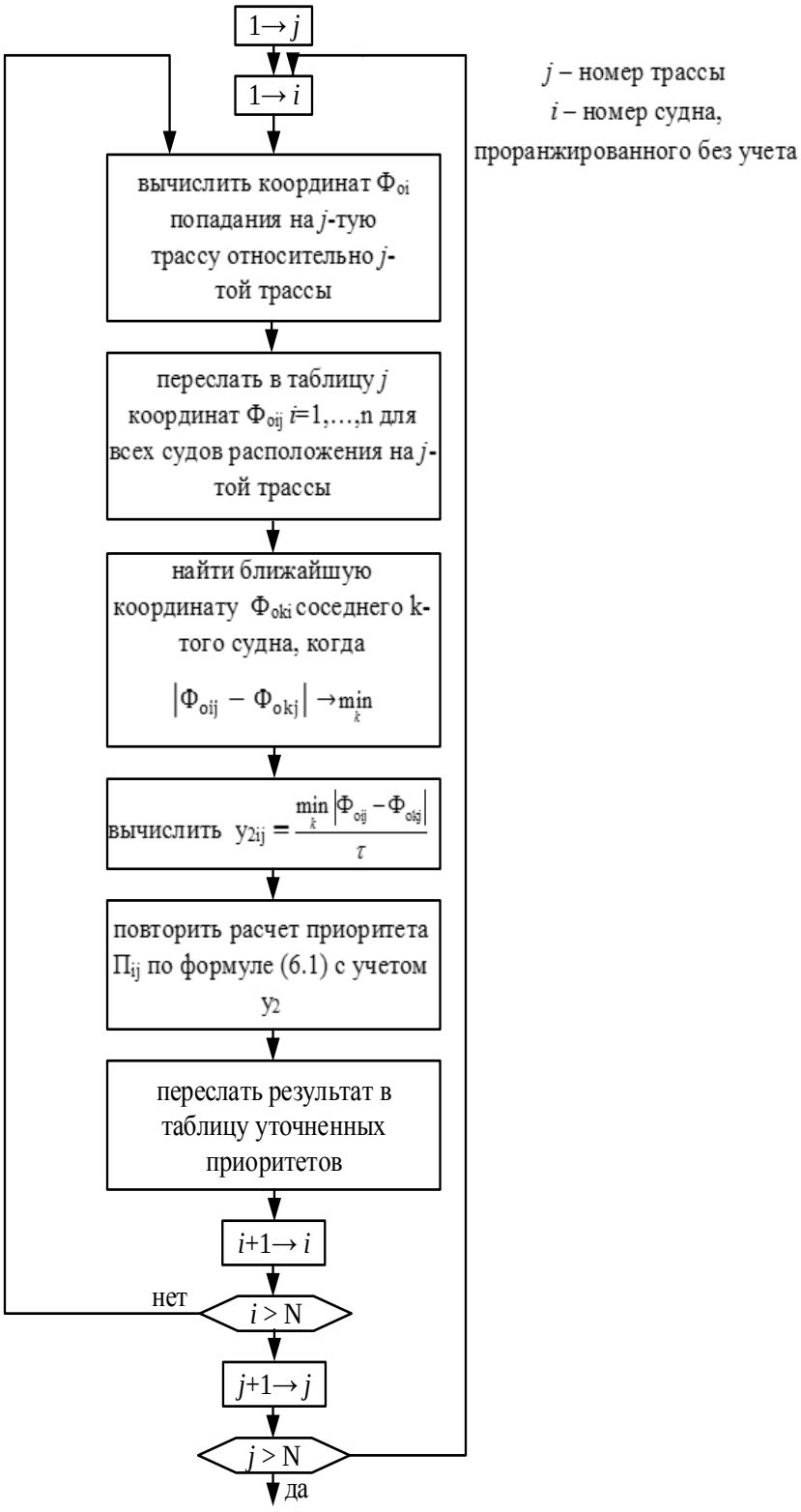


Рис.6.1. Блок схема алгоритма последовательного формирования приоритетных списков судов при заходе на посадку

-очную таблицу 1, хранящую все вновь вычисленные координаты расположения судов для*j*-той трассы.

Затем вновь вычисленная координата сравнивается с уже имеющимися значениями координат соседних судов, и среди них ищется ближайшее *k*-тое судно по критерию

$$\left|\Phi_{oij} - \Phi_{okj}\right| \to \min_{k} \qquad (6.2)$$

Далее найденное значение позволяет найти величинуy_2, а значит – вычислить скорректированное значение приоритета *П*по формуле (6.1), но уже с учетом y_2.

Полученные решения позволяют обновить таблицу первоначальных приоритетов сначала для одной трассы, затем – для всех трасс.В итоге формируются списки первоочередных судов для каждой трассы с учетом экономичности и безопасности их захода на посадку.

6.3 Пример распределения 20 воздушных судов в Московском аэроузле

Рассмотрим задачу формирования нужных списков при следующих исходных данных, представленных в таблицах 6.1 и 6.2. Картина расположения судов показана на рис 6.2.

Таблица 6.1. Входные данные программы о ВПП

№ ВПП	1	2	3	4
Ψ_{oj} (град.)	136	194	238	246

Таблица. 6.1 содержит информацию о курсах четырех ВПП, в таблице 6.2 находятся исходные параметры движения 20 самолетов. Результаты вычисления приоритетов по формуле (6.1) без учета близости судовна трассе (как по отношению к сзадилетящему,так и впередилетящему судну), используя формулу (6.1) в случае $y_2 = \frac{x_2}{r} > 1$, показаны в виде таблицы 6.3.

Таблица 6.2. Входные данные программы о воздушных судах

№ ЛА	ζ_k (км)	Z_k (км)	Ψ_k (град.)
1	0	-300	100
2	100	-250	40
3	200	-150	316
4	280	-40	120
5	300	60	140
6	200	220	225
7	60	280	270
8	-40	260	260
9	-120	220	280
10	-220	200	10
11	-240	100	0
12	-280	-20	310
13	-200	-180	40
14	-140	-260	90
15	-60	-260	85
16	***240***	-50	***165***
17	***240***	-80	***165***
18	240	260	242
19	320	310	238
20	160	260	250

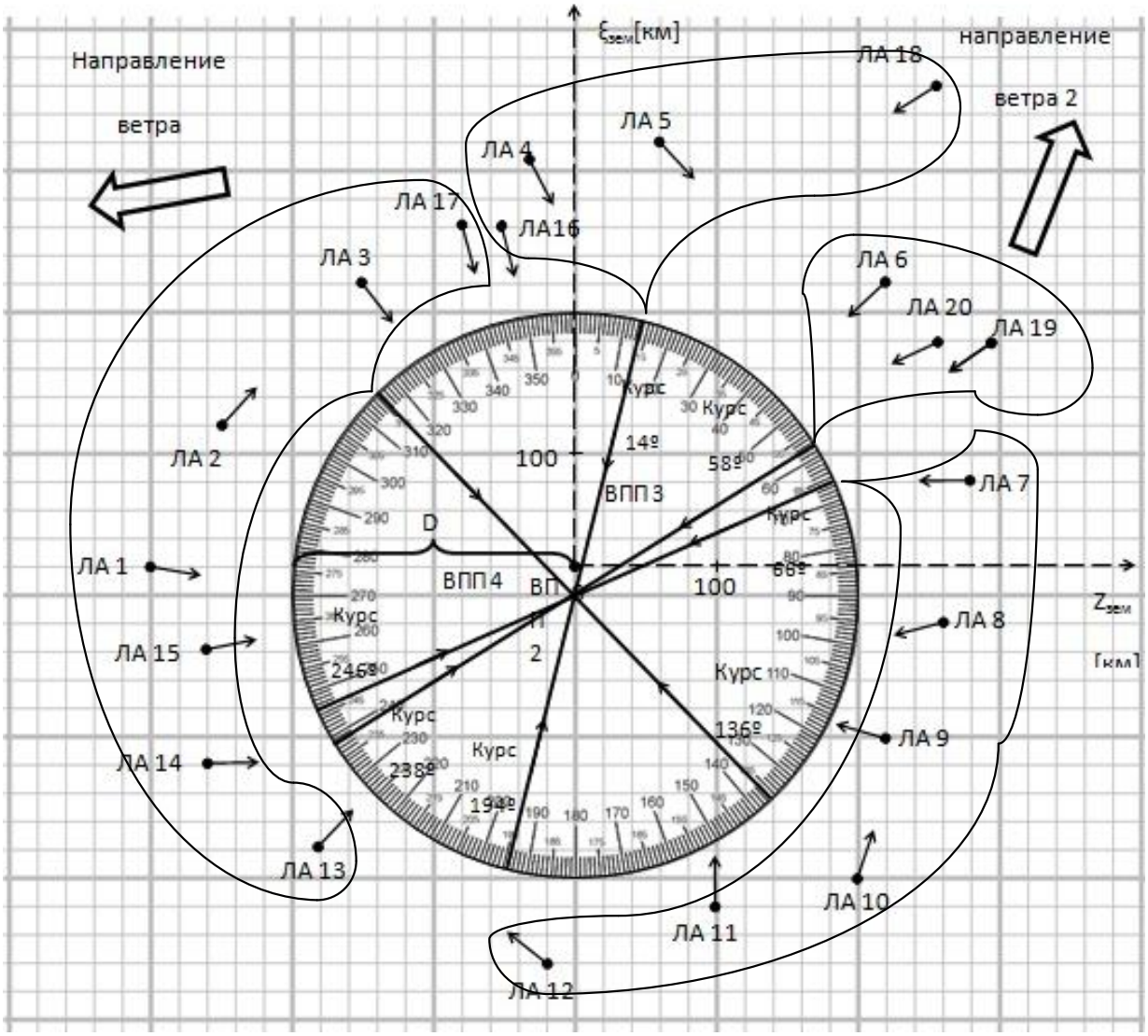


Рис 6.2. Картинка захода на посадку для самолетов по приоритету

Таблица.6.3. Первоначальные приоритеты, вычисленные без учета y_2

	ВПП 1	ВПП 2	ВПП 3	ВПП 4
ЛА 1	-8.7	-11.03	-10.55	-10.6
ЛА 2	-6.1	-9.1	-10.75	-10.63
ЛА 3	-0.6	-7.3	-10.9	-10.8
ЛА 4	-5.3	-3.7	-9.4	-8.9
ЛА 5	-8.2	-1.3	-9.1	-8.9

ЛА 6	-10.9	-5.5	-2.4	-4.0
ЛА 7	-10.87	-8.8	-3.7	-3.3
ЛА 8	-10.56	-8.7	-5.5	-5.3
ЛА 9	-10.35	-10.91	-8.7	-8.6
ЛА 10	-11.16	-11.8	-11.05	-11.04
ЛА 11	-11.37	-11.5	-10.89	-10.87
ЛА 12	-10.7	-11.37	-11.0	-10.8
ЛА 13	-10.5	-11.56	-11.44	-11.5
ЛА 14	-10.2	-11.69	-11.03	-11.04
ЛА 15	-9.1	-10.85	-10.22	-10.18
ЛА 16	-6.1	***-5.7***	-9.4	-9.2
ЛА 17	***-6.5***	-6.6	-9.6	-9.6
ЛА 18	210	215.1	216.2	213.4
ЛА 19	-11.17	-8.7	***-2.11***	-2.15
ЛА 20	-11.09	-8.5	***-1.6***	-1.8

В этой таблице воздушные суда пока что по приоритету не рассортированы. Нужно дополнительно заметить, что для вычисления нужной координаты y_2нужно каким-то образом найти координату q_1попадания судна на трассе. В данной работе также, как и при описании бокового движения судна был применен приближенный подход к нужной оценке. В частности, при курсе судна $\psi_i = 0$, совпадающем с посадочным курсом, было принято допущение, чтовеличина q_1как одного из катетов

вдвое меньше второго катета $|x_1|$, т. е судно приближается к линии пути со средним курсом примерно 30°, т. е при $\psi_i = 0$ имеем $q_1 = 0.5\, x_1$.

Если же имеет место несовпадение курсов, Ψ_1 то длина пути входа в эшелон увеличивается, и соответственно увеличивается катет q_1.

$$q_1 \cong 0,5x_1 + k\left|\Psi_i\right|$$

где, $kx\dfrac{\Psi}{360}R = \dfrac{\Psi V^2}{360a} = \dfrac{\Psi.100^2}{360.5} = 10700\Psi \approx 0.7 < 1$

Необходимо также подчеркнуть важное обстоятельство – чтобы определить расстояния между судами, необходимо их предварительно расставить, а для этого нужно знать состав судов, претендующих на полет по заданной трассе. Именно для этого сформирована таблица 6.3. первоначальных приоритетов. После их уточнения с помощью координат y_{2i} повторно должен формироваться окончательный список.

Таблица 6.4. Список самолетов в эшелоне для ВПП-1

ЛА, заходящие на ВПП1 (Курс 136)							
3	17	15	1	2	14	13	-

Их относительное расположение на линии пути для ВПП-1 характеризуется координатой q_1, и перечень этих координат представлен в таблице 6.4.

Таблица 6.5. Самолёты, которые заходят на 136 курс, и их значения q_1

ЛА	17	3	15	1	14	2	13
q_1	-228.214	-268.906	-137.451	-208.398	-79.9036	-266.438	18.8295

С помощью координаты q_1 для каждого самолета можно установить относительное местоположение самолетов на трассе и затем определить дистанцию до ближайшего самолета, впереди или сзади летящего, что позволяет определить минимальную дистанцию x_2 , представленную в таблице 6.

Таблица 6.6. Самолёты, которые заходят на 136 курс, и их значения x_2

ЛА	17	3	15	1	14	2	13
X_2	21	1	31	21	61	1	500

Координата y_2 связана с координатой x_2 простым соотношением $y_2 = x_2 / r$.Чтобы количественно оценить разницу в вычислении приоритетов с учетом координаты y_2 и без нее, сравним результаты расчетов приоритетов в обеих случаях, показанные в таблицах 7 и 8.

Таблица 6. 7. Самолеты, которые заходят на 136 курс, и их приоритеты без учета y_2

ЛА	17	3	1	15	14	2	13
Приоритеты без учета y_2	-6.5	-0.6	-9.5	-9.1	-108.8	-6.1	-105

Таблица 6.8. Самолёты, которые заходят на 136 курс, и их приоритеты с учетом y_2

ЛА	17	3	1	15	14	2	13
Приоритеты с учетом y_2	-6.5	**-120.6**	-8.7	-9.1	-10.2	**-126.1**	-10.5

Как видно, приоритеты отличаются у самолетов №2 и №3, а именно у них дистанция до ближайшего судна, т.е. друг до друга, меньше допустимой, поэтому $y_2 < 1$,значит первая часть формулы в квадратных скобках, зависящая от y_2, правильно реагирует на ее учет. После внесения поправки из-за y_2 самолет №3 попадает либо в тромбон ВПП-1, либо в план захода на посадку на ВПП-2, и выбывает из списка для ВПП-1.

В итоге формируются окончательные списки приоритетных судов для каждой из трасс, так показано итоговой таблицей 6.9.

Таблица 6.9. Список самолетов в эшелоне для четырех ВПП вместе с очередями

ЛА, заходящие на ВПП-1 (Курс 136)						
17	1	15	14	2	13	3
ЛА, заходящие на ВПП-2 (Курс 194)						
18	5	4	16	-	-	
ЛА, заходящие на ВПП-3 (Курс 238)						
19	20	6	-	-	-	
ЛА, заходящие на ВПП-4 (Курс 246)						
7	8	9	10	11	12	

Видно, что эти списки имеют разную длину из-за внезапного изменения направления ветра и “векторения” движения воздушных судов. Если максимальная длина *M* этих списков ограничена, то часть самолетов не будет допущена на вход в эшелон, и они будут отправлены в очередь – соответствующий тромбон. Например, пусть *M*=4, тогда для захода на ВПП-1 в эшелон не попадут самолеты № 2, 13, 3 и они полетят в очередь в свой

тромбон трассы 1. Так же для заходана ВПП-4 в эшелон не попадут самолеты № 11, 12 и они полетят в очередь в тромбон трассы 4.

6.4 Алгоритм определенияпервоочередности приземления судов на каждом ВПП

6.4.1 Постановка задачи

Ставится задача эффективного планирования очерёдности движения прибывающих воздушных судов при внезапном изменении условий посадки (технические, метеорологические и т.д.) хотя бы по одному аэродрому, с выводом воздушных судов на новые стандартные маршруты прибытия[37].

1. Задано ограниченное число стандартных маршрутов прибытия в московском аэроузле (считаем, что в любой момент времени для каждого ВС можно задать курс в соответствии с требованием стандартного маршрута прибытия) ($Y^t_{k(i)}$);
2. Задано количество воздушных судов, прибывающих на аэродромы Московского аэроузла в соответствии с первоначальным планом (K_0);Каждое воздушное судно характеризуется четырьмя координатами
   - высота (H_i);
   - ближайшее расстояние до «нового» STAR (Z_i);
   - расстояние от воздушного судна до (faf\fap)конечной точки захода на посадку (X_i);
   - курс полета ВС, отсчитываемый против часовой стрелки относительно $Y^t_{k(i)}$.
3. Скорость полета воздушных судов для упрощения решения задачи считается одинаковой, соответственно время достижения судном конечной цели определяется длиной пути до конечной точки X_i.
4. Задано значение безопасной дистанции между ВС (D);
5. Задан минимальный радиусr движения ВС в развороте;

6. Координата Y_i – выхода на «новый» STAR рассчитывается и имеет ограничения $0<Y_i<X_i$ т.е. ВС не должно двигаться в противоположную сторону от конечной точки посадки faf , в противном случае выполняется ожидание типа «вираж» с учетом расчетного времени (T_z) ухода на запасной аэродром;
7. Учитывается оптимальный высотный режим для прилета на конкретный аэродром $H_{min}<H_i<H_{max}$

Требуется :

- определить эффективную очерёдность захода на посадку группы ВС на «новые» STAR с соблюдением условия безопасности.

- в случае прогнозирования невыполнения ограничений составить список воздушных судов, которым необходимо рекомендовать уход на запасной аэродром до наступления расчетного времени ухода по каждому воздушному судну в списке.

Разработку алгоритма программы начнем с описания заданных параметров.

k=1..n – итератор для номера воздушного судна, находящегося в ожидании разрешения войти в эшелон и совершить посадку

j=1..p – итератор для номера посадочной трассы

z_k – продольная координатаk-ого судна. Ось продольных координат направлена по линии посадочной трассы

z_k – боковая координата k-ого судна

Ψ_k– курсовой угол k-ого судна относительно земного севера

Θ_j – курсовой уголj-ой трассы относительно земного севера

y_{4k}– доля потраченного топлива k-ого судна

Описанные параметры являются числовыми массивами.

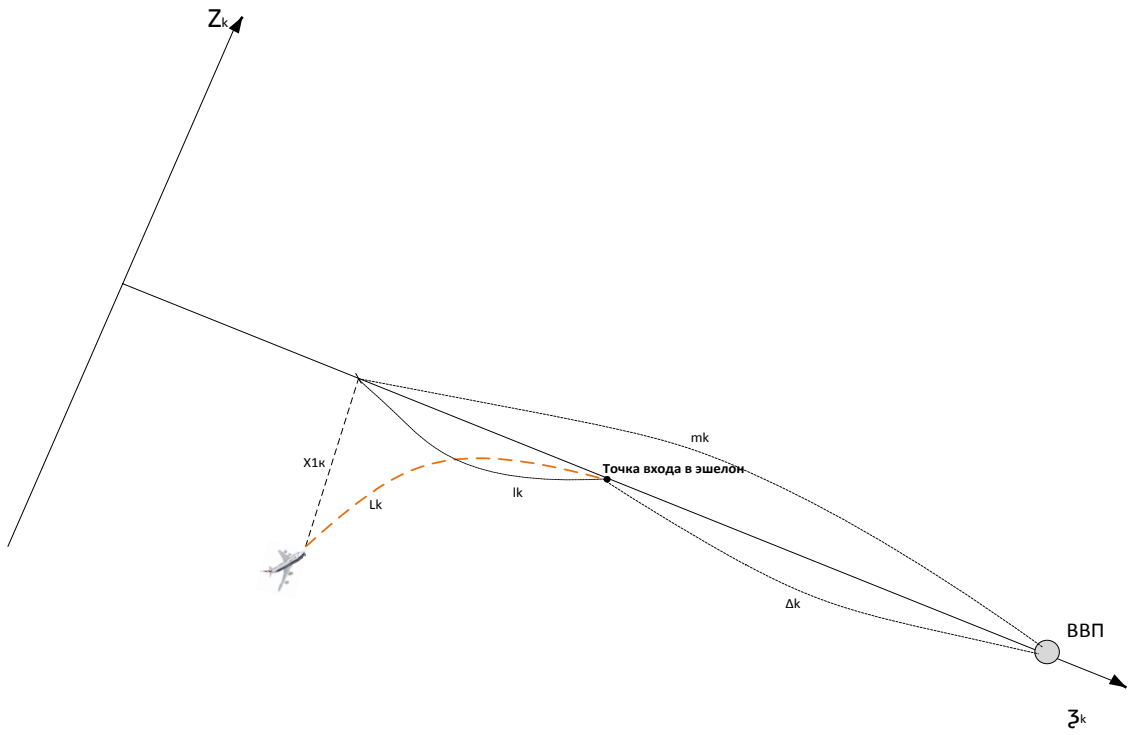


Рис.6.3 Летная ситуация вблизи*j*-ой трассы для *k*-ого судна

Пользуясь рис. 6.3, можно найти следующие параметры:

$x_3 = |\Psi_k - \theta_j|$ - разность курсовых углов трассы и воздушного судна

$x_1 = |Z_k|$ - расстояние между трассой и воздушным судном

$m_k = |\zeta_k|$ - расстояние от ближайшей точки на трассе до ВПП

$l_k = x_3 * \sqrt{3} + R * \frac{x_3}{60}$ – расстояние от ближайшей точки на трассе до точки входа в эшелон (R=10)

$L_k = l_k$ – длина пути при входе судна в эшелон

При этом если $x_1 > R$ выражения для $l_{kи}$ и L_k примут следующий вид:

$$\begin{aligned} L_k &= x_1 - R + l_k \\ l_k &= R * (\sqrt{3} + \frac{x_3}{60}) \end{aligned} \qquad (6.3)$$

Для нахождения x_2 потребуются следующие дополнительный параметрM_k – расстояние до ближайшего судна в эшелоне для каждого судна. Тогда $x_2 = |M_k + (L_k - L_i)|$, где *i*-номер судна, ближайшего к*k*-ому в эшелоне.

Также при разработке алгоритма следует учесть ограничения на значения параметров. Если разность курсовых углов посадочной трассы и воздушного судна x_3>90°, то осуществлять посадку на заданную трассу данного судна нецелесообразно из соображений экономичности и безопасности полета (за исключением аварийных случаев, которые в данномслучае не рассматриваются). Расстояние от ближайшей точки на трассе до ВПП ограничено, т.еm_k>R, в противном случае для безопасности полета воздушному судну должен быть присвоен минимальный приоритет для посадки[44].

6.4.2 Формирование общего алгоритма назначения очередности сучетом удаленности от аэродрома.

Считаем, что решив задачу для одного маршрута STAR(Стандартный маршрут прибытия) можно решить эту задачу для множества подобных маршрутов. В качестве иллюстрации на рис 6.4 изображено множество случайно расположенных в пространстве пяти воздушных судов. Каждое из них следует со случайным курсом. Ось Z ориентирована по сторонам света (север, юг, запад, восток) и направлена в сторону *faf* ВПП. Аналогично ориентирована ось x и направлена в сторону ближайшей точки стандартного маршрута[63, 64].

Выход на стандартный маршрут производится путем предварительной оценки длины и необходимости маневрирования с соблюдением условий безопасности воздушного движения. На рисунке 6.5 представлена укрупнённая блок-схема алгоритма определения очередности прилёта с выходом на стандартный маршрут.

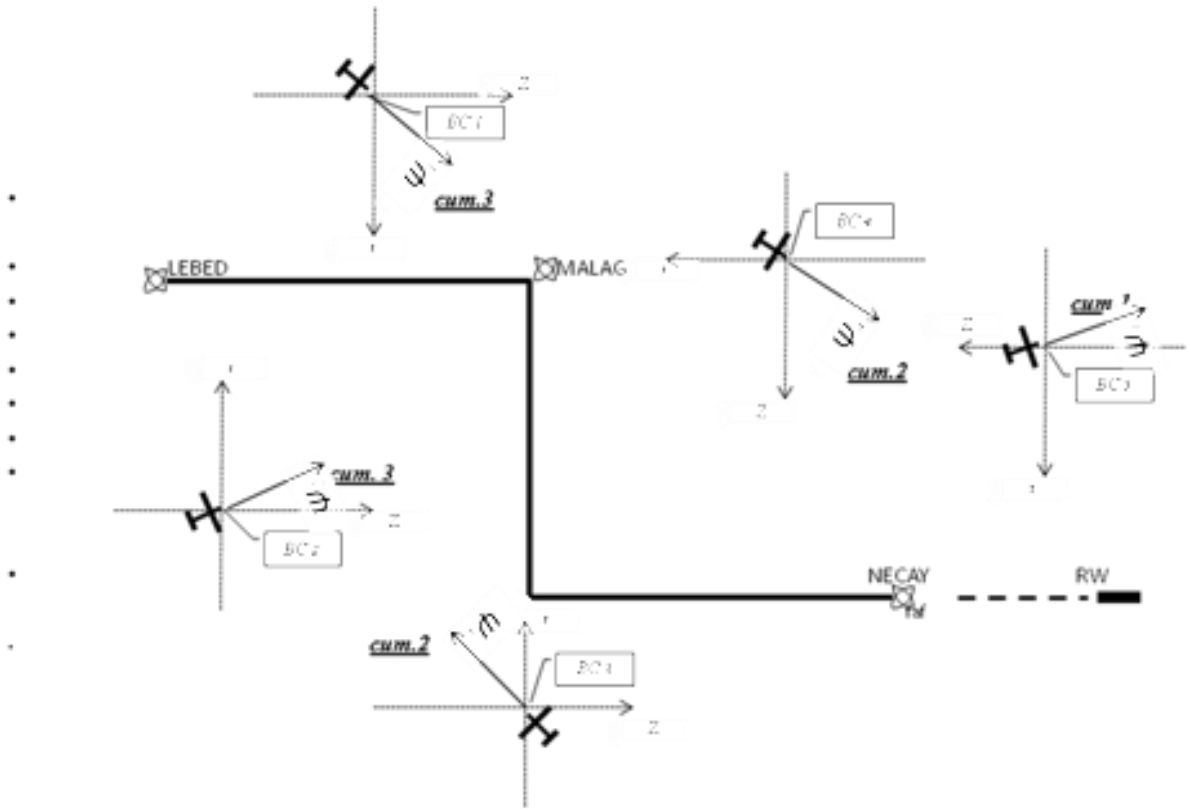


Рис.6.4 . Расположение воздушных судов в начальный момент времени t_0 относительно STAR (LEBED, MALAG, NECAY)

В блоке 1 происходит предварительная оценка длины пути оптимального входа в стандартный маршрут без учета условий безопасности воздушного движения. Множество анализируемых воздушных судов сортируется по трём качествам – «пригодные ВС» (разворот по курсу следования меньше или равно 30 град.–ситуация.3) , «ВС требующие дополнительной оценки» (разворот по курсу следования более 30 град. - ситуация.2), «непригодные» (воздушные суда не успевающие снизиться до приемлемой высоты, или требующие разворота на угол более 360 град. Также к ним относятся воздушные суда, не успевающие произвести посадку до наступления момента времени ухода на запасной аэродром).В блоке 2 производится предварительное ранжирование ВС по критерию минимума длины пути.

В блоке 3 уточняется приоритет очередности при внесении условия соблюдения безопасных интервалов эшелонирования между ВС

анализируемого множества, после чего также может возникнуть подмножество «непригодных» ВС Блок 4 повторно ранжирует ВС с учётом безопасности воздушного движения и эффективности построение очереди прилетающих ВС.

Группа ВС, характеризуется ситуацией №3 (рис.6.4 ВС1,ВС2), когда ВС способны выйти на STARбез существенного дополнительного маневра. Длина пути L_i каждого *i-го* ВС этой группы определяется прямолинейными отрезками, начиная с кратчайшего расстояния до STARс прибавлением оставшейся части стандартного маршрута. В результате работы первого блока формируются две таблицы:

- таблица 1 (состоит из ВС, которым отказано в обслуживании);

- таблица 2 (состоит из ВС, характеризующихся ситуацией №3.) В таблице в случайном порядке располагаются номера ВС и соответствующие значения длинны участка путиL_i и суммарной длины оставшегося пути отрезков стандартного маршрута $(L_{(i+1)} + \sum y_{i+1}) > (L_i + \sum y_i)$.

Работа второго блока предварительно устанавливает очередь по критерию минимальной длины пути $(L_{(i+1)} + \sum y_{i+1}) > (L_i + \sum y_i)$. Результаты размещены в таблице 3.

В блоке 3 производится контроль и рассредоточение ВС на безопасную продольную дистанцию *D*. С учетом безопасности воздушного движения формируется таблица 4. При необходимости производится повторное ранжирование. В конечном счёте на выходе результат соответствует пятой таблице.

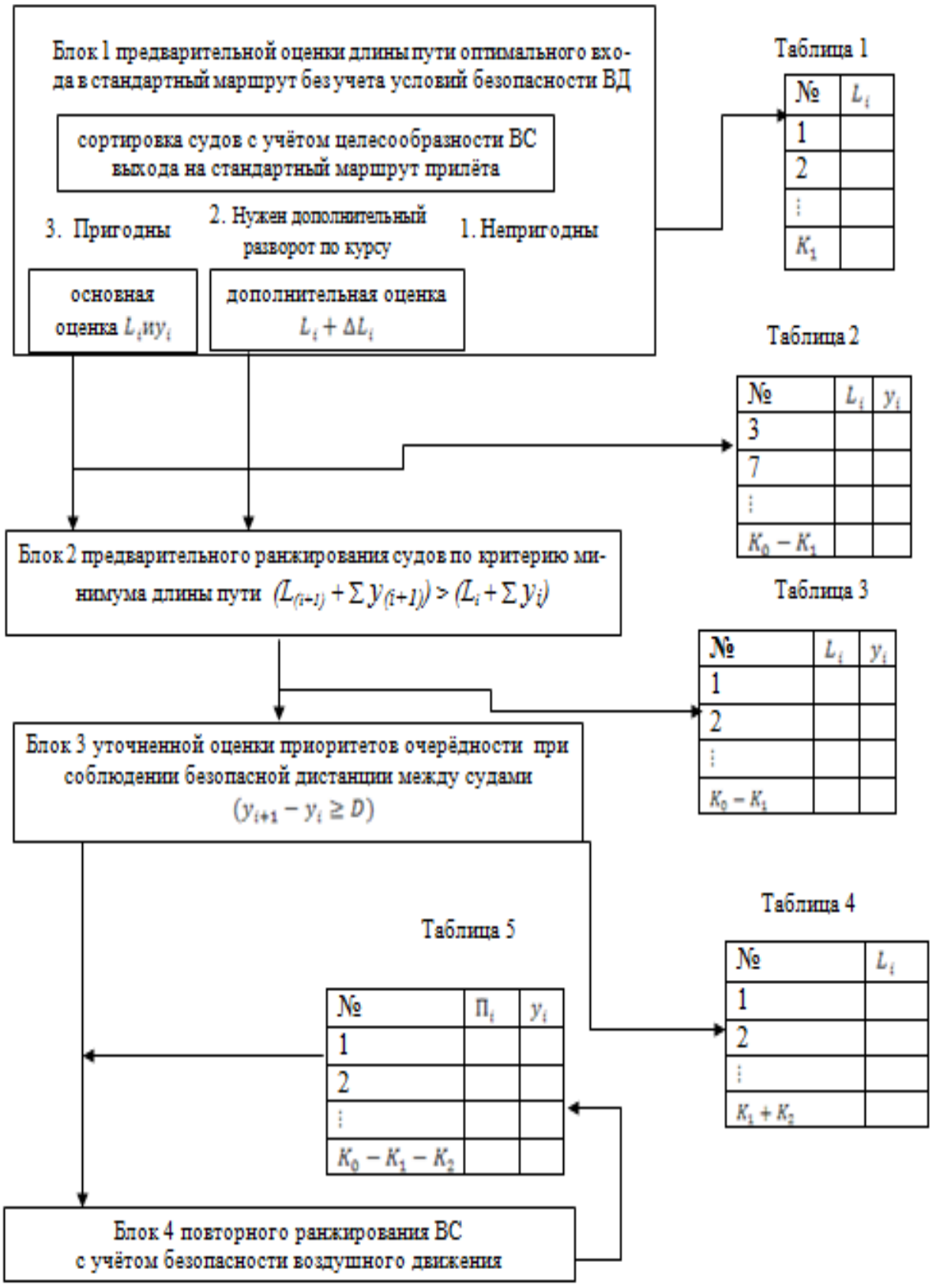


Рис.6.5. Укрупненная блок-схема алгоритма определения очередности прилёта с выходом на стандартный маршрут

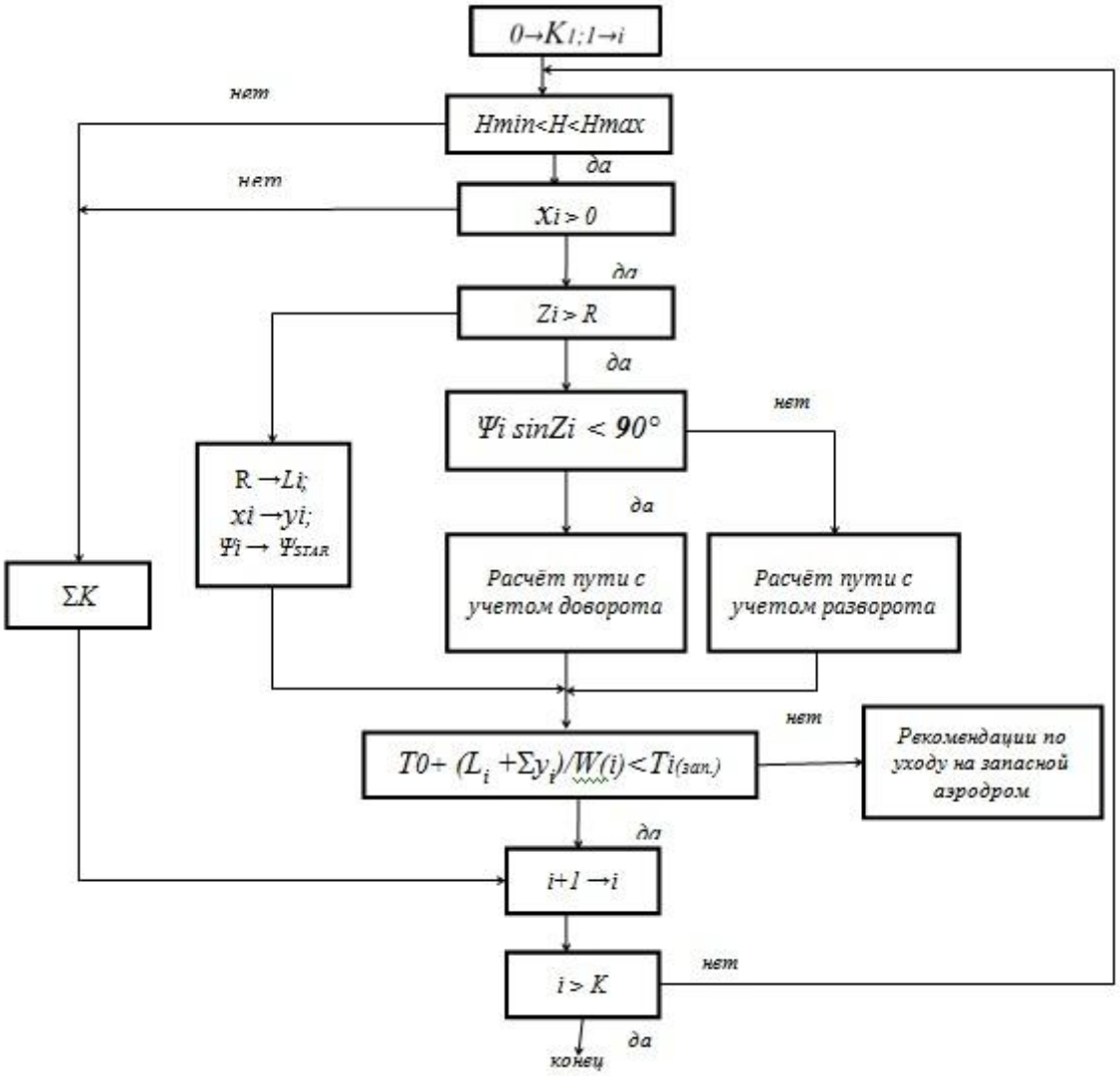


Рис. 6.6. блок-схема предварительной оценки длины пути оптимального входа в стандартный маршрут без учета условий безопасности воздушного движения.

