

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

В. М. Белоконов, И. В. Белоконов

Системный анализ космических миссий

Электронный лабораторный практикум

САМАРА

2010

Авторы: БЕЛОКОНОВ Виталий Михайлович,
БЕЛОКОНОВ Игорь Витальевич

Настоящий лабораторный практикум по дисциплине «Системный анализ космических миссий» призван дать практические навыки в проведении комплексного анализа практически любых типовых миссий в околоземном космическом пространстве.

Задачи в практикуме представляют комплекс взаимосвязанных задач по анализу выполнимости космической миссии, начиная с этапа выведения космического аппарата до этапа доставки груза на Землю.

Основной целью практикума является закрепление теоретических знаний по дисциплине и усвоение учащимися основных инженерных методик, используемых при системном анализе космических миссий.

Практикум также способствует усвоению количественных оценок проектно-баллистических характеристик для типичных условий функционирования космических аппаратов, и может быть использован при выполнении выпускной магистерской работы.

Лабораторный практикум предназначен для магистрантов, обучающихся по магистерской программе «Космические информационные системы и наноспутники. Навигация и дистанционное зондирование» по направлению 010900.68 «Прикладные математика и физика».

Лабораторный практикум разработан на межвузовской кафедре космических исследований.

СОДЕРЖАНИЕ

№	Название раздела	Стр.
	ВВЕДЕНИЕ	4
1	Концепция лабораторного практикума	6
1.1	Задание и исходные данные для выполнения практикума по анализу космической миссии	6
1.2	Последовательность выполнения анализа космической миссии	8
1.3	Анализ нештатных вариантов реализации космической миссии	13
2	Описание и инструкция по применению программного комплекса START для анализа этапа выведения на промежуточную орбиту	14
2.1	Блок-схема программного комплекса START	14
2.2	Описание интерфейса программного комплекса START	15
2.3	Листинг программы	25
3	Инструкция по применению программы STK для системного анализа космической миссии	58
3.1	Начальные навыки работы с ПК STK	58
3.2	Анализ орбитальных миссий с ПК STK	82
Приложение 1	Расчетные методики и модели для анализа выполнимости выведения заданной полезной нагрузки выбранной РКН	91
Приложение 2	Расчетные методики и справочные данные для анализа орбитальных этапов космической миссии	106

ВВЕДЕНИЕ

Настоящий лабораторный практикум по дисциплине «Системный анализ космических миссий» призван дать практические навыки в проведении комплексного анализа практически любых типовых миссий в околоземном космическом пространстве.

Задачи в практикуме представляют комплекс взаимосвязанных задач по анализу выполнимости космической миссии, начиная с этапа выведения космического аппарата до этапа доставки груза на Землю.

Основной целью практикума является закрепление теоретических знаний по дисциплине и усвоение учащимися основных инженерных методик, используемых при системном анализе космических миссий.

Практикум также способствует усвоению количественных оценок проектно-баллистических характеристик для типичных условий функционирования космических аппаратов, и может быть использован при выпускной магистерской работы.

Практикум поддерживает практически все вопросы, отраженные в конспекте лекций по данной дисциплине.

Практикум состоит из трех частей и двух приложений.

В первой части описывается концепция практикума, детально излагается последовательность решаемых задач, структура исходных данных, описывающих космическую миссию. Исходными данными при выполнении практикума являются проектные характеристики ракеты-носителя космического назначения, требования к космическому аппарату, определяющие его целевое назначение и тактико-технические характеристики (в том числе параметры рабочей орбиты), основные проектные характеристики космического аппарата и его бортовых систем.

Во второй части приводится описание, а также инструкция по применению программного комплекса START для анализа использования выбранного типа ракеты-носителя космического назначения (РКН) для доставки заданной полезной нагрузки (космического аппарата) на орбиту выведения. Программный комплекс START написан на языке высокого уровня JAVA. Этот язык позволяет создать удобный интерфейс и эффективно организовать учебный процесс не только в очной, но и в дистанционной форме.

В Приложении 1 кратко излагаются расчетные методики и модели для анализа выполнимости выведения заданной полезной нагрузки выбранной РКН и описывается последовательность выполнения различного типа заданий.

В третьей части описываются возможности и приводится инструкция по применению специализированного лицензионного пакета прикладных программ STK (Satellite Tool Kit) для системного анализа космической миссии. Этот пакет используется для анализа следующих этапов:

- анализа баллистической схемы полета в штатном режиме функционирования космического аппарата,
- оценки потребного запаса топлива на борту космического аппарата,
- анализа баллистической схемы полета в аварийном режиме.

При анализе орбитальных этапов решаются следующие задачи:

- обоснование выбора типа рабочей орбиты и ее основных элементов,
- оценка эволюции орбиты от действия атмосферы Земли (в том числе расчет времени существования космического аппарата и т. д.) и от нецентральности поля притяжения,
- расчет маневра перевода космического аппарата с орбиты выведения на рабочую орбиту,
- расчет маневров поддержания рабочей орбиты (обеспечение заданной высоты полета, обеспечение движения в заданном диапазоне высот, корректирование элементов орбиты) на интервале времени активного существования на орбите,
- расчет трассы полета и вспомогательной баллистической информации (условий взаимной видимости космического аппарата и наземного пункта, наблюдаемости районов земной поверхности с борта космического аппарата, условий освещенности космического аппарата),
- расчет тормозного импульса, точки его приложения и характеристики внеатмосферного участка полета, предшествующего спуску космического аппарата в атмосферу,
- выбор требуемых условий входа и расчет движения в атмосфере спускаемого аппарата.

В Приложении 2 кратко излагаются основные теоретические положения и описывается последовательность выполнения различных типовых заданий, связанных с анализом отдельных этапов космической миссии. Особенное внимание уделено навыкам проверки возможностей осуществления контроля полета, планирования сеансов получения телеметрической информации, оценки условий видимости и освещенности различных районов земной поверхности.

При подготовке практикума использовались материалы, подготовленные ранее профессором Тимбаем И.А., аспирантом Травиным В.С.

1. Концепция лабораторного практикума

1.1. Задание и исходные данные для выполнения практикума по анализу космической миссии

1.1.1 Целевое назначение и основные параметры космической миссии

Целевое назначение космического аппарата (спутник метеорологический, исследования природных ресурсов Земли, связи навигационный и т. Д.) _____

Полная масса космического аппарата после отделения от ракеты носителя m_0 в кг _____

Масса полезной нагрузки $m_{\text{нн}}$ в кг (масса космического аппарата незаполненного топливом) _____

Площадь миделевого сечения космического аппарата $S_{\text{кд}}$ в м^2 _____

Площадь миделевого сечения спускаемого аппарата $S_{\text{сд}}$ в м^2 _____

Коэффициент аэродинамического сопротивления в разреженной атмосфере космического аппарата C_x _____

Максимальное аэродинамическое качество при спуске в атмосфере K_{max} _____

Удельная тяга комбинированной двигательной установки системы управления движением $P_{\text{уд}}$ в м/с _____

Угол обзора бортовой аппаратуры $\delta_{\text{ка}}$ в град _____

Ресурс времени работы бортовой аппаратуры $T_{\text{ба}}$ в сут _____

1.1.2 Параметры реализовавшейся орбиты выведения

Вариант задания через элементы орбиты:

Высота перигея $H_{\text{п}}^{(0)}$ в км _____

Высота апогея $H_{\text{а}}^{(0)}$ в км _____

Наклонение орбиты $i^{(0)}$ в град _____

Аргумент перигея $\omega^{(0)}$ _____

Долгота восходящего узла орбиты $\Omega^{(0)}$ в град _____

Угол истинной аномалии в точке выведения $\vartheta^{(0)}$ в град (в начальный момент времени) _____

Вариант задания через характеристики конца активного участка выведения:

координаты точки конца активного участка движения в стартовой системе координат

$x_{\text{ск}}$ в км _____

$y_{\text{ск}}$ в км _____

$z_{\text{ск}}$ в км _____

проекция вектора скорости в стартовой системе координат

$V_{x_{\text{СК}}}$ в м/с _____

$V_{y_{\text{СК}}}$ в м/с _____

$V_{z_{\text{СК}}}$ в м/с _____

1.1.3 Параметры рабочей орбиты и дополнительные требования к движению

Параметры рабочей орбиты:

высота апогея $H_a^{(p)}$ в км _____

высота перигея $H_{\Pi}^{(p)}$ в км _____

наклонение орбиты $i^{(p)}$ в град _____

аргумент перигея $\omega^{(0)}$ в град _____

1.1.4 Дополнительные требования, определяющие частоту проведения коррекции движения

Для круговой орбиты:

допустимое изменение долготы восходящего узла орбиты $\Delta\Omega^*$ в град _____

допустимое падение высоты полета ΔH^* в км _____

Для эллиптической орбиты:

допустимое изменение долготы восходящего узла орбиты $\Delta\Omega^*$ в

град _____

допустимое падение высоты перигея ΔH_{Π}^* в км _____

допустимое изменение аргумента перигея $\Delta\omega^*$ в град _____

1.1.5 Дополнительные ограничения на орбитальное движение:

продолжительность пребывания на орбите выведения $\Delta t_{\text{ОБ}}$ в _____ сут

максимальное время перехода с орбиты выведения на рабочую орбиту $\Delta t_{\text{пер}}^*$ в ч. _____

1.1.6 Данные для расчета вспомогательной баллистической информации

Географическая широта наземного пункта $\varphi_{\text{нп}}$ в град _____

Географическая долгота наземного пункта $\lambda_{\text{нп}}$ в град _____

Минимальный угол возвышения космического аппарата для его видимости с наземного пункта Y_{min} в град. _____

Угол склонения Солнца $\delta_{\text{с}}$ в град. _____

Географическая широта пункта старта ракеты - носителя $\varphi_{\text{ст}}$ _____ в град

Географическая долгота пункта старта ракеты-носителя $\lambda_{\text{ст}}$ _____ в град. _____

1.1.7 Данные для расчета маневра сближения с орбитальной станцией

Начальное угловое расстояние между космическим аппаратом на орбите выведения и орбитальной станцией на рабочей орбите u в град _____

Проекции вектора положения космического аппарата относительно орбитальной станции после завершения дальнего сближения (промах по положению)

$$\tilde{x}_0 \text{ в м,}$$

$$\tilde{y}_0 \text{ в м,}$$

$$\tilde{z}_0 \text{ в м.}$$

Проекции вектора скорости полет космического аппарата относительно орбитальной станции после завершения дальнего сближения (промах по скорости)

$$\tilde{V}_{x0} \text{ в м/с,}$$

$$\tilde{V}_{y0} \text{ в м/с,}$$

$$\tilde{V}_{z0} \text{ в м/с.}$$

1.1.8 Параметры предпусковой орбиты и ограничения на контролируемые характеристики движения в атмосфере

Наклонение $i^{(сп)}$ в град _____

Высота полета $H^{(сп)}$ в км _____

Географическая широта точки спуска φ_{τ} в град _____

Географическая долгота точки спуска λ_{τ} в град _____

Максимальное допустимое значение перегрузки $n_{доп}^*$ _____

Максимальное допустимое значение удельного теплового потока в критической точке спускаемого аппарата $Q_{доп}^*$ в $\frac{\text{кДж}}{\text{с}\cdot\text{м}^2}$ _____

Максимальная допустимая температура конструкции спускаемого аппарата $T_{СА}^0$ в °С. _____

1.2 Последовательность анализа космической миссии

1.2.1 Подготовка исходных данных

Перед началом выполнения анализа космической миссии требуется внимательно ознакомиться с исходными данными.