На рис 6.6 представлена предварительная оценка длины пути, затрачиваемого воздушным судном для оптимального попадания в стандартный маршрут прибытия и движения по данному маршруту до точки faf/fap.

В каждом конкретном случае вычисляются три слагаемых:

- L_1 – путь или время дополнительного разворота по курсу;
- L_2 - путь до достижения точки входа на стандартный маршрут;

- L_3 - путь по стандартному маршруту до его конечной точки.

Алгоритм вычисления каждого из этих значений имеет ряд особенностей и в данной работе не рассматривается[37].

Следующим этапом производится ранжирование воздушных судов по критерию минимума длины пути для определения очерёдности посадки. После сортировки данных о всех ВС в верхней строке таблицы 2 располагается воздушное судно, заходящее на посадку первым.

В блоке №3 (рис. 6.5) уточняется длина пути каждого воздушного судна с учетом безопасной дистанции в горизонтальной и вертикальной плоскости относительно остальных ВС. В том случае, если безопасные интервалы горизонтального и вертикального эшелонирования между какими - либо двумя воздушными судами меньше допустимого, длина пути $ВС_{i+1}$ увеличивается за счет отворота от оптимального курса в сторону от $ВС_i$на угол 30 градусов, после чего производится пересчет с учетом задержки времени разворота.

Влияние условий безопасности воздушного движения на очередность прилёта существенна. Поэтому после каждого пересчета производится повторное ранжирование воздушных судов в очереди прилёта[62].

6.5 Выводы по главе 6

На основании проведенных в данной главе исследований можно сделать следующие выводы:

1. Полученный алгоритм назначения приоритетов позволяет сформировать списки распределения судов по трассам с помощью двух итераций. На первойитерации вычисляются первоначальные приоритеты судов для каждой трассы без учета их близости друг к другу при движении в воздушном эшелоне. На второй итерации полученный состав судов по каждой трассе используется для их «расстановки» в предполагаемых точках трассы, после чего окончательно уточняются их приоритеты.
2. При ограниченном числе судов в каждом эшелоне автоматически определяется состав наименее приоритетных судов, отправляемых в очередь – тромбон. Предложенный подход обеспечит автоматизированную поддержку действиям авиадиспетчерской службе при заходе на посадку.
3. Предложен алгоритм определения очерёдности прилета для случайно расположенных в пространстве воздушных судов с выходом на стандартный маршрут прилета. Расчёт приоритетов ведется исходя из условий встраивания в стандартный маршрут с учетом предыдущего ВС.
4. Планирование очерёдности имеет два этапа – предварительная оценка длины пути до конечной точки маршрута без учёта безопасной дистанции между воздушными судами, и конечное ранжирование с учетом безопасных интервалов между судами.
5. Алгоритм позволяет ещё на начальном этапе движения ВС определить целесообразность его встраивания в прилетной поток с

учетом времени ухода на запасной аэродром. Данная способность особенно актуальна для сбойных ситуаций при управлении полетом.

Глава 7. Оперативный контроль безопасности попутного движения судов в эшелоне

В данной главе решается важная задача автоматического контроля безопасности попутного движения воздушных судов в заданном эшелоне, которая наряду с обычным управлением полетом с постоянной скоростью по заданной линии пути должна формировать сигнал предупредительной тревоги и принять необходимые меры при опасном сближении судов. Такая ситуация, в частности возникает, когда аварийное судно после перелета на заданную линию пути внезапно входит в эшелон на слишком близкой дистанции от других судов, которые уже летят в эшелоне[35, 39, 45].

7.1 Постановка задачи управления попутным движением

На рис 7.1 показана картина движения двух воздушных судов в эшелоне, характеризуемая координатами попутного движения x_1 и l_1, которые определяют главный показатель – дистанцию между судами $l_1 - x_1$.

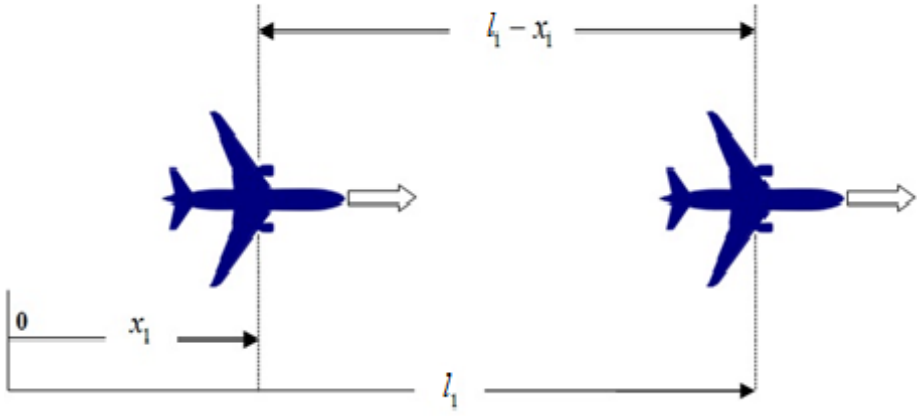


Рис.7.1. Картина сближения воздушных судов при попутном движении

Задача решается при следующей постановке[46]

Дано:

1. Заданы уравнения движения сзадилетящего судна

$$\begin{cases} x_1^{'} = x_2 + w_1 \\ x_2^{'} = -a_1 x_2 + b_1 u_1 \end{cases} \quad (7.1)$$

2. Задано другое впередилетящего судно, двигающееся по закону :

$$\begin{cases} l_1^{'} = l_2 + w_2 \\ l_2^{'} = -a_2 l_2 + b_2 u_2 \end{cases} \quad (7.2)$$

Рассмотрим случай, когда другое впередилетящее судно движется с непредсказуемой постоянной скоростью w_2; также пусть $d_1 = 1, d_2 = 0, l_2^{'} = 0, w_1 = 0$. Тогда решим задачу на основе следующей системы дифференциальных уравнений:

$$\begin{cases} x_1^{'} = x_2 \\ x_2^{'} = -a_1 x_2 + b_1 u_1 \\ l_1^{'} = w_2 \end{cases} \quad (7.3)$$

где x_1 - координата попутного движения судна, x_2 - попутная скорость судна, l_1 - координата попутного движения другого судна, w_2 - скорость движения другого судна.

3. Задан интегральный критерий качества

$$J = \int_0^{t_k} f_0(\overline{x}, u_1, t)dt$$

где, $f_0 = r_0 \frac{u_1^2}{2} + r_1 \frac{1}{2}\left[(l_1 - x_1) - (D + Nw_2)\right]^2 + r_2 \frac{1}{2}(x_2 - w_2)^2 + M_2(x_2 - w_2) - M_1(l_1 - x_1)$ (7.4)

- подынтегральное выражение функционала *J*, учитывающегоштраф r_0 за квадрат управления, т.е. за потраченную мощность при управлениии или энергию, штраф r_1 за приближение к другому судну, штраф r_2 за отклонение скоростей; *D* - заданное безопасное расстояние между управляемым объектом и другим судом при одинаковой скорости движения; $D + Nw_2$ -

минимальная безопасная дистанция между двумя судами при заданном значении коэффициента N; a_1, a_2- динамические коэффициенты объекта управления; M_1- коэффициент, дополнительно учитывающий отклонение траектории движения двух судов и M_2 - коэффициент, дополнительно учитывающий отклонение их скоростей движения.

Требуется решить прямую задачу оптимального управления, т.е. нужно найти функцию управления $u_1 = f(x_1, x_2)$, а затем – искомую функцию риска опасного сближения судов при их оптимальном движении.

7.2 Дополнительное замечание о коэффициентах штрафа интегрального критерия качества попутного движения

В формуле (7.4) подынтегрального выражения f_0присутствует 5 коэффициентов – r_0,r_1,r_2, M_1 и M_2. Обычно один из них (как правило это r_0) принимается за единицу, а другие назначаются относительно этого значения.

Наиболее трудным является вопрос назначения коэффициентов r_1и r_2штраф за отклонения по положению и скорости. Если вообще не учитывать относительную скорость (x_2-w_2) движения судов, то приr_2=M_1=M_2=0и при

$$[(l_1 - x_1) - (D + NW_2)]^2 = D^2 \qquad (7.5)$$

коэффициент r_1 можно трактовать как максимальную стоимость ущербаS_0при столкновении судов на весьма малой скорости, поделенную на D^2.

$$r_1 = \frac{S_0}{D^2}$$

Однако скорость сближения судов имеет немаловажную роль. Из правил дорожного движения наземного транспорта известно, что сама безопасная дистанция D есть функция скорости. Чем выше скорость движения, тем больше требуемая дистанция ($D + NW_2$) между движущимся попутно транспортом. Скорость полета несравненно выше наземной

скорости, и поэтому значимость коэффициента r_2 в авиации должна быть выше при попутном движении. Соотношение $\frac{r_1}{r_2}$ должно быть уточнено в результате моделирования.

Слагаемые в (7.4), имеющие множители M_1 и M_2, имеющие одинаковую роль, организуя следующие пары

$$0.5r_1[(l_1 - x_1) - (D + NW_2)]^2 - M_1(l_1 - x_1);$$
$$0.5r_2(x_1 - W_2)^2 + M_2(x_2 - W_2).$$

Смысл этих пар состоит в том, чтобы устранить одинаковость штрафа при одинаковых по модулю, но имеющих разные знаки отклонений – соответственно $(l_1\text{-}x_1)$ и $(x_2\text{-}W_2)^2$и подчеркнуть пониженную опасность при $l_1 > x_1$ и $x_2 > W_2$. Поэтому вводятся дополнительные слагаемые, содержащие ненулевые значения коэффициентов M_1 и M_2.

Вид штрафных функций в этом случае корректируется, как это показано на рис 7.2.

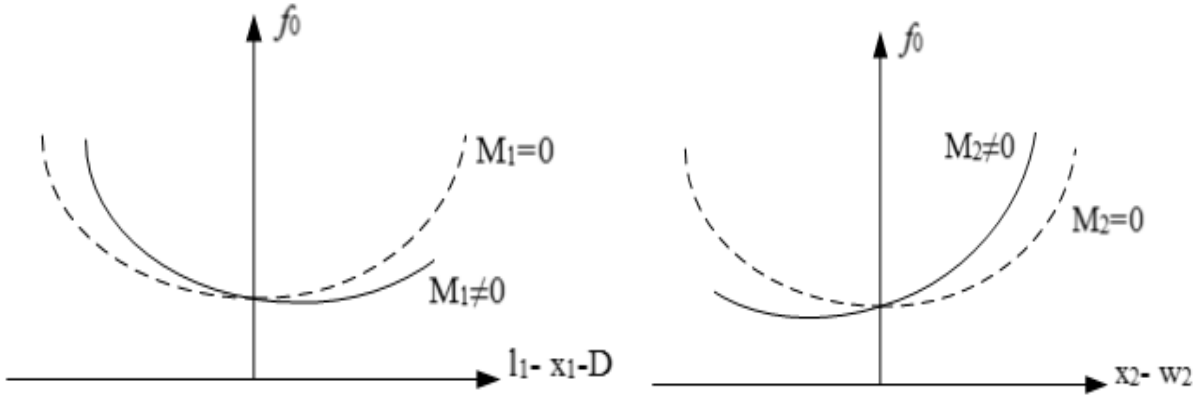


Рис. 7.2 Несимметричные составляющие функции штрафов за отклонения по положению на трассе и попутной скорости судов

7.3 Решение задачи синтеза управления и контроля безопасности попутного движения судов

Решение задачи проведем для двух случаев. В первом простом случае опишем попутное движение судов динамической системой второго порядка при управлении скоростью движения[48]

$$\begin{cases} x_1' = x_2 - \rho x_1^2 \\ x_2' = -a x_2 + u_1 \\ l' = w \end{cases} \tag{7.6}$$

где *W*–скорость движения впередилетящего судна, изменяющаяся непредсказуемы образом; x_1–скорость движения сзадилетящего судна; ρ - коэффициент лобового сопротивления; x_2 – значение тяги двигателя вдоль продольной оси ЛА.

В качестве упрощенного функционала качества возьмем интегральный функционал[49].

$$J = \int_0^{t_k} f_0(\bar{x}, u_1, t)dt \tag{7.7}$$

$$f_0 = r_0 \frac{u_1^2}{2} + r_1 \frac{1}{2}(x_1 - w)^2$$

Соотношения (7.6 – 7.7) указывают, что траекторное движение пока не рассматривается, а в функционале *J*штрафуется только мощность на управления и разность скоростей попутного движения двух судов. Зато учитывается лобовое сопротивление, пропорциональное квадрату скорости, поэтому первое уравнение в (7.6) есть нелинейное дифференциальное уравнение Риккати.

Стремление предельно упростить динамическую модель движения судна и управления им обусловлено тем, что используемый затем метод

АКОР приведет к решению задачи оптимального синтеза в квадратурах только в том случае, если размерность задачи будет невысока. Тогда **окажется возможным достижение** (пусть приближенным способом) **главной цели исследования в этой главе – вычисление в реальном масштабе времени функции риска** при контроле безопасности попутного движения судов.

Далее применим известный подход к решению задачи для условий (7. 6 – 7. 7). Запишем функцию Беллмана ε степенным полиномом второго порядка.

$$\varepsilon = \beta_1 x_1 + \beta_2 x_2 + \gamma_1 \frac{x_1^2}{2} + \gamma_2 \frac{x_2^2}{2} + \psi_{12} x_1 x_2;$$

Затем найдем частные производные

$$\frac{\partial \varepsilon}{\partial x_1} = \beta_1 + \gamma_1 x_1 + \psi_{12} x_2;$$

$$\frac{\partial \varepsilon}{\partial x_2} = \beta_2 + \gamma_2 x_2 + \psi_{12} x_1;$$

и подставим их в исходное уравнение Беллмана(7.6) для данной задачи

$$-\frac{\partial \varepsilon}{\partial t} = r_0 \frac{u_2{}^2}{2} + r_1 \frac{(x_1 - w)^2}{2} + (\beta_1 + \gamma_1 x_1 + \psi_{12} x_2)(x_2 - \rho x_1^2) + (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1)(-a_2 x_2 + u_1) \quad (7.8)$$

Решая это уравнение известным методом АКОР и представляя процесс установившегося движения при $\frac{\partial \varepsilon}{\partial t} = 0$, можно получить приближенное значение сигнала оптимального управления u_1.

$$u_1 \approx \frac{r_1 W}{\psi_{12}} - \psi_{12} x_1 - \frac{\psi_{12}}{a} x_2 \quad (7.9)$$

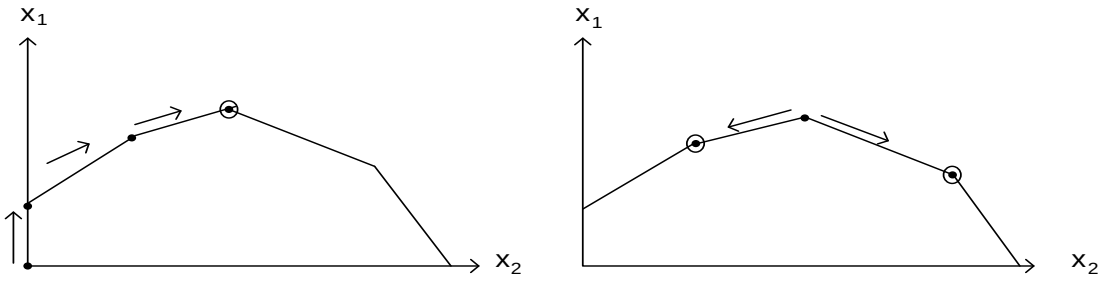
где, $\psi_{12} = \sqrt{r_1}$ при $r_0 = 1$

Моделирование описанной выше простой системы показало, что в случае незначительных изменений скорости полета на $\pm 20\%$ квадратическую зависимость лобового сопротивления от скорости можно не учитывать, линеаризуя динамику полета при рассмотрении второго более сложного случая.

Перейдем теперь к уравнениям (7.1 – 7.2) траекторного движения судов, описываемого с помощью трех переменных.

Тогда решим поставленную задачу с помощью динамического программирования[1], совершив следующие действия.

1. Функция Беллмана ε и ее производные записываются таким образом :



(7.10)

2. Запишем уравнение Беллмана и представим в нем функцию ε степенным полиномом:

$$-\frac{\partial \varepsilon}{\partial t} = \min_{u}\left\{ f_0 + \sum \frac{\partial \varepsilon}{\partial x_i} x_i^{'} \right\};$$

$$
\begin{aligned}
&-\frac{\partial \varepsilon}{\partial t} = r_0 \frac{u_2^{\,2}}{2} + r_1 \frac{\left[(l_1 - x_1) - (D + Nw_2)\right]^2}{2} + r_2 \frac{(x_2 - w_2)^2}{2} - M_1(l_1 - x_1) + \\
&+M_2(x_2 - w_2) + (\beta_1 + \gamma_1 x_1 + \psi_{12} x_2 + \psi_{13} l_1) x_1^{'} + (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1) x_2^{'} + \\
&+(\beta_3 + \gamma_3 l_1 + \psi_{13} x_1 + \psi_{23} x_2) l_1^{'}; \\
&-\frac{\partial \varepsilon}{\partial t} = r_0 \frac{u_2^{\,2}}{2} + r_1 \frac{\left[(l_1 - x_1) - (D + Nw_2)\right]^2}{2} + r_2 \frac{(x_2 - w_2)^2}{2} - M_1(l_1 - x_1) + \\
&+M_2(x_2 - w_2) + (\beta_1 + \gamma_1 x_1 + \psi_{12} x_2 + \psi_{13} l_1) x_2 + (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)(-a_2 x_2 + b_2 u_2) + \\
&+(\beta_3 + \gamma_3 l_1 + \psi_{13} x_1 + \psi_{23} x_2) w_2 .
\end{aligned}
\qquad (7.11)
$$

3. Оптимизируя функцию Беллмана по параметру u_2 , получаем таким образом:

$$f(u_1) = r_0 \frac{u_1^2}{2} + (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1) b_1 u_1 \tag{7.12}$$

$$f'(u_1) = r_0 u_1 + (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1) b_1 = 0;$$
$$\Rightarrow u_{1опт.} = -\frac{b_1}{r_0}(\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1) \tag{7.13}$$

4. Подставим $u_{2опт.}$ (7.13) в выражение (7.12) и получим :

$$f(u_{1опт.}) = r_0 \frac{b_1^2}{2r_0^2}(\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)^2 - (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)^2 \frac{b_1^2}{r_0};$$
$$f(u_{1опт.}) = -\frac{b_1^2}{2r_0}(\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)^2; \tag{7.14}$$

5. Подставив функцию $f(u_{1опт.})$ (7.14) в уравнение Беллмана (7.11) и представив правую часть уравнения Беллмана степенным рядом, получаем :

$$-\frac{\partial \varepsilon}{\partial t} = \frac{r_1}{2}[(l_1 - x_1) - (D + N w_2)]^2 + \frac{r_2}{2}(x_2 - w_2)^2 - M_1(l_1 - x_1) + M_2(x_2 - w_2) +$$
$$+(\beta_1 + \gamma_1 x_1 + \psi_{12} x_2 + \psi_{13} l_1) x_2 - (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1) a_1 x_2 +$$
$$+(\beta_3 + \gamma_3 l_1 + \psi_{13} x_1 + \psi_{23} x_2) w_2 - \frac{b_1^2}{2r_0}(\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)^2;$$

$$-\frac{\partial \varepsilon}{\partial t} = \left[r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2 - \frac{b_2^2}{r_0} \psi_{12} \beta_2 \right] x_1 + (-r_2 w_2 + M_2 + \beta_1 - \beta_2 a_1 + \psi_{23} w_2 - \frac{b_1^2}{r_0} \beta_2 \gamma_2) x_2 +$$
$$+(-r_1 D - r_1 N w_2 - M_1 + \gamma_3 w_2 - \frac{b_1^2}{r_0} \beta_2 \psi_{23}) l_1 + (r_1 - \frac{b_1^2}{r_0} \psi_{12}^2) \frac{x_1^2}{2} + (r_2 + 2\psi_{12} - 2\gamma_2 a_2 - \frac{b_1^2}{r_0} \gamma_2^2) \frac{x_2^2}{2} +$$
$$+(r_1 - \frac{b_1^2}{r_0} \psi_{23}^2) \frac{l_1^2}{2} + (\gamma_1 - a_1 \psi_{12} - \frac{b_1^2}{r_0} \gamma_1 \psi_{12}) x_1 x_2 + (-r_1 - \frac{b_1^2}{r_0} \psi_{12} \psi_{23}) x_1 l_1 + \tag{7.15}$$
$$+(\psi_{13} - \psi_{23} a_1 - \frac{b_1^2}{r_0} \psi_{23} \gamma_2) x_2 l_1 + [\frac{r_1}{2}(D + N w_2)^2 + \frac{r_2}{2} w_2^2 + M_2 w_2 + \beta_3 w_2 - \frac{b_1^2}{r_0} \beta_2^2]$$

6. Приравнивая сомножители при одинаковых степенях и группируя их по степеням, получим систему дифференциальных уравнений [48]

$$\dot{\beta}_1 x_1 = \left[r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2 - \frac{b_2^2}{r_0} \psi_{12} \beta_2 \right] x_1;$$

$$\dot{\beta}_2 x_2 = (-r_2 w_2 + M_2 + \beta_1 - \beta_2 a_2 + \psi_{23} w_2 - \frac{b_2^2}{r_0} \beta_2 \gamma_2) x_2;$$

$$\dot{\beta}_3 l_1 = (-r_1 D - r_1 N w_2 - M_1 + \gamma_3 w_2 - \frac{b_1^2}{r_0} \beta_2 \psi_{23}) l_1;$$

$$\dot{\gamma}_1 \frac{x_1^2}{2} = (r_1 - \frac{b_1^2}{r_0} \psi_{12}{}^2) \frac{x_1{}^2}{2}; \qquad (7.16)$$

$$\dot{\gamma}_2 \frac{x_2^2}{2} = (r_2 + 2\psi_{12} - 2\gamma_2 a_1 - \frac{b_1^2}{r_0} \gamma_2{}^2) \frac{x_2^2}{2};$$

$$\dot{\gamma}_3 \frac{l_1^2}{2} = (r_1 - \frac{b_1^2}{r_0} \psi_{23}{}^2) \frac{l_1^2}{2};$$

$$\dot{\psi}_{12} x_1 x_2 = (\gamma_1 - \frac{b_1{}^2}{r_0} \gamma_2 \psi_{12} + \gamma_1 d - a_1 \psi_{12}) x_1 x_2;$$

$$\dot{\psi}_{13} x_1 l_1 = (-r_1 - \frac{b_1^2}{r_0} \psi_{12} \psi_{23}) x_1 l_1;$$

$$\dot{\psi}_{23} x_2 l_1 = (\psi_{13} - \psi_{23} a_2 - \frac{b_1^2}{r_0} \psi_{23} \gamma_2) x_2 l_1;$$

7. Заменим дифференциальные уравнения алгебраическими при: $-\frac{\partial \varepsilon}{\partial t} = 0$

$$r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2 - \frac{b_1^2}{r_0} \psi_{12} \beta_2 = 0;$$

$$-r_2 w_2 + M_2 + \beta_1 - \beta_2 a_1 + \psi_{23} w_2 - \frac{b_1^2}{r_0} \beta_2 \gamma_2 = 0;$$

$$-r_1 D - r_1 N w_2 - M_1 + \gamma_3 w_2 - \frac{b_1^2}{r_0} \beta_2 \psi_{23} = 0;$$

$$(r_1 - \frac{b_1^2}{r_0} \psi_{12}{}^2) = 0;$$

$$r_2 + 2\psi_{12} - 2\gamma_2 a_2 - \frac{b_1^2}{r_0} \gamma_2{}^2 = 0;$$

$$r_1 - \frac{b_1^2}{r_0} \psi_{23}{}^2 = 0;$$

$$\begin{aligned}
&\gamma_1 - \frac{b_1^2}{r_0}\gamma_2\psi_{12} + \gamma_1 d - a_1\psi_{12} = 0;\\
&-r_1 - \frac{b_1^2}{r_0}\psi_{12}\psi_{23} = 0;\\
&\psi_{13} - \psi_{23}a_1 - \frac{b_1^2}{r_0}\psi_{23}\gamma_2 = 0;
\end{aligned} \qquad (7.17)$$

8.После преобразования всех уравнений (7.17), их взаимной замены и упрощения 5-го уравнения этой системы, если пренебрежем составным элементом $-\frac{b_2^2}{r_0}\gamma_2^2 \approx 0$, окончательно получим нижеследующее решение :

$$\begin{cases}
\beta_1 = r_2 w_2 + (a_1 + \frac{b_1^2}{r_0})\beta_2 - M_2 + w_2\psi_{23};\\
\beta_2 = \frac{r_0(r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2}{b_1^2\psi_{12}};\\
\gamma_1 = \psi_{12}(a_1 + \frac{b_1^2}{r_0}\gamma_2);\\
\gamma_2 = \frac{r_2 + 2\psi_{12}}{2a_2};\\
\gamma_3 = \frac{(\frac{b_1^2}{r_0}\beta_2\psi_{23} + r_1 D + r_1 N w_2 + M_1)}{w_2};\\
\psi_{12} = \sqrt{r_1 r_0}\,/\,b_1;\\
\psi_{13} = \psi_{23}(a_1 + \frac{b_1^2}{r_0}\gamma_2);\\
\psi_{23} = -\psi_{12};
\end{cases} \qquad (7.18)$$

9. Подставим четыре составляющих $\beta_2, \gamma_2, \psi_{12}, \psi_{23}$ решения (7.18) в выражение $u_{1опт.} = -\frac{b_1}{r_0}(\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)$ и получим:

$$u_1 = -\frac{b_1}{r_0}\left\{\frac{r_0(r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2}{b_2^2\psi_{12}} + \frac{r_2 + 2\psi_{12}}{2a_2}x_2 + \sqrt{r_1 r_0}\,/\,b_1 x_1 - \psi_{12} l_1\right\} \qquad (7.19)$$

10. Подставив полученную функцию u_1 в выражение (7.1), получим окончательный результат синтеза

$$\begin{cases} x_1^{'} = x_2 \\ x_2^{'} = -a_1 x_2 + b_2 u_2 = \\ l_1^{'} = w_2 \end{cases} \begin{cases} x_1^{'} = x_2 \\ x_2^{'} = -a_1 x_2 - \dfrac{b_1^{\,2}}{r_0}\left\{ \dfrac{r_0(r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2}{b_1^{\,2}\psi_{12}} + \dfrac{r_2 + 2\psi_{12}}{2a_1} x_2 + \sqrt{r_1 r_0}\,/\, b_1 x_1 - \psi_{12} l_1 \right\} \\ l_1^{'} = w_2 \end{cases}$$

$$= \begin{cases} x_1^{'} = x_2 \\ x_2^{'} = -(a_1 + \dfrac{b_2^{\,2}}{r_0}\dfrac{r_2 + 2\psi_{12}}{2a_1}) x_2 - \dfrac{b_2^{\,2}}{r_0}\sqrt{r_1 r_0}\,/\, b_1 x_1 - \dfrac{b_1^{\,2}}{r_0}\psi_{12} l_1 - \dfrac{b_1^{\,2}}{r_0}\dfrac{r_0(r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2}{b_1^{\,2}\psi_{12}} \\ l_1^{'} = w_2 \end{cases} \qquad (7.20)$$

7.4 Результаты моделирования попутного движения

Моделирование системы управления попутным движением проводилось при следующих условиях[31]:

$$D = 6000m, a_2 = b_2 = 2, C_1 = 0$$

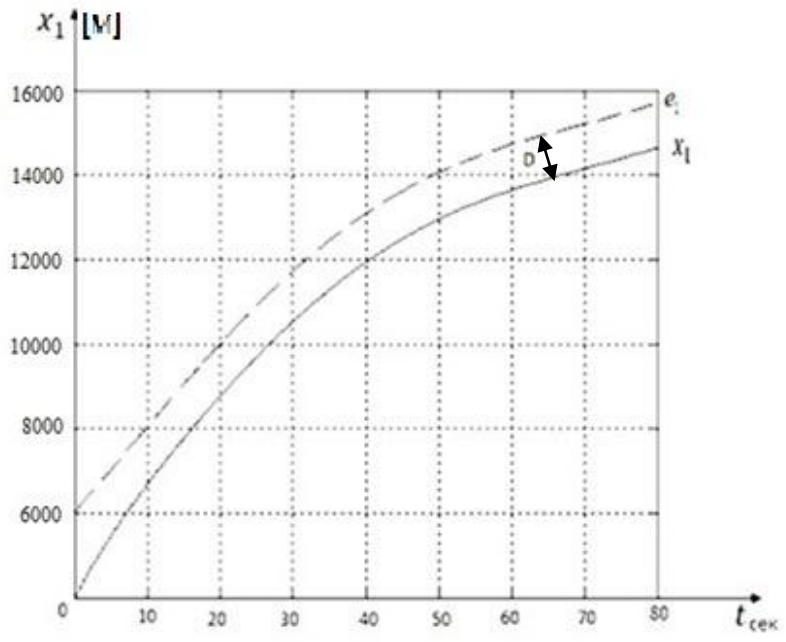


Рис 7.3. Результаты моделирования при попутном движении двух воздушных судов; - - - впередилетящее судно; ———сзадилетящее основное судно.

Результаты моделирования при попутном движении двух воздушных судов в случае равномерного замедления скорости полета впередилетящего судна показаны на рис 7.3.

Видно, что оптимальное управление обеспечивает сохранение безопасной дистанции *D* между судами. При этом замедление скорости впередилетящего судна происходило с продольным ускорением примерно $2\,{}^{м}\!/_{сек^2}$ при исходом начальном расстоянии $l_{10} - x_{10} = 6000\ м$, что соответствует безопасной стартовой позиции*D*летящих судов.

После того, как было найдено оптимальное управление, решим задачу контроля безопасности движения, вычислив специальную функцию риска.

Длянахождения функции риска опасного сближения судов воспользуемся правой частью уравнения Беллмана (7.11) . Тогда подставим $u_{1опт.}$ (7.13) в выражение (7.11) и получим :

$$f(u_{1опт.}) = r_0 \frac{b_1^{\,2}}{2r_0^{\,2}}(\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)^2 - (\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)^2 \frac{b_1^{\,2}}{r_0};$$
$$f(u_{1опт.}) = -\frac{b_1^{\,2}}{2r_0}(\beta_2 + \gamma_2 x_2 + \psi_{12} x_1 + \psi_{23} l_1)^2; \qquad (7.21)$$

Подставив функцию $f(u_{1опт.})$ (7. 21) в уравнение Беллмана (7.11) и представив правую часть уравнения Беллмана степенным рядом, получаем :

$$-\frac{\partial \varepsilon}{\partial t} = F = \left[r_1 D + r_1 N w_2 + M_1 + \psi_{13} w_2 - \frac{b_2^2}{r_0}\psi_{12}\beta_2 \right] x_1 + (-r_2 w_2 + M_2 + \beta_1 - \beta_2 a_1 + \psi_{23} w_2 - \frac{b_1^2}{r_0}\beta_2\gamma_2) x_2 +$$
$$+(-r_1 D - r_1 N w_2 - M_1 + \gamma_3 w_2 - \frac{b_1^2}{r_0}\beta_2\psi_{23}) l_1 + (r_1 - \frac{b_1^2}{r_0}\psi_{12}^{\,2})\frac{x_1^2}{2} + (r_2 + 2\psi_{12} - 2\gamma_2 a_2 - \frac{b_1^2}{r_0}\gamma_2^{\,2})\frac{x_2^2}{2} +$$
$$+(r_1 - \frac{b_1^2}{r_0}\psi_{23}^{\,2})\frac{l_1^2}{2} + (\gamma_1 - a_1\psi_{12} - \frac{b_1^2}{r_0}\gamma_1\psi_{12}) x_1 x_2 + (-r_1 - \frac{b_1^2}{r_0}\psi_{12}\psi_{23}) x_1 l_1 + \qquad (7.22)$$
$$+(\psi_{13} - \psi_{23} a_1 - \frac{b_1^2}{r_0}\psi_{23}\gamma_2) x_2 l_1 + [\frac{r_1}{2}(D + N w_2)^2 + \frac{r_2}{2} w_2^2 + M_2 w_2 + \beta_3 w_2 - \frac{b_1^2}{r_0}\beta_2^2]$$

Правая частьFуравнения (7.22)есть степенной полином второго порядка и по определению метода динамического программирования является функцией риска. Если она превышает заданный порогF_0, то формируется сигнал тревоги, требующий увеличения дистанции между судами. В упрощенном виде эта формула может быть представлена следующим образом, удобным для технической реализации на борту при $r_2 >> r_1$[50].

$$F = \frac{r_1(x_1 + D - l_1)^2}{2} + r_2 \frac{(x_2 - l_2)^2}{2} - \frac{u_1^2}{2} \qquad (7.23)$$

Нужно обратить внимание, что передаточные числа оптимального регулятора для сигнала управления уточнялись при моделировании экспериментально, и на их основании оказались известными при скорости полета $100 \, м/сек$ коэффициенты штрафа, равные r_1=1 , r_2=16.**Таким образом, штраф за несоблюдение скорости имеет превалирующее значение по сравнению со штрафом за несоблюдение дистанции.** По сравнению с известными в [38, 50] результатами функции риска для поперечного и встречного движения. В последних важнее является коэффициент штрафаr_1.

Совместное использование результатов контроля безопасности и управления полетом можно реализовать в виде двухуровневой структуры, показанной на рис 7.4.

Согласно этой структуре, на верхнем уровне системы управления сзадилетящего судна функционирует блок оперативного контроля безопасности попутного движения, в которой поступают сигналы измерения координат движения двух судов по положению и скорости – x_1, x_2, l_1, l_2, при этом вопрос информационной точности измерений в работе не рассматривается[44]. Блок вычисления функции риска в реальном масштабе времени формирует сигнал F, сравниваемый с порогом. В случае его превышения начинается координированное управление скоростью попутного движения впередилетящего судна. В противном случае реакция на полет

впередилетящего судна отсутствует. и сзадилетящее судно летит с постоянной скоростью без перегрузок. Совместный контроль и управление полетом повышает безопасность полета и может быть использован в первую очередь на борту воздушных судов.

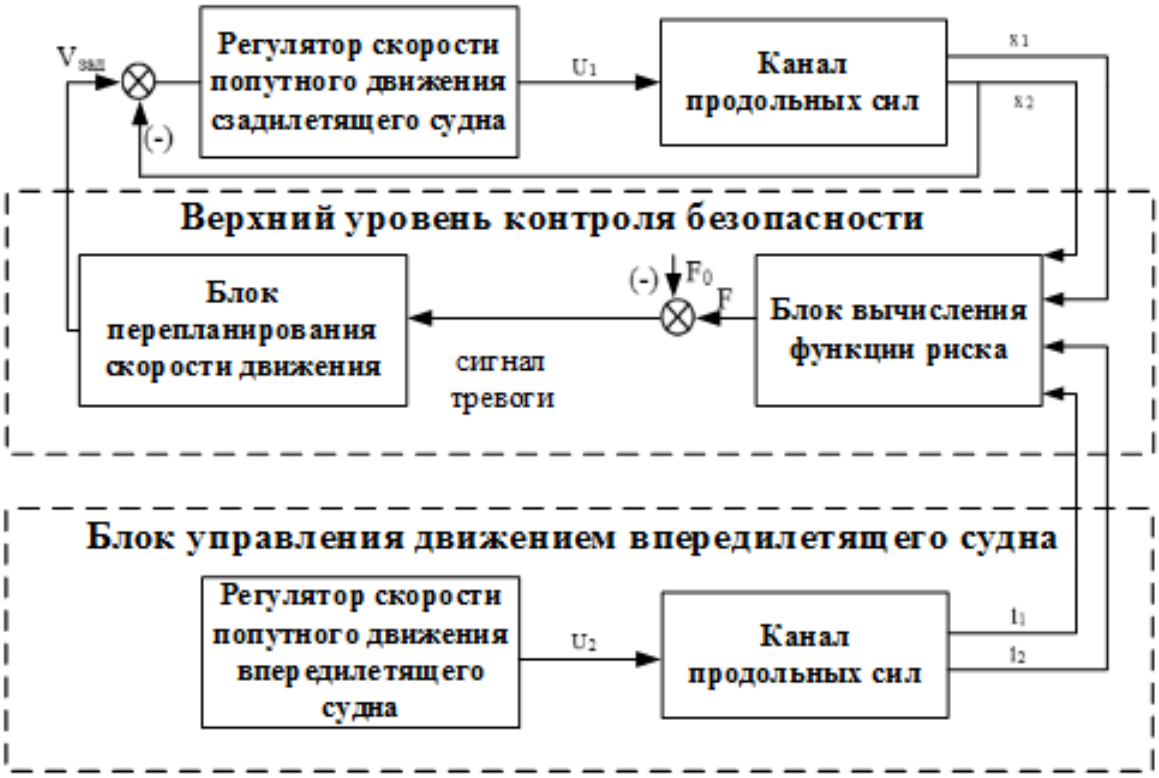


Рис 7.4. Двухуровневая структура принятия решений при контроле безопасности и управлении скоростью попутного движения воздушных судов

При отсутствии управления попутным движением сзадилетящего судна, т.е при u_1=0, расстояние между судами начинает постепенно сокращаться,как показано в таблице 7.1, и к 15 секунде отклонение Δx от заданногодостигает значения, на 503 метров меньшего, чем необходимо для безопасного полета.

Таблица 7.1. данных попутного движения судов без координированного управления при u_1=0.

t	0	10	20	30
Δx_1	0	7	226	503
F	0	1000	3×10^4	1.2×10^5

Соответственно функция риска F, начинает резко расти, и в целом его опасное значение превышаем 10^5.

При координации движения судов, т.е при $u_1 \neq 0$, отклонение Δx_1 траекторий невелико и показано в таблице 7.2., а функция риска возрастает незначительно до уровня 1200. Поэтому в принципе можно установить порог F_0=200.

Таблица 7.2. данных попутного движения судов при координированном управлении

t	0	10	20	30
Δx_1	0	6	26	30
F	0	200	1100	1200

Также были рассмотрен случай, когда начальная дистанция Δx_1 между судами неравна заданному значению D=6000м. На рис 7.5 показаны траектории попутного движения, когда в начале суда находятся на расстоянии, больше, чем дистанция D.

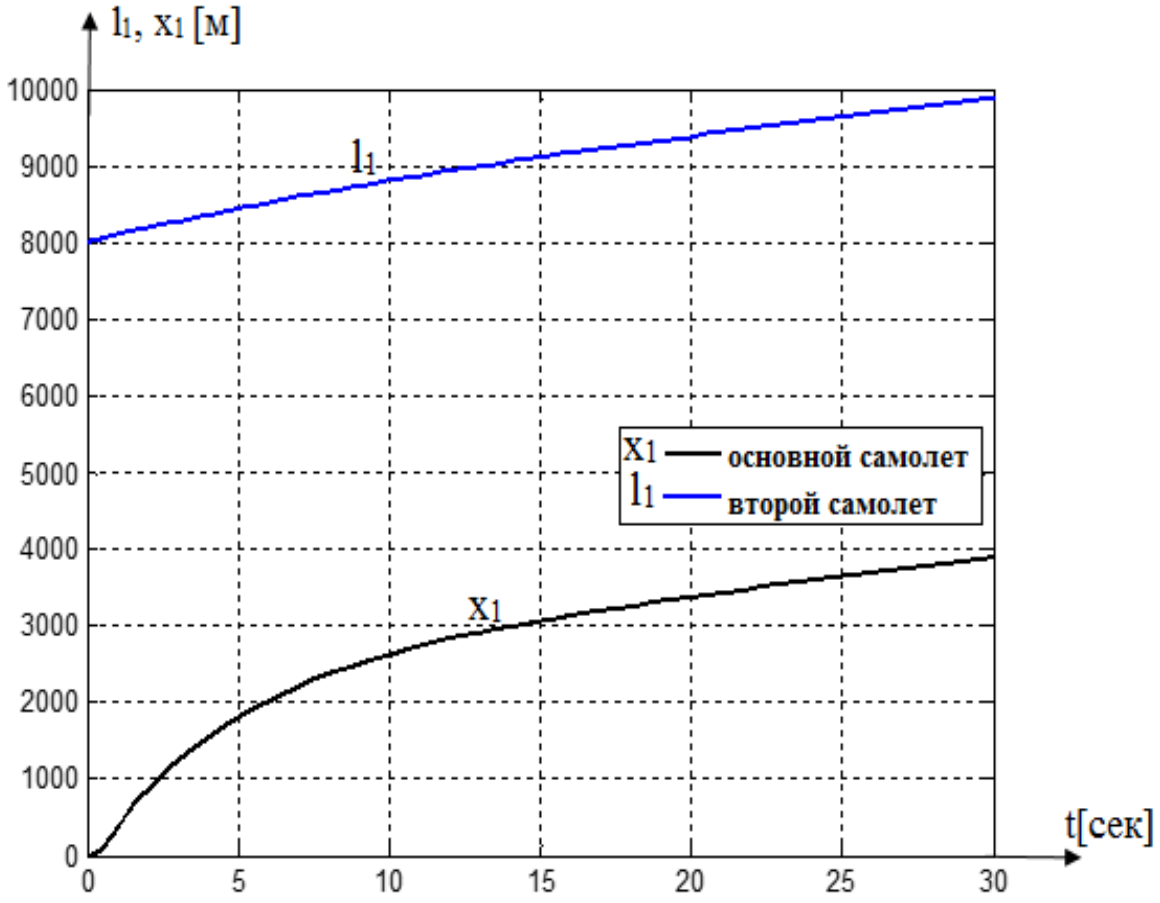


Рис. 7.5. Траектории движения судов при начальном расстоянии $\Delta x_1 = 8000$ м между ними, превышающем безопасную дистанцию *D*=6000м

Видно, что дистанция между судами постепенно приходит в норму, При *t*=30 имеем $\Delta x_1 = 500$ т.е постепенно $(l_1 - x_1) \to D$.

Следующим является случай, когда в стартовой позиции дистанция $(l_1 - x_1)$ между судами меньше заданного значения*D*. Обычно это происходит, когда аварийный самолет вне очереди входит в эшелон, если есть окно. Траектории движения судов показаны на рис 7.6. Видно, что постепенно дистанция между ними приходит в норму. Значит, с помощью предложенногосовместного контроля безопасности и управления попутного движения судов можно избежать угрозы их столкновения.

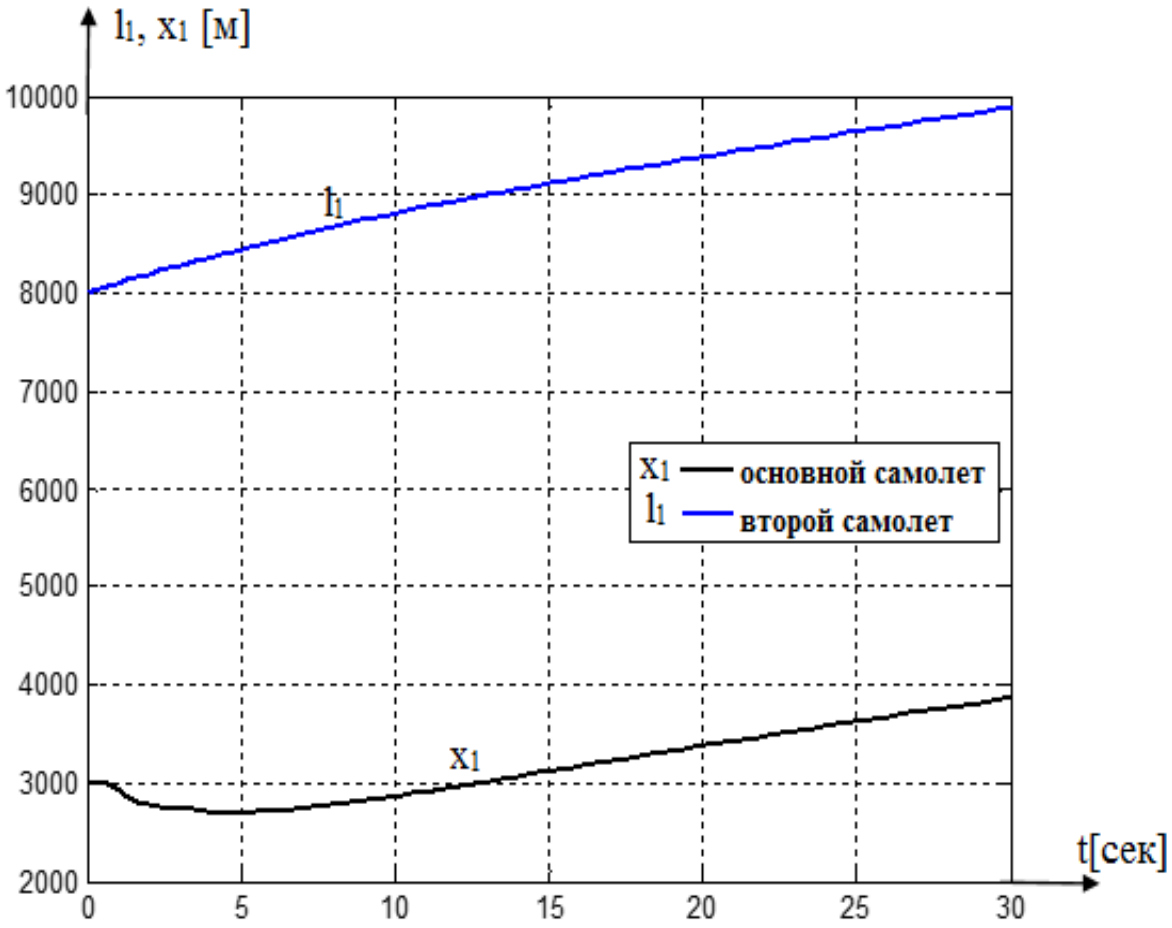


Рис. 7.6. Траектории движения судов при начальном расстоянии $\Delta x_1 = 5000$ *м* между ними, меньшем безопасной дистанции*D*=6000м

7.5 Выводы по главе 7

На основании проведенных в данной главе исследований можно сделать следующие выводы :

1. Результаты синтеза оптимального управления полетом сзадилетящего судна и моделирования попутного движения двух судов в эшелоне показали, что найденный алгоритм линейного управления скоростью полета успешно справляется с задачей обеспечения безопасности полета в целом.
2. С помощью правой части уравнения Беллмана найдено упрощенное алгебраическое выражение функции риска опасного сближения судов, позволяющее в реальном масштабе времени осуществлять контроль безопасности полета и сформировать в необходимых случаях сигнал тревоги для принятия необходимых предупредительных мер.
3. Показано, что при оценке риска важность отклонения по скорости движения судов имеет для попутного движения превалирующий характер по сравнению с важностью отклонения между имеющимся расстояниемот заданной безопасной дистанции междусудами.
4. Полученные результаты синтеза имеют сугубо приближенный характер, динамическая модель движения должна быть усложнена, а сам закон управления скоростью должен быть уточнен. Однако найденная аналитическая форма оценка риска оправдывает получение нужных формул вычисления в квадратурах коэффициентов β_i, r_i, ψ_{ik} функции Беллмана, что позволило в целом добиться возможности в реальном масштабе времени контролировать безопасность попутного движения судов. Найденную формулу (7. 23) функции риска нужно проверить при моделировании более «реального» объекта, чему посвящена глава 9.

5. Замечание о назначении коэффициентов штрафов r_i, M_i в критерии оптимальности. В данной работе был использован следующий подход. Вначале при r_0=1 все остальные коэффициенты были взяты одинаковыми и не меньше, чем единица, например

$$r_i = M_i = 4$$

Затем с их помощью при синтезе оптимального управления методом АКОР были определены передаточные числа линейного регулятора и проведено первоначальное моделирование на ЭВМ попутного движения. Анализируя качество переходных процессов и допустимость по амплитуде полученного управления, передаточные числа экспериментальным путем корректировались. Наконец, решая обратную задачу определения коэффициентов штрафов по новым передаточным числам, удалось установить, что коэффициент r_2штрафа за несоблюдение скорости значительно выше, чем r_1. Поэтому в итоге получены следующие оценки.

$$r_0 = 1; r_1 = 1; M_1 = 1; r_2 = 16; M_2 = 16$$

Глава 8. Исследование системы массового обслуживания пассажиров в аэропорту после прилета

Данная глава посвящена специфической задаче расчета вероятностных характеристик обслуживания пассажиров после прилета с позиций теории массового обслуживания. При этом возможна ситуация обычного бесприоритетного обслуживания пассажиров и наоборот, когда часть пассажиров также нуждается во внеочередном обслуживании. В обеих случаях вероятность отказа в приоритетном обслуживании как в воздухе, так и на земле должна быть сведена к минимуму.

В данной работе предложена новая методика расчета **приоритетной** системы массового обслуживания (СМО), которая позволяет определить полную группу вероятностных состояний многоканальной СМО с ожиданием в очереди.

8.1 Постановка задачи

Ниже рассмотрена задача бесприоритетного и приоритетного обслуживания пассажиров в аэропорту, как это показано на рис 8.1, при следующей постановке[58].

Дано:

1. Под одной заявкой понимается группа пассажиров, прилетевшая на одном самолете.
2. Случайный поток прилетающих самолетов, как и их обслуживание в аэропорту, подчиняется закону Пуассона. Интенсивность прилета аварийных самолетов равна λ_1, а обычных самолетов - λ_2, при условии, что $\lambda_2 >> \lambda_1$
3. Важные заявки поступают вне очереди в первый освободившийся канал

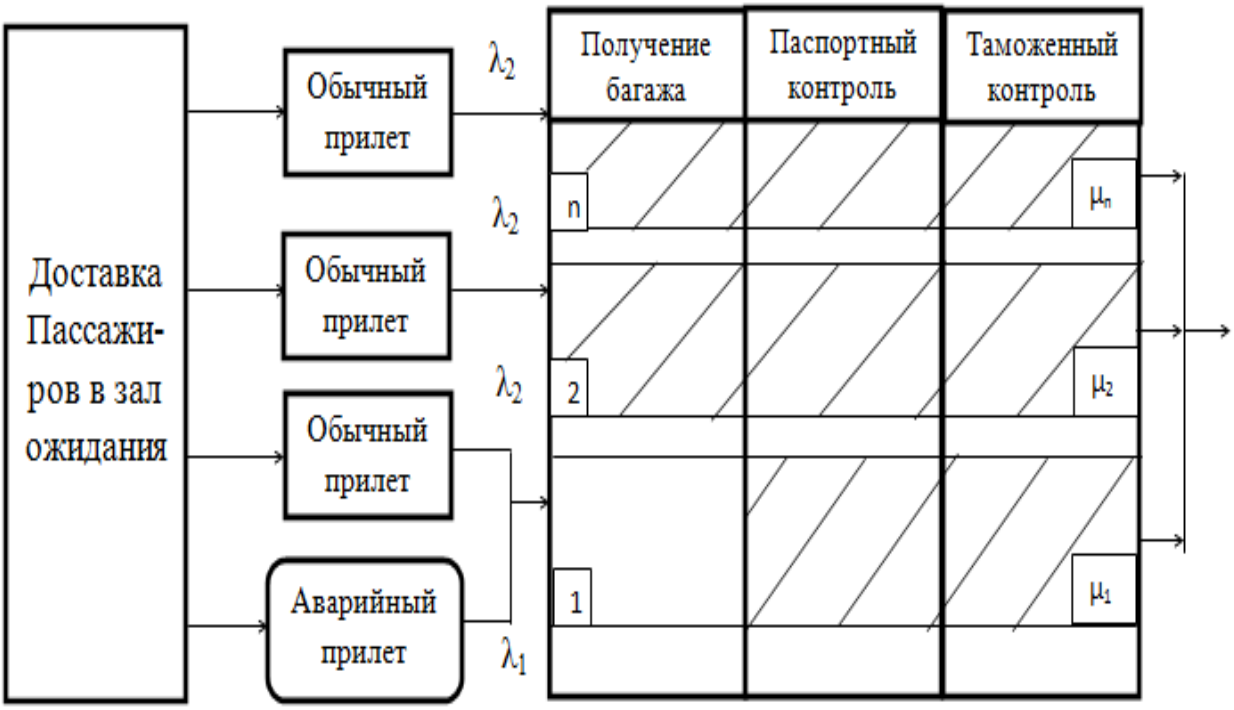


Рис.8.1. Схема многоканальной системы обслуживания пассажиров в аэропорту

4. В системе (либо в канале, либо в очереди) одновременно может быть только одна приоритетная заявка после аварийного прилета
5. Длина очереди *l* принята равной числу каналов *n*, что соответствует площади общего зала ожидания пассажиров, пропорциональной числу каналов обслуживания.

Требуется:

1. Рассчитать вероятностные состояния приоритетной СМО, в первую очередь вероятность отказа в обслуживании обычных и приоритетных заявок
2. Сравнить полученный результат с известным бесприоритетным обслуживанием и выбрать число каналов *n*, обеспечивающее требуемую вероятность отказа в обслуживании.

8.2 Расчет вероятностного состояния системы бесприоритетногообслуживания пассажиров,попавших в очередь

Вначале разберем многоканальную систему бесприоритетного обслуживания с очередью, также число мест в очереди примем равным максимальным числу каналов обслуживания. Это число обозначим через *n*.

Пусть на вход многоканальной системы массового обслуживания (СМО) поступает поток неприоритетных заявок с интенсивностью λ_2 ; интенсивность обслуживания (для одного канала) μ; число мест *n* ;

Состояние системы будем нумеровать по числу заявок, связанных с системой:

X_0 – оба каналы свободны, система свободна от заявок;

$X_{1,0}$ - в системе обслуживается одна заявка;

$X_{2,0}$ - в системе обслуживается 2 заявки;

.
.
.

$X_{n,0}$ – в системе обслуживается *n* заявок и все *n* каналов заняты;

$X_{n,1}$ – заняты все *n* каналов и одна заявка стоит в очереди;

$X_{n,2}$ – заняты все *n* каналов, и 2 заявки стоят в очереди;

.
.
.

$X_{n,n}$- заняты все *n* каналов, и *n* заявок стоят в очереди, т.е. и в очереди мест нет.

Поток заявок производится с интенсивностью λ_2 , а поток обслуживания с интенсивностью, равной μ, умноженной на число занятых каналов:

Напишем выражения для предельных вероятностей состояний, сразу же обозначая $\lambda_2/\mu = \rho_2$:

$$\left.\begin{aligned} P_1 &= \frac{\rho_2}{1!} P_0, \\ P_2 &= \frac{\rho_2^{\,2}}{2!} P_0, \\ P_{n+1} &= \frac{\rho_2^{\,n}}{n * n!} P_0, \\ P_{n+2} &= \frac{\rho_2^{\,n+2}}{n^2 * n!} P_0, \\ P_{n+n} &= \frac{\rho_2^{\,n+n}}{n^n * n!} P_0, \end{aligned}\right\} \qquad (8.1)$$

где вероятность простоя P_0 с учетом формулы для геометрической прогрессии равна

$$\begin{aligned} P_0 &= [1+\frac{\rho_2}{1!}+\frac{\rho_2^{\,2}}{2!}+\frac{\rho_2^{\,3}}{3!}+\frac{\rho_2^{\,n}}{n!}\left\{1+\frac{\rho_2}{n}+(\frac{\rho_2}{n})^2+...+(\frac{\rho_2}{n})^n\right\}]^{-1}, \\ P_0 &= [\sum_{i=0}^{n-1}\frac{\rho_2^i}{i!}+\frac{\rho_2^n}{n!}\frac{\left\{1-(\frac{\rho_2}{n})^{n+1}\right\}}{(1-\frac{\rho_2}{n})}]^{-1}. \end{aligned} \qquad (8.2)$$

Подставляя P_0 в формулу (8.1), можно получить остальные вероятностные состояния многоканальной СМО с ожиданием, в том числе интересующую нас вероятность отказа.

$$P_i = \frac{\rho^i P_0}{i!}; \quad i = 1,..n \qquad (8.3)$$

С ожиданием в очереди:

$$\begin{aligned} P_{n+\omega} &= \frac{\rho^\omega P_n}{\prod_{m=1}^{\omega}(n+m\beta)}; \quad \omega = 1,..S \\ P_0 &+ \sum_{i=1}^{n} P_i + \sum_{\omega=1}^{S} P_{n+\omega} = 1 \end{aligned} \qquad (8.4)$$

Формулы (8.3) и (8.4) для бесприоритетного обслуживания известны в виде общих **формул Эрланга, но они показаны для методического анализа и получения новых формул,** приведенных ниже для приоритетного обслуживания и сравнения их друг с другом.

8.3 Расчетвероятностного состояния системыприоритетного обслуживания.

При нахождении формул вероятностного состояния СМО будем исходить из следующих положений и допущений[59].

Проведём расчет вероятностных состояний приоритетной СМО при следующих допущениях и обозначениях:

1. Пусть есть два потока заявок с интенсивностью λ_1 – приоритетных, $\lambda_2 >> \lambda_1$ – обычных.
2. Пусть максимальная длина очереди S равна числу каналов n.
3. Пусть вместо привычных вероятностей P_0 , P_i, (i=1,…,n); P_{n+w}(w=1,…,S) имеютсяследующие вероятности:

P_0 – вероятность того, что в системе нет никаких заявок, т.е. она свободна.

Z_i – вероятность того, что i-каналов занято обслуживанием обычных заявок, приоритетных заявок нет, а заявки в очереди отсутствуют(i=1,…,n).

ζ_i – вероятность того, что в одном из каналов обслуживается приоритетная заявка, в остальных (i-1) канала – обычные заявки, а заявки в очереди отсутствуют (i=1,…,n).

Z_{n+l} – вероятность того, что все каналы заняты обычными заявками, и в очереди есть также обычные заявки (l=1,…,S илиl=1,…,n).

ζ_{n+l} - вероятность того, что в системе либо в канале, либо короткое время – в очереди длинойlимеется одна приоритетная заявка (l=1,…,n).

Z_{n+n} - вероятность того, что все каналы и места в очереди заняты обычными заявками, что соответствует вероятности отказа в бесприоритетном обслуживании.

ζ_{n+n}– вероятность отказа в обслуживании приоритетных заявок, когда все каналы заняты, причем в систему уже попала одна приоритетная заявка.

4. В системе может одновременно находиться только одна приоритетная заявка.
5. Пусть $\nu = \mu$, т.е. β=1.
6. Полной группе событий соответствует условие

$$P_0 + \sum_{i=1}^{n} (\mathrm{Z_i} + \zeta_i) + \sum_{e=1}^{n} (\mathrm{Z}_{n+e} + \zeta_{n+e}) = 1$$

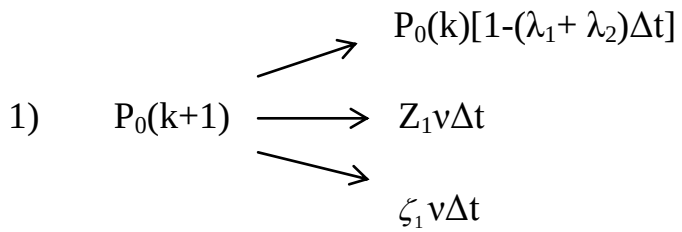
7. Используя общеизвестную методику расчета, представим условия перехода из одного состояния в другое показанными ниже схемами из группы стрелок для различных значений n, после чего можно получить формулы переходных вероятностей путем составления разностных, затем дифференциальных уравнений перехода из одного состояния в соседние.

Сначала рассмотрим случай *n*=1, для которого можно представить следующие действия. Приступим к составлению уравнений вероятностного перехода с одного состояния в другое. Очевидно, что эти состояния должны быть соседними, т.е. они отличаются на одну заявку, больше или меньше (в канале или в очереди).

Полной группе событий соответствует условие :

$$P_0 + Z_1 + \zeta_1 + Z_{1+1} + \zeta_{1+1} = 1 \qquad (8.5)$$

Расчет вероятности начнем с оценки вероятности P_0 того, что система свободна в следующий момент (k+1). Пользуясь формулами, имеющимися в [53, 54] можно изобразить следующую схему



где ν - интенсивность пребывания заявок в очереди, обратная среднему времени задержки в ней заявки.

В этой схеме есть 3 слагаемых:

- первое слагаемое – состояние, когда ни одна заявка не пришла в систему;

- второе слагаемое – состояние, когда в канале имеется обычная заявка, и она успела обслужиться за время Δt;

- третье слагаемое – состояние, когда в канале имеется важная заявка и она успела обслужиться.

$$P_0(k+1) - P_0(k) = P_0 = (\lambda_1 + \lambda_2)P_0 + Z_1\nu + \zeta_1\nu = 0$$

после принятых обозначений можно написать следующее уравнение:

$$Z_1 + \zeta_1 = (\rho_1 + \rho_2)P_0 \qquad (8.6)$$

2) Рассмотрим следующее событие, относящиеся к вероятности состояния $P_{1,0}$ т.е. обслуживания простой заявки в канале

Поэтому для установившегося режима можно написать:

$$P_0 = Z_1\mu + \zeta_1\mu - (\lambda_1 + \lambda_2)P_0 \qquad (8.7)$$

$$\dot{Z}_1 = \zeta_1\mu + P_0\lambda_2 - Z_1(\lambda_1 + \lambda_2 + \mu) \text{ или } 2Z_{1+1} = Z_1(1 + \rho_1 + \rho_2) - P_0\rho_2 \qquad (8.8)$$

Рассмотрим еще одно событие, учитывающее присутствие в системе важной заявке

3) $\zeta_1(k+1)\Delta t$ → $P_0\lambda_1\Delta t$; → $\zeta_1\left[1-(\lambda_1+\lambda_2+\mu)\Delta t\right]$; → $\zeta_{1+1}2\mu\Delta t$

Получим формулы:

$$\dot{\zeta}_{1+1}=\zeta_{1+1}2\mu+P_0\lambda_1-\zeta_1(\mu+\lambda_1+\lambda_2)$$

$$2\zeta_{1+1}=\zeta_1(1+\rho_1+\rho_2)-\rho_0\rho_1 \qquad (8.9)$$

Получим 4 уравнения при 5 искомых вероятностях, т.е. одного условия нехватает. При этом переход из ζ_{1+1} в Z_{1+1} невозможен, т.к. сразу должно произойти два маловероятных события. Поэтому сделаем кроме перечисленных выше 6 допущений ещё одно, **седьмое общее допущение**

$$\zeta_{n+1}\cong\frac{\rho_1}{\rho_2}Z_{n+1} \qquad (8.10)$$

Это допущение указывает, что в случае весьма оживленного потока событий, когда все каналы заняты и в системе уже есть важная заявка, а в очереди появилась первая заявка, то вероятность ζ_{n+1} отказа при появлении новой заявки явно меньше, чем вероятность Z_{n+1} того, что в системе имеется только (n+1) простых заявок. Тогда используем повторяющийся ниже прием, выразив все вероятности через ζ_1 и P_0.

$$Z_1=\zeta_1(\rho_1+\rho_2)-P_0;\ Z_{1+1}=0,5\left[\rho_1+(\rho_1+\rho_2)^2\right]P_0-(1+P_1+P_2)\zeta_1$$

$$\zeta_{1+1}=0,5\left[\zeta_1(1+P_1+P_2)-P_0P_1\right]$$

После чего можно определить искомые вероятности в квадратурах

$$P_0=\frac{1}{1+P_1+P_2+0,5(P_1+P_2)^2} \qquad (8.11)$$

$$\zeta_1=\rho_1\rho_0;\ Z_1=\rho_2\rho_0;\ \zeta_{1+1}=0,5\rho_1(P_1+P_2)P_0;\ Z_{1+1}=0,5\rho_2(1+\rho_2)P_0$$

а выигрыш Bв вероятности отказа в обслуживании приоритетной и бесприоритетной заявки согласно допущению (5) равен

$$B = \frac{Z_{1+1}}{\zeta_{1+1}} = \frac{\rho_2}{\rho_1} \qquad (8.12)$$

Теперь рассмотрим случай *n*=2

В начале перечислим все возможные состояния x_i событий, образующих полную группу

x_0 – система свободна от заявок;

$x_{1,0}$ – в системе обслуживается одна простая заявка;

$x_{2,0}$ – в системе обслуживаются две заявки, т.е. все каналы заняты;

$x_{0,1}$ – в канале обслуживается одна важная заявка;

$x_{1,1}$ – в канале обслуживается одна важная заявка, а другая простая.