Если параметры рабочей орбиты не полностью заданы, например, не задано наклонение орбиты, то необходимо их выбрать самостоятельно, основываясь на целевом назначении космического аппарата и пользуясь нижеприведенными рекомендациями.

Промежуточная орбита выведения может задаваться набором элементов или характеристиками конца активного участка полета в стартовой системе координат. В отдельных случаях ее требуется самому рассчитать.

Если принята вторая форма задания исходных данных, то необходимо по приведенной методике произвести расчет начальных условий движения в абсолютной геоцентрической системе координат и найти элементы орбиты выведения.

1.2.2 Анализ баллистической схемы полета в штатном режиме функционирования космического аппарата

Расчет движения космического аппарата на заданном интервале пребывания на орбите выведения

1) Рассчитывается эволюция орбиты выведения, вызванная нецентральностью поля притяжения ($\Delta\Omega, \Delta\varpi$) на интервале времени $\Delta t_{\text{ов}}$ 2)

Рассчитывается эволюция орбиты, вызванная влиянием атмосферы ($\Delta r, \Delta l, \Delta T, \Delta V_r, \Delta V_n$), на интервале времени $\Delta t_{\text{ов}}$.

3) Рассчитывается маневр коррекции по поддержанию орбиты выведения (для круговых и околокруговых орбит - поддержание высоты полета) и определяются полные затраты характеристической скорости $\Delta V_{\text{хар}}^{(\text{ов})}$.

4) Рассчитывается трасса полета космического аппарата и стандартная баллистическая информация (зона взаимной видимости и освещенности космического аппарата);

- на интервале $\Delta t_{\text{ов}}$ с использованием программы STK, при этом координаты наземного пункта выбираются из исходных данных;

- на интервале двух витков полета - вручную, при этом в качестве наземного пункта выбирается пункт старта и дополнительно вычисляются геометрические характеристики обзора земной поверхности.

1.2.3 Анализ маневра перехода с орбиты выведения на рабочую орбиту (без фазирования)

Маневр перехода рассчитывается в классе двухимпульсных и трехимпульсных энергетически оптимальных маневров. В случае трсхимпульсного маневра радиус апоцентра переходной эллиптической орбиты ограничен сверху из условия не превышения времени перехода заданной величины $\Delta t_{\text{пер}}^*$.

В случае, если рабочая орбита является эллиптической, переход осуществляется или в ее апоцентр, или в ее перицентр. При этом расчетные формулы определения импульсов скорости выводятся самостоятельно с использованием интегралов энергии и площадей.

Рассчитываются полные затраты характеристической скорости и время полета для каждого варианта перехода.

1.2.4 Анализ маневра сближения с орбитальной станцией

Вначале рассматривается маневр дальнего сближения (переход с орбиты выведения на рабочую орбиту с фазированием). Данный расчет выполняется только для случая перехода между круговыми орбитами.

Если выбран в качестве оптимального двухимпульсный перелет, то определяется требуемый угол фазирования. При задании начального углового расстояния между маневрирующим космическим аппаратом на орбите выведения и орбитальной станцией на рабочей орбите также вычисляется время ожидания до установления требуемого угла фазирования по соотношениям (96) - (99).

Если выбран в качестве оптимального трехимпульсный биэллиптический перелет, то определяется необходимая величина радиуса апоцентра переходной эллиптической орбиты из условия фазирования для заданного начального угла рассогласования между космическими аппаратами и рассчитываются заново характеристики перелета.

Предполагая, что в результате маневра дальнего сближения космический аппарат переходит в малую окрестность орбитальной станции, рассчитывается завершающий маневр этапа сближения. Считается, что продолжительность сближения не превышает периода обращения по орбите. Находится оптимальная продолжительность и рассчитываются параметры маневра из условия минимальности величины характеристической скорости.

1.2.5 Анализ корректирующих маневров для поддержания рабочей орбиты

Для случая круговой орбиты выведения рассчитывается маневр поддержания движения космического аппарата в заданном диапазоне высот и долготы восходящего узла орбиты в заданном диапазоне значений.

Для случая эллиптической орбиты выведения рассчитываются маневры коррекции (в предположении их несвязности) в отдельности для поддержания высоты перигея, аргумента перигея, долготы в заданном интервале значений согласно требованиям, сформулированным в исходных данных. При этом оценивается частота проведения коррекции путем расчета эволюции орбиты, рассчитывается количество маневров коррекции и требуемые затраты характеристической скорости на интервале времени активного существования $T_{БА}$ космического аппарата на рабочей орбите.

Результаты расчетов представляются в виде табл. 1 для круговой орбиты и в виде табл.2 для эллиптической орбиты.

Таблица 1

Коррекция элемента орбиты	Условная частота коррекции в 1/сутки	Затраты характеристической скорости на 1 коррекцию в м/с	Количество коррекций на интервале $T_{БА}$	Затраты характеристической скорости на интервале $T_{БА}$ в м/с
Ω				
H				
Полные затраты характеристической скорости в м/с				

Таблица 2

Коррекция элемента орбиты	Частота коррекции в 1/сутки	Затраты характеристической скорости на 1 коррекцию в м/с	Количество коррекций на интервале $T_{БА}$	Затраты характеристической скорости на интервале $T_{БА}$ в м/с
Ω				

Н				
Полные затраты характеристической скорости в м/с				

1.2.6 Анализ маневров на этапе доставки полезной нагрузки на Землю

Маневрирование на этапе доставки информации на Землю включает в себя маневр перехода космического аппарата с рабочей орбиты на предпусковую орбиту и маневр торможения для осуществления входа в плотные слои атмосферы с заданными условиями входа.

Максимальный по модулю угол входа $|\theta_{вх}|$ приближенно можно оценить для заданного типа спускаемого аппарата, подставляя вместо n_{max} , Q_{max} , $T_{кр max}^*$ их предельно допустимые значения. При этом скорость входа полагается равной ~ 8 км/с.

Предпусковая орбита считается круговой. Если высота предпусковой орбиты $H^{(сп)}$ задана, то рассчитываются параметры маневра перехода. Затем рассчитывается маневр схода с предпусковой орбиты.

Если высота предпусковой орбиты не задана, то она выбирается из условия минимизации полных затрат характеристической скорости на маневр перехода на предпусковую орбиту и на маневр схода с предпусковой орбиты. Оптимальная высота предпусковой орбиты может находиться любым образом, например, перебором в выбранном диапазоне высот.

1.2.7 Анализ этапа спуска с орбиты на Землю

Для выбранных величины тормозного импульса скорости ΔV_T и угла его ориентации α определяются скорость и угол входа в плотные слои атмосферы, угловая дальность и время полета на внеатмосферном участке.

Далее по заданным условиям входа в атмосферу рассчитывается траектория снижения, считая, что движение осуществляется в плоскости предпусковой орбиты при максимальном аэродинамическом качестве.

При этом проверяется выполнение всех ограничений на контролируемые характеристики движения, а также рассчитываются ориентировочные координаты точки достижения спускаемым аппаратом поверхности Земли (без участка движения на парашюте). В случае, если ограничения на траекторию спуска не выполняются; необходимо предложить меры, обеспечивающие их выполнение.

1.2.8 Расчет полных затрат топлива на реализацию рассматриваемой космической миссии

На основании результатов расчета характеристической скорости на все виды маневрирования, полученные в предыдущих разделах, рассчитывается табл. 3.

Найденные полные затраты топлива m_T на реализацию баллистической схемы полета сравниваются с запасом топлива $m_{тр}$, имеющимся на борту космического аппарата:

$$m_{тр} = m_0 - m_{ин}.$$

Если $m_{\text{т}} \leq m_{\text{тр}}$, то рассмотренная баллистическая схема полета осуществима и часть топлива можно было бы перевести в полезную нагрузку или увеличить время активного существования космического аппарата. В этом случае необходимо оценить на сколько суток может быть продлено активное существование космического аппарата.

Таблица 3

Затраты характеристической скорости $V_{ХАФ}$ в м/с		Затраты топлива в кг	Продолжи- тельность работы двигателя в сек.
1	на маневр поддержания орбиты выведения		
2	на маневр перехода с орбиты выведения на рабочую орбиту		
3	на маневр сближения		
4	на маневр поддержания рабочей орбиты		
5	на маневр перехода с рабочей орбиты на предпусковую орбиту		
6	на маневр торможения при сходе с предпусковой орбиты		
Полные затраты			

Если $m_T > m_{тр}$ то данная баллистическая схема не осуществима и необходимо предложить варианты выполнения целевой задачи, например, оценить время активного существования космического аппарата исходя из имеющихся на борту запасов топлива, или определить массу полезной нагрузки, для которой может быть реализована принятая баллистическая схема полета.

1.3 Анализ нештатных вариантов реализации космической миссии

В практикуме учащийся должен оценить две возможные аварийные ситуации.

В а р и а н т 1. После отделения космического аппарата от ракеты-носителя связь с ним установить не удастся (или вышла из строя корректирующая двигательная установка) и космический аппарат совершает неориентированное пассивное движение.

В этом случае необходимо:

- подсчитать время существования космического аппарата для минимального, среднего и максимального индексов солнечной активности,

- определить высоту критической орбиты,

- рассчитать и построить графики изменения высоты полета от времени до достижения критической высоты.

После достижения критической высоты, которая практически совпадает условной границей плотной атмосферы, космический аппарат начинает движение по пологой траектории в атмосфере, где должно произойти его разрушение.

Для анализа процесса разрушения необходимо знать перегрузки и тепловые потоки при движении. Поэтому рассчитывается движение космического аппарата (с неотделенной спускаемой частью) в атмосфере до момента падения на Землю. При этом полагается, что происходит баллистический спуск ($K=0$), а угол входа принимается равным $\sim -0,1^\circ$.

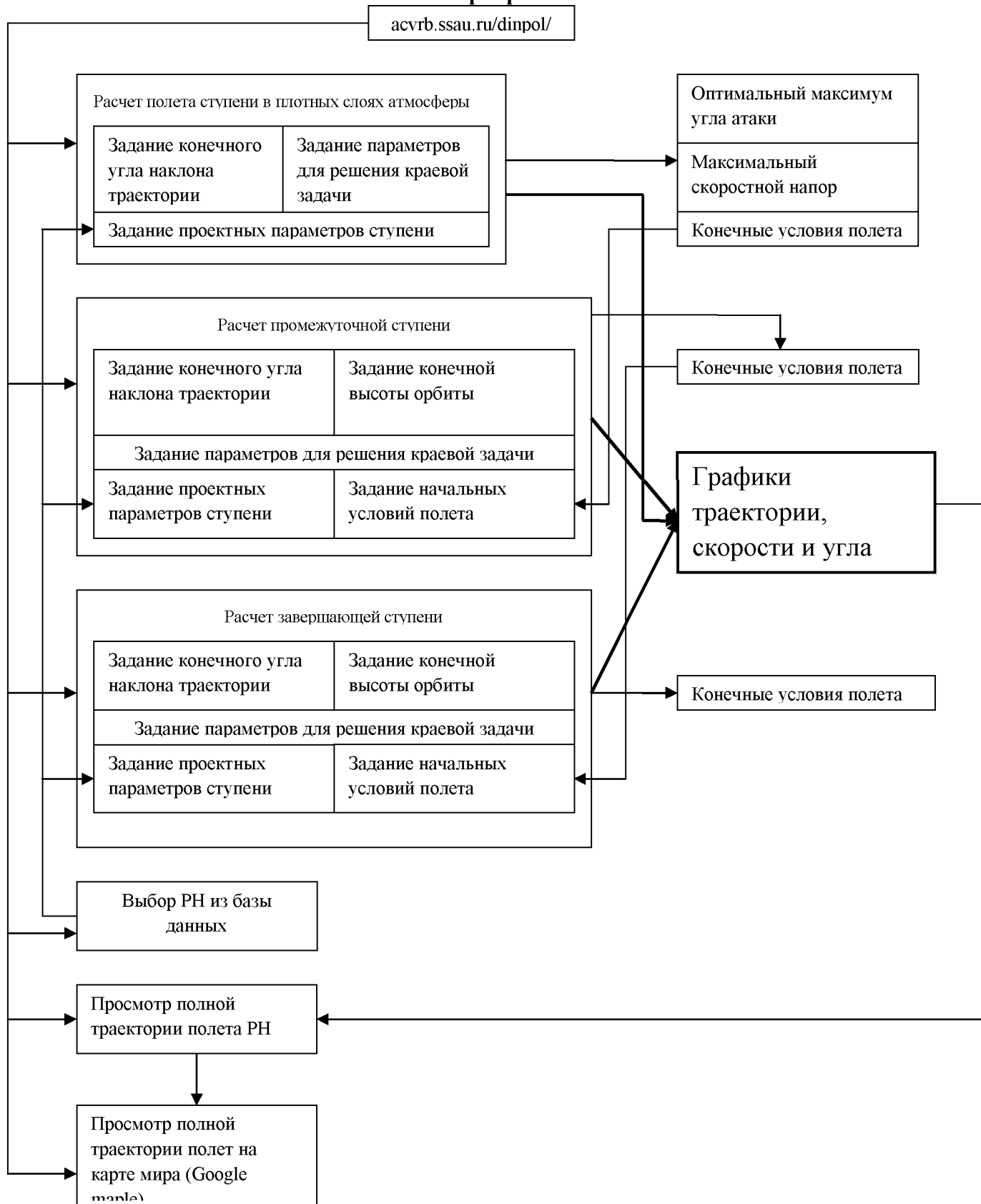
В а р и а н т 2. На заключительном этапе полета после схода спускаемого аппарата с орбиты вышла из строя система управления движения. В этом случае спускаемый аппарат закручивается относительно продольной оси и осуществляется баллистический спуск в атмосфере.

Траекторные и контролируемые характеристики движения (перегрузки и удельные тепловые потоки) также рассчитываются. Причем начальные условия движения (условия входа в плотные слои атмосферы) выбираются на основании рекомендаций.

В результате расчета аварийных режимов должны быть сформулированы требования к конструкции космического аппарата и спускаемой его части с точки зрения их самоликвидации при нештатной ситуации (принудительное разрушение или саморазрушение).

2. Описание и инструкция по применению программного комплекса START для анализа этапа выведения КА на промежуточную орбиту

2.1 Блок-схема программного комплекса START



2.2. Описание интерфейса программного комплекса START

Программный комплекс (ПК) START допускает размещение в Интернете для удобства его использования учащимися. В результате любой учащийся может иметь доступ к модулям расчета траектории выведения. Кроме этого ПК позволяет традиционно установить в компьютерном классе на одном компьютере, соединенным в локальную сеть с другими компьютерами. При загрузке начальной страницы, открывается окно для расчета атмосферной ступени, копия экрана приведена на рисунке 1. Левый столбик содержит меню пользователя, с помощью которого можно выбрать интересующий вид расчета, либо использовать дополнительные функции такие, как просмотр полной траектории ракеты, выбор текущего ракетносителя и просмотр проекции траектории полета ракеты на карту Google Maple. Чтобы просмотреть полную траекторию полета, надо последовательно пройти по всем модулям расчета ступеней ракеты. При расчете атмосферных ступеней конечные данные автоматически переносятся в начальные данные для расчета промежуточной ступени (если трехступенчатая ракета) или в начальные данные завершающей ступени (для двухступенчатой ракеты).

Расчет траектории начинается с нажатия кнопки «Моделирование».

Расчет ступеней

Атмосфер. ступени

Первая ступень

Первая и вторая

Вне атмосфер. ступени

Промежуточная ступень

Промежуточная ступень с пассивным участком

Завершающие ступени

Завершающая ступень

Ракета

Полная траектория

Двухступенчатая

Союз

Моделирование->

Задание параметров решения краевой задачи:

Шаг интегрирования,с: 0.1

Шаг варьирования по углу атаки,град: 1

Допустимая погрешность,град: 0.1

Начальное приближение максимума угла атаки,град: 0.898

Конечный угол наклона траектории,град: 25

Проектные параметры ступени

Стартовая масса,кг: 310000.0

Масса топлива, кг: 181800.0

Площадь модели,м2: 21.5

Тяга ДУ (в пустоте),Н: 4774020.0

Удельная тяга ДУ(в пустоте), м/с: 3100

Коэффициент высотности сопла: 1.175

Скорость начала аэрод. разворота: 50.0

Скорость окончания аэрод. разворота: 270.0

Аэродинамические параметры

Max: 0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4

Ска: 0.215 0.215 0.214 0.212 0.218 0.269 0.466 0.524 0.516 0.476 0.461 0.429 0.408

Рисунок 1- Главное окно программного комплекса

Описание интерфейса иллюстрируется на примере анализа возможности выведения полезной нагрузки заданной массы трехступенчатой ракетой «Союз-У» (рис. 2)

РН "Союз-У"	Основные проектные параметры	
	Первой ступени	
	Начальная масса, т:	304
	Тяга двигателей (в пустоте), кН	4093
	Удельный импульс (в пустоте), м/с	3100
	Масса топлива, т	157
	Кэфф. высотности сопла:	1.218
	Площадь мицеля, м2	11.282
	Второй ступени	
	Начальная масса, т:	131.8
	Тяга двигателей (в пустоте), кН	997
	Удельный импульс (в пустоте), м/с	3100
	Масса топлива, т	93
	Третьей ступени	
	Начальная масса, т:	32.3
	Тяга двигателей (в пустоте), кН	297
	Удельный импульс (в пустоте), м/с	3300
	Масса топлива, т	22.9

Рисунок 2- Ввод проектных параметров ракетносителя

При нажатии на ссылку «Первая ступень» появится окно расчета траектории движения ступени в плотных слоях атмосферы. В качестве входных параметров для расчета программы управления и траектории полета необходимо задать исходные данные:

- Параметры решения краевой задачи;
- Проектные параметры ступени;
- Конечный угол наклона траектории.

Копия экрана показана на рисунке 3:

Задание параметров решения краевой задачи:

Шаг интегрирования,с:	0.1
Шаг варьирования по углу атаки,град:	1
Допустимая погрешность,град:	0.1
Начальное приближение максимума угла атаки,град:	0.898
Конечный угол наклона траектории,град:	25

Проектные параметры ступени

Стартовая масса,кг:

Масса топлива, кг:

Площадь мицеля,м2:

Тяга ДУ (в пустоте),Н:

Удельная тяга ДУ(в пустоте), м/с:

Коэффициент высотности сопла:

Скорость начала аэрод. разворота:

Скорость окончания аэрод. разворота:

Аэродинамические параметры

Max:	0	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.8	2.0	2.2	2.4
Cxa:	0.215	0.215	0.214	0.212	0.218	0.269	0.466	0.524	0.516	0.476	0.461	0.429	0.408
Max:	2.6	2.8	3.0	3.2	3.4	3.6	3.8	4.0	4.2	4.4	4.6	4.8	5.0
Cxa:	0.367	0.376	0.364	0.339	0.33	0.321	0.313	0.306	0.299	0.292	0.285	0.279	0.274
Cya,град:	0.173												

Рисунок 3- Задание начальных данных для расчета атмосферной ступени

Результаты расчёта:

- Параметры полёта ракеты на момент старта,
- Параметры полёта ракеты на момент окончания вертикального участка,
- Параметры полёта ракеты на момент достижения максимума угла атаки,
- Параметры полёта ракеты на момент окончания участка аэродинамического разворота,

- Параметры полёта ракеты на момент достижения максимума скоростного напора,
- Параметры полёта ракеты на момент окончания работы ступени,
- Графики траектории угла атаки и скорости.