При рассмотрении ситуации с очередями нужно иметь ввиду, когда все каналы заняты простыми заявками:

$x_{2,0,1,0}$ – в канале обслуживаются две простые заявки, а в очереди стоит одна простая заявка;

$x_{2,0,2,0}$ – в канале обслуживаются две простые заявки, и в очереди стоят две; простые заявки

$X_{2,0,0,1}$ – в канале обслуживаются две простые заявки, а в очереди стоит одна важная;

$X_{2,0,1,1}$ - канал занят простыми заявками, а в очереди стоят одна важная ,одна простая.

Рассмотрим состояние, когда в канале обслуживается одна важная заявка. Тогда в очереди не должно быть ни одной важной заявки, потому что в системе может присутствовать только одна заявка: либо в канале, либо в

очереди. Поэтому для двухканальной системы n=2 с ожиданием получаются следующие вероятности состояний приl=n:

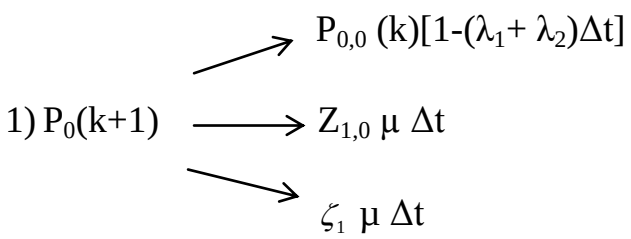
$X_{1,1,1,0}$ – в системе обслуживается одна важная и одна простая заявки, а в очереди стоит одна простая заявка;

$X_{1,1,2,0}$ - в системе обслуживается одна важная и одна простая заявки, а в очереди стоят две простые.

Полной группе событий соответствует формула

$$P_0 + Z_1 + \zeta_1 + Z_2 + \zeta_2 + Z_{1+1} + \zeta_{1+1} + Z_{2+2} + \zeta_{2+2} = 1 \qquad (8.13)$$

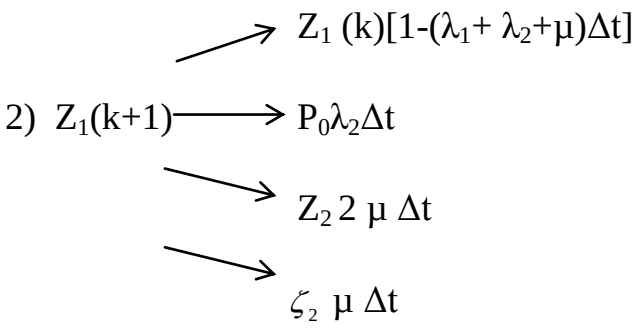
Можно представить 7 следующих схем :



-первое слагаемое – описывает состояние, когда ни одна заявка не пришла в очередь и ни одна не обслужилась из нее;

- второе слагаемое – описывает, когда из канала обслуживания обслужилась одна простая заявка;

-третье слагаемое – описывает, когда из канала обслуживания обслужилась одна важная заявка.



гдеν - величина, обратная среднему времени пребывания в очереди.

μ - средняя интенсивность обслуживания заявки в одном канале.

-первое слагаемое – описывает состояние, когда ни одна заявка не пришла в систему и ни одна не обслужилась;

- второе слагаемое – описывает, когда в систему пришла обычная заявка;

-третье слагаемое – вероятность того, что простая заявка в системе обслужилась.

- четвертое слагаемое – описывает, когда одна важная заявка обслужилась, а вторая простая заявка осталась в канале обслуживания.

$$3)\ \zeta_1(k+1)\Delta t \longrightarrow \begin{cases} P_0\lambda_1\Delta t \\ \zeta_1\left[1-\left(\lambda_1+\lambda_2-M\right)\Delta t\right] \\ \zeta_2 2M\Delta t \end{cases}$$

В этой схеме первое слагаемое учитывает появление одной важной заявки, когда два канала свободны.

- второе слагаемое соответствует вероятности сохранения ситуации, когда в системе обслуживается только одна важная заявка.

- третье слагаемое соответствует ситуации, когда в каналах была одна простая и одна важная заявка, и одна простая заявка успела обслужиться.

Рассмотрим следующее событие, относящиеся к вероятности состояния Z_2 т.е. обслуживания двух простых заявок в канале

$$4)\ P_{2,0}(k+1) \longrightarrow \begin{cases} Z_2(k)[1-(\lambda_1+\lambda_2+2\mu)\Delta t] \\ Z_1\lambda_2\Delta t \\ \zeta_{2+1}\ \mu\Delta t \end{cases}$$

-первое слагаемое – описывает состояние, когда ни одна заявка не пришла в систему и ни одна не обслужилась, т.е. как были две простые заявки, так и остались в момент времени *k*;

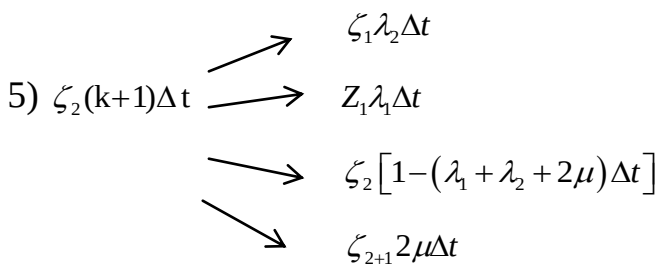
- второе слагаемое – описывает, когда в систему пришла обычная заявка, т.е. была одна простая заявка и пришла еще одна;

-третье слагаемое – вероятность того, что одна важная заявка в системе обслужилась, и в системе остались две простые заявки.

Тогда можно получить

$$\dot{Z}_2 = \zeta_{2+1}\mu + Z_1\lambda_2 - Z_{2,0}(\lambda_1 + \lambda_2 + 2\mu) = 0 \qquad (8.14)$$

Рассмотрим следующее событие, относящиеся к вероятности состояния ζ_2 при обслуживании важной и простой заявок в каналах



- первое слагаемое - описывает, когда в систему поступила важная заявка и когда в канале уже имелась важная заявка;

-второе слагаемое – описывает, когда в канал поступила важная заявка, и когда в канале уже имелась обычная заявка;

-третье слагаемое – описывает состояние, когда ни одна заявка не пришла в систему и ни одна не обслужилась, т.е. как были одна важная заявка и одна простая так и остались в момент времени *k*;

-четвертое слагаемое – вероятность того, что простая заявка в системе обслужилась и в системе остались одна простая и одна важная заявка.

Рассмотрим еще одно событие, когда в системе есть две простые заявки в каналах, и одна простая заявка имеется в очереди. Этому случаю соответствует схема 6.

$$Z_2\lambda_2\Delta t$$

6) $Z_{2+1}(k+1)\Delta t \rightarrow Z_{2+1}\left[1-\left(\lambda_1+\lambda_2-2\mu\right)\Delta t\right]$

$$Z_{2+2}3\mu\Delta t$$

$$\zeta_{2+2}2\mu\Delta t$$

Наконец рассмотрим последнюю схему 7, когда в системе есть одна важная заявка, и одна заявка имеется в очереди.

$$\zeta_2\lambda_2\Delta t$$

7) $\zeta_{2+1}(k+1)\Delta t \rightarrow Z_2\lambda_1\Delta t$

$$\zeta_{2+1}\left[1-\left(\lambda_1+\lambda_2-2M\right)\Delta t\right]$$

Добавив к формуле (8.13) и к семи уравнениям, полученным из 7 схем, ещё одно допущение $\zeta_{2+1}=\frac{\rho_1}{\rho_2}Z_{2+1}$ согласно (8.10), получим все вероятности, выраженные через P_0 и ζ_1 для случая n=2.

$$Z_1=P_0\left(\rho_1+\rho_2\right)-\zeta_1;\ Z_2=P_0\left[0,5\left(P_1+P_2\right)^2+\rho_1\left(1+P_2\right)-\zeta_1\left(1+P_1+\rho_2\right)\right]$$

$$Z_{1+1}=0,5\left[Z_1\left(1+P_1+P_2\right)-P_2P_0\right];\ \zeta_{1+1}=0,5\left[\zeta_1\left(1+P_1+P_2\right)-P_0\left(P_1+P_2\right)\right] \qquad (8.15)$$

$$\zeta_2=\left(1+\rho_1+\rho_2\right)\zeta_1+P_0\left(1+P_2\right)\rho_1;\ Z_{2+1}\approx 0,5\left(P_1+P_2\right)^2P_0-\zeta_1\left(1+\rho_2\right)$$

$$\zeta_{2+1}=\left(1+\rho_2\right)\left(\zeta_1-\rho_1P_0\right);\ Z_{2+2}\cong 0,5\left(P_1+P_2\right)^2P_0-\zeta_1\left(1+\rho_2\right);$$

$$\zeta_{2+2}\cong 0,25\left(1+P_2\right)\left(\zeta_1-\rho_1P_0\right)$$

После чего можно также определить все вероятности в квадратурах, из которых нам важно знать вероятности ζ_{2+2} и Z_{2+2}, а значит – прежде всего значения ζ_1 и P_0.

$$\zeta_1 = \rho_1 P_0 \left[1 + \frac{(\rho_1 + \rho_2)}{2(1+\rho_2)} \right] \tag{8.16}$$

$$\zeta_{2+1} = \frac{\rho_1^3 P_0 (\rho_1 + \rho_2)}{4\rho_2};\ Z_{2+1} = \frac{P_1^2 P_0 (P_1 + P_2)}{2(1+P_1)}$$

$$\zeta_{2+2} = \frac{2P_1 P_2 P_0}{15};\ Z_{2+2} \approx \frac{\rho_2^2 P_0}{2};\ B = 4\frac{\rho_2}{\rho_1} \tag{8.17}$$

Проведя аналогичные рассуждения для *n*=3, удалось установить, что в первом приближении можно записать формулы.

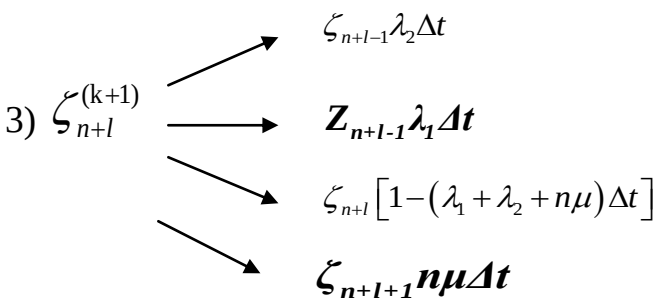
$$\zeta_{3+3} = \frac{9}{20}(\rho_1 + P_2)^2 \rho_1 P_0;\ Z_{3+3} = 2{,}5(P_1 + \rho_2)^3 P_0;\ B \cong 5{,}5\frac{\rho_2 + \rho_1}{\rho_1} \tag{8.18}$$

8.4 Обобщение условий перехода вероятностного состояния системы ввиде алгебраических формул

При дальнейшем увеличении значения *n* и фиксированной длине очереди *l*были обнаружены другие рекуррентные формулы и схемы.

1) $Z_n^{(k+1)} \longrightarrow$
- $Z_{n-1}(\lambda_1 + \lambda_2)\Delta t$
- $Z_n(k)\left[1 - (\lambda_1 + \lambda_2 - n\mu)\Delta t\right]$
- $\boldsymbol{(Z_{n+1} + \zeta_{n+1})(n+1)\mu\Delta t = P_{n+1}(n+1)\mu\Delta t}$

2) $Z_{n+l}^{(k+1)} \longrightarrow$
- $Z_{n+l-1}\lambda_2\Delta t$
- $Z_{n+l}\left[1 - (\lambda_1 + \lambda_2 - nM)\Delta t\right]$
- $\boldsymbol{Z_{n+l+1}(n+1)\mu\Delta t}$
- $\boldsymbol{Z_{n+l+1}2\mu\Delta t}$



Из приведенных схем видно, что при $l>0$ схемы перехода в приоритетной СМО существенно отличаются от используемых в формулах Эрланга из-за несовпадения выделенных более жирно **элементов с вероятностью** P_{n+1}.

Проведя рассуждения для вероятности состояния системы при $n=1$, $n=2$, $n=3$, $n=4$ и. т. д, можно получить рекуррентные соотношения для различных практических важных случаев.

Для установившегося состояния можно написать следующие частные уравнения, считая, что $\beta=1$

$$Z_{1+1}(1+\beta)=Z_1(1+\rho_1+\rho_2)-P_0\rho_2$$

$$Z_{2+1}(2+\beta)=Z_2(2+\rho_1+\rho_2)-P_1\rho_2 \qquad (8.19)$$

$$\xi_{1+1}(1+\beta)=\xi_1(\rho_2+\rho_1+2)-P_0\left(\rho_2+\rho_1\right)$$

С помощью (8.19) получены следующие более общие рекуррентные формулы:

$$Z_i+\zeta_i=\frac{P_0\left(\rho_1+\rho_2\right)^i}{i!}+\rho_1\zeta_1\frac{i-2}{i}\text{ при } i\geq 2 \text{ и } l=0$$

$$\zeta_{n+1}=\left(1+\rho_2\right)\zeta_1-\rho_1\left(1+\rho_2\right)P_0 \qquad (8.\ 20)$$

$$Z_{n+1}=P_0\left[\rho_1\left(1+\rho_2\right)+\frac{\left(P_1+P_2\right)^n}{n!}\right]-\zeta_1\left(1+P_2\right)$$

Все это позволило получить обобщенные формулы для ζ_{n+l} и Z_{n+l}, и в первую очередь рассчитать вероятности отказа в приоритетном и бесприоритетном обслуживании ζ_{n+n} и Z_{n+n}

В формулах (8.16 – 8.20) при увеличении номера очереди *l* обнаруживаются следующие общие элементы.

$$\left.\begin{aligned} Б &= n-1+\frac{p_1}{2p_2}\frac{n^{n-1}(1+n+n^n)}{(n+1)!} \\ A &= \frac{(p_1+p_2)^{n-1}}{n!}\frac{Б}{Б+p_1}p_1P_0 \\ M &= \frac{(p_1+p_2)^{n-1}}{n!}p_1P_0 \end{aligned}\right\} \qquad (8.21)$$

С помощью этих элементов можно наконец получить при $\nu = м$ следующую группу общих формул для $n \geq 2$, необходимую для расчета приоритетной СМО[59].

$$\left.\begin{aligned} Z_{n+l} &= \frac{n^{z_{l-1}}(1+l+l^2)(M-A)}{(l+1)!\Pi_{m=1}{}^{l}(n-1+n*m)} \\ \zeta_{n+l} &= \frac{2n^{l}[(l-1+\frac{p_1 l}{n})A-(l-1)M]}{\Pi_{m=1}{}^{l}(n-1+m*n)} \end{aligned}\right\} \qquad (8.22)$$

При*l*= *n* можно вычислить интересующие нас вероятности отказа в обслуживании Z_{n+n} и ζ_{n+n}.

$$\zeta_{n+n} = \frac{4n^n\rho_1^2}{(\rho_1+\rho_2)\prod_{m=1}^{n}(n-1+mn)}P_n$$

$$Z_{n+n} = \left[(\rho_1+\rho_2)^n + \frac{\rho_1^2 n^{2n}}{(\rho_1+\rho_2)n!}\right]\frac{P_n}{\prod_{m=1}^{n}(n-1+mn)} \qquad (8.23)$$

где, $P_n = \dfrac{(\rho_1 + \rho_2)^n}{n!} P_0$

При $\rho_1 = 0$ формулы (8.23) совпадают с известными формулами Эрланга.

В завершение осталось получить рекуррентное соотношение для оценки выигрыша $B = \dfrac{\zeta_{n+n}}{Z_{n+n}}$. Подставляя из формул (8.23) вычисленные вероятности ζ_{n+n} и Z_{n+n} для различных значений *n*, можно убедиться, что при *n*>3 справедлива формула оценки выигрыша *B* в отказе обслуживания важных и обычных заявок.

$$B = \frac{n^{n-1}\left(1+n+n^2\right)\rho_2}{2(n-1)n!.\rho_1} \qquad (8.24)$$

В частности при $\rho_1 = 0{,}1;\ \rho_2 = 0{,}7$ и *n*=4 имеем $\dfrac{BP_1}{P_2} = 5{,}5$; при *n*=6 имеем $\dfrac{BP_1}{P_2} = 6$, т.е. даже при *n*=4,5 использование приоритетного обслуживания примерно в 5 раз лучше бесприоритетного.

Построим график логарифмической зависимости вероятности отказа в обслуживании пассажиров от числа каналов в аэропорту, а также зависимость вероятности отказов при бесприоритетном обслуживании. (см рис. 8.2).

Сравнивая графики 1 и 2 на рис 8.2, можно убедиться, что вероятность отказа в обслуживании пассажиров с аварийного самолета более чем в 5 раз ниже при приоритетном обслуживании, чем бесприоритетном. Значит, этим достигается максимальная безопасность обслуживания пассажиров после их прилета[59].

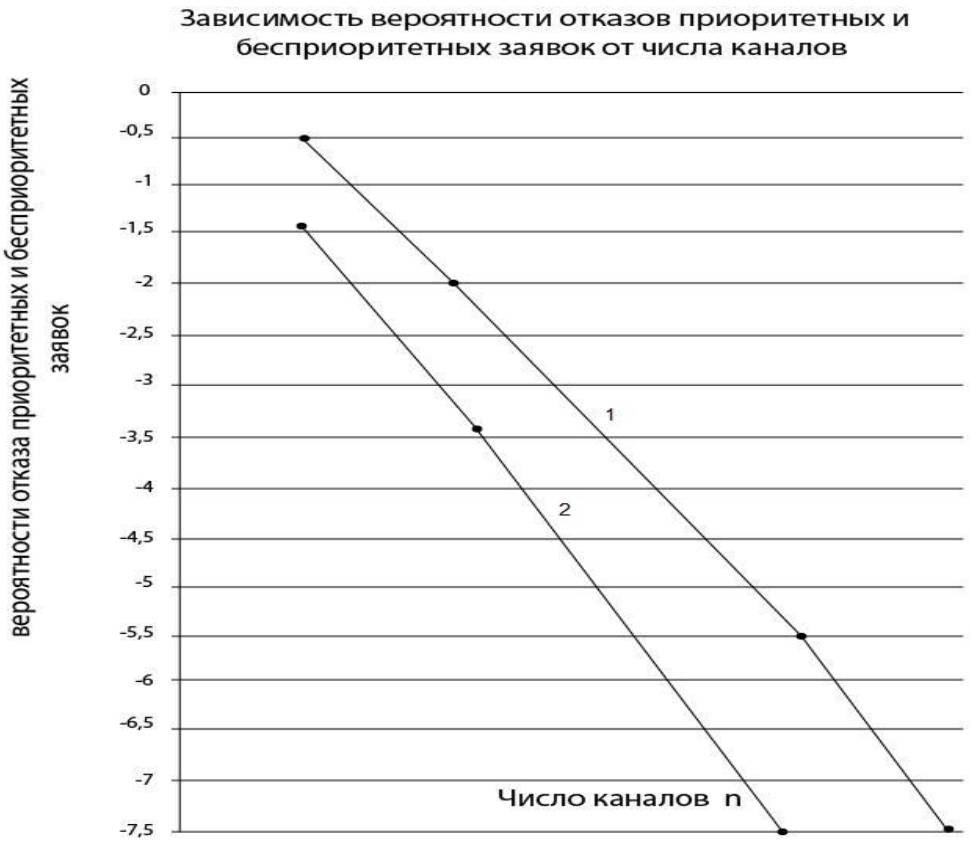


Рис. 8.2 . Зависимость вероятности отказа от числа каналов; график 1- в бесприоритетном обслуживании, график 2 – в приоритетном обслуживании.

Также графики показывают, что при заданной вероятности отказа можно назначать требуемое число каналов. Например, если рассмотреть случай многоканального обслуживания пассажиров в аэропорту, то при требуемой вероятности $P_{n+n} = 10^{-4}$ получается, что пассажиры после аварийного прилета будут быстро обслужены, как только освободится один из трех каналов.

8.5 Выбор числа каналов обслуживания пассажиров в аэропорту по критерию минимальной средней стоимости с учетом реальной взаимопомощи между каналами

Полученные выше формулы бесприоритетного и приоритетного обслуживания, так же как и формулы Эрланга, найдены в предположении,

что передача заявки из своего занятого канала «в чужой» свободный канал не связана с дополнительными потерями времени. Однако при обслуживании пассажиров это не так – при переходе пассажиров в другой канал необходимо определенное время, тогда как в идеальном случае интенсивность μ обслуживания каждой заявки после передачи в любой канал будет одинакова[60].

В данной работе рассмотрен другой случай, когда пассажиры, прилетевшие в аэропорт (в первую очередь аварийным самолетом), передаются в другой освободившийся канал (если занят свой) с естественной дополнительной потерей времени, характеризуемой снижением скорости обслуживания $\Delta\mu < \mu$.

Формулы расчета СМО с реальной взаимопомощью между каналами, полученные в кандидатской диссертации [61], выглядят так.

Вероятности занятости каналов СМО равны:

$$P_i \cong \frac{\rho^i P_0}{i!\prod_{l=1}^{i}\left[1-\frac{\Delta\mu(l-1)}{\mu l\,\mathrm{n}}\right]};\quad i=1,..n \tag{8.25}$$

Для случая, когда все каналы заняты, уравнение перехода включает факт вероятного обслуживания единичной заявки в очереди, преимущественно в

“ чужом ” канале, что означает

$$P_{n+1} \approx \frac{\rho P_n}{n+\beta-\frac{\Delta\mu(n-1)}{\mu n}}$$

Для любого числа ω заявок в очереди, можно записать[51]

$$P_{n+\omega} \cong \frac{\rho^{\omega} P_n}{\prod_{m=1}^{\omega}\left[n+m\beta-\frac{\Delta\mu(n-1)}{\mu n}\right]} \tag{8.26}$$

Формулы (8.14 – 8.15) позволяют провести необходимые расчеты средних статистических характеристик СМО, в том числе и средней стоимости бесприоритетного обслуживания при любом назначенном или выбираемом числе каналов. В свою очередь, средняя стоимость Z всех затрат на обслуживание пассажиров может быть, как показано в [61], может быть определена формулой

$$Z = C_0(k-n) + C_1\sum_{i=0}^{n-1}(n-i)P_i + C_2\sum_{i=1}^{n} iP_i + C_3\sum_{\omega=1}^{S}\omega P_{n+\omega} \qquad (8.27)$$

При этом коэффициенты C_j затрат удовлетворяют общему условию

$$C_0 \le C_1 \le C_2 \le C_3 \qquad (8.28)$$

где C_0 - минимальные затраты, приходящиеся на один невключенной канал; C_1 - затраты на обслуживание одного включенного, но неработающего канала; C_2 - затраты на обслуживание одного включенного работающего канала; C_3 - затраты при отказе в обслуживании.

Нужно подчеркнуть, что при заданных технических параметрах каналов $\mu, \Delta\mu, \beta, S$ вероятности P_i являются функциям входной интенсивности потока λ и числа n включенных в работу каналов, подлежащих возможному выбору.

Теперь в представленные формулы необходимо внести поправки. В качестве первой поправки внесём в формулу (8.4) уточнение – вместо длины очереди S нужно подставить выбираемое число каналов nсогласно принятому в данной главе допущению о равенстве длины очереди числу каналов.

При этом, как это принято и в [59], считается, что при увеличении числа прилетевших пассажиров число n включенных каналов будет расти, и между этими каналами будет организована взаимопомощь. Под этим подразумевается, что часть пассажиров будет транспортироваться из своего

зала в другие, и на это уходит заданное дополнительно среднее время, Поэтому будем считать заданными следующие параметры, например

$$\frac{\Delta\mu}{\mu} = 0,3; \quad \beta = \frac{\nu}{\mu} = 0,1; \quad S = 3$$

При этих условиях требуется найти оптимальное число *k* включенных в работу линий в зависимости от скорости *λ(t)*пребывания пассажиров, исходя из условия минимальной стоимости обслуживания в аэропорту.

В качестве второй поправки внесем коррективу в определение коэффициента загрузки ρ одного канала с учетом того, что кроме обычных заявок вне очереди будут обслуживаться и приоритетные – пассажиры, прилетевшие на аварийном самолете. Можно предложить следующую компромиссную аппроксимацию

$$\rho^* \cong \frac{\lambda_2}{\mu} + \frac{\lambda_1}{\mu - \Delta\mu} = \rho\left[1 - \frac{\lambda_1\mu}{\lambda_2(\mu - \Delta\mu)}\right] \qquad (8.29)$$

Смысл этой формулы состоит в том, что первое слагаемое учитывает процесс обслуживания обычных заявок, а второе слагаемое – только приоритетных заявок, которые как только освободится любой канал, пассажиры без очереди будут туда направлены. Но при большом числе каналов *n*более чем вероятно попадание в “чужой” канал, чем в ближайший, и поэтому время перемещения пассажиров учитывается снижением скорости обслуживания. Отдельно нужно заметить, что в отсутствие приоритетных заявок при $\lambda_1=0$ новые формулы совпадают с прежними. С учетом этих поправок можно написать новые формулы.

При этих допущениях повторим логику расчетов нужных вероятностей и целевой функции, как это было сделано в [51]. С этой целью зададимся следующими коэффициентами C_j стоимости с учетом условия (8.17)

$$C_0=0,25; \quad C_1=0,5; \quad C_2=C_3=1$$

Расчеты целевой функции по полученным выше формулам представим в виде графиков на рис 8.3.

На рис 8.3 для сравнения непрерывными кривыми показаны оценки затрат при бесприоритетном обслуживании, а пунктиром – с учетом внеочередного обслуживания пассажиров с аварийных самолетов при допущении, что $\frac{\lambda_1}{\lambda_2} = 0{,}15$.

Из рис 8.3 видно, что во-первых, число включенных каналов обслуживания в аэропорту должно увеличиваться по мере увеличения

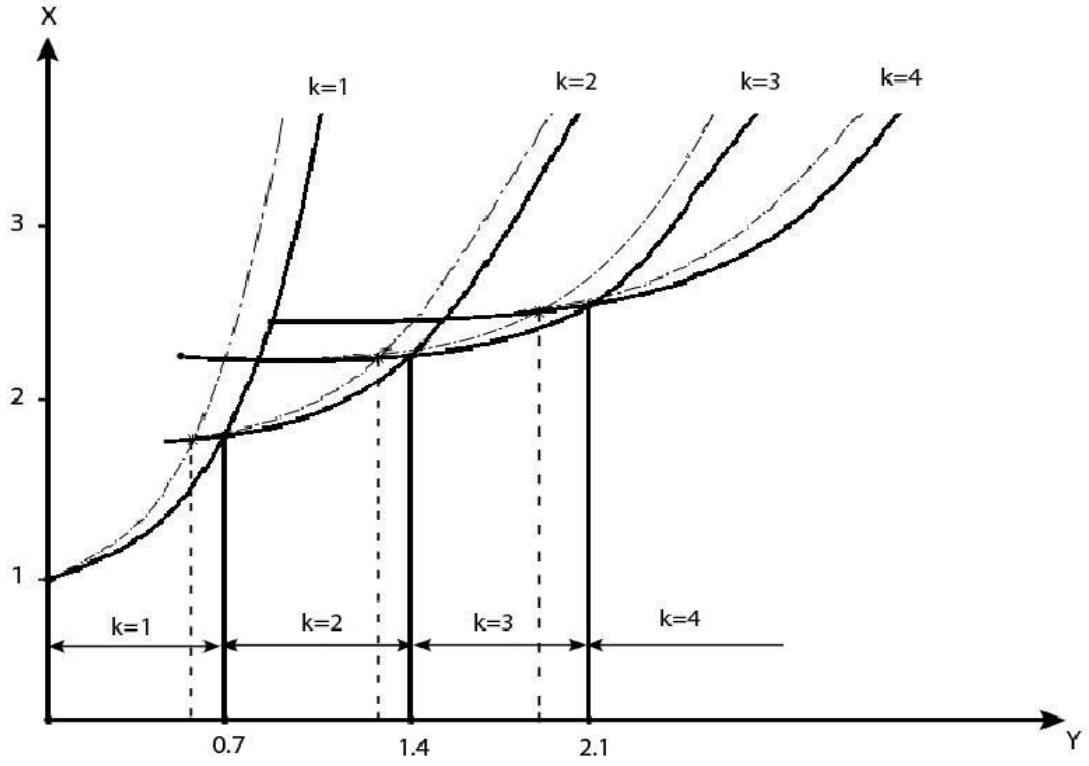


Рис. 8.3. Области включения оптимального числа работающих каналов обслуживания пассажиров в зависимости от входного потока

интенсивности потока прилетающих самолетов. Во-вторых, средние затраты Zна обслуживание в целом растут. В-третьих, оптимальное число каналов, которые нужно включить в работу, незначительно, но также растет, если есть информация о приближении к аэродрому самолета в аварийном состоянии.

Кроме того, в формулу (8.27) можно внести третью поправку. Дело в том, что в её вычислении можно учесть вероятность ζ_{n+n}отказа в обслуживании приоритетных заявок отдельным штрафом C_4, т.е. использовать новую формулу.

$$Z = C_0(\mathrm{k}-\mathrm{n}) + C_1\sum_{i=0}^{n-1}(\mathrm{n}-\mathrm{i})\rho_{\mathrm{i}} + C_2\sum_{i=1}^{n} i\rho_{\mathrm{i}} + C_3\sum_{w=1}^{n} w\rho_{n+w} + C_4\zeta_{n+n} \qquad (8.30)$$

где ζ_{n+n}вычисляется по формуле (8.23). Оказалось, что если взять $C_4 >> C_3$, например $C_4 = 4C_3$, то кроме увеличения затрат, в целомрасчетное число n включенных каналов должно увеличиться примерно на единицу.

8.6 Выводы по главе 8

На основании проведенных в данной главе исследований можно сделать следующие выводы:

1. Найдены рекуретные соотношения для вычисления вероятностных состояний многоканальной приоритетной СМО с ожиданием, для случая идеальной взаимопомощи между каналами. Это позволяет определить в квадратурах вероятности простоя и отказа в обслуживании обычных и приоритетных заявок.
2. Получены новые формулы расчета многоканальных однофазных **приоритетных** СМО с ожиданием для случая оказания реальной взаимопомощи между каналами, обслуживающими пассажиров.
3. Сформулирована задача оптимизация выбора числа работающих каналов приоритетной СМО по критерию минимальной стоимости обслуживания с учетом простоя, занятости каналов и дополнительных затрат при появлении пассажиров, прилетевших на аварийном самолете.
4. Показано, что число обслуживаемых каналов в аэропорту должно

   увеличиваться по мере повышения числа подлетающих самолетов, а в случае прилета аварийного самолета это число должно быть увеличено на единицу.
5. Установлено, что занятость каналов в приоритетных СМО незначительно выше, чем в приоритетных, поэтому вероятность простоя P_0 в приоритетной СМО немного меньше.

6. Достоверность полученных формул для расчета приоритетных СМО подтверждается тем, что при $\rho_1 = 0$ они совпадают с известными формулами Эрланга.