Копия экрана, отражающего результаты расчета, показана на рисунке 4.

Данные моделирования

	Время,с	X,м	Y,м	Скорость,м/с	Тета,град	q,	а,град	пх	пу
1	0	0	0	0	90	0	0	1,335	0
2	13,4	0	321	50	90	1 458	-0	1,428	-0
3	21,1	8	838	85	87,54	3 908	-1,593	1,5	-0,05
4	48,2	1 166	5 214	270	65,968	20 456	0	1,86	0
5	66,3	4 673	10 794	470	51,389	26 800	0	2,107	0
6	118,051	45 602	38 129	1 615	25,04	5 237	0	3,811	0

Количество итераций: 11

Тета- угол наклона траектории в стартовой системе координат

q-скоростной напор

а-угол атаки

пх-перегрузка по оси x

пу-перегрузка по оси y

1 - Момент старта

2 - Момент окончания вертикального участка полета

3 - Момент достижения максимума угла атаки

4 - Момент окончания участка аэродинамического разворота

5 - Момент достижения максимума скоростного напора

6 - Момент окончания работы ступени

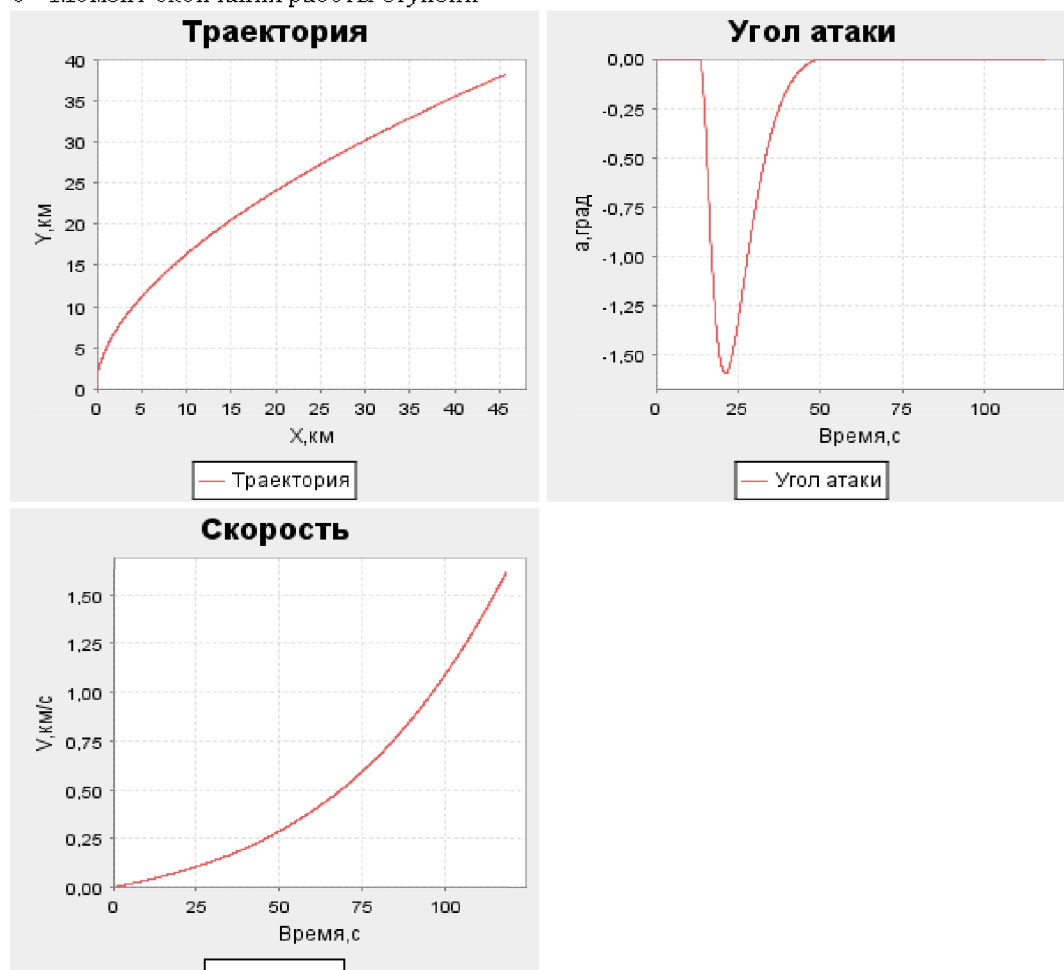


Рисунок 4 -Результаты моделирования движения атмосферной ступени

Так как рассматриваемый вариант ракеты имеет три ступени, то переходим к расчету промежуточной ступени, копия экрана показана на рисунке 5. Для расчета необходимо задать следующие входные данные:

- Параметры решения краевой задачи,
- Проектные параметры ступени,
- Начальные условия полёта.

Конечный результат моделирования атмосферной ступени автоматически перенесется в начальные условия полета для промежуточной ступени.

Задание параметров решения краевой задачи:

Шаг интегрирования,с:	0.1
Шаг варьирования по углу тангажа,град:	1
Допустимая погрешность наклона траектории,град:	0.1
Конечный угол наклона траектории,град:	10

Проектные параметры ступени

Стартовая масса,кг:

Масса топлива, кг:

Тяга ДУ (в вакууме),Н:

Удельная тяга ДУ(в пустоте), м/с:

Время сброса головного обтекателя, с:

Масса головного обтекателя, кг:

Начальные условия

Координата х,м:

Координата у,м:

Начальная скорость по х,м/с:

Начальная скорость по у,м/с:

Рисунок 5- Входные данные для моделирования движения промежуточной ступени

Копия экрана полученных результатов показан на рисунке 6.

Данные моделирования

Проекция x в конечный момент времени,м: 902 985,156

Проекция y в конечный момент времени,м: 82 501,633

Горизонтальная составляющая скорости,м/с: 5 724,756

Вертикальная составляющая скорости,м/с: 203,394

Модуль полной скорости,м/с: 5 728,368

Высота орбиты в конечный момент,м: 145 369,044

Угол наклона траектории,град: 10,001

Оптимальный параметр управления по углу тангажа β ,град: 26,33

Время работы ступени,с: 280,07

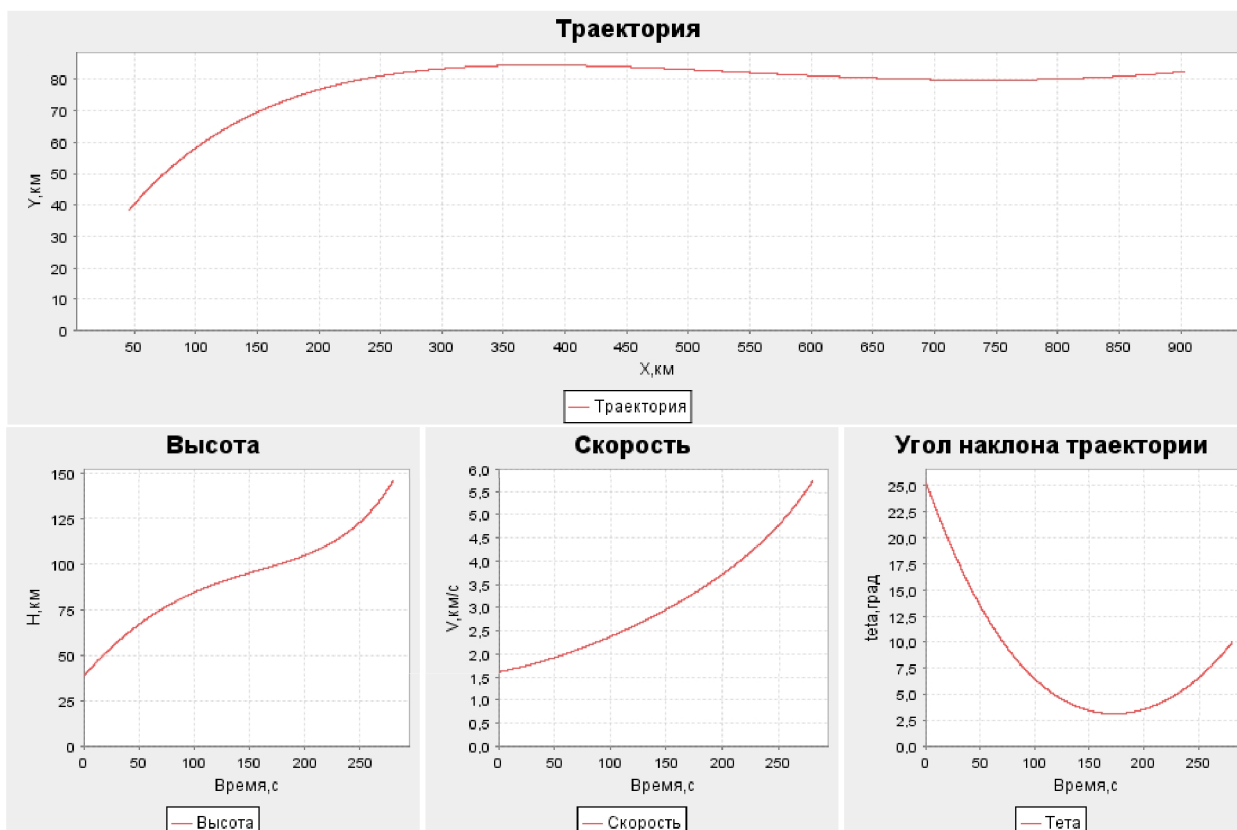


Рисунок 6- Результат моделирования движения промежуточной ступени

На рисунке 7 приведена копия экрана задания начальных параметров для расчета третьей (завершающей) ступени.

Задание параметров решения краевой задачи:

Шаг варьирования по углу тангажа,град:	1
Шаг варьирования по $\dot{b}$:	0.1
Шаг интегрирования,с:	1
Допустимая погрешность наклона траектории,град:	0.1
Допустимая погрешность по высоте,м:	10
Конечный угол наклона траектории,град:	0
Высота орбиты выведения,м:	200000

Проектные параметры ступени

Стартовая масса,кг:

Масса топлива, кг:

Тяга ДУ (в вакууме),Н:

Удельная тяга ДУ(в пустоте), м/с:

Время сброса головного обтекателя, с:

Масса головного обтекателя, кг:

Начальные условия

Координата x ,м:

Координата y ,м:

Начальная скорость по x ,м/с:

Начальная скорость по y ,м/с:

Рисунок 7- Начальные данные для расчета завершающей ступени

На рисунке 8 показана копия экрана результатов моделирования полета завершающей ступени, а на рисунке 9 показана полная траектория полета.

Данные моделирования

Проекция x в конечный момент времени,м: 1 521 903,838
Проекция y в конечный момент времени,м: 21 337,682
Горизонтальная составляющая скорости,м/с: 7 498,897
Вертикальная составляющая скорости,м/с: -1 785,356
Модуль полной скорости,м/с: 7 708,499
Высота орбиты в конечный момент,м: 200 009,993
Угол наклона траектории,град: -0

Оптимальный параметр управления по углу тангажа β ,град: -12,824
Оптимальный параметр управления по углу тангажа ν ,град: -0,672
Количество итераций: 2 044
время работы ступени,с: 94,435

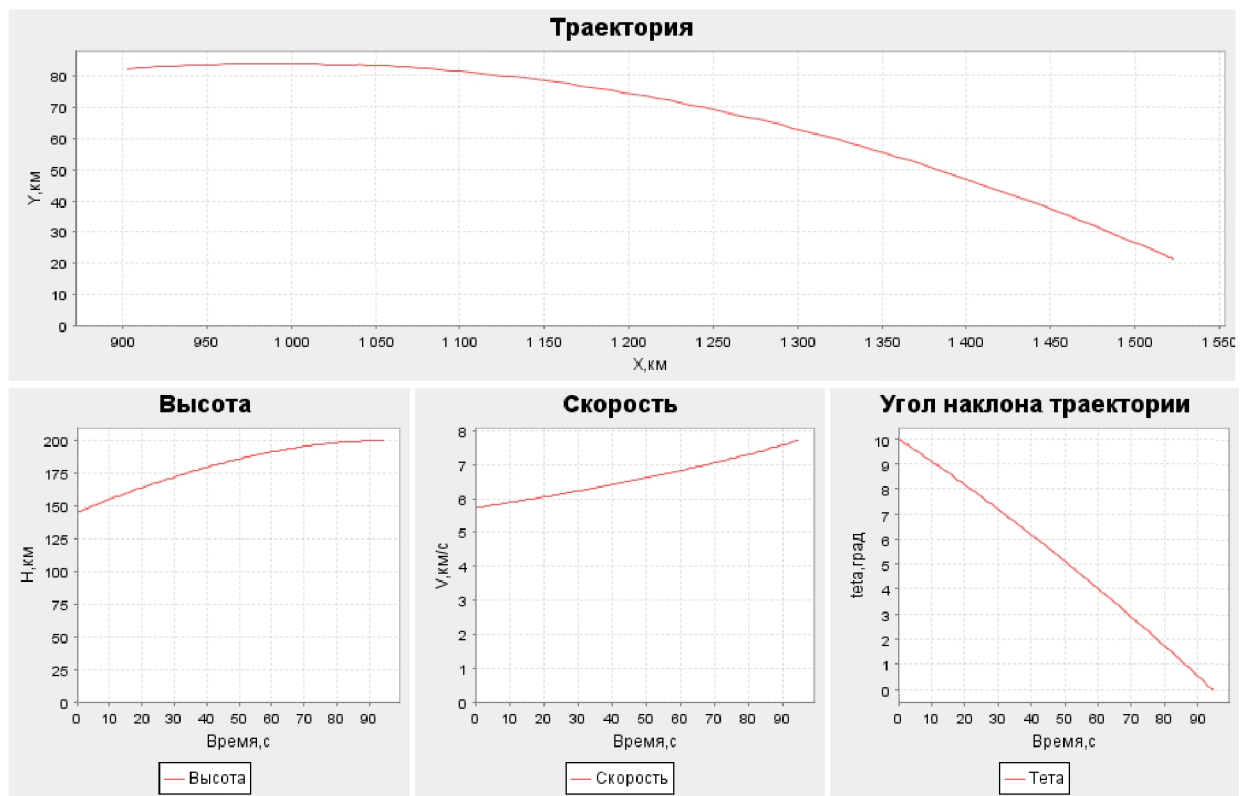


Рисунок 8- Результаты моделирования полета завершающей ступени

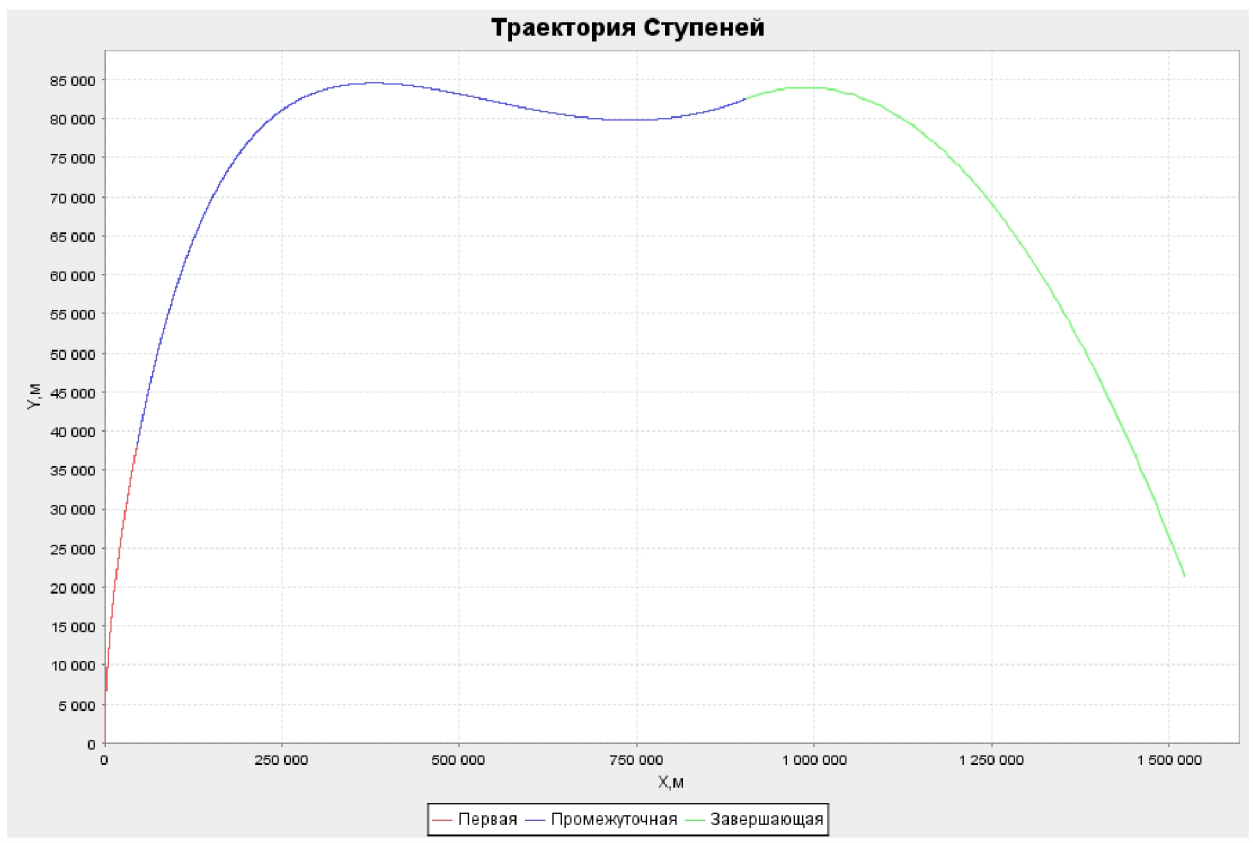


Рисунок 9- Полная траектория полета ракеты-носителя «Союз»

Для получения информации об используемых терминах, методиках и моделях следует обратиться к Приложению 1 данного практикума.