Глава 9. Моделирование на ЭВМ системы обслуживания воздушных самолетов и результаты внедрения

9.1 Описание программы выбора посадочных курсов

Программа выбора посадочных курсов моделировалась на языке C++ для различных значений угла ветра. Моделирование алгоритма работы системы производилось, используя VisualStudio 2010.

Результат работы программы – таблица значений исходных курсов и альтернативных курсов, куда предполагается отправить авиалайнер в условиях изменившейся ветровой обстановки. Полученные на выходе результаты можно сравнить с ручным счетом.

Входные данные:

Массив посадочных курсов - задан константами в коде

Радиус окружности - принят 100км

WindDirection - направление ветра задается с клавиатуры

WindSpeed - скорость ветра задается с клавиатуры

MaximumWindSpeed - допустимая скорость ветра на посадке

Существует два разных понятия определения направления ветра. Ветер, который дует, например, четко с востока в метеорологии будет иметь направление 90 градусов (ветер метеорологический), а в авиации направление ветра будет обратное – 270 градусов (ветер авиационный). Это сделано для того, чтобы при взлете или посадке экипажу было легче определить отклонение направления ветра относительно курса полосы. Разрабатываемая в данной работе программа использует как раз понятие авиационного ветра.

Текст программы приведен в приложении 1. Результаты работы программы приведены в виде тестов.

```
Landing Courses:  66 136 14 58 246 316 194 238
TABLE 1
(0)  0   70   52   8   180   110   128   172
(1)  70   0   122   78   110   180   58   102
(2)  52   122   0   44   128   58   180   136
(3)  8   78   44   0   172   102   136   180
(4)  180   110   128   172   0   70   52   8
(5)  110   180   58   102   70   0   122   78
(6)  128   58   180   136   52   122   0   44
(7)  172   102   136   180   8   78   44   0
TABLE 2
(K)  4   5   2   7
(M)  246   316   14   238

TABLE 3
(M)   4   5   2   7
(P0)  314.158   38.3971   90.7568   300.195
(P1)  191.985   314.158   106.465   178.023
(P2)  223.401   101.229   0   33.9091
(P3)  75.0489   178.023   76.7942   314.158
(P4)  0   24.4345   223.401   13.9626
(P5)  122.173   0   101.229   136.135
(P6)  90.7568   212.929   314.158   10.9706
(P7)  13.9626   136.135   237.364   0

TABLE 4
(i)     0   1   2   3   4   5   6   7
(Mmin)  5   2   5   5   5   5   7   7
```

Тест 1 при угле ветра 120̊

В первом примере задан угол ветра 120 градусов. На выходе программа выдает список курсов полос аэродромов и альтернативный курс к каждой конкретной полосе.

Таблица 9.1. Результаты определения посадочных курсов в тесте 1

Курс полосы	66	136	14	58	246	316	194	238
Новый курс полосы	316	14	316	316	316	316	238	238

Как видно, программа на выходе выдает рекомендации о смене посадочного курса полосы в условиях изменившейся ветровой обстановки с учетом потерь топлива при перелете к другому аэродрому.

Во втором примере рассмотрен угол ветра в 270 градусов.

Таблица 9.2. Результаты определения посадочных курсов в тесте 2

Курс полосы	66	136	14	58	246	316	194	238
Новый курс полосы	66	136	58	58	136	66	58	136

```
Landing Courses:  66 136 14 58 246 316 194 238
TABLE 1
(0)  0   70   52   8   180   110   128   172
(1)  70   0   122   78   110   180   58   102
(2)  52   122   0   44   128   58   180   136
(3)  8   78   44   0   172   102   136   180
(4)  180   110   128   172   0   70   52   8
(5)  110   180   58   102   70   0   122   78
(6)  128   58   180   136   52   122   0   44
(7)  172   102   136   180   8   78   44   0
TABLE 2
(K)  0   1   2   3
(M)  66   136   14   58

TABLE 3
(M)   0   1   2   3
(P0)  0   122.173   90.7568   13.9626
(P1)  122.173   0   106.465   136.135
(P2)  90.7568   212.929   0   25.5981
(P3)  13.9626   136.135   76.7942   0
(P4)  314.158   191.985   223.401   300.195
(P5)  191.985   314.158   101.229   178.023
(P6)  223.401   101.229   314.158   237.364
(P7)  300.195   178.023   237.364   314.158

TABLE 4
(i)     0   1   2   3   4   5   6   7
(Mmin)  0   1   3   3   1   0   3   1
```

Тест 2 при угле ветра 270̊

Результаты моделирования в приведенных тестах подтвердили правильность работы программы.

9.2 Описание программы назначения приоритетов попадания самолетов на каждую из трасс

Моделирование программы назначения динамических приоритетов проводилось на языке С++ при проверке формул (6.1) с помощью ряда тестов. Текс программы приведен в приложении 2.

Результаты работы программы приведены в виде трех тестов.

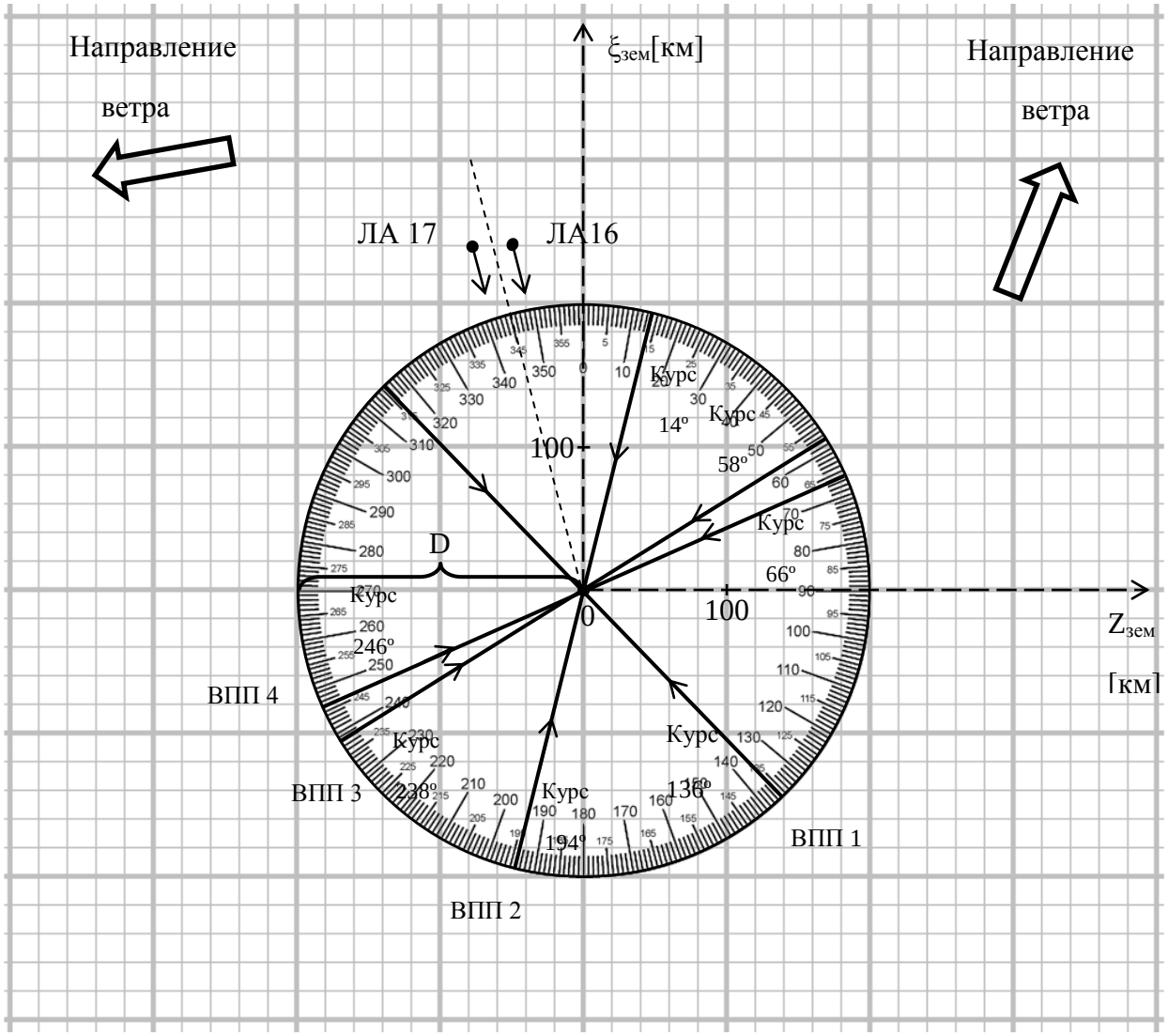


Рис.9.1. Заход на посадку для 16-того и 17-того самолета по приоритету

В первом случае пусть 16-тый и 17-тый самолеты находятся близко друг к другу. У 16-того самолета $Z=-50$, у 17-ого $Z=-80$, а значения ξ,ψ - это одинаковые координаты $\xi=240,\psi=165$. Это означает, что самолеты

находятся рядом с биссектрисой между двумя линиями пути – 194 и 136 курсами, как показано на рис.9.1. Тогда 16-тый самолет должен садиться на 194 курс во Внуково, 17-тый самолет должен садиться на 136 курс в Домодедово.

Расчеты приоритетов с помощью формулы (6.1) подтверждают этот факт. Как видно из таблицы 9.3 приоритетов, чем больше значение числа П, тем выше приоритет, значит минимальный штраф самолета 16 принадлежит ВПП-2, самолета 17 – ВПП-1. поэтому часть формулы (6.1), учитывающая различные значения $Z_к$ у самолета, верная.

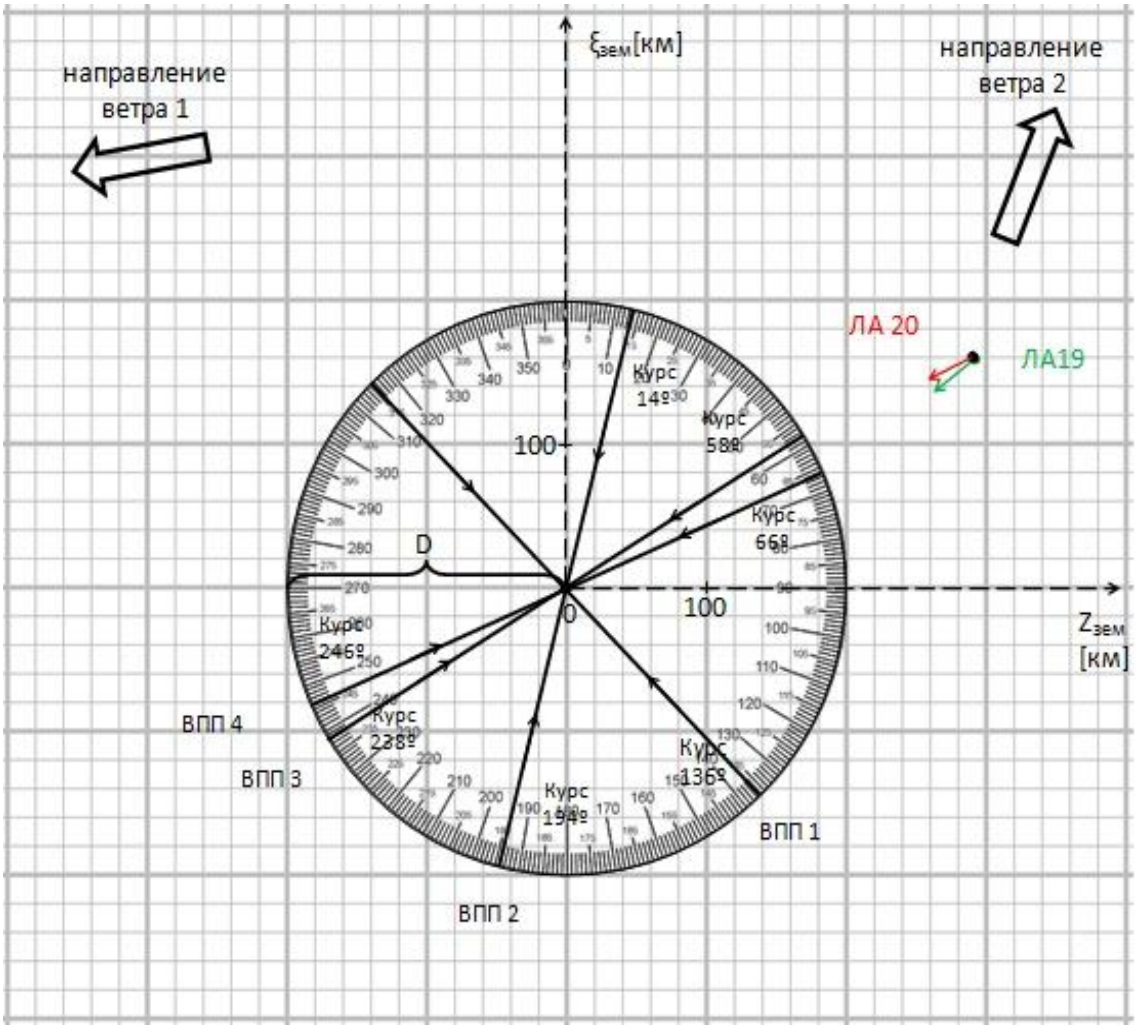


Рис.9.2. Заход на посадку для 19-того и 20-того самолета по приоритету.

Во втором случае пусть у самолетов будут разными только курсы. Сравним действия 19 и 20 самолетов. Тогда согласно рис.6 19-тый самолет

при $\zeta = 160, Z = 300, \psi = 238$ должен заходить на 246 курс, а 20-тый самолет при координатах $\zeta = 160,\ Z = 260,\ \psi = 250$ должен заходить на другой 238 курс.

Из таблицы 9.3 видно, что минимальный штраф самолета 19, а значит максимальный приоритет принадлежит ВПП-3, самолета 20 – приоритет тоже принадлежит ВПП–3, но П20>П19 . Значит часть формулы (6.1), учитывающая с помощью $y_3 = \varphi - \varphi_{тра}$ различные курсы – тоже верная.

Таблица.9.3. Приоритеты, вычисленные с учетом y_1, y_2, y_3, y_4

	ВПП 1	ВПП 2	ВПП 3	ВПП 4
ЛА 1	-8.7	-11.03	-10.55	-10.6
ЛА 2	**-126.1**	-9.1	-10.75	-10.63
ЛА 3	**-120.6**	-7.3	-10.9	-10.8
ЛА 4	-5.3	-3.7	-9.4	-8.9
ЛА 5	-8.2	-1.3	-9.1	-8.9
ЛА 6	-10.9	-5.5	-2.4	-4.0
ЛА 7	-10.87	-8.8	-3.7	-3.3
ЛА 8	-10.56	-8.7	-5.5	-5.3
ЛА 9	-10.35	-10.91	-8.7	-8.6
ЛА 10	-11.16	-11.8	-11.05	-11.04
ЛА 11	-11.37	-11.5	-10.89	-10.87
ЛА 12	-10.7	-11.37	-11.0	-10.8
ЛА 13	-10.5	-11.56	-11.44	-11.5
ЛА 14	-10.2	-11.69	-11.03	-11.04

ЛА 15	-9.1	-10.85	-10.22	-10.18
ЛА 16	-6.1	***-5.7***	-9.4	-9.2
ЛА 17	***-6.5***	-6.6	-9.6	-9.6
ЛА 18	210	215.1	216.2	213.4
ЛА 19	-11.17	-8.7	***-2.11***	-2.15
ЛА 20	-11.09	-8.5	***-1.6***	-1.8

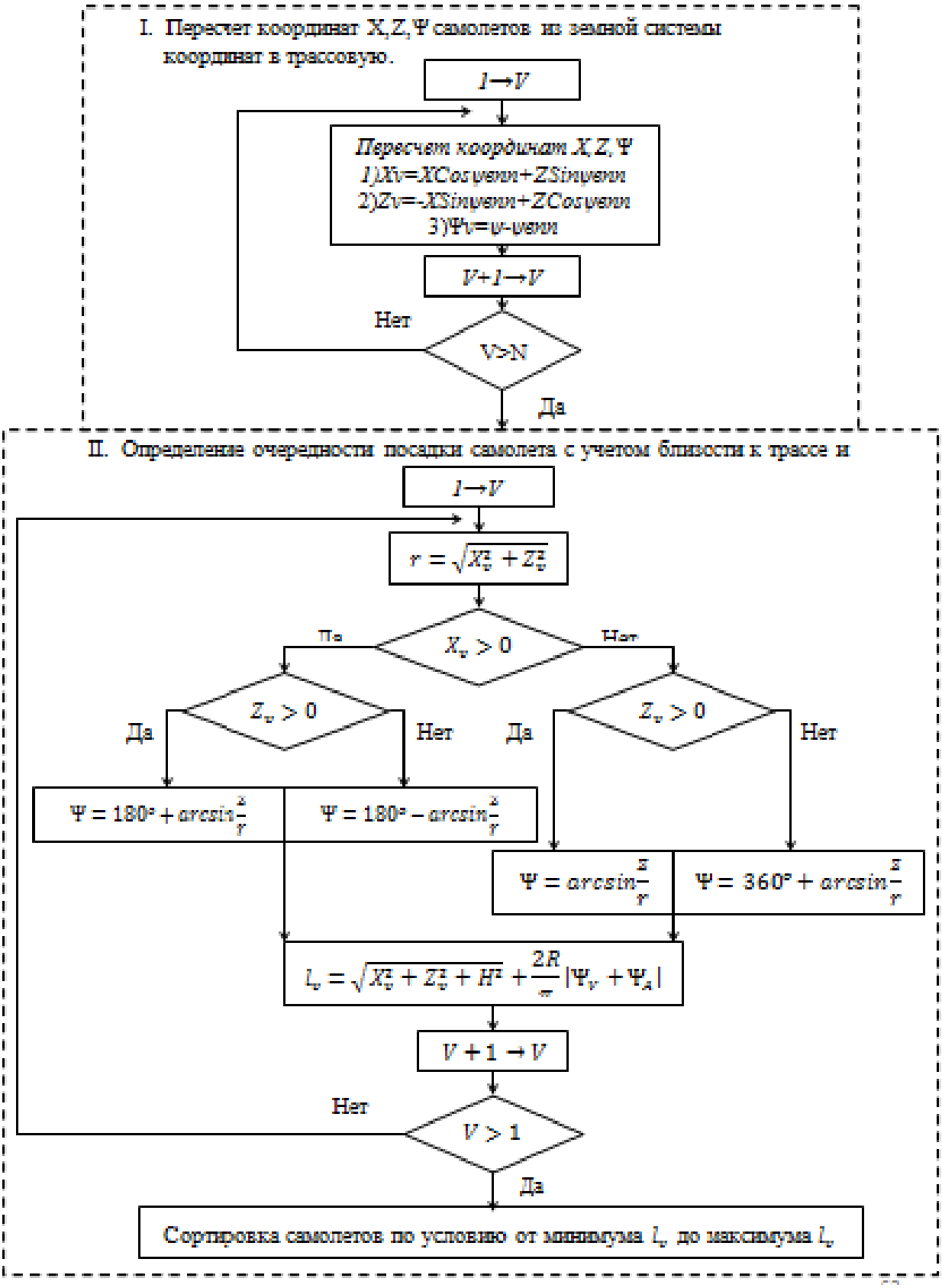
Нужно подчеркнуть, что проверенные части формулы (6.1) в круглых скобках учитывают экономичность полета самолетов при входе в эшелон.

В третьем случае осталось проверить правильность учета в формуле (6.1) фактора доли израсходованного топлива – если она велика, то самолет считается аварийным, и его надо сажать в первую очередь, т.е. его приоритет должен быть очень высок. Судя по расчетам, к таким самолетам относится №18, если его значение $y_4 = 0.9$. Поэтому, если этот факт не учитывать, его приоритет в списке самолетов, заходящих на ВПП-3, занимает шестое место, т.к. его приоритет П=-84. Если этот показатель учесть, то согласно таблице 9.3 приоритет этого самолета возрастаети становится равным П=215, и он займет первое место в списке и обязательно попадет в эшелон ВПП-3. Значит, в конце концов вся формула (6.1) реализуется в программе верно.

Примечание – формула (6.1) используется для судов попавщих в заднюю полусферу, а для остальных судов в программе вычисления приоритетов используется другая формула расчета.

9.3 Описание программы выбора первоочередности приземления судов

Ниже представлен более подробно алгоритм моделирования, состоящий из трех частей для выполнения поставленной задачи. Текст программы представлен в приложении 3.


I. Пересчет координат X,Z,Ψ самолетов из земной системы координат в трассовую.
1→V
Пересчет координат X,Z,Ψ
1)Xv=XCosψвпп+ZSinψвпп
2)Zv=-XSinψвпп+ZCosψвпп
3)Ψv=ψ-ψвпп
V+1→V
Нет
V>N
Да
II. Определение очередности посадки самолета с учетом близости к трассе и
1→V
$r = \sqrt{X_v^2 + Z_v^2}$
Да
$X_v > 0$
Нет
$Z_v > 0$
Да
Нет
$Z_v > 0$
Да
Нет
$\Psi = 180^\circ + arcsin\frac{z}{r}$
$\Psi = 180^\circ - arcsin\frac{z}{r}$
$\Psi = arcsin\frac{z}{r}$
$\Psi = 360^\circ + arcsin\frac{z}{r}$
$l_v = \sqrt{X_v^2 + Z_v^2 + H^2} + \frac{2R}{\pi}|\Psi_v + \Psi_A|$
$V + 1 \rightarrow V$
Нет
$V > 1$
Да
Сортировка самолетов по условию от минимума l_v до максимума l_v

III. Определение точек попадания самолетов на трассу и вычисление нужного времени посадки.

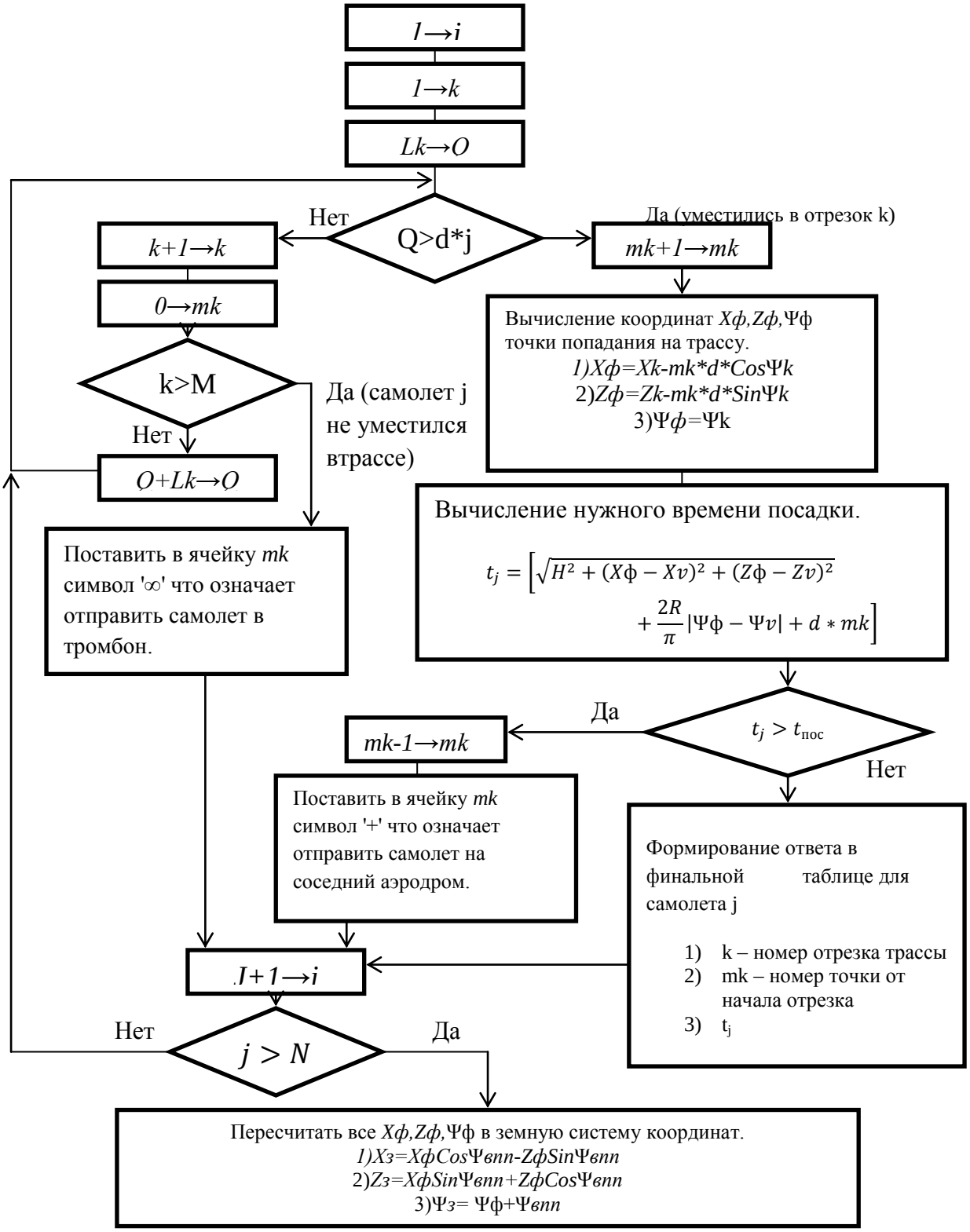


Рис 9.3. Блок-схема программы определения перевоочередности посадки воздушных судов

9.3.1 Описание программы на ЭВМ.

Моделирование алгоритма работы системы производилось в среде C++, используя VisualStudio 2013.

Результат работы программы – таблица, включающая в себя номер очереди (J), номер самолета по критерию экономичности (V), номер отрезка трассы (K), номер точки на трассе(m_k), координаты точки попадания на трассу($X_з$, $Z_з$), курс на отрезке трассы ($\Psi_з$), нужное время посадки (t_j). Полученные на выходе результаты можно сравнить с ручным счетом.

Входные данные:

1) Координаты начала снижения по глиссаде и курс первого отрезка в земной системе координат($X_{впп}$, $Z_{впп}$,$\Psi_{впп}$).
2) Таблица 1 включающая в себя координатыX, Z, Ψ, H, $t_{пос}$ самолетов.
3) Таблица 2 данных об отрезках трассы в трассовой системе координат (L_k, X_k, Z_k, Ψ_k).
4) Константы(включены в код программы):
    - r=6км – дистанция безопасного движения между судами.
    - V_o=0.1км/с – посадочная скорость полета.
    - R=5км – радиус разворота.

9.3.2 Результаты моделирования.

Задаем координаты начала снижения по глиссаде и курс первого отрезка в земной системе координат:

$X_{впп}$=1 $Z_{впп}$=3 $\Psi_{впп}$=15.

Вводим нужное количество самолетов и их значения:

Таблица 9.4. Исходные данные о воздушных судах

№	X	Z	Ψ	H	$t_{пос}$
1	350	-26	242	5	9675
2	260	-50	270	3	11232
3	200	36	162	7	7000
4	336	38	137	2	10453
5	290	-32	225	1	10567
6	200	70	334	9	12500
7	360	-29	242	10	55555

Вводим данные об отрезках трассы в трассовой системе координат:

Таблица 9.5. Исходные данные об отрезках трассы

№	L_k	X_k	Z_k	Ψ_k
1	18	0	0	0
2	18	20	-60	0

Введя все данные в программу для соответствующих переменных и запустив ее, получены следующие результаты:

Таблица 9.6. Результаты очередности захода судов на посадку и координаты их попадания на трассу

J	1	2	3	4	5	6	7
V	3	6	2	5	4	1	7
K	1	1	1	2	2	2	

m_k	1	2	3	1	2	+	∞
$X_з$	4.588	9.116	13.674	28.382	32.94		
$Z_з$	-3.902	-7.803	-11.705	54.685	50.784		
Ψз	15	15	15	15	15		
T_j	6735	12340	10790	9501	7037	10620	
$X_ф$	-6	-12	-18	14	8	2	
$Z_ф$	0	0	0	-60	-60	-60	
$Ψ_ф$	0	0	0	0	0	0	

По результатам работы можно сделать общий вывод-разработанная в данной работе программа может быть использована для частичного решения проблемы загруженности диспетчеров в Московском узловом диспетчерском районе.

9.4 Моделирование на ЭВМ системы контроля и управления безопасным попутным движением воздушных судов

В данной главе более подробно изложены результаты моделирования попутного движения судов, поскольку в главе 7 представлены лишь данные о траекторном движении при упрощенной динамике горизонтального полета. Ниже приведены результаты, анализирующие изменение скорости полета этих судов при координации движения и без неё.

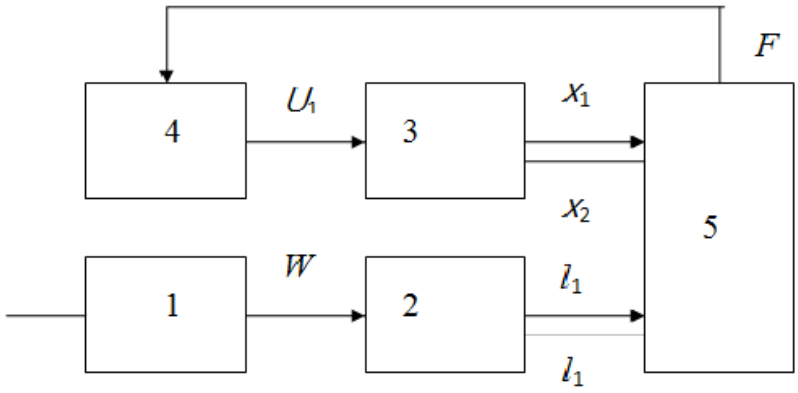


Рис.9.4 Общая блок-схема моделирования системы контроля и управления попутным движением судов

Моделирование проводилось в среде Matlab. Общая блок-схема системы представлена на рис.9.4.

На этой схеме показано 5 блоков, моделирующих движение двух воздушных судов. На рис 9.5 представлена модель1 неравномерного изменения скорости полета впередилетящего судна. Эта модель имитирует периодические эпизоды спада и увеличения скорости полета судна, находящегося в аварийном состоянии.

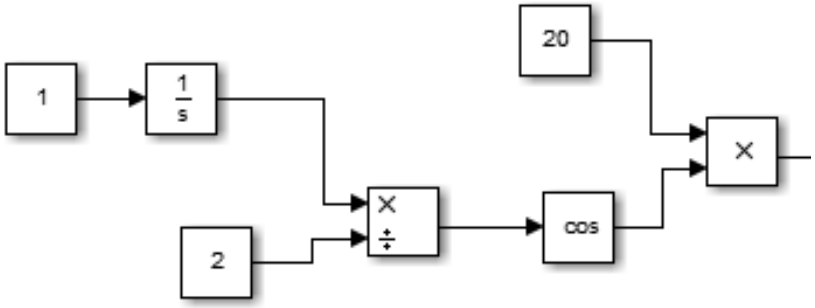


Рис 9.5. Модель1 колебаний скорости полета впереди летящего судна

На рис 9.6 представлена упрощенная модель 2 движения впередилетящего судна, которое наряду с равномерным замедлением скорости изменяет ее неравномерно при колебаниях продольной перегрузки.

Для воспроизведения этих колебаний была использована модель, представленная на рис 9.5 и реализующая возмущение w_1, действующее на впередилетящее судно в виде гармонической функции

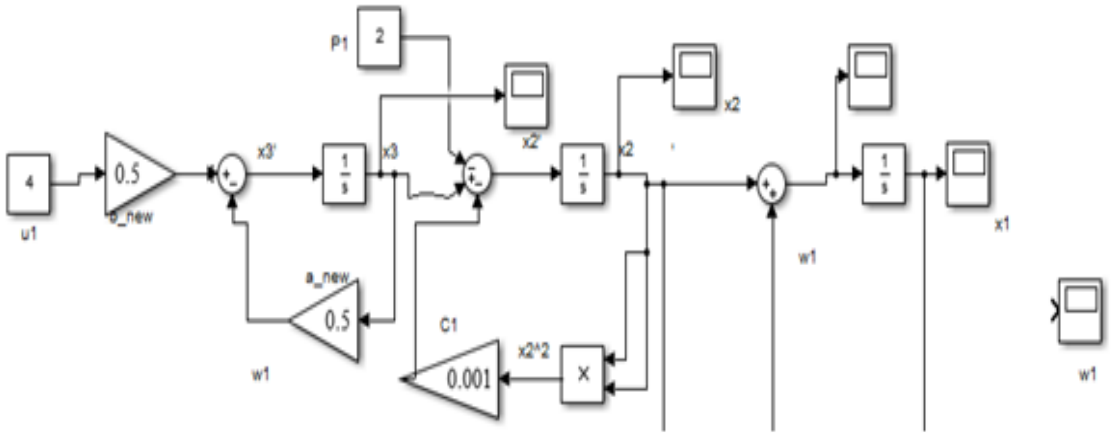


Рис 9.6.Модель 2 движения впередилетящего судна

На рис 9.7 представлена модель 3 движения сзадилетящего судна, на которое действует сигнал управления U_1, сформированный с учетом контроля безопасности попутного движения.