2.3 Листинг программы START

```
package rocket.servlets;

import base.stage.StageAtm;
import base.stage.StageVacuum;
import base.vector.VectorAtm;
import base.vector.VectorVacuum;
import base.earth.EarthSphere;
import base.earth.Earth;
import base.bean.BeanStageAtm;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
import javax.servlet.RequestDispatcher;
import java.io.IOException;

import rocket.bean.*;

public class ServletRocket extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException {
        if
        ((request.getAttribute("paramStageAtm1")!=null)&&(request.getAttribute("paramIntegrationatm")!=null)&&(request.g
        etAttribute("paramBoundaryStageAtm1")!=null)&&(request.getAttribute("paramStageEnd")==null)){
            ParamStageAtm paramStageAtm1 = (ParamStageAtm) request.getAttribute("paramStageAtm1");
            ParamIntegration paramintegrationatm = (ParamIntegration) request.getAttribute("paramIntegrationatm");
            ParamBoundary paramBoundaryStageAtm = (ParamBoundary)
            request.getAttribute("paramBoundaryStageAtm");
            StageAtm stageatm = new
            StageAtm(paramStageAtm1.getP(),paramStageAtm1.getP()/paramStageAtm1.getPud(),paramStageAtm1.getM0(),p
            aramStageAtm1.getMf(),paramStageAtm1.getH(),paramStageAtm1.getS(),paramStageAtm1.getCxa(),paramStageA
            tm1.getCya(),paramStageAtm1.getVstart(),paramStageAtm1.getVend());
            VectorAtm vectoratm=new VectorAtm();
            double
            iterate=Method.Gold(stageatm,vectoratm,paramBoundaryStageAtm.getStepRad(),paramintegrationatm.getStepinteg
            ration(),paramBoundaryStageAtm.getInitRad(),paramBoundaryStageAtm.getValueendRad(),paramBoundaryStageA
            tm.getEpselonRad());
            //-----
            request.getSession().setAttribute("stageatm",vectoratm);
            request.setAttribute("iterate",Double.valueOf(iterate));

            RequestDispatcher dispatcher = request.getRequestDispatcher("/?frame=stageatmresult");
            dispatcher.forward(request,response);
        }
        if
        (((request.getAttribute("paramStageInt")!=null)&&(request.getAttribute("paramIntegrationint")!=null)&&(request.g
        etAttribute("paramBoundaryStageInt")!=null)&&(request.getAttribute("paramInitInt")!=null))) {
            ParamStageVacuum paramStageInt = (ParamStageVacuum) request.getAttribute("paramStageInt");
            ParamIntegration paramintegrationint = (ParamIntegration) request.getAttribute("paramIntegrationint");
            ParamBoundary paramBoundaryStageInt = (ParamBoundary)
            request.getAttribute("paramBoundaryStageInt");
            ParamInit paramInitInt= (ParamInit) request.getAttribute("paramInitInt");
            StageVacuum stagevacuum= new
            StageVacuum(paramStageInt.getP(),paramStageInt.getP()/paramStageInt.getPud(),paramStageInt.getM0(),paramSta
            geInt.getMf(),paramStageInt.getTimefaultcowl(),paramStageInt.getMassfaultcowl());
            VectorVacuum vectorinterim=new VectorVacuum();
            double
            phi=Method.Parabola(stagevacuum,paramInitInt.getX(),paramInitInt.getY(),paramInitInt.getVx(),paramInitInt.getV
```

```

y(),vectorinterim,paramintegrationint.getStepintegration(),paramBoundaryStageInt.getStepRad(),paramBoundarySta
geInt.getValueendRad(),0);
    request.setAttribute("phi",new Double(phi));
    request.setAttribute("timework",new Double(stagevacuum.getTimework()));
    request.getSession().setAttribute("stagevacuuminterim",vectorinterim);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/?frame=stagevacuuminterimresult");
    dispatcher.forward(request,response);
}
if
(((request.getAttribute("paramStageInt")!=null)&&(request.getAttribute("paramIntegrationint")!=null)&&(request.g
etAttribute("paramBoundaryStageIntPassiv")!=null)&&(request.getAttribute("paramInitInt")!=null))) {
    ParamStageVacuum paramStageInt = (ParamStageVacuum) request.getAttribute("paramStageInt");
    ParamIntegration paramintegrationint = (ParamIntegration) request.getAttribute("paramIntegrationint");
    ParamBoundary paramBoundaryStageInt = (ParamBoundary)
request.getAttribute("paramBoundaryStageInt");
    ParamInit paramInitInt = (ParamInit) request.getAttribute("paramInitInt");
    StageVacuum stagevacuum= new
StageVacuum(paramStageInt.getP(),paramStageInt.getP()/paramStageInt.getPud(),paramStageInt.getM0(),paramSta
geInt.getMf(),paramStageInt.getTimefaultcowl(),paramStageInt.getMassfaultcowl());
    VectorVacuum vectorinterim=new VectorVacuum();
    double
phi=Method.Parabola(stagevacuum,paramInitInt.getX(),paramInitInt.getY(),paramInitInt.getVx(),paramInitInt.getV
y(),vectorinterim,paramintegrationint.getStepintegration(),paramBoundaryStageInt.getStepRad(),0,paramBoundaryS
tageInt.getValueend());
    request.setAttribute("phi",new Double(phi));
    request.getSession().setAttribute("stagevacuuminterim",vectorinterim);

    RequestDispatcher dispatcher = request.getRequestDispatcher("/?frame=stagevacuuminterimresult");
    dispatcher.forward(request,response);
}
if
(((request.getAttribute("paramStageEnd")!=null)&&(request.getAttribute("paramIntegrationend")!=null)&&(request
.getAttribute("paramBoundaryStageEndTeta")!=null)&&(request.getAttribute("paramBoundaryStageEndAltitude")
!=null)&&(request.getAttribute("paramInitEnd")!=null))) {
    ParamStageVacuum paramStageEnd = (ParamStageVacuum) request.getAttribute("paramStageEnd");
    ParamIntegration paramintegrationend = (ParamIntegration) request.getAttribute("paramIntegrationend");
    ParamBoundary paramBoundaryStageEndTeta = (ParamBoundary)
request.getAttribute("paramBoundaryStageEndTeta");
    ParamBoundary paramBoundaryStageEndAltitude = (ParamBoundary)
request.getAttribute("paramBoundaryStageEndAltitude");
    ParamInit paramInitEnd= (ParamInit) request.getAttribute("paramInitEnd");
    StageVacuum stagevacuum= new
StageVacuum(paramStageEnd.getP(),paramStageEnd.getP()/paramStageEnd.getPud(),paramStageEnd.getM0(),para
mStageEnd.getMf(),paramStageEnd.getTimefaultcowl(),paramStageEnd.getMassfaultcowl());
    VectorVacuum vectorend=new VectorVacuum();
    double[]
result=Method.Newton(stagevacuum,paramInitEnd.getX(),paramInitEnd.getY(),paramInitEnd.getVx(),paramInitEn
d.getVy(),vectorend,paramintegrationend.getStepintegration(),paramBoundaryStageEndTeta.getStepRad(),paramBo
undaryStageEndAltitude.getStep(),paramBoundaryStageEndTeta.getValueendRad(),paramBoundaryStageEndAltitu
de.getValueend(),paramBoundaryStageEndTeta.getEpsilonRad(),paramBoundaryStageEndAltitude.getEpsilon());
    request.setAttribute("iterate",Double.valueOf(result[1]));
    request.setAttribute("phi",new Double(result[0]));
    request.setAttribute("b",new Double(result[2]));
    request.setAttribute("timework",new Double(stagevacuum.getTimework()));
    request.getSession().setAttribute("stagevacuumend",vectorend);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/?frame=stagevacuumendresult");
    dispatcher.forward(request,response);
}
if
((request.getAttribute("paramorbit")!=null)&&(request.getAttribute("paramStageAtm1")!=null)&&(request.getAttri
bute("paramIntegrationatm")!=null)&&(request.getAttribute("paramBoundaryStageAtm")!=null)&&(request.getAtt
ribute("paramStageEnd")!=null)&&(request.getAttribute("paramIntegrationend")!=null)&&(request.getAttribute("p
aramBoundaryStageEndTeta")!=null)&&(request.getAttribute("paramBoundaryStageEndAltitude")!=null)) {

```

```

//Для первой ступени
ParamStageAtm paramStageAtm1 = (ParamStageAtm) request.getAttribute("paramStageAtm1");
ParamIntegration paramintegrationatm = (ParamIntegration) request.getAttribute("paramIntegrationatm");
ParamBoundary paramBoundaryStageAtm = (ParamBoundary)
request.getAttribute("paramBoundaryStageAtm");
StageAtm stageatm = new
StageAtm(paramStageAtm1.getP(),paramStageAtm1.getP()/paramStageAtm1.getPud(),paramStageAtm1.getM0(),p
aramStageAtm1.getMf(),paramStageAtm1.getH(),paramStageAtm1.getS(),paramStageAtm1.getCxa(),paramStageA
tm1.getCya(),paramStageAtm1.getVstart(),paramStageAtm1.getVend());
//Для завершающей ступени
ParamStageVacuum paramStageEnd = (ParamStageVacuum) request.getAttribute("paramStageEnd");
ParamIntegration paramintegrationend = (ParamIntegration) request.getAttribute("paramIntegrationend");
ParamBoundary paramBoundaryStageEndTeta = (ParamBoundary)
request.getAttribute("paramBoundaryStageEndTeta");

ParamBoundary paramBoundaryStageEndAltitude = (ParamBoundary)
request.getAttribute("paramBoundaryStageEndAltitude");

StageVacuum stagevacuum= new
StageVacuum(paramStageEnd.getP(),paramStageEnd.getP()/paramStageEnd.getPud(),paramStageEnd.getM0(),para
mStageEnd.getMf(),paramStageEnd.getTimefaultcowl(),paramStageEnd.getMassfaultcowl());

ParamOrbit paramorbit =(ParamOrbit )request.getAttribute("paramorbit");
//Поиск оптимального значения

double tetaatm_start,tetaatm_end,epselon,alfa,betta,falfa=0,fbetta=0;
boolean flagbetta=true;
boolean flagalfa=true;
VectorAtm vectoratm=new VectorAtm();
VectorVacuum vectorend=new VectorVacuum();
tetaatm_start=10/57.3;
tetaatm_end=60/57.3;
epselon=0.1/57.3;
double[] result= new double[3];

while (true){
    alfa=tetaatm_start+(2/(3+Math.sqrt(5)))*(tetaatm_end-tetaatm_start);
    betta=tetaatm_start+(2/(1+Math.sqrt(5)))*(tetaatm_end-tetaatm_start);

    if (flagalfa){
        vectoratm=new VectorAtm();

Method. Gold(stageatm,vectoratm,paramBoundaryStageAtm.getStepRad(),paramintegrationatm.getStepintegration()
,paramBoundaryStageAtm.getInitRad(),alfa,paramBoundaryStageAtm.getEpselonRad());

        vectorend=new VectorVacuum();

result=Method. Newton(stagevacuum,vectoratm.getXend(),vectoratm.getYend(),vectoratm.getVxend(),vectoratm.get
Vyend(),vectorend,paramintegrationend.getStepintegration(),paramBoundaryStageEndTeta.getStepRad(),paramBou
ndaryStageEndAltitude.getStep(),paramBoundaryStageEndTeta.getValueendRad(),paramBoundaryStageEndAltitud
e.getValueend(),paramBoundaryStageEndTeta.getEpselonRad(),paramBoundaryStageEndAltitude.getEpselon());

        falfa=vectorend.getVend();
    }
    if (flagbetta){
        vectoratm=new VectorAtm();

Method. Gold(stageatm,vectoratm,paramBoundaryStageAtm.getStepRad(),paramintegrationatm.getStepintegration()
,paramBoundaryStageAtm.getInitRad(),betta,paramBoundaryStageAtm.getEpselonRad());

        vectorend=new VectorVacuum();
        result =
Method. Newton(stagevacuum,vectoratm.getXend(),vectoratm.getYend(),vectoratm.getVxend(),vectoratm.getVyend

```

```

(),vectorend,paramintegrationend.getStepintegration(),paramBoundaryStageEndTeta.getStepRad(),paramBoundaryS
tageEndAltitude.getStep(),paramBoundaryStageEndTeta.getValueendRad(),paramBoundaryStageEndAltitude.getV
alueend(),paramBoundaryStageEndTeta.getEpselonRad(),paramBoundaryStageEndAltitude.getEpselon());

```

```

        fbeta=vectorend.getVend();
    }

    if (fbeta>falfa){
        tetaatm_start=alfa;
        flagalfa=false;
        flagbeta=true;
        falfa=fbetta;
    }else{
        tetaatm_end=betta;
        flagalfa=true;
        flagbeta=false;
        fbeta=falfa;
    }
    if (Math.abs(tetaatm_start-tetaatm_end)<epselon) break;
}
double A=Math.asin(Math.cos(paramorbit.getI())/Math.cos(paramorbit.getLatitudestart()));
double vorb=Math.sqrt(Earth.mu/(EarthSphere.r+paramBoundaryStageEndAltitude.getValueend()));
double vzeml=Earth.w*EarthSphere.r*Math.cos(paramorbit.getLatitudestart());
double vtreb=Math.sqrt(vorb*vorb+vzeml*vzeml-2*vorb*vzeml*Math.sin(A));

request.setAttribute("vtreb",new Double(vtreb));

request.setAttribute("vend",Double.valueOf(vectorend.getVend()));

request.setAttribute("amax",Double.valueOf(((BeanStageAtm)vectoratm.getVector().get(vectoratm.getIndexmaxa()
)).getA()));
request.setAttribute("teta",Double.valueOf(tetaatm_start));
request.setAttribute("phi",new Double(result[0]));
request.setAttribute("b",new Double(result[2]));

RequestDispatcher dispatcher = request.getRequestDispatcher("/?frame=rocketresult");
dispatcher.forward(request,response);
}
}

```

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {

```

```

    }
}
package rocket.servlets;

```

```

import base.rungekutta.RungeKuttaStageVacuum;
import base.rungekutta.RungeKuttaStageAtmLoad;
import base.earth.EarthSphere;
import base.stage.StageVacuum;
import base.stage.StageAtm;
import base.vector.VectorVacuum;
import base.vector.VectorAtm;
import base.math.M;
import methods.linsolve.LineSolveLU;
import methods.boundary.GoldSection;

```

```

public class Method {
    public static double[] Newton(StageVacuum stagevacuum,double xstart,double ystart,double vxstart,double
vystart,VectorVacuum vectorvacuum,double stepintegration,double stepvariationphi,double stepvariationb,double
tetaend,double altitudeend,double epselonteta,double epselonheight){
        //Начальные условия

```

```

double[] y= new double[5];
//Выбор начального угла тангажа
double phi=Math.atan(vystart/vxstart);
//Выбор начального праметра b
double b=(-1)*vystart/vxstart;
//Инициализация блока интегрирования
RungeKuttaStageVacuum rungevaccum;
//Счетчик итераций
int iterate=1;
while (iterate<=2999){
    //Инициализация блока интегрирования
    rungevaccum= new RungeKuttaStageVacuum(stagevacuum,phi,b);
    //Начальные условия
    y[1]=xstart;
    y[2]=ystart;
    y[3]=vxstart;
    y[4]=vystart;
    //Интегрируем
    vectorvacuum.getVector().clear();
    //сохраняем в полной таблице начальные данные
    rungevaccum.HelpVariable(0,y,vectorvacuum);
    rungevaccum.SolveSystemRungeKutta(0,stagevacuum.getTimework(),stepintegration,y,vectorvacuum);
    double teta=Math.atan(y[4]/y[3])+EarthSphere.AngleCKStart(y[1],y[2]);
    double height=EarthSphere.HCKStart(y[1],y[2]);
    if ((Math.abs(teta-tetaend)<epselonteta)&&(Math.abs(height-altitudeend)<epselonheight)) break;
    //Инициализация блока интегрирования
    rungevaccum= new RungeKuttaStageVacuum(stagevacuum,phi+stepvariationphi,b);
    //Начальные условия
    y[1]=xstart;
    y[2]=ystart;
    y[3]=vxstart;
    y[4]=vystart;
    //Интегрируем
    rungevaccum.SolveSystemRungeKutta(0,stagevacuum.getTimework(),stepintegration,y,null);
    double dtetaphi=(Math.atan(y[4]/y[3])+EarthSphere.AngleCKStart(y[1],y[2]))/(stepvariationphi);
    double dheightphi=(EarthSphere.HCKStart(y[1],y[2]))/(stepvariationphi);
    rungevaccum= new RungeKuttaStageVacuum(stagevacuum,phi,b+stepvariationb);
    //Начальные условия
    y[1]=xstart;
    y[2]=ystart;
    y[3]=vxstart;
    y[4]=vystart;
    //Интегрируем
    rungevaccum.SolveSystemRungeKutta(0,stagevacuum.getTimework(),stepintegration,y,null);
    double dtetab=(Math.atan(y[4]/y[3])+EarthSphere.AngleCKStart(y[1],y[2]))/(stepvariationb);
    double dheightb=(EarthSphere.HCKStart(y[1],y[2]))/(stepvariationb);

    double[][] aa={{dtetaphi,dtetab},{dheightphi,dheightb}};
    double[] bb={tetaend-teta,altitudeend-height};
    double[] delta=LineSolveLU.Solve2(aa,bb);
    phi=phi+delta[0];
    b=b+delta[1];
    iterate++;
}
double[] result= new double[3];
result[0]=phi;
result[1]=iterate;
result[2]=b;
return result;
}

public static double Parabola(StageVacuum stagevacuum,double xstart,double ystart,double vxstart,double
vystart,VectorVacuum vectorvacuum,double stepintegration,double stepvariationphi,double tetaend,double
altitudeend){

```

```

//Начальные условия
double[] y= new double[5];
//Возьмем три разных угла тангажа
double[] phimatrix= new double[3];
phimatrix[0]=Math.atan(vystart/vxstart);//угол тангажа с предыдущей ступени
phimatrix[1]=phimatrix[0]+stepvariationphi;//угол тангажа с предыдущей ступени плюс шаг вариации
phimatrix[2]=phimatrix[0]-stepvariationphi;//угол тангажа с предыдущей ступени минус шаг вариации
//В этой матрице будем сохранять углы наклона траектории
double[] tetamatrix= new double[3];
//цикл решения краевой задачи
for(int iterate=1;iterate<=3;iterate++){
    //Инициализация блока интегрирования
    RungeKuttaStageVacuum rungevaccum= new RungeKuttaStageVacuum(stagevacuum,phimatrix[iterate-
1],0);
    //Начальные условия
    y[1]=xstart;
    y[2]=ystart;
    y[3]=vxstart;
    y[4]=vystart;
    rungevaccum.SolveSystemRungeKutta(0,stagevacuum.getTimework(),stepintegration,y,null);
    tetamatrix[iterate-1]=Math.atan(y[4]/y[3])+EarthSphere.AngleCKStart(y[1],y[2]);
}
//Решение системы линейных уравнений
double[][] amatrix={{phimatrix[0]*phimatrix[0],phimatrix[0],1},
                    {phimatrix[1]*phimatrix[1],phimatrix[1],1},
                    {phimatrix[2]*phimatrix[2],phimatrix[2],1}};
double[] endmatrix=LineSolveLU.Solve(amatrix,tetamatrix);
//Определение коэффициентов a,b,c
double aa=endmatrix[0];
double bb=endmatrix[1];
double cc=endmatrix[2]-tetaend;
//Решение квадратного уравнения
double phi=(-bb+Math.sqrt(M.v2(bb)-4*aa*cc))/(2*aa);
RungeKuttaStageVacuum rungevaccum= new RungeKuttaStageVacuum(stagevacuum,phi,0);
y[1]=xstart;
y[2]=ystart;
y[3]=vxstart;
y[4]=vystart;
rungevaccum.SolveSystemRungeKutta(0,stagevacuum.getTimework(),stepintegration,y,vectorvacuum);
return phi;
}

```

```

public static double Gold(StageAtm stageatm,VectorAtm vectoratm,double stepvariationattack,double
stepintegration,double amax,double tetaend,double epselonteta){

```

```

    //Начальные условия
    double[] y= new double[5];
    //Начальные условия для краевой задачи
    double[] opt ={stepvariationattack,amax,0,tetaend,epselonteta,0};
    //Инициализация блока интегрирования
    RungeKuttaStageAtmLoad rungeatm;
    //Счетчик итераций
    int iterate=1;
    //Решение краевой задачи-----
    while (iterate<=100){
        //Инициализация блока интегрирования
        rungeatm = new RungeKuttaStageAtmLoad(stageatm,opt[1]);
        //Начальные условия
        y[0]=0;y[1]=0;y[2]=0;y[3]=0;y[4]=M.pina2();
        //Полная таблица результата
        vectoratm.clear();
        //сохраняем в полной таблице начальные данные
        rungeatm.HelpVariable(0,y,vectoratm);
        //Интегрируем
    }
}

```

```

    rungeatm.SolveSystemRungeKutta(0,stageatm.getTimework(),stepintegration,y,vectoratm);
    //Решаем краевую задачу, если нашли решения то выходим из цикла
    opt[2]=y[4];
    if (GoldSection.Value(opt)) break;
    //Увеличиваем счетчик итераций
    iterate++;
}
//-----
return iterate;
}

```

```

import methods.rungekutta.RungeKutta;
import base.math.M;
import base.earth.Environment;
import base.stage.StageAtm;
import base.earth.EarthSphere;
import base.earth.EarthTable;
import base.earth.Earth;
import base.vector.VectorAtm;

```

//класс для решения диф. уравнений движения атмосферной ступени в перегрузках

```

public class RungeKuttaStageAtmLoad extends RungeKutta{
    private StageAtm stageatm;//проектные параметры ступени
    private double amax;//максимальный угол атаки

    private double current_p; //текущая тяга двигателей
    private double current_a; //текущий угол атаки
    private double current_xa; //текущая сила лобового сопротивления
    private double current_ya; //текущая подъемная сила
    private double current_m; //текущая масса ступени
    private double current_nx; //текущая перегрузка по оси x
    private double current_ny; //текущая перегрузка по оси y
    private double current_q; //текущий скоростной напор

    public RungeKuttaStageAtmLoad(StageAtm stageatm, double amax) {
        this.stageatm = stageatm;
        this.amax = amax;
    }