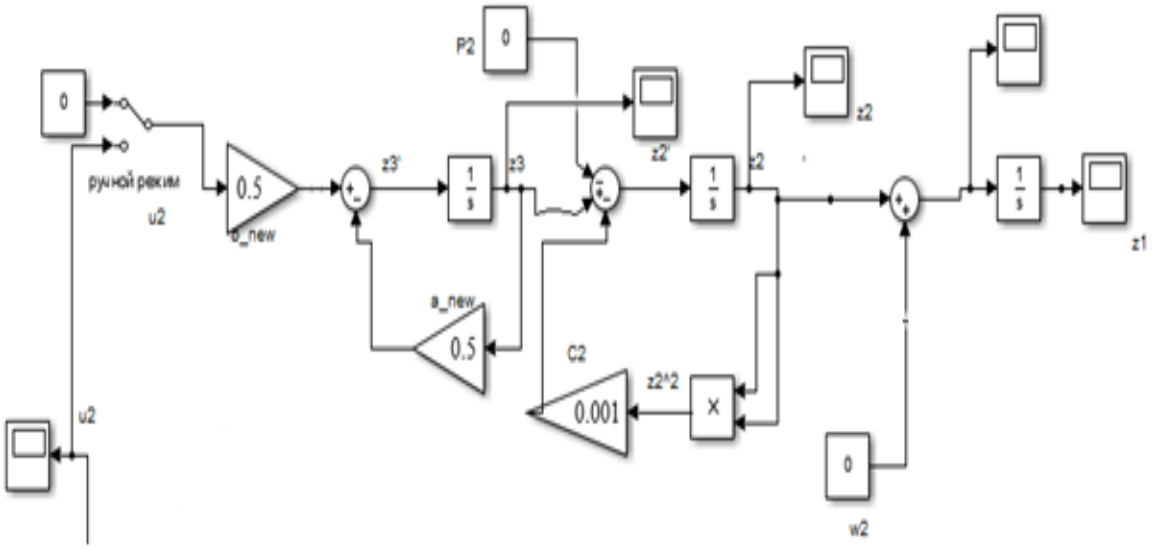


Рис 9.7. Модель 3 движения сзадилетящего судна

Этот сигнал формируется на выходе регулятора, модель 4 которого представлена на рис.9.8.

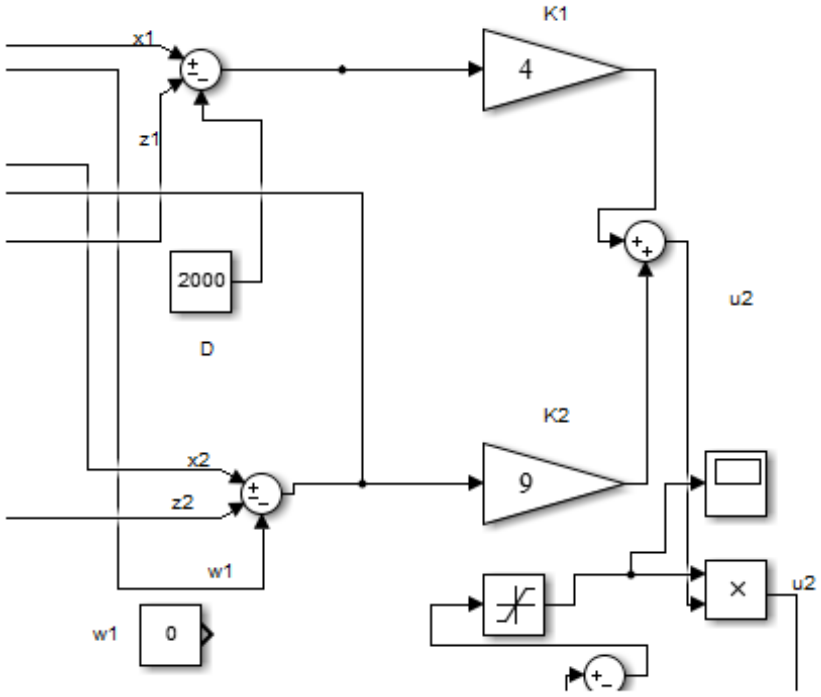


Рис 9.8. Модель 4 регулятора управления попутным движением сзадилетящего судна

Особенность этого регулятора состоит в том, что ненулевой сигнал управления возникает лишь тогда, когда функция риска $F>F_0$, в противном случае этот сигнал обнуляется. Значит, если $F_0=0$, то регулятор воздействует на объект всегда. Если $F_0 \to \infty$, то сигнал управления $U_1=0$, т.е. сзадилетящее судно никак не реагирует на изменения полета впередилетящего судна.

Сама функция риска вычисляется по формуле (7.23) и набрана в среде Matlab в виде схемы, показанной на рис.9.9

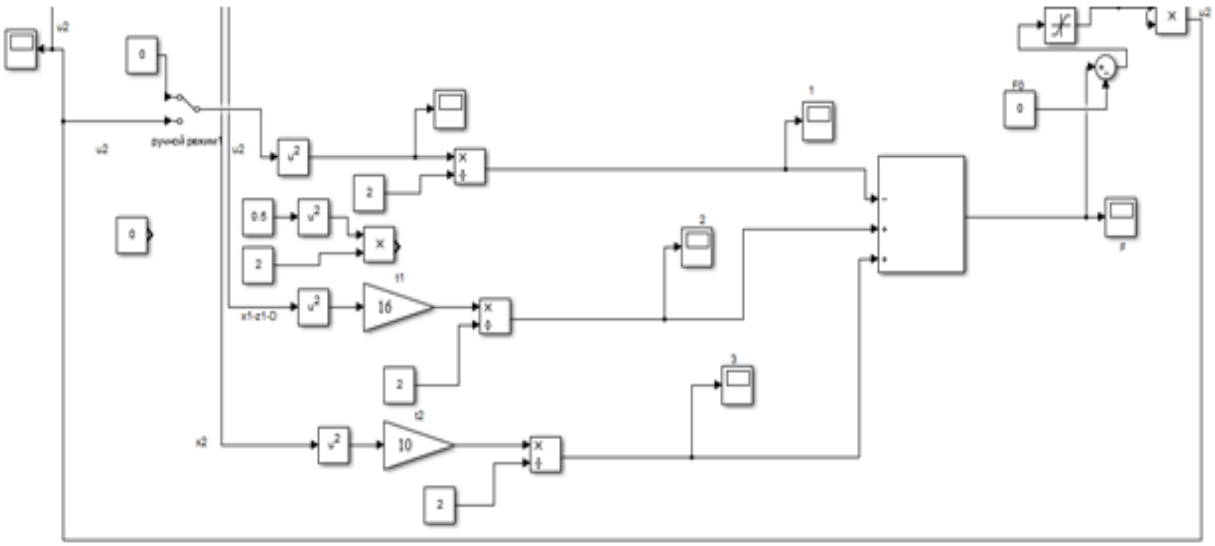


Рис 9.9. Модель 5 вычислителя функции риска

Моделирование полета двух судов проводилось для двух напряжённых режимов, когда впередилетящее судно является аварийным и имеющим технические неисправности. Назовём первый режим тяжелым, второй – аварийным.

Пусть **в первом тяжелом режиме** скорость впередилетящего судна снижается в среднем почти в два раза при средней продольной перегрузке торможения, равной примерно 0,3g., а амплитуда неравномерных колебаний скорости составляет 15-20 м/сек, как это показано на рис 9.10.

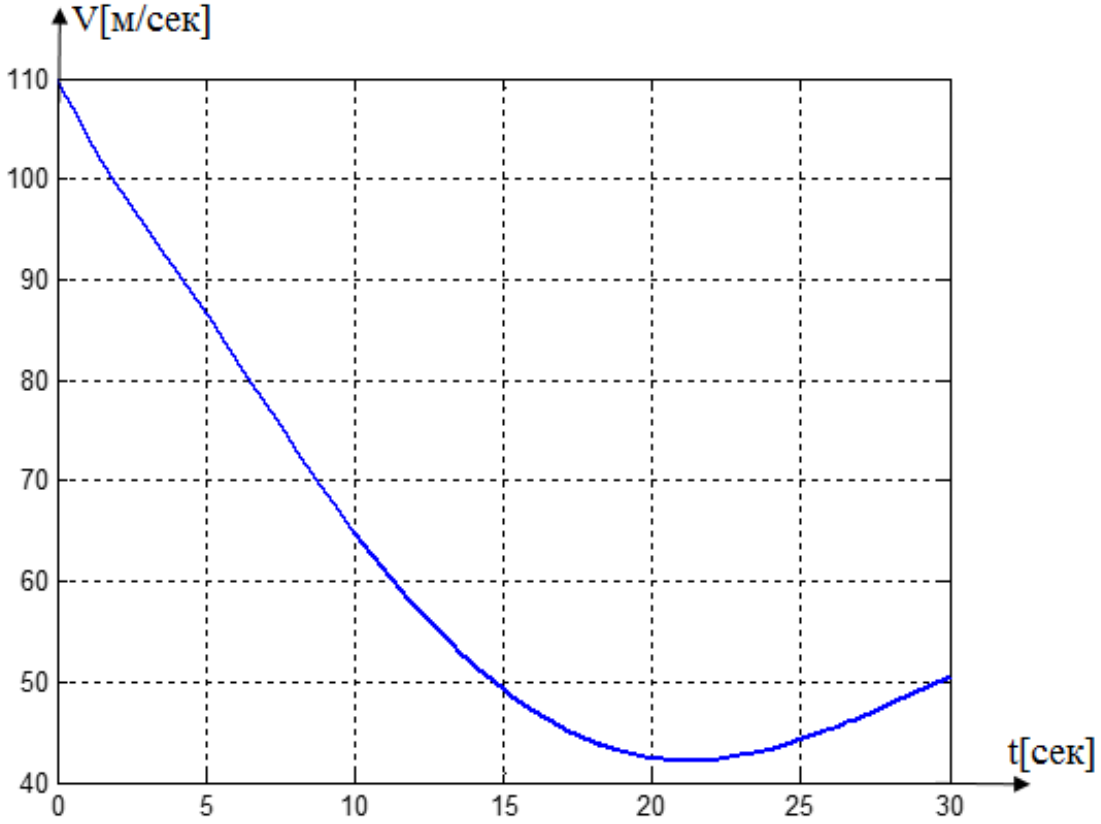


Рис 9.10. Процесс изменения скорости в тяжелом режиме полета впередилетящего судна

Рассмотрим для этого режима два случая - при отсутствии координации попутного движения и при активных действиях.

В случае $F_0 \to \infty$ активные действия исключены, т.к. $U_1=0$. В этом случае траектории движения судов, показанные на рис 9.11, начинают постепенно отличаться, и вместо безопасной начальной дистанции D=6000м возникает разница (l_1- x_1), отличающаяся от D на 1200 метров.

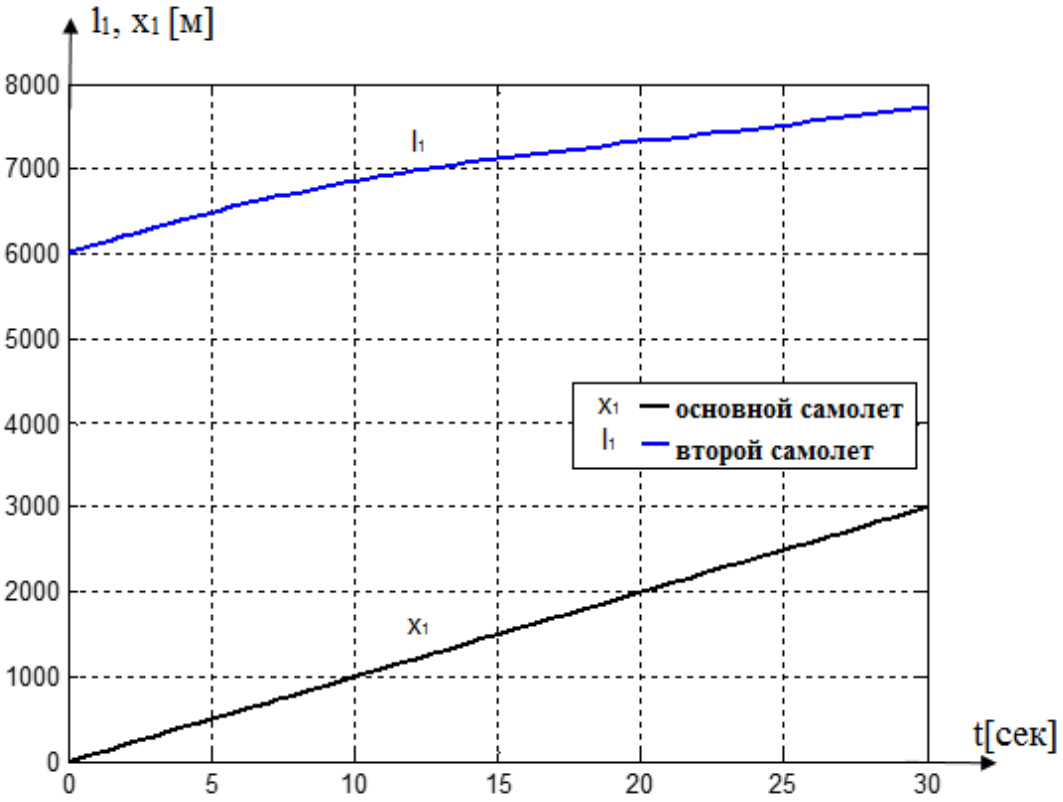


Рис 9.11. Траектории попутного движения судов без активных действий

При этом функция риска F резко возрастает, начиная с пятнадцатой секунды и явно является сигналом тревоги, как показано на рис. 9.12. В случае назначения выбранного порога $F_0=10^4$ начинаются активные действия по координации попутного движения, т.к. функция риска F начинает вырабатывать сигнал предупредительной тревоги при $F>F_0$, как это представлено на рис 9.12.

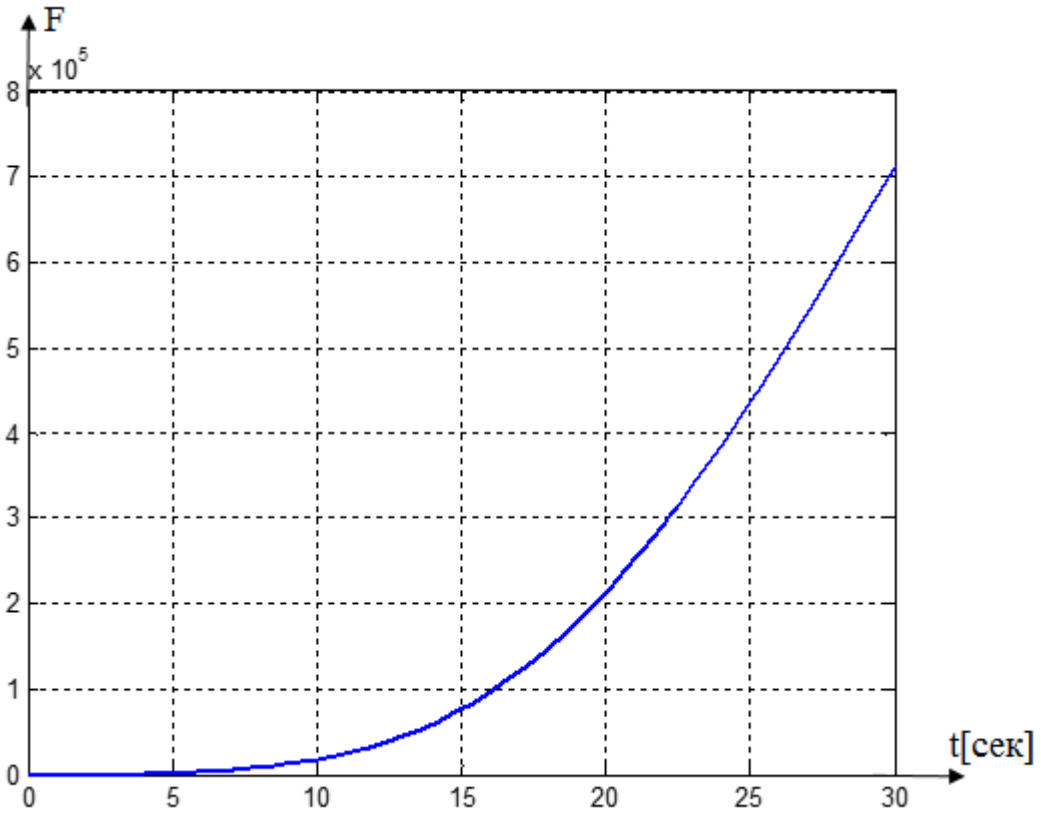


Рис 9.12. Функция риска опасного сближения судов без использования активных действий

Из рис 9.12 видно, что активные действия предпринимаются в течение примерно 40% общего времени, а “пиковые” значения тревоги, когда функция F достигает максимумов, коррелированы с максимальным спадом скорости впередилетящего судна, если обратить более пристальное внимание на график рис 9.10.

Траектории движения двух судов при их координированном управлении представлены на рис9.13, и разница (l_1- x_1) по сравнению с D=6000м отличается на незначительную величину не более 175 м.

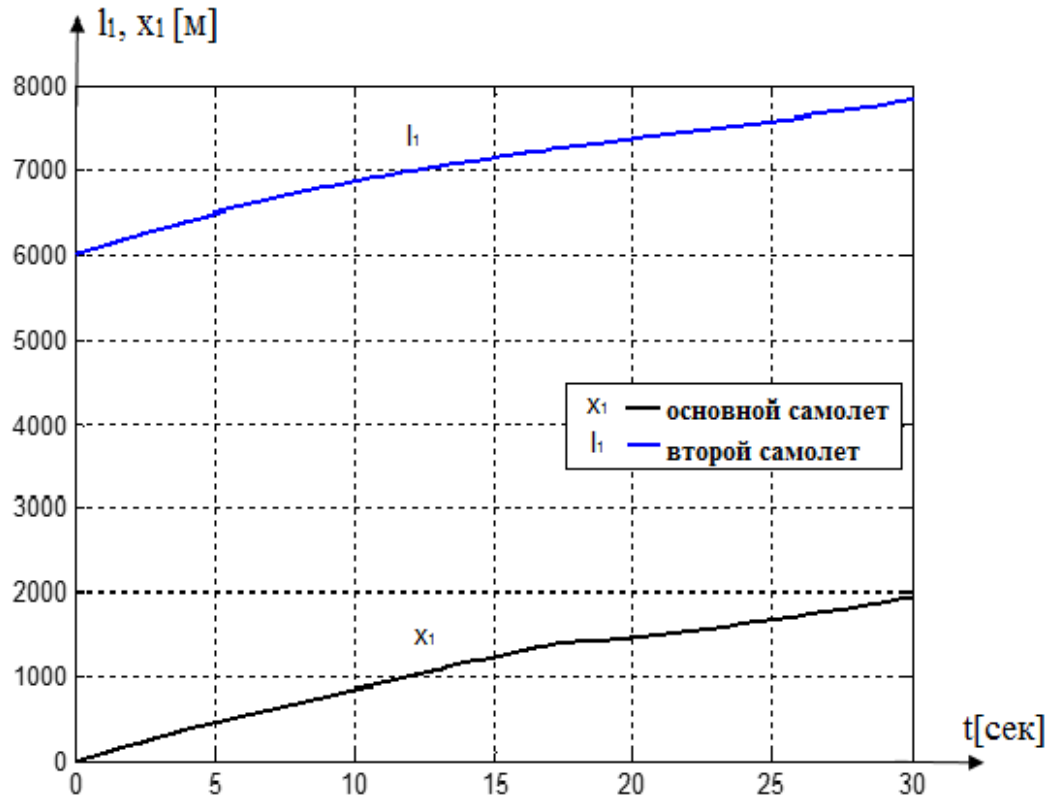


Рис 9.13. Траектории попутного движения судов при контроле безопасности и управлении сзадилетящим судом

Во втором аварийном режиме полетная ситуация у впередилетящего судна еще более усугубляется – скорость замедляется в среднем в три раза при средней продольной перегрузке торможения, примерно равной 0,4g, как это показано на рис 9.14.

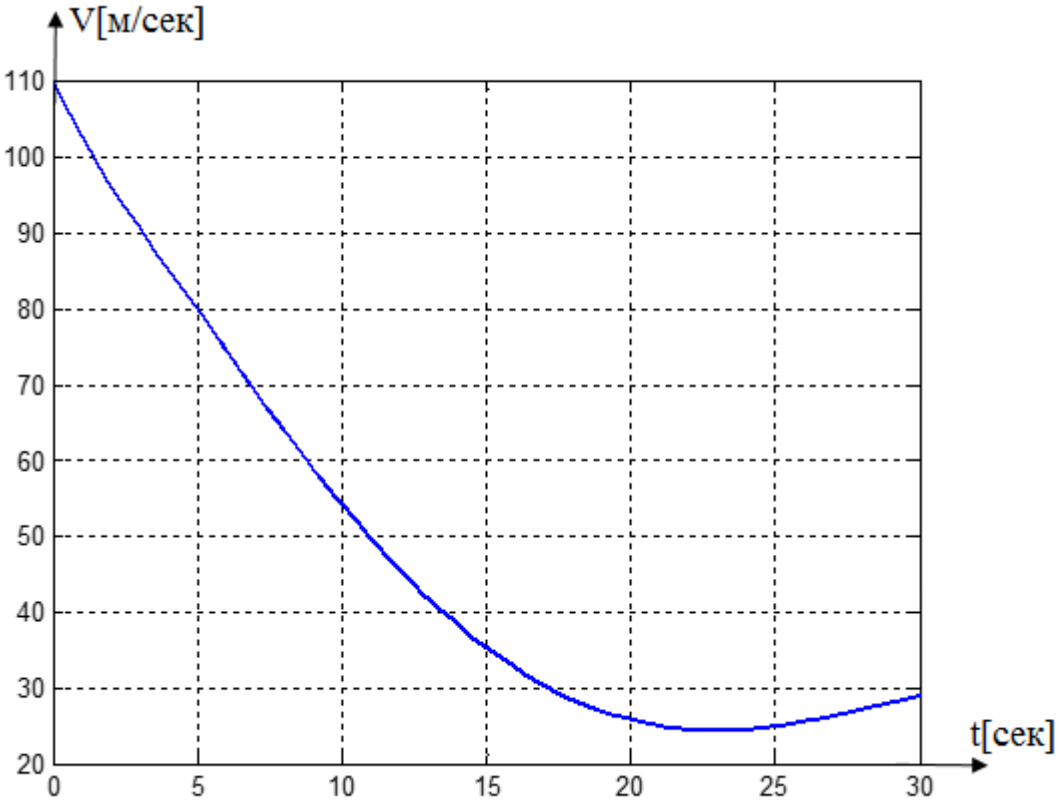


Рис 9.14. Процесс изменения скорости в аварийном режиме полета впередилетящего судна

Рассмотрим также два случая – при отсутствии активных действий при U_1=0, и также при ненулевом управлении скоростью сзадилетящего судна в промежутки времени, когда функция риска$F>F_0$.

При $F_0\to\infty$, т.е. при $U_1 = 0$ траектории движения двух судов показаны на рис 9.15. Видно, что эти траектории отличаются при t>20сек от траекторий на рис.9.11, и вместо дистанции D=6000м в конце процесса разница (l_1- x_1) отличается на величину порядка 1500 м.

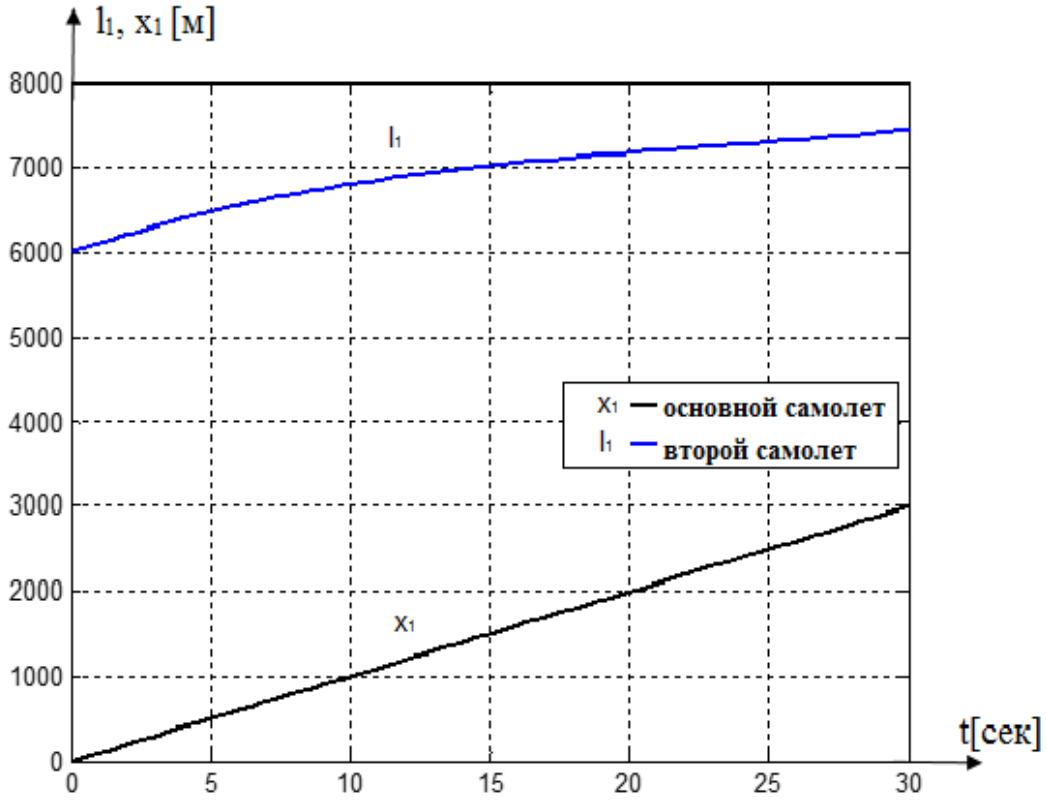


Рис 9.15. Траектории попутного движения двух судов в аварийном режиме полета без использования активных действий

При этом функция риска F становится сигналом аварийной тревоги, уже начиная сt=10 сек, как это показано на рис 9.16.

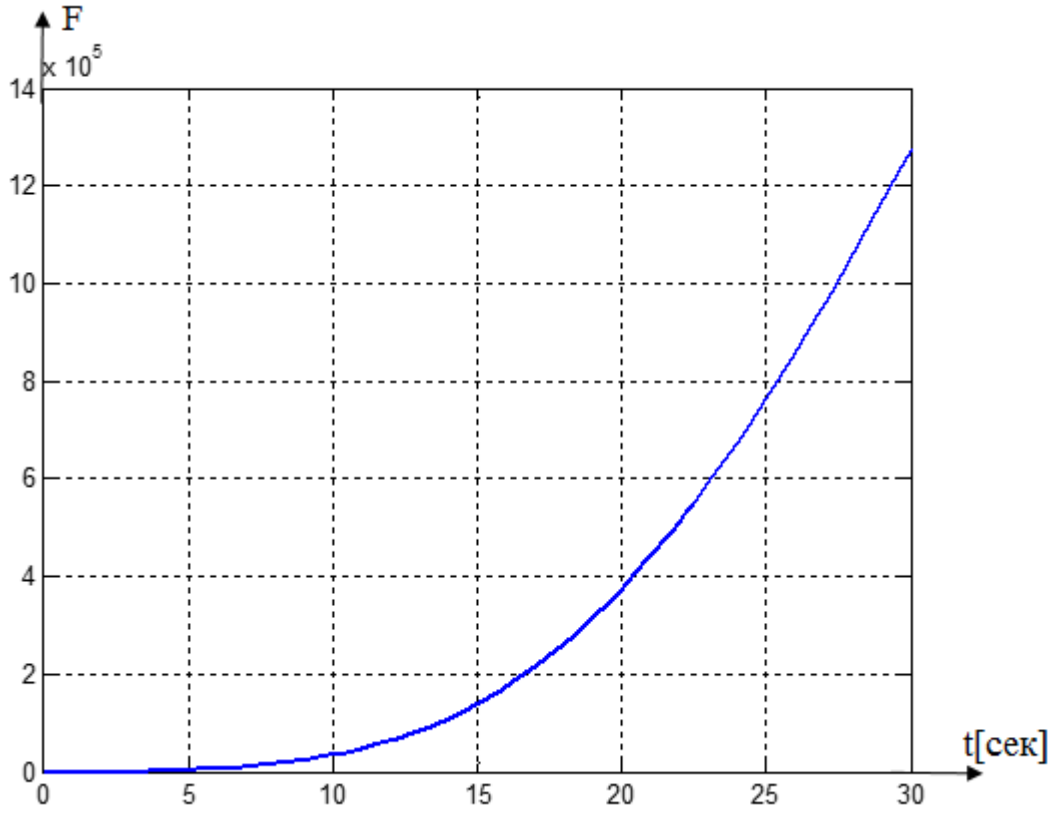


Рис 9.16. Функция риска опасного сближения судов в аварийном режиме полета без использования активных действий

В случае использования назначенного порога $F_0=10^4$ и координированного управления сзадилетящим судном в нужные интервалы времени, траектории попутного движения судов становятся лучше, чем при $U_1=0$, т.к. разница в дистанции между судами при t = 30сек по сравнению с требуемой составляет примерно 70 м. Эти траектории показаны на рис 9.17.

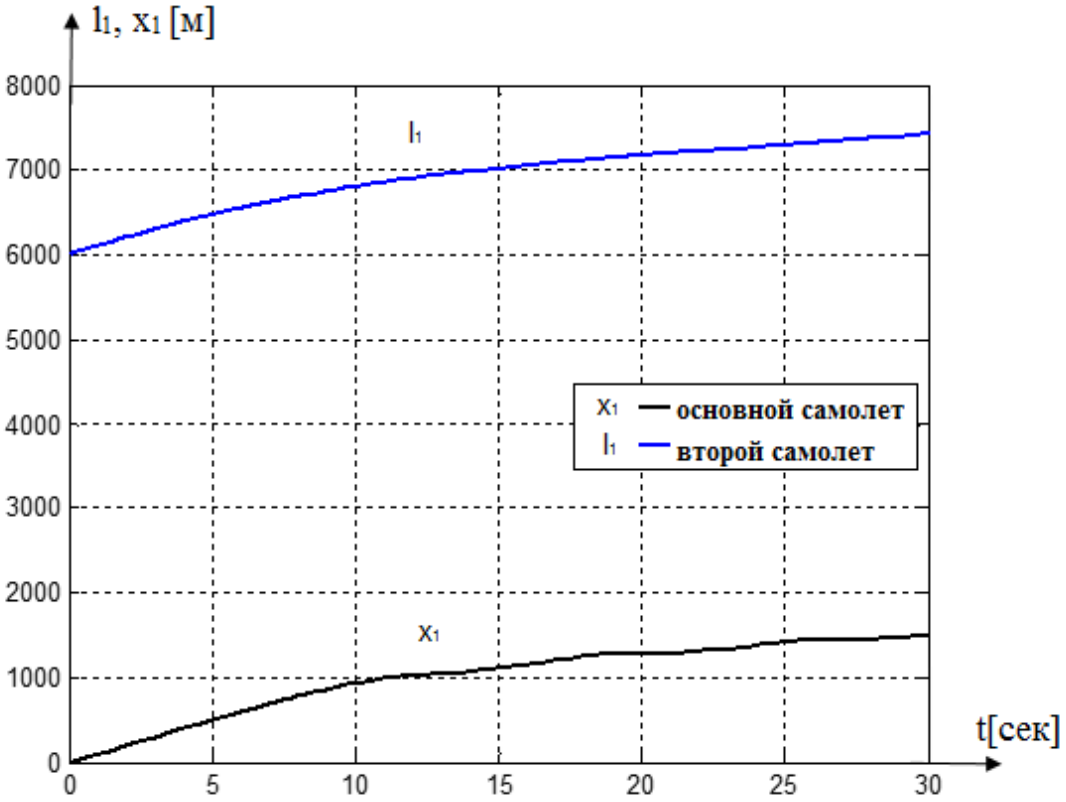


Рис 9.17. Траектории попутного движения судов при контроле безопасности полета в аварийном режиме и использовании активных действий

Достигнутое улучшение обусловлено тем, что время активных действий сзадилетящего судна существенно увеличилось. Как видно из рис 9.18, это время достигло примерно 60% от общего времени полета, что указывает на пользу одновременного контроля безопасности полета и координированного управления движением сзадилетящего судна.

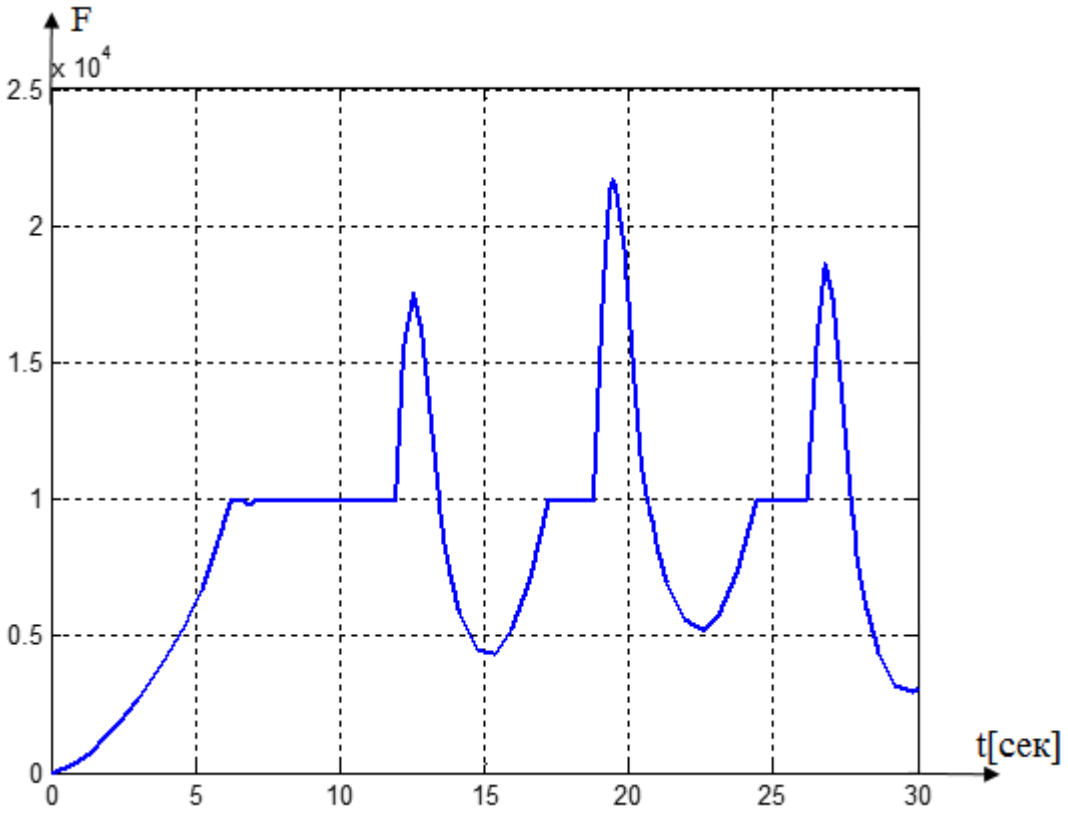


Рис 9.18. Функция риска опасного сближения судов в аварийном режиме полета при использовании активных действий

Таким образом, полученные результаты моделирования соответствуют физическому смыслу предлагаемых активных действий и подтвердили эффективность предложенного подхода при попутном движении судов.

9.5 Выводы по главе 9

На основании проведенных в данной главе исследований можно сделать следующие выводы.

1. Моделирование алгоритмов назначения посадочных курсов в зависимости от ветра и вычисления динамических приоритетов воздушных судов для каждой из трасс подтвердило правильность их работы и возможность использования в Московском аэроузле.
2. Моделирование двухуровневой системы контроля и оптимального управления попутным движением воздушных судов показало, что в аварийной полетной ситуации ее использование обеспечивает соблюдение нужной дистанции между ними в воздушном эшелоне.
3. Результаты внедрения предложенного подхода, отмеченные российской компанией «Трансаэро» указывают, что разработанные алгоритмы и программы позволяют быстро перераспределять воздушные судна в Московском аэроуле при гарантированной безопасности и повышенной экономичности полетов при заходе на посадку.
4. Результаты внедрения найденного способа приоритетного обслуживания судов в учебный процесс в МАИ и МГТУ ГА подчеркивают способность одновременного учета пространственного и технического состояния судов, что практически важно.

Заключение

На основании проведенных исследований можно сделать следующие выводы.

1. Предложен единый параметрический критерий экономичности и безопасности управления полета, в котором весовые коэффициенты их значимости найдены с помощью решения обратной задачи линейного программирования.

2. Сформирован алгоритм назначения динамических приоритетов для воздушных судов при их заходе на заданную трассу, учитывающий не только их близость к трассе, но и оставшийся запас топлива и их техническое состояние.
3. Найденные формулы приоритетного распределения судов между трассами удобны для расчетов в реальном времени и позволяют управлять потоком прилетающих судов, идущих на посадку и попадающих в очередь в соответствующий тромбон.
4. Показано, что при бесприоритетном обслуживании самолетов без учета оставшегося на борту топлива, нужная малая вероятность отказа при заходе в заданный эшелон воздушного движения требует использования тромбона значительного размера, который ограничен другими причинами. При приоритетном обслуживании вероятность отказа и отправки на другой аэродром аварийных самолетов с малым запасом топлива в 10 раз меньше по сравнению с бесприоритетной системой обслуживания.

5. Предложен алгоритм определения первоочередности прилета в аэропорт для случайно расположенных в пространстве воздушных судов с выходом на стандартный маршрут прилета.
6. Получена объединенная двухуровневая структура контроля и управления полетом, обеспечивающая при использовании координат

относительного движения судов необходимую безопасность полета в эшелоне при автоматической подсказке летчику или в диспетчерской наземной службе.

7. Получены новые формулы для вероятного состояния системы приоритетного обслуживания, которые в отличие от формул Эрланга позволяют определить отдельно вероятности отказа в бесприоритетном и приоритетном обслуживании. Показан существенный выигрыш во времени обслуживания пассажиров после аварийного прилета.
8. Разработанные в данной работе программы на ЭВМ могут быть использованы для частичного решения проблемы загруженности диспетчеров в Московском аэроузле и других аэропортах, что подтверждается актами о внедрении.
9. Полученные результаты использованы при проведении учебного процесса на кафедре 301 МАИ по дисциплине «Эргатические системы управления» в рамках магистерской подготовки по направлению 220400 «Управление в технических системах» по программе «Управление и информационные технологии в технических системах», а также внедрены в тренажерном центре авиадиспетской службы в МГТУ ГА, о чем имеются акты о внедрении.

Список использованных источников

1. Малыгин Б. В.,Нечаев Е.Е., “Обеспечение безопасности полётов при управлении воздушным движением”, М., МГТУ ГА, 2011г , 88 с.

2. Малыгин Б. В., “Схемы «Тромбон» и «Веер» - альтернатива векторению при формировании очереди на посадку на аэродромах московской воздушной зоны” , Научный вестник МГТУ ГА, №160, М., МГТУ ГА, 2010, стр5.

3. Малыгин Б. В., “Принципы оценки факторов риска в области управления технологической безопасностью при УВД”, МГТУ ГА, №159, М., МГТУ ГА, 2010, стр.81-85.

4. Коновалов А. Е., Юркин Ю. А., «Средства поддержки принятия решения диспетчерами УВД, Научный вестник МГТУ ГА, №198(12), М., МГТУ ГА, 2013, стр 118-124.

5. Михалев И.А., Окоемов Б.Н., Павлина И.Г., Чикулаев М.С., Кисилев Ю.Ф. «Системы автоматического и директорного управления самолетом». - М.: Машиностроение, 1974, с.3.

6. Е. К. Щербаков, В. З. Епишкин«Разработка, анализ вариантов совершенствования и выбор оптимальной структуры воздушного пространства московской воздушной зоны и московского района ЕС ОрВД (5-ый этап) книга 1/2, г. М., ФГУП ГосНИИ «Аэронавигация», 2011г.

7. Горбачев Ю.В., Тин Пхон Чжо, Рыбников С.И., Степаньянц Г.А. «Назначение динамических приоритетов при обслуживании самолетов с произвольном курсом во время захода на посадку и полета в строю» г. М., Труды МАИ, № 49, 2011.

8. Ю.С. Гришанин, Г.Н Лебедев, А.В Липатов, Г.А Степаньянц “Теория оптимальных систем ”– изд-во МАИ 1999. -320 с.: ил.

9. Солодухин В.А. Обратная задача оптимизации и объективизация эффективности принятия решений в системе УВД. Методы и модели анализа процессов УВД. Л.:АГА,1981.С.27-30.

10. И.М Соболь , Р. Б Ставников « Выбор оптимальных параметров в задачах со многими критериями »2-е изд-во перераб и доп - М: Дрофа , 2006-175.

11. Ким Д. П. Теория автоматического управления. Т. 2. Многомерные, нелинейные, оптимальные и адаптивные системы: Учеб. пособие. - М.: ФИЗМАТЛИТ, 2004. - 464 с.

12. А.В. Аттетков, С.В. Галкин, В.С. Зарубин. Методы оптимизации: Учеб. для вузов. - 2-е изд., стереотип. - М.: Изд-во МГТУ им. Н.Э. Баумана, 2003. - 440 с.

13. Зуховицкий С.И, Авдеева Л.И. Линейное выпуклое программирование.

изд-во « Наука» 1964.

14. Юдин Д Б., Гольштейн Е.Г Линейное программирование. Физматгиз.

1963.

15. Гасс С. Линейное программирование –М.: Физматгаз,1961.

16. Кюнци Г. П., Крелле В. Нелинейное программирование - М. : Сов. радио, 1965.

17. Атманов С.А. Линейное программирование. - М.: Наука, Физматгиз,1981.

18. Солодовников А. Введение в линейную алгебру и линейное

программирование. – М.: Просвещение, 1966.

19. Дегтярев Ю.И. Методы оптимизации: Учеб. Пособие для вузов – М.:

сов. Радио, 1980.- 272 с., ил.

20. Канторович Л.В. Экономический расчет наилучшего использования

ресурсов -М.: Изд-во АН СССР, 1960.

21. Болтянский В. Г, Математик и оптимальное управление.- М.: Знание 1968.

22. Понтрягин Л. С. и др., Математичекая теория оптимальных процессов.- М. ИИЛ, 1961.

23. Фельдбаум А. А. Основы теории оптимальных автоматических систем. - М.: Наука, 1966.

24. Александров А.Г. Оптимальные и адаптивные системы: Учеб. Пособие для вузов по спец. «Автоматика и упр. в техн. системах».- М: Высш.шк., 1989. – 263 с.: ил.

25. Иванов В.А., Фалдин Н.В. Теория оптимальных систем автоматического управления. – М.: Наука , 1981, 336 с.

26. Беллман Р. Динамическое программирование. Изд - во иностранной литературы, 1960.

27.Летов А. М., Динамика полета и управления. М., Наука, 1964.

28. Чаки Ф. Современная теория управления. Нелинейные, оптимальные адаптивные системы.- М.: Мир. 1975.

29. Тин Пхон Чжо «Определение относительной значимости экономичности и безопасности воздушного движения с помощью обратной задачи линейного программирования», г. М., Труды МАИ, № 78 , 2014.

30. Лебедев Г.Н. Методы принятия оперативных решений в задачах управления и контроля. - М.: Изд. МАИ, 1992. - 120 с.

31. Лебедев Г.Н., Тин Пхон Чжо, Чан Ван Туен «Решение задачи динамического программирования при безопасном попутном движении воздушных судов» г. М. Труды МАИ, № 54, 2012.

32. Тин Пхон Чжо «Проблема приоритетного обслуживания самолетов гражданской авиации при их заходе на посадку в воздушном пространстве Московского аэроузла» // Труды XXI международный нучно-технический семинар “Современные технологии в задачах управления, автоматики и

обработки информации” 18-25 сентября, 2012 г., г. Алушта. М., МГУ. Сборник тезисов докладов, С.27.

33. Тин Пхон Чжо, Зо Мин Тайк «Проблемы обслуживания самолетов гражданской авиации при заходе на посадку и пассажиров в аэропорту после прилета при ограничении очередей» // Труды XXI международный нучно-технический семинар “Современные технологии в задачах управления, автоматики и обработки информации” 18-24 сентября, 2013 г., г. Алушта .М., МГУ. Сборник тезисов докладов, С.22.

34. Малыгин В.В., Тин Пхон Чжо «Задача бесприоритетного обслуживания самолётов при их попадании в тромбон во время захода на посадку» , г. М., №198(12) 2013, Научный вестник МГТУ ГА, Стр : 37-40.

35. Тин Пхон Чжо «Автоматизированная система управления и контроля безопасности попутного и поперечного движения группы воздушных судов при заходе на посадку», г. Пенза, Вестник Пензенского государственного университета, №1 , 2014, Стр 72 - 80.

36. Лебедев Г.Н., Тин Пхон Чжо, Зо Мин Тайк, Дао Нгок Тхай «Автоматизированная система управления безопасном движением наземного и воздушного транспорта при их сближении» г. Орел.,ОГУ, “Фундаментальные и прикладные проблемы техники и технологии”, №1 Январь-февраль (303) 2014, Стр: 109-115.

37. Тин Пхон Чжо «Автоматизация оперативного распределения воздушных судов между трассами захода на посадку в Московском аэроуле при внезапном изменении метеоусловий», г. М., № 3, 2014, Научный вестник МАИ, Стр:128 -140.

38. Лебедев Г.Н., Чан Ван Туен, Ву Суан Хьюнг, «Контроль и управление безопасным движением транспорта при встречном движении». - Мехатроника, автоматизация, управление , №8, 2011, стр. 56-61.

39. Лебедев Г. Н., Тин Пхон Чжо, Зо Мин Тайк, Хахулин Г. Ф., Малыгин В. Б.“Оптимальное управление и контроль безопасности поперечного движения речных и воздушных судов при пересечении их маршрутов”, М., “Новые технологии”, “Мехатроника, автоматизация, управление”, 2012, №12, стр. 50-55.

40. Лебедев Г. Н., Зо Мин Тайк, “Синтез оптимального управления боковым движением воздушных или речных судов при пересечении их маршрутов под произвольным углом”, М., “Новые технологии”, “Мехатроника, автоматизация, управление”, 2014, №5, стр. 61-68.

41. Винер Н. Кибернетика.- М.: Сов. Радио, 1968.

42. Чураков Е. П. Оптимальные и адаптивные системы.- М.: Энергоатомиздат, 1987. - 256 с.

43. Кузин Л. Т. Основы кибернетики.— Т. 1 и 2. — М.; Энергия, 1973.

44. Лебедев Г. Н., Тин Пхон Чжо, Зо Мин Тайк «Интегрированная система управления и контроля безопасности попутного движения воздушных или речных судов» // V Международная научно-техническая конференция «Информационные технологии в науке, образовании и производстве», 17-18 мая, 2012 г., г. Орел., ИТНОП (информационные технологии в науке, образовании и производстве. С. 25.

45. Лебедев Г. Н., Тин Пхон Чжо «Система управления безопасным движением транспортных средств при их сближении»// Всероссийская научно-техническая конференция “Проблемы и решения построения систем ориентации, навигации и управления подвижным объектами” 3-5 октября 2012 г. Тула ТулГУ, С.10.

46. Тин Пхон Чжо «Управление и контроль безопасности движения самолетов при входе заданный воздушный эшелон посадки» // XXIX Международная научно-техническая конференция «Проблемы

автоматизации в технических системах», 23-25 апреля, 2013 г., г. Пенза., Изд-во: ПГУ 2013г. С. 59-62.

47. Горбачев Ю.В., Лебедев Г. Н., Тин Пхон Чжо «Система приоритетного обслуживания самолетов гражданской авиации при их заходе на посадку по заданной линии пути» » г. М., Труды МАИ, № 49, 2012.

48. Лебедев Г.Н., Тин Пхон Чжо, Зо Мин Тайк, «Система обеспечения безопасности при попутном движении воздушных или речных судов и пересечении их маршрутов» г. Тула., №7 2012, Известия Тульского государственного института, Стр: 246 – 254.

49. Лебедев Г.Н., Зо Мин Тайк, Тин Пхон Чжо, Медведев А. М., «Управление полетом пассажирских самолетов при пересечении их маршрутов во время захода на посадку» » г. М., Труды МАИ, № 63, 2013.

50. Тин Пхон Чжо, Зо Мин Тайк, Ву Суан Хыонг, «Автоматический контроль безопасности сближения двух управляемых воздушных судов при пересечении их маршрутов», г. М., № 198(12) 2013, Научный Вестник МГТУ ГА, Стр: 51-59.

51. Лебедев Г . Н, Тин Пхон Чжо, “ Оценка эффективности организации взаимопомощи в многоканальных авиакосмических системах”, «Мехатроника, автоматизация, управление , изд-во Новые технологии», стр 63-68, № 7, 2009 г.

52. Венцель Е.С. Исследование операций.М.,«Советское радио»Москва,1972.

53. Венцель Е.С. Теория вероятностей. Изд-во « Наука» 1969.

54. Гнеденко Б. В, Коваленко И. Н. Введение в теорию массового обслуживания. Изд-во « Наука » 1966.

55. Феллер, В. Введение в теорию вероятностей и ее приложения. Том. Пер. С англ. Под рад Е.Б Дынкина. М., Мир, 1964. 498 с.

56. Харрари Ф. Теория графов. – М.: Мир, 1973.

57. Тин Пхон Чжо «К вопросу о приоритетном обслуживании воздушных судов при заходе на посадку» // Труды XXIII международный нучно-технический семинар “Современные технологии в задачах управления, автоматики и обработки информации” 14-20 сентября, 2014 г., г. Алушта. М., МГУ. Сборник тезисов докладов, С. 21.

58. Тин Пхон Чжо., Гасанзаде К. И., Горбачев Ю.В. «Приоритетное обслуживание самолетов при заходе на посадку и пассажиров после их прилета» г. М., Труды МАИ, № 63, 2013.

59. Тин Пхон Чжо «Сравнение с помощью теории массового обслуживания бесприоритетной и приоритетной систем управления заходом самолетов на посадку» » г. М., Труды МАИ, № 63, 2013.

60. Тютрин А.В, Организация взаимопомощи между каналами в многоканальной системе контроля, М; Изд-МАИ, 1979.

61. Тин Пхон Чжо, «Обслуживание пассажирских самолетов при заходе на посадку и пассажиров в аэропорту с помощью методов оптимизации, Кандидатская диссертация». М., МАИ, 2010, с – 115.

62. Лебедев Г.Н., Малыгин В.Б., Нечаев Е.Е.,Тин Пхон Чжо «Использование системы приоритетного обслуживания при внедрении автоматизированного управления прилетом-вылетом в воздушном пространстве Московского аэроузла» г. М., №180(6) 2012, Научный вестник МГТУ ГА, Стр : 254-259.

63. Малыгин В.В., Тин Пхон Чжо, Турков А. Н., «Методика определения технологических возможностей диспетчера по управлению группой воздушных судов.», г. М., № 198(12) 2013, Научный вестник МГТУ ГА, Стр: 41-44.

64. Тин Пхон Чжо., Малыгин В.Б., Михайлин Д. А, «Система приоритетного обслуживания при внедрении автоматизированного управления прилетом в

воздушном пространстве Московского аэроузла», г. М., №198(12) 2013, Научный вестник МГТУ ГА, Стр : 45-50.

65. http://lliric.narod.ru/4/page_4.1.htm

66. http://aeromodel.narod.ru/posadka.html

67. http://life-prog.ru/1_988_printsip-maksimuma-lspontryagina-v-teorii-optimalnih-sistem.html

68. Dimitri P. Bertsekas., Dynamic programming and optimal control (third edition), ISBN 1-886529-86-4, Athena Scientific , Belmont, Massachusetts, 2005, 541 pages.

69. Dimitri P. Bertsekas, with Angelia Nedic and Asuman E. Ozdaglar, Convex Analysis and Optimization, ISBN 1-88652945-0, 2003, 560 pages.

70. Dimitri P. Bertsekas and John N. Tsitsiklis, Introduction to Probability, ISBN 1-886529-40-X, 2002, 430 pages.

71. Dimitri P. Bertsekas, Dynamic Programming and Optimal Control, Two-Volume Set, ISBN 1-886529-08-6, 2005, 840 pages

72. Dimitri P. Bertsekas and Steven E. Shreve, Stochastic Optirnal Control: The Discrete Time Case, ISBN 1-886529-03-5, 1996, 330 pages.

73. Dimitris Bertsimas and John N. Tsitsiklis, Introduction to Linear Optimization, ISBN 1-886529-19-1, 1997, 608 pages

74. Dimitri P. Bertsekas and John N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, ISBN 1-88652901- 9, 1997, 718 pages.

75. Dimitri P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, ISBN 1-88f1529-04-3, 1996, 410 pages.

76. George B. Dantzig, Linear programming and Extensions, R-366-PR, The Rand corporation, California, 1963, 611 pages.

77. George B. Dantzig and Mukund N. Thapa. *Linear programming 1: Introduction*. Springer-Verlag, 1997, 361 pages.

78. Richard W. Cottle, ed. *The Basic George B. Dantzig*. Stanford Business Books, Stanford University Press, Stanford, California, 2003, 400 pages.

79. Combinatorial Optimization: Polyhedra and Efficiency, Springer, Berlin, ISBN 3-540-44389-4, 2003, Pages 437-440.

80. Dimitri P. Bertsekas, Nonlinear Programming, 2nd Edition, ISBN 1-886529-00-0, 1999, 791 pages.

81. Yamada, K., "Diffusion Approximation for Open State-Dependent Queueing Networks in the Heavy Traffic Situation". The Annals of Applied Probability 5 (4): (1995), 958 pages.

82. Pierre Bremaud., Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues (Texts in Applied Mathematics), Springer, USA, 441 pages.

Приложение 1

Текстпрограммывыборапосадочныхкурсов

```
#include "LandingCourseAsu.h"
using namespace LandingAsu;
LandingCourseAsu::LandingCourseAsu()
{

}
LandingCourseAsu::~LandingCourseAsu()
{

}
void LandingCourseAsu::Initialize(double windDir, double windSpeed, double
maxWindSpeed)
{
	for (int i = 0; i < sizeof(landingCourseAngles)/sizeof(double); i++)
	{
		landingCourses.append(landingCourseAngles[i]);
	}
	windDirection = windDir;
	this->windSpeed = windSpeed;
	this->maxWindSpeed = maxWindSpeed;
}
voidLandingCourseAsu::MakeTable1()
{

	//////////////////////////////////////////////////////////////////
	// Создаем динамический массив-таблицу для возможных курсов
подлета //
```

```
	//////////////////////////////////////////////////////////////////
	TableOne = new double*[landingCourses.size()];

	for (int i = 0; i < landingCourses.size(); i++)
	{
		TableOne[i] = new double[landingCourses.size()];
	}

	// Формируем таблицу
	for ( int i = 0; i < landingCourses.size(); i++ )
	{
		for ( int j = 0; j < landingCourses.size(); j++ )
		{
			//////////////////////////////////////////////////////////////////////////
			// Предварительный расчет потерь топлива как угол между посадочными курсами ВПП //
			//////////////////////////////////////////////////////////////////////////
			double dAng = abs(landingCourses[i] - landingCourses[j]);

			if(dAng > 180.0)
				dAng = 360.0 - dAng;

			TableOne[i][j] = dAng;
		}
	}
}

voidLandingCourseAsu::MakeTable2()
{
	//////////////////////////////////////////////////////////////////////
```

```
        // Создаем динамический массив-таблицу для посадочных курсов с учетом ветра //
        ////////////////////////////////////////////////////////////////////////////
        tableTwo = new double*[landingCourses.size() / 2];

        for (int i = 0; i < landingCourses.size() / 2; i++)
        {
                tableTwo[i] = new double[2];

                if(landingCourses[i] > windDirection - 90.0 &&
                  landingCourses[i] < windDirection + 90.0)
                {
                        tableTwo[i][0] = i + landingCourses.size() / 2;
                        tableTwo[i][1] = landingCourses[i + landingCourses.size() / 2];
                }
                else
                {
                        tableTwo[i][0] = i;
                        tableTwo[i][1] = landingCourses[i];
                }

        }
}

voidLandingCourseAsu::MakeTable3()
{
        ////////////////////////////////////////////////////////////////////////////
        // Создаем динамический массив-таблицу для оценки расходов как длину дуги окружности //
        ////////////////////////////////////////////////////////////////////////////
```

```
        tableThree = new double*[landingCourses.size() / 2];

        for (int i = 0; i < landingCourses.size() / 2; i++)
        {
                tableThree[i] = new double[landingCourses.size() + 1];

                tableThree[i][0] = tableTwo[i][0];

                for (int j  = 1; j < landingCourses.size() + 1; j++)
                {
                        int M = (int)tableTwo[i][0];
                        tableThree[i][j] = TO_RAD(TableOne[M][j - 1]) *
crossAreaRadius;
                }
        }


        //////////////////////////////////////////
        // Блок поощрения посадки на свой аэродром //
        //////////////////////////////////////////
        for (int j  = 1; j < landingCourses.size() + 1; j++)
        {
                for (int i = 0; i < landingCourses.size() / 2; i++)
                {
                        if( tableThree[i][0] == j ||
                                tableThree[i][0] == (j + landingCourses.size() / 2) )
                        {
                                tableThree[i][j] = tableThree[i][j] / tableThree[i][0];
                        }
```

```
            }
        }

}

void LandingCourseAsu::MakeTable4()
{
        tableFour = new double*[landingCourses.size()];

        for (int i = 0; i < landingCourses.size(); i++)
        {
                tableFour[i]    = new double[2];
                tableFour[i][0] = i;

        }

        ////////////////////////////////////////////////////////////////////
        // Определяем оптимальный курс посадки для каждой группы самолетов //
        ////////////////////////////////////////////////////////////////////

        for (int j = 1; j < landingCourses.size() + 1; j++)
        {
                double Min  = tableThree[0][j];
                double Mmin = 0;

                for (int i = 0; i < landingCourses.size() / 2; i++)
                {
                        if(tableThree[i][j] < Min)
                        {
```

```
                    Min = tableThree[i][j];
                    Mmin = tableThree[i][0];
                }
            }

            tableFour[j - 1][1] = Mmin;
        }
    }
```

Приложение 2

Текст программыназначения динамических приоритетов воздушных судов

MAIN

```
// Main_Diplom.cpp: определяет точку входа для консольного приложения.
//
#include"stdafx.h"
#include<iostream>
#include<cmath>
#include<clocale>
#include<Windows.h>
#include<conio.h>
#include"ASU.h"

usingnamespace std;

int N_s = 4; //количество трас
intK_s = 20; //количество самолетов
doublewD_s = 20; //направление ветра
double Teta_s[4] = {136, 14, 238, 66}; // углы трасс
double Ksi_s[20] = {0, 100, 200, 280, 200, 200, 60, -40, -120, -220, -240, -280, -200, -140, -60, 240, 240, 320, 160, 160};
double Z_s[20] = {-300, -250, -150, -40, 60, 220, 280, 260, 220, 200, 100, -20, -180, -260, -260, -50, -80, 260, 310, 260};
```

```
double Psi_s[20] = {100, 40, 316, 120, 140, 225, 270, 260,  280, 10, 0, 310, 40, 90, 85, 165, 165, 242 , 238, 250};
double y4_s[20] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.89, 0, 0}; //топливо
double H_s[20] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};  //высота

int main()
{
	setlocale(LC_CTYPE, "rus");
	ASU MAU = ASU(N_s, K_s);
	MAU.Vvod(wD_s, Ksi_s, Z_s, Psi_s, y4_s, H_s); //ввод данных
	MAU.Veter(Teta_s); //Определение посадочных курсов
	//MAU.X1value(Ksi_s,Z_s,Psi_s);//X1 value
	//MAU.X3value(Ksi_s,Z_s,Psi_s);//X3 value
	//MAU.Y1value(K_s,N_s);//Y1 value
	//MAU.KoordKsi(Ksi_s,Z_s,Psi_s);//Ksi[k] value
	//MAU.KoordPsi(Ksi_s,Z_s,Psi_s);//Psi[k] value
	//MAU.KoordZ(Ksi_s,Z_s,Psi_s);//Z[k] value
	//Расчет без учета x2
	for (int j = 0; j < MAU.N ; j++)
	{
		MAU.Koord(j, Ksi_s, Z_s, Psi_s); //пересчет координат
		for (int k = 0; k < MAU.K; k++)
			MAU.Ras4(k, j); //расчет x1, x3, m, l, L
```

```
            for (int k = 0; k < MAU.K; k++)
            {
                MAU.If_For_Prior(k); //определение по какой формуле расчитывается приоритет
                MAU.Prior0(k, j); //расчет приоритета
            }
        }
        MAU.Table0();  //Формирование списков

        //расчет с учетом x2
        //cout << "----------------------------------------\n\n \t\t\t  <X2> Value \n\n";
        for (int j = 0; j < MAU.N ; j++)
        {
            //cout << "\tВПП " << j+1 <<"\t";
            //cout << "\n\n";
            MAU.Koord(j, Ksi_s, Z_s, Psi_s);  //пересчет координат
            for (int k = 0; k < MAU.K; k++)
                MAU.Ras4(k, j);  //расчет x1, x3, m, l, L
            for (int k = 0; k < MAU.K; k++)
            {
                MAU.If_For_Prior(k);  //определение по какой формуле расчитывается приоритет
                //cout<<j+1<<endl<<endl;
```

```
                MAU.X2(k);  //расчет x2


                MAU.Prior(k, j); //расчет приоритета
            }
        }
        MAU.Table();  //Формирование списков


        MAU.Show_Tables();  //вывод на экран


        system("pause");
        return 0;
}
```

Листинг объявления класса ASU.h:

ASU.h

```
#ifndef ASU_H_
#define ASU_H_
class ASU
{
private:
        int R;      //дляусловия
        int r;
        int D;
        double windDir;
        double* Kursi;
```

```
double* Teta;
double* Ksi; //дляусловия
double* Z;
double* Psi;
double* y4;
double* H;
double** PRIOR0;
double** PRIOR;
int** TABLE0;
int** TABLE;
double* x1;
double* x3; //дляусловия
double* m;  //дляусловия
double* l;
double* L;
double** M;
double* x2;
double* y1;
double* y2;
int ch;

void Minimum(double* A, int n, double* B);
void show1(double* A, int n);
void show1(int* A, int n);
```

```
    void show2(double** A, int n1, int n2);
public:
    int N; //дляцикла
    int K; //дляцикла

    ASU(int NN, int KK);
    ~ASU();
    void Vvod(double& wD, double* KSI, double* ZK, double* PSI, double*
Y4, double* HH);
    void Veter(double* TETA);
    void Koord(int j, double* KSI, double* ZK, double* PSI);
    void KoordKsi(double* KSI, double* ZK, double* PSI);//ksi[k] value
    void KoordZ(double* KSI, double* ZK, double* PSI);//Z[k] value
    void KoordPsi(double* KSI, double* ZK, double* PSI);//Psi[k] value
    void RasX1(int k,int j);//X1 value
    void RasX3(int k,int j);//X3 value
    void Y1value(int k,int j);//Y1 value
    void Ras4(int k, int j);
    void If_For_Prior(int k);
    void X2(int k);
    void Prior0(int k, int j);
    void Prior(int k, int j);
    void Table0();
    void Table();
```

```
        void Show_Details(int j);
        void Show_Tables();
};
#endif
```

Листингметодовкласса ASU.cpp:

ASU.CPP

```
#include"stdafx.h"
#include"ASU.h"
#include<iostream>
#include<clocale>
#define _USE_MATH_DEFINES
#include<math.h>

usingnamespace std;

struct AA

{
        int num,kypc;
        double prio;

};

ASU::ASU(int NN, int KK)
```

```
{
        N = NN;
        K = KK;
        Kursi = newdouble[2*N];
        Teta = newdouble[N];
        Ksi = newdouble[K];
        Z = newdouble[K];
        Psi = newdouble[K];
        y4 = newdouble[K];
        H = newdouble[K];

        PRIOR0 = newdouble*[N];
        for (int i = 0; i < N; i++)
                PRIOR0[i] = newdouble[K];
        TABLE0 = newdouble*[N];
        for (int i = 0; i < N; i++)
                TABLE0[i] = newdouble[K];
        for (int i = 0; i < N; i++)
                for (int j = 0; j < K; j++)
                        TABLE0[i][j] = 0;
        PRIOR = newdouble*[N];
        for (int i = 0; i < N; i++)
                PRIOR[i] = newdouble[K];
        TABLE = newdouble*[N];
```

```
        for (int i = 0; i < N; i++)
                TABLE[i] = newdouble[K];
        for (int i = 0; i < N; i++)
                for (int j = 0; j < K; j++)
                        TABLE[i][j] = 0;


        x1 = newdouble[K];
        m = newdouble[K];
        l = newdouble[K];
        L = newdouble[K];
        x2 = newdouble[K];
        x3 = newdouble[K];
        y1 = newdouble[K];//y1
        y2 = newdouble[K];//y2
        M = newdouble*[K];
        for (int i = 0; i < K; i++)
                M[i] = newdouble[K];
}


ASU::~ASU()
{
        delete [] Kursi;
        delete [] Teta;
        delete [] Ksi;
```

```
delete [] Z;
delete [] Psi;
delete [] y4;
delete [] H;

for (int i = 0; i < N; i++)
        delete [] PRIOR0[i];
delete [] PRIOR0;
for (int i = 0; i < N; i++)
        delete [] TABLE0[i];
delete [] TABLE0;
for (int i = 0; i < N; i++)
        delete [] PRIOR[i];
delete [] PRIOR;
for (int i = 0; i < N; i++)
        delete [] TABLE[i];
delete [] TABLE;
delete [] x1;
delete [] m;
delete [] l;
delete [] L;
delete [] x2;
delete [] x3;
delete [] y1;
```

```
        for (int i = 0; i < K; i++)
                delete [] M[i];
        delete [] M;
}

void ASU::Vvod(double& wD, double* KSI, double* ZK, double* PSI, double*
Y4, double* HH)
{
        windDir = wD;
        for (int k = 0; k < K; k++)
        {
                Ksi[k] = KSI[k];
                Z[k] = ZK[k];
                Psi[k] = PSI[k];
                y4[k] = Y4[k];
                H[k] = HH[k];
        }
        R = 10;
        r = 6;
        D = 200;

}
```

```
void ASU::Veter(double* TETA)
{
	for (int j = 0; j < N; j++)
		if (TETA[j] >= 180)
		{
			Kursi[j] = TETA[j] - 180;
			Kursi[j+N] = TETA[j];
		}
		else
		{
			Kursi[j] = TETA[j];
			Kursi[j+N] = TETA[j] + 180;
		}

	double delPsi;
	for (int j = 0; j < N; j++)
	{
		delPsi = abs(Kursi[j] - windDir);
		if (delPsi > 180)
			delPsi = 360 - delPsi;

		if (delPsi > 90)
			Teta[j] = Kursi[j];
		else
```

```
                Teta[j] = Kursi[j+N];
        }
}

void ASU::Koord(int j, double* KSI, double* ZK, double* PSI)
{
        for (int k = 0; k < K; k++)
        {
                Ksi[k] = KSI[k]*cos(Teta[j]*M_PI/180) +
ZK[k]*sin(Teta[j]*M_PI/180);
                Z[k] = -KSI[k]*sin(Teta[j]*M_PI/180) +
ZK[k]*cos(Teta[j]*M_PI/180);
                Psi[k] = PSI[k] - Teta[j];

        }
}
void ASU::KoordKsi(double* KSI, double* ZK, double* PSI)//For Ksi[k] value
{
        cout <<"----------------------------------------\n\n \t\t\t  <Ksi> Value \n\n";

        for (int i = 0; i < N; i++)
                cout <<"\tВПП "<< i+1 <<"\t";
        cout <<"\n\n";
        for (int k = 0; k < K; k++)
```

```
        {
                cout <<"ЛА "<< k+1 <<"\t";
                for (int j = 0; j < N; j++)
                {
                        Ksi[k] = KSI[k]*cos(Teta[j]*M_PI/180) +
ZK[k]*sin(Teta[j]*M_PI/180);
                        cout<<Ksi[k]<<" \t";
                }
                cout <<"\n\n";
        }
        cout <<"\n\n\n\n";

}

void ASU::KoordZ(double* KSI, double* ZK, double* PSI)//For Z[k] value
{
        cout <<"-----------------------------------------------\n\n \t\t\t  <Z> Value \n\n";

        for (int i = 0; i < N; i++)
                cout <<"\tВПП "<< i+1 <<"\t";
        cout <<"\n\n";
        for (int k = 0; k < K; k++)
        {
                cout <<"ЛА "<< k+1 <<"\t";
```

```
            for (int j = 0; j < N; j++)
            {
                Z[k] = -KSI[k]*sin(Teta[j]*M_PI/180) +
ZK[k]*cos(Teta[j]*M_PI/180);
                if(k==3 || j==2)
                {
                cout<<Z[k]<<"    \t ";
                }
                else
                    cout<<Z[k]<<"     \t";
            }
            cout <<"\n\n";
        }
        cout <<"\n\n\n\n";

}

void ASU::KoordPsi(double* KSI, double* ZK, double* PSI)//For Psi[k] value
{
        cout <<"-----------------------------------------------\n\n \t\t\t  <Psi> Value
\n\n";

        for (int i = 0; i < N; i++)
            cout <<"\tВПП "<< i+1 <<"\t";
```

```
        cout <<"\n\n";
        for (int k = 0; k < K; k++)
        {
                cout <<"ЛА "<< k+1 <<"\t";
                for (int j = 0; j < N; j++)
                {
                        Psi[k] = PSI[k] - Teta[j];
                        cout<<Psi[k]<<" \t\t";
                }
                cout <<"\n\n";
        }
        cout <<"\n\n\n\n";

}

void ASU::Ras4(int k, int j)
{

        if (abs(Psi[k]) > 180)
                x3[k] = 360 - abs(Psi[k]);
        else
                x3[k] = abs(Psi[k]);

           x1[k] = abs(Z[k]);
```

```
        m[k] = abs(Ksi[k]);
        if (x1[k] > R)
        {
                l[k] = (2*R)+(R*abs(x3[k]))/90;
                L[k] = abs(x1[k]) + R + (R*abs(x3[k]))/40;
        }
        else
        {
                l[k] = x1[k]*sqrt(3.0)+R*x3[k]/60;
                L[k] = l[k];
        }
}

void ASU::X1value(double* KSI, double* ZK, double* PSI)//For X1 value
{

        cout <<"-----------------------------------------\n\n \t\t\t  <X1> Value \n\n";

        for (int i = 0; i < N; i++)
                cout <<"\tВПП "<< i+1 <<"\t";
        cout <<"\n\n";
        for (int k = 0; k < K; k++)
        {
```

```
                cout <<"ЛА "<< k+1 <<"\t";
                for (int j = 0; j < N; j++)
                {
                        Z[k] = -KSI[k]*sin(Teta[j]*M_PI/180) +
ZK[k]*cos(Teta[j]*M_PI/180);
                        x1[k]=abs(Z[k]);
                        if(k==3 || j==2)
                        {
                        cout<<x1[k]<<"    \t ";
                        }
                        else
                                cout<<x1[k]<<"     \t";
                }
                cout <<"\n\n";
        }
        cout <<"\n\n\n\n";

}

void ASU::X3value(double* KSI, double* ZK, double* PSI)//For X3 value
{
        cout <<"-----------------------------------------\n\n \t\t\t <X3> Value \n\n";

        for (int i = 0; i < N; i++)
```

```
            cout <<"\tВПП "<< i+1 <<"\t";
        cout <<"\n\n";
        for (int k = 0; k < K; k++)
        {
            cout <<"ЛА "<< k+1 <<"\t";
            for (int j = 0; j < N; j++)
            {
                Psi[k] = PSI[k] - Teta[j];
                if (abs(Psi[k]) > 180)
                    x3[k] = 360 - abs(Psi[k]);
                else
                    x3[k] = abs(Psi[k]);
                cout<<x3[k]<<" \t\t";
            }
            cout <<"\n\n";
        }
        cout <<"\n\n\n\n";

}

void ASU::Y1value(int K,int N)//For Y1 value
{
        cout <<"----------------------РАСЧЕТ----------------------\n\n \t\t\t  <Y1
vlaue>\n\n";
```

```
        for (int i = 0; i < N; i++)
                cout <<"\tВПП "<< i+1 <<"\t";
        cout <<"\n\n";

        for(int k=0;k<K;k++)
        {
                cout <<"ЛА "<< k+1 <<"\t";

                for(int j=0;j<N;j++)
                {
                        y1[k]= x1[k]/r;
                        if(k==14 || k==15)
                        cout<<y1[k]<<" \t\t";
                        else
                        cout<<y1[k]<<" \t";
                }
                cout<<"\n\n";
        }
        cout <<"\n\n\n";
}

void ASU::If_For_Prior(int k)
{
```

```
        if (Ksi[k] > 0)
                ch = 1;
        else
        {
                if (x3[k] > 90)
                        ch = 2;
                else
                        ch = 3;
        }
}

void ASU::X2(int k)
{

        if (ch == 1 || ch == 2)
        {
                for (int i = 0; i < K; i++)
                        M[k][i] = 500;
                x2[k] = 500;
                int y2=x2[k]/6;
                //cout << "ЛА " << k+1 << "\t";
                //cout <<y2<<"\t\t"<<endl;
                //cout<<x2[k]<<"\t\t"<<endl;
                //cout << "\n\n";
```

```
}
else
{
        double DELTA_k;
        double DELTA_i;
        double CI[2];
        double min_M;

        double mi;
        double x1i;
        double x3i;

        for (int i = 0; i < K; i++)
        {
                DELTA_k = m[k] - l[k];
                DELTA_i = m[i] - l[i];
                M[k][i] = abs(DELTA_k - DELTA_i);  //abs!
                if (x3[i] > 90 || M[k][i] == 0)
                        M[k][i] = 500;
        }
        Minimum(M[k], K, CI);
        min_M = CI[0];
        if (min_M == 500)
```

```
                min_M = 0;

            for( int i = 0 ; i < K ; i++)//For new formula
            {
                mi = m[k] - m[i];
                x1i = x1[k] - x1[i];
                x3i = x3[k] - x3[i];
            }

            x2[k] = abs( mi + x1i + (R/90)*x3i);//This is new formula

            //x2[k] = abs(min_M + (L[k] - L[(int)CI[1]]));
            //x2[k] = abs(min_M ) + abs(L[k] - l[(int)CI[1]]);

            int y2=x2[k]/6;
            //cout << "ЛА " << k+1 << "\t";
            //cout <<y2<<"\t\t"<<endl;
            //cout<<x2[k]<<"\t\t"<<endl;
            //cout << "\n\n";
        }

}

void ASU::Prior0(int k, int j) //расчет приоритета БЕЗ учета x2
```

```
{
        switch (ch)
        {
                case 1 :PRIOR0[j][k] = 5*abs(x3[k]*M_PI/180 - M_PI)*R +
sqrt(pow(m[k] + D, 2) + pow(x1[k], 2)) + M_PI*R/2; //формула для передней
полусферы
                        break;

                case 2 : PRIOR0[j][k] = 10*(x3[k]*M_PI/180)*R;
                        break;

                case 3 :
                        double Fy1;
                        double y1;
                        double y2;

                        y1 = x1[k]/r;

                        if (y1 <= 1.2)
                                Fy1 = x1[k];
                        else
                        {
                                Fy1 = 1.2;
```

```
                    PRIOR0[j][k] = (-1+0.06*abs(y1)+0.006*(180-abs(x3[k]-
180)))+(200*exp(-3*abs(y2))-(250*y4[k]));
//PRIOR0[j][k] = (-1+0.06*abs(y1)+0.006*(180-abs(x3[k]-180)))+(0.5*exp(-
3*abs(y2))-(600*y4[k]));
                    // PRIOR0[j][k] = 1.45*Fy1 - 0.008*pow(x3[k], 2) -
0.2*pow(y1, 2);  //формула для задней полусферы без учета x2
                    }
        }
} ;

voidASU::Prior(intk, intj) //расчет приоритета С учетом x2
{
        switch (ch)
        {
                case 1 : PRIOR[j][k] = 5*abs(x3[k]*M_PI/180 - M_PI)*R +
sqrt(pow(m[k] + D, 2) + pow(x1[k], 2)) + M_PI*R/2; //формула для передней
полусферы
                    break;

                case 2 : PRIOR[j][k] = 10*(x3[k]*M_PI/180)*R;
                    break;

                case 3 :
                    double Fy1;
                    double y1;
```

```
                double y2;


                y1 = x1[k]/r;
                y2 = x2[k]/r;
                if (y1 <= 1.2)
                        Fy1 = x1[k];
                else
                {
                        Fy1 = 1.2;
PRIOR0[j][k] = (-1.4-0.5*abs(y1)-0.01*(180-abs(x3[k]-180)))-(200*exp(-3*abs(y2))+(250*y4[k]));
                //PRIOR0[j][k] = (-1+0.06*abs(y1)+0.006*(180-abs(x3[k]-180)))+(200*exp(-3*abs(y2))-(250*y4[k]));
                //PRIOR[j][k] = -1+0.06*abs(y1)+0.006*(180-abs(x3[k]-180))+(0.5*exp(-3*abs(y2))-1.2*y4[k]);
                //PRIOR[j][k] = 1.45*Fy1 + 0.013*y2*x3[k] - 0.008*pow(x3[k], 2) - 0.06*pow(y2, 2) - 0.2*pow(y1,2); //формула для задней полусферы
                }break;

        }
}

void ASU::Table0()
{
```

```
AA TAB0[20][30];
double* Pr = newdouble[N];
int* NumTr = newint[K];
double* Numvalue = newdouble[K];
double CI[2];
int i;

for (int k = 0; k < K ; k++)
{
        for (int j = 0; j < N ; j++)
                Pr[j] = PRIOR0[j][k];
        Minimum(Pr, N, CI);
        NumTr[k] = (int)CI[1];
        Numvalue[k] = abs(CI[0]);
}
for (int j = 0; j < N ; j++)
{
        i = 0;
        for (int k = 0; k < K ; k++)
        {
                if (NumTr[k] == j)
                {
                        TAB0[j][i].num=k+1;
                        TAB0[j][i].kypc=NumTr[k];
```

```
                    TAB0[j][i].prio=Numvalue[k];
                    ++i;
                }
            }
        }

for(int k=0; k<N; k++)
        {
            for(int i =0; i<17-1; i++)
            {
                for(int j=i+1;j<17;j++)
                {
                    if(TAB0[k][i].prio > TAB0[k][j].prio &&
TAB0[k][j].prio!= 0)
                    {
                        double temp=TAB0[k][i].prio;
                        int  temp1=TAB0[k][i].num;
                        int  temp2=TAB0[k][i].kypc;

                        TAB0[k][i].prio=TAB0[k][j].prio;
                        TAB0[k][i].num=TAB0[k][j].num;
                        TAB0[k][i].kypc=TAB0[k][j].kypc;

                        TAB0[k][j].prio=temp;
```

```
                                TAB0[k][j].num=temp1;
                                TAB0[k][j].kypc=temp2;
                            }
                        }
                    }
                }

                for(int i=0 ; i<N ; i++)
                {
                    for(int j=0 ; j<K ; j++)
                    {
                        if(TAB0[i][j].num!=0)
                        TABLE0[i][j]=TAB0[i][j].num;
                        //TABLE0[i][j]=TAB0[i][j].prio;
                    }
                }

                delete [] Pr;
                delete [] NumTr;
}

void ASU::Table()
{
                AA TAB[20][30];
```

```
double* Pr = newdouble[N];
int* NumTr = newint[K];
double* Numvalue = newdouble[K];
double CI[2];
int i;

for (int k = 0; k < K ; k++)
{
        for (int j = 0; j < N ; j++)
                Pr[j] = PRIOR[j][k];
        Minimum(Pr, N, CI);
        NumTr[k] = (int)CI[1];
        Numvalue[k] = (double) CI[0];
        //cout<<k+1<<"\t"<<NumTr[k]<<"\t"<<Numvalue[k]<<endl;
}
for (int j = 0; j < N ; j++)
{
        i = 0;
        for (int k = 0; k < K ; k++)
        {
                if (NumTr[k] == j)
                {
                        TAB[j][i].num=k+1;
                        TAB[j][i].kypc=NumTr[k];
```

```
                    TAB[j][i].prio=Numvalue[k];

//cin>>TAB[j][i].num>>TAB[j][i].kypc>>TAB[j][i].prio;
                    ++i;
                }
            }
        }

        for(int k=0; k<N; k++)//program ngal sin kyi lite
        {
            for(int i =0; i<17-1; i++)
            {
                for(int j=i+1;j<17;j++)
                {
                    if(TAB[k][i].prio > TAB[k][j].prio && TAB[k][j].prio!=
0)
                    {
                        double temp=TAB[k][i].prio;
                        int  temp1=TAB[k][i].num;
                        int  temp2=TAB[k][i].kypc;

                        TAB[k][i].prio=TAB[k][j].prio;
                        TAB[k][i].num=TAB[k][j].num;
                        TAB[k][i].kypc=TAB[k][j].kypc;
```

```
                        TAB[k][j].prio=temp;
                        TAB[k][j].num=temp1;
                        TAB[k][j].kypc=temp2;
                    }
                }
            }
        }

for(int i=0 ; i<N ; i++)
{
        for(int j=0 ; j<K ; j++)
        {
                if(TAB[i][j].num!=0)
                {
                TABLE[i][j]=TAB[i][j].num;
                //cout<<TABLE[i][j]<<endl;
                }
                //TABLE[i][j]=TAB[i][j].prio;
        }
}

delete [] Pr;
delete [] NumTr;
```

```
}

void ASU::Show_Details(int j)
{
	cout <<"\t\t\tВПП №"<< j + 1 <<" (Курс "<< Teta[j] <<"):\n\n";
	cout<<"Координаты x3 ЛА:\n";
	show1(x3, K);
	cout<<"\nКоординаты x1 ЛА:\n";
	show1(x1, K);
	cout<<"\nКоординаты KSI ЛА:\n";
	show1(Ksi, K);
	//cout<< "\nРасстояние от ближайшеё точки на трассе до точки входа в эшелон ЛА (lk):\n";
	//show1(l, K);
	//cout<< "\nДлина пути при входе в эшелон ЛА (Lk):\n";
	//show1(L, K);
	//cout<< "\nМатрица разниц расстояний от ВПП до точки входа в эшелон:\n";
	//show2(M, K, K);
	//cout<< "\nКоординаты x2 ЛА:\n";
	//show1(x2, K);
	cout<<"\nПриоритеты входа в воздушный эшелон ЛА:\n";
	show1(PRIOR[j], K);
	cout <<"\n\n\n\n";
```

```
}

void ASU::Show_Tables()
{
	cout.precision(5);
	//double* A = new double[K];

	cout<<"------------------------РАСЧЕТ БЕЗ УЧЕТА y2----------------------
\n\n \t\t\tТАБЛИЦА ПРИОРИТЕТОВ\n\n\t";
	for (int i = 0; i < N; i++)
	{
		if(i==0)cout<<"	";
		cout <<"ВПП "<< i+1 <<"\t\t";
	}
	cout <<"\n\n";
	for (int k = 0; k < K; k++)
	{

		cout <<"  ЛА "<< k+1 <<"  \t";
		for (int j = 0; j < N; j++)
		{

			cout << fabs(PRIOR0[j][k]) <<"\t\t";
		}
```

```
            cout <<"\n\n";
        }
        cout <<"\n\n\n\n";

        for (int j = 0; j < N; j++)
        {
            cout<<"   ЛА, заходящие на ВПП "<<j + 1 <<" (Курс "<<Teta[j]
<<")\n\n";
            for (int i = 0; i < K; i++)
            {
                if (TABLE0[j][i] == 0)
                {
                    if (i == 0)
                        cout<<"\tНи один ЛА на ВПП не садится";
                    break;
                }
                elseif(TABLE0[j][i]>0)
                cout <<"\t"<< TABLE0[j][i];
            }
            cout <<"\n\n";
        }
```

```
        cout <<"----------------------РАСЧЕТСУЧЕТОМ y2----------------------
\n\n \t\t\tТАБЛИЦАПРИОРИТЕТОВ\n\n\t";
        for (int i = 0; i < N; i++)
        {
                if(i==0)cout<<"        ";
                cout <<"ВПП "<< i+1 <<"\t\t";
        }
        cout <<"\n\n";
        for (int k = 0; k < K; k++)
        {
                cout <<"  ЛА "<< k+1 <<"  \t";
                for (int j = 0; j < N; j++)
                {
                        cout << fabs(PRIOR[j][k]) <<"\t\t";
                }
                cout <<"\n\n";
        }
        cout <<"\n\n\n\n";

        for (int j = 0; j < N; j++)
        {
                cout<<"  ЛА, заходящие на ВПП "<<j + 1 <<" (Курс "<<Teta[j]
<<")\n\n";
                for (int i = 0; i < K; i++)
```

```
            {
                if (TABLE[j][i] == 0)
                {
                    if (i == 0)
                        cout<<"\tНи один ЛА на ВПП не садится";
                    break;
                }
                elseif(TABLE[j][i]>0)
                cout <<"\t"<< TABLE[j][i];

            }
            cout <<"\n\n";
        }
}


void ASU::Minimum(double* A, int n, double* B)
{
    double Min = A[0];
    int I = 0;
    for (int i = 1; i < n; i++)
    {
        if (fabs(A[i]) < abs(Min))
        {
```

```
                Min = A[i];
                I = i;
            }
        }
        B[0] = Min;
        B[1] = I;
}
void ASU::show1(double* A, int n)
{
        for (int i = 0; i < n; i++)
            cout <<"\t"<< A[i];
        cout << endl;
}
void ASU::show1(int* A, int n)
{
        for (int i = 0; i < n; i++)
            cout <<"\t"<< A[i];
        cout << endl;
}
void ASU::show2(double** A, int n1, int n2)
{
        for (int i = 0; i < n1; i++)
        {
            for (int j = 0; j < n2; j++)
```

```
                cout <<"\t"<< A[i][j];
            cout << endl;
        }
    }
```

Приложение 3

Текст программы определения очередности посадки судов на ВПП

```
#include"stdafx.h"
#include<iostream>
#include"conio.h"
#include<cmath>
usingnamespace std;
structtehdata{
        float V;
        int R;
        int r;
        int sh;
};
structdatasam{
        int nn;
        double x;
        double z;
        double psi;
        int H;
        int t;
        char priz;
};
structvpp{
        int t;
```

```
        int x;
        int z;
        int psi;
        int l_tr;
        int an;
        double h;
};
structlength{
        double L_v;
        int num;
        int t;
};
int main()
{
        tehdata param={0.1,4,6,15};
        //param.R=4; param.r=6; param.V=0.1;
        //cout << param.R << " " << param.V << " " << param.r << "\n";
        //printf("%d", param.R);

        vpp trass[]={
                1,50,60,90,30,68,2,
                2,30,40,135,28.25,113,1.5,
                3,30,15,90,25,135,1,
                4,45,15,0,15,89,1
```

```
};
int t=4;
/*cout <<"\nt x z psi dlina an H\n";
for (int i=0; i<t; i++)
{
	cout <<trass[i].t<<" "<<trass[i].x<<" "<<trass[i].z<<"
"<<trass[i].psi<<"  "<<trass[i].l_tr<<"  "<<trass[i].an<<" "<<trass[i].h<<"\n";
}*/
datasam ishdata[]={
	1,55,65,90,3,0,30,
	2,50,75,90,2.5,0,30,
	3,55,90,100,2,0,30,
	4,40,65,110,1.5,0,30,
	5,60,75,80,2,0,30,
	6,80,75,75,2.5,0,30,
	7,60,55,120,2,0,30,
	8,40,80,150,1,0,30,
	9,50,45,130,1.5,0,30,
	10,40,35,140,2,0,30,
	11,30,20,90,1,0,30,
	12,35,60,80,1.5,0,30,
	13,30,50,100,0.8,0,30,
	14,25,40,75,1,0,30,
	15,37,15,0,1,0,30,
```

```
        16,40,5,5,1,0,30
    };
    int s=16;
    cout <<"\nnn x  z psi H  t\n";
    for (int i=0; i<s; i++)
    {
        cout <<ishdata[i].nn<<" "<<ishdata[i].x<<" "<<ishdata[i].z<<"
"<<ishdata[i].psi<<"  "<<ishdata[i].H<<" "<<ishdata[i].t<<"\n";
    }
    lengthdlina[16];

    //Определение принадлежности самолета к определенному участку
трассы
    double PI=3.14159;
    for(int i=0; i<s; i++)
    {
        int j=0;
begin:

        int sr;
        double c,x_p,z_p;
        sr=(360 - trass[j].psi + trass[j].an) % 360;
        c=(ishdata[i].z-trass[j].z)/(ishdata[i].x-trass[j].x);
        x_p=ishdata[i].x-trass[j].x;
```

```
z_p=ishdata[i].z-trass[j].z;


if ((sr>=0)&&(sr<90))
{
    if ((x_p>=0)&&(z_p<=0)) {j++; goto begin;}
    if
((x_p<=0)&&(z_p>=0)||(x_p>=0)&&(c>=tan(sr*PI/180))||(x_p<=0)&&(c<=tan(sr
*PI/180))) ishdata[i].t=j+1;
    else {j++; goto begin;}
}
if ((sr>=90)&&(sr<180))
{
    if ((x_p>=0)&&(z_p>=0)) {j++; goto begin;}
    if ((x_p>=0)&&(z_p>=0)) {j++; goto begin;}
    if
((x_p<=0)&&(z_p<=0)||(x_p>=0)&&(c<=tan(sr*PI/180))||(x_p<=0)&&(c>=tan(sr
*PI/180))) ishdata[i].t=j+1;
    else {j++; goto begin;}
}
if ((sr>=180)&&(sr<270))
{
    if ((x_p<=0)&&(z_p>=0)) {j++; goto begin;}
    if
((x_p>=0)&&(z_p<=0)||(x_p>=0)&&(c<=tan(sr*PI/180))||(x_p<=0)&&(c>=tan(sr
*PI/180))) ishdata[i].t=j+1;
```

```
                else {j++; goto begin;}
            }
            if ((sr>=270)&&(sr<360))
            {
                if ((x_p<=0)&&(z_p<=0)) {j++; goto begin;}
                if
((x_p>=0)&&(z_p>=0)||(x_p>=0)&&(c>=tan(sr*PI/180))||(x_p<=0)&&(c<=tan(sr
*PI/180))) ishdata[i].t=j+1;
                else {j++; gotobegin;}
            }
        }
        //проверкарезультата
        /*for (int i=0; i<s; i++)
        {
            cout <<ishdata[i].t <<" ";
        }*/
        datasamcurrent[16]; //координатывтрассовойситемекоординат

        //переход в трассовую систему координат
        doublex_n,z_n,h_n;
        intpsi_n;
        //где i+1 номер самолета
        //cout <<"\n  x_n    z_n     psi_n\n";
        for (int i=0; i<s; i++)
```

```
{
        int nt;
        nt=ishdata[i].t-1;
        x_n=-(ishdata[i].z-trass[nt].z)*sin(-trass[nt].psi*PI/180)-(ishdata[i].x-trass[nt].x)*cos(-trass[nt].psi*PI/180);
        z_n=(ishdata[i].z-trass[nt].z)*cos(-trass[nt].psi*PI/180)-(ishdata[i].x-trass[nt].x)*sin(-trass[nt].psi*PI/180);
        psi_n=ishdata[i].psi-trass[nt].psi;
        h_n=abs(ishdata[i].H-trass[nt].h);
        //cout <<x_n<<" "<<z_n<<" "<<psi_n<<"\n";
        current[i].x=x_n; current[i].z=z_n; current[i].psi=psi_n; current[i].nn=ishdata[i].nn; current[i].H=h_n; current[i].t=ishdata[i].t;
}
cout <<"\nnn x  z psi H  t\n";
for (int i=0; i<s; i++)
{
        cout <<current[i].nn<<" "<<current[i].x<<" "<<current[i].z<<" "<<current[i].psi<<"  "<<current[i].H<<" "<<current[i].t<<"\n";
}
double r,L_v;
double psi_a;
for (int i=0; i<s; i++)
{
        r=sqrt(pow(current[i].x,2) + pow(current[i].z,2));
        if (current[i].x >= 0)
```

```
            {
                if (current[i].z >= 0)
                    psi_a=(int)asin(current[i].z/r);
                else
                   psi_a=PI - (int)asin(current[i].z/r);
            }
            else
            {
                if (current[i].z >= 0)
                    psi_a=PI + (int)asin(current[i].z/r);
                else
                   psi_a=2*PI + (int)asin(current[i].z/r);
            }
            cout<<psi_a<<" ";
            //Определение длины пути L_v до точки начала снижения.

        L_v=sqrt(pow(current[i].H,2)+pow(current[i].x,2)+pow(current[i].z,2))+(2/
PI)*param.R*abs(current[i].psi*PI/180 - psi_a);
            //cout << L_v <<"  ";
            dlina[i].L_v=L_v; dlina[i].num=current[i].nn; dlina[i].t=current[i].t;
        }

        cout <<"\nnum L_v t\n";
        for (int i=0; i<s; i++)
```

```
{
        cout <<dlina[i].num<<"   "<<dlina[i].L_v<<" "<<dlina[i].t<<"\n";
}
//Сортировка самолётов по условию от minL_v до maxL_v и в порядке увеличения номера трассы

double lv;
int num,nt;
for (inti=0; i<s; i++)                    //Сортировка по трассам в порядке их следования
{
        for (int j=s-1; j>=i; j--)
        {
                if(dlina[j-1].t > dlina[j].t)
                {
                        nt=dlina[j-1].t; dlina[j-1].t=dlina[j].t; dlina[j].t=nt;
                        num=dlina[j-1].num; dlina[j-1].num=dlina[j].num; dlina[j].num=num;
                        lv=dlina[j-1].L_v; dlina[j-1].L_v=dlina[j].L_v; dlina[j].L_v=lv;
                }
        }
}

lv=0; num=0; nt=0;
```

```
for (int i=0; i<s; i++)
   {
      for (int j=s-1; j>=i; j--)
         {
            if (dlina[j-1].t == dlina[j].t)
            {
          if (dlina[j-1].L_v > dlina[j].L_v)
             {
                lv=dlina[j-1].L_v; dlina[j-1].L_v=dlina[j].L_v;
dlina[j].L_v=lv;
                num=dlina[j-1].num; dlina[j-1].num=dlina[j].num;
dlina[j].num=num;
                   nt=dlina[j-1].t; dlina[j-1].t=dlina[j].t;
dlina[j].t=nt;
          }
            }
         }
   }

   //Результат сортировки
   cout<<"\nnumL_vt\n";
   for (int i=0; i<s; i++)
   {
      cout <<dlina[i].num<<"   "<<dlina[i].L_v<<" "<<dlina[i].t<<"\n";
```

```
}

//Далее производим расчет по 3 алгоритму
//Определение координат точек попадания самолетов на трассу

double AA, b, dl;
datasam newcoord[16];
num=dlina[0].num-1;
newcoord[0].x=0; newcoord[0].z=0; newcoord[0].nn=dlina[0].num;
newcoord[0].t=dlina[0].t; newcoord[0].H=current[num].H;
newcoord[0].psi=current[num].psi;

for (int i=1; i<s; i++)
{
    double x_f, z_f;
    int n_s,n_s1, n_t;
    n_s=dlina[i].num-1;
    n_s1=dlina[i+1].num-1;
    n_t=dlina[i].t-1;
    if (dlina[i-1].t == dlina[i].t )
    {
        if ((current[n_s].z < 0.001) && (current[n_s].z >- 0.001)&&(i <
s-1))  //Еслинатрассеужеестьсамолет
        {
```

```
AA=pow(current[n_s1].z,2) + pow(current[n_s1].H,2) +
pow((current[n_s1].psi*PI/180 - abs(current[n_s1].z)/current[n_s1].x)*param.R,2);

cout <<"\n i= "<< i <<"\nAA "<<AA<<"\n";

b=(current[n_s].x+current[n_s1].x -
2*param.r)*(current[n_s].x-current[n_s1].x + param.r); cout <<"sam
"<<current[n_s+1].nn<<" b "<<b<<"\n";

dl=(current[n_s].x+current[n_s1].x+param.r)/2 -
AA/(2*(current[n_s].x - current[n_s1].x + param.r)); cout <<"dl "<<dl<<"\n";

if (AA > b)

{

newcoord[i].x=dl - param.r; newcoord[i].z=0;
newcoord[i].nn=n_s1; newcoord[i].psi=0; newcoord[i].H=0;
newcoord[i].t=dlina[i].t;

}

}

else

{

newcoord[i].x=newcoord[i-1].x+param.r; newcoord[i].z=0;
newcoord[i].nn=n_s+1; newcoord[i].psi=0; newcoord[i].H=0;
newcoord[i].t=dlina[i].t;


}

}

else

{
```

```
                newcoord[i].x=0; newcoord[i].z=0;
newcoord[i].nn=dlina[i].num; newcoord[i].t=dlina[i].t; newcoord[i].H=0;
newcoord[i].psi=0;
            }
            if (newcoord[i].x > trass[n_t].l_tr)
            {
                newcoord[i].priz='T'; newcoord[i].x=newcoord[i-1].x;
newcoord[i].z=0; newcoord[i].nn=dlina[i].num; newcoord[i].t=dlina[i].t;
newcoord[i].H=0; newcoord[i].psi=0;
            }
        }

        cout <<"\nnum x z psi H t\n";
        for (int i=0; i<s; i++)
        {
            cout <<newcoord[i].nn<<"  "<<newcoord[i].x<<"
"<<newcoord[i].z<<" "<<newcoord[i].psi<<" "<<newcoord[i].H<<"
"<<newcoord[i].t<<" "<<newcoord[i].priz<<"\n";
        }
        //Переход из трассовой системы координат в исходную систему
координат
        for (int i=0; i<s; i++)
        {
            int nt;
            nt=newcoord[i].t-1;
```

```
            x_n = newcoord[i].z*sin(trass[nt].psi*PI/180) -
(newcoord[i].x)*cos(trass[nt].psi*PI/180) + trass[nt].x;
            z_n = newcoord[i].z*cos(trass[nt].psi*PI/180) +
newcoord[i].x*sin(trass[nt].psi*PI/180) + trass[nt].z;
            psi_n=newcoord[i].psi + trass[nt].psi;
            h_n=abs(newcoord[i].H + trass[nt].h);
            //cout <<x_n<<" "<<z_n<<" "<<psi_n<<"\n";
            newcoord[i].x=x_n; newcoord[i].z=z_n; newcoord[i].psi=psi_n;
newcoord[i].H=h_n;
        }


        cout <<"\nn nnum x z psi H t\n";
        for (int i=0; i<s; i++)
        {
            cout <<i+1<<" "<<newcoord[i].nn<<"  "<<newcoord[i].x<<"
"<<newcoord[i].z<<" "<<newcoord[i].psi<<" "<<newcoord[i].H<<"
"<<newcoord[i].t<<" "<<newcoord[i].priz<<"\n";
        }
        system("pause");
        return 0;
}
```