    //-y[1] координата X в стартовой системе координат
    //-y[2] координата Y в стартовой системе координат
    //-y[3] скорость V в траекторной системе координат
    //-y[4] угол наклона траектории
    public double F(int i,double x, double[] y){
        switch (i) {
            case 1:
                return y[3]*M.c(y[4]);
            case 2:
                return y[3]*Math.sin(y[4]);
            case 3:
                return EarthSphere.g(y[2])*(current_nx-Math.sin(y[4]));
            case 4:
                if (y[3] == 0) return 0;
                if (y[3] != 0) return (EarthSphere.g(y[2])/y[3])*(current_ny-Math.cos(y[4]));
        }
        return 0;
    }

    public void SystemRungeKuttaStep(double x,double h, double[] y, VectorAtm table){
        HelpVariable(x,y,null);
        super.SystemRungeKuttaStep(x, h, y);
        HelpVariable(x+h,y,table);
    }
}

```



```

    }

    public void HelpVariable(double x, double[] y, VectorAtm table ){
        //TODO вопрос по модели, определение числа маха
        current_a=stageatm.angleofattack(amax,x,y[3]);
        current_p=stageatm.traction(EarthTable.pressure(0),EarthTable.pressure(y[2]));
        current_q=Environment.incidentflowH(y[2],y[3]);
        current_xa=Environment.drag(stageatm.getCxa(Earth.mah(y[2],y[3])),current_q,stageatm.getS());
        current_ya=Environment.lift(stageatm.getCya(),current_q,stageatm.getS());
        current_m=stageatm.mass(x);
        current_nx=Environment.axleload(current_p-current_xa,current_m,EarthSphere.g(y[2]));
        current_ny=Environment.axleload(current_ya+current_p,current_m,EarthSphere.g(y[2]))*Math.sin(current_a);
        if (table !=
null)table.add(x,y[1],y[2],y[3],y[4],current_a,current_nx,current_ny,current_p,current_xa,current_ya,current_m,current_q,stageatm.getVstart(),stageatm.getVend());

    }

    //метод решения системы диф. уравнений
    //xinit-начальное значение переменной по которой интегрируем
    //xend-конечное значение переменной по которой интегрируем
    //h-шаг интегрирования
    //y-массив начальных данных
    public void SolveSystemRungeKutta(double xinit, double xend, double h, double[] y, VectorAtm table){
        double xcurrent=xinit;
        double hcurrent=h;
        while (true){
            //если мы попали в окрестность для выхода то выходим из цикла
            if (Math.abs(xend-xcurrent)<0.001) break;
            //корректируем шаг на выходе
            if ((xcurrent+hcurrent)>xend) hcurrent=hcurrent/2;
            if ((xcurrent+hcurrent)>xend) hcurrent=hcurrent/4;
            if ((xcurrent+hcurrent)>xend) hcurrent=hcurrent/5;
            //-----
            //расчитываем значение переменных на значении xcurrent+hcurrent
            SystemRungeKuttaStep(xcurrent,hcurrent,y,table);
            //корректируем текущие значение
            xcurrent=xcurrent+hcurrent;
        }
    }

}

package base.rungekutta;

import methods.rungekutta.RungeKutta;
import base.stage.StageVacuum;
import base.earth.EarthSphere;
import base.vector.VectorVacuum;

//класс для решения диф. уравнений движения ступени в не атмосферы
public class RungeKuttaStageVacuum extends RungeKutta{

    private StageVacuum stagevacuum;//проектные параметры ступени
    private double phi0;//начальный угол тангажа
    private double b;//параметр b

    private double current_p;    //текущая тяга двигателей
    private double current_phi;  //текущий угол тангажа
    private double current_g;    //текущие Земное ускорение
    private double current_epselon; //текущий угол пути

    public RungeKuttaStageVacuum(StageVacuum stagevacuum, double phi0, double b) {

```

```

        this.stagevacuum = stagevacuum;
        this.phi0 = phi0;
        this.b = b;
    }

    public double F(int i, double x, double[] y){
        switch (i) {
            case 1:
                return y[3];
            case 2:
                return y[4];
            case 3:
                return current_p*Math.cos(current_phi)-current_g*Math.sin(current_epselon);
            case 4:
                return current_p*Math.sin(current_phi)-current_g*Math.cos(current_epselon);
        }
        return 0;
    }

    public void SystemRungeKuttaStep(double x, double h, double[] y, VectorVacuum table){
        HelpVariable(x, y, null);
        super.SystemRungeKuttaStep(x, h, y);
        HelpVariable(x+h, y, table);
    }

    public void HelpVariable(double x, double[] y, VectorVacuum table){
        current_phi=Math.atan(Math.tan(phi0)+b*x/stagevacuum.getTimework());
        current_g=EarthSphere.g(EarthSphere.HCKStart(y[1], y[2]));
        current_p=stagevacuum.acceleration(x);
        current_epselon=EarthSphere.AngleCKStart(y[1], y[2]);
        if (table != null)
            table.add(x, y[1], y[2], y[3], y[4], EarthSphere.HCKStart(y[1], y[2]), Math.atan(y[4]/y[3])+EarthSphere.AngleCKStart(y[1], y[2]));
    }

    //метод решения системы диф. уравнений
    //xinit-начальное значение переменной по которой интегрируем
    //xend-конечное значение переменной по которой интегрируем
    //h-шаг интегрирования
    //y-массив начальных данных
    public void SolveSystemRungeKutta(double xinit, double xend, double h, double[] y, VectorVacuum table){
        double xcurrent=xinit;
        double hcurrent=h;
        while (true){
            //если мы попали в окрестность для выхода то выходим из цикла
            if (Math.abs(xend-xcurrent)<0.001) break;
            //корректируем шаг на выходе
            if ((xcurrent+hcurrent)>xend) hcurrent=hcurrent/2;
            if ((xcurrent+hcurrent)>xend) hcurrent=hcurrent/4;
            if ((xcurrent+hcurrent)>xend) hcurrent=hcurrent/5;
            //-----
            //расчитываем значение переменных на значении xcurrent+hcurrent
            this.SystemRungeKuttaStep(xcurrent, hcurrent, y, table);
            //корректируем текущие значения
            xcurrent=xcurrent+hcurrent;
        }
    }
}

package base.stage;

import methods.newton.Newton;

public class NewtonAtmTime extends Newton{

```

```

private double p;
private double g;
private double dm;
private double m0;
private double v;
private double t1;
private double cxa;
private double s;
private double qcr;

public NewtonAtmTime(double p, double g, double dm, double m0, double v, double t1, double cxa, double s,
double qcr) {
    this.p = p;
    this.g = g;
    this.dm = dm;
    this.m0 = m0;
    this.v = v;
    this.t1 = t1;
    this.cxa = cxa;
    this.s = s;
    this.qcr = qcr;
}

public double F(double x){
    double xa=cxa*s*qcr;
    double m=m0-dm*x;
    double m1=m0-dm*t1;
    return (Math.log(m)-Math.log(m1))*(xa/dm-p/dm)-g*(x-t1)-v;
}

public double dF(double x){
    double xa=cxa*s*qcr;
    double m=m0-dm*x;
    return (p-xa)*(1/m)-g;
}
}

package methods.linsolve;
//Класс для решения системы линейных уравнений с помощью LU-разложения
public class LineSolveLU {
    //-----Для матрицы 3 на 3-----
    //LU разложение матрицы
    //a-входная матрица параметров
    //возвращаемая матрица- это матрица содержащая LU
    public static double[][] LU(double [][] a){
        double[][] lu={{0,0,0},{0,0,0},{0,0,0}};

        lu[0][0]= a[0][0]; //первая строка u
        lu[0][1]= a[0][1];
        lu[0][2]= a[0][2];

        lu[1][0]=a[1][0]/lu[0][0];
        lu[1][1]=a[1][1]-lu[1][0]*lu[0][1];
        lu[1][2]=a[1][2]-lu[1][0]*lu[0][2];

        lu[2][0]=a[2][0]/lu[0][0];
        lu[2][1]=(a[2][1]-lu[2][0]*lu[0][1])/lu[1][1];
        lu[2][2]=a[2][2]-lu[2][0]*lu[0][2]-lu[2][1]*lu[1][2];

        return lu;
    }
    //Возвращает решение системы линейных уравнений
    //a-матрица коэффициентов
    //b-матрица правых частей

```

```

public static double[] Solve(double [][] a,double[] b){
    double[][] lu=LU(a);
    double[] z={0,0,0};
    double[] x={0,0,0};
    z[0]=b[0];
    z[1]=b[1]-lu[1][0]*z[0];
    z[2]=b[2]-lu[2][0]*z[0]-lu[2][1]*z[1];

    x[2]=z[2]/lu[2][2];
    x[1]=(z[1]-lu[1][2]*x[2])/lu[1][1];
    x[0]=(z[0]-lu[0][1]*x[1]-lu[0][2]*x[2])/lu[0][0];

    return x;
}
//-----
//-----Для матрицы 2 на 2-----
//LU разложение матрицы
//a-входная матрица параметров
//возвращаемая матрица- это матрица содержащая LU
public static double[][] LU2(double [][] a){
    double[][] lu={{0,0},{0,0}};

    lu[0][0]= a[0][0]; //первая строка u
    lu[0][1]= a[0][1];

    lu[1][0]=a[1][0]/lu[0][0];
    lu[1][1]=a[1][1]-lu[1][0]*lu[0][1];

    return lu;
}
//Возвращает решение системы линейных уравнений
//a-матрица коэффициентов
//b-матрица правых частей
public static double[] Solve2(double [][] a,double[] b){
    double[][] lu=LU2(a);
    double[] z={0,0};
    double[] x={0,0};
    z[0]=b[0];
    z[1]=b[1]-lu[1][0]*z[0];

    x[1]=z[1]/lu[1][1];
    x[0]=(z[0]-lu[0][1]*x[1])/lu[0][0];

    return x;
}
//-----
}
package methods.boundary;

import base.math.M;

//класс реализует методы золотого сечения
public class GoldSection {
    //метод для поиска конкретного значения
    //opt[0]-шаг варьирования
    //opt[1]-параметр
    //opt[2]-получившееся значение
    //opt[3]-значение которого надо достиг
    //opt[4]-допустимая погрешность
    //opt[5]-предыдущая разность
    //Если результат получен то возвращает true
    public static boolean Value(double[] opt){
        //Вычисляем разность

```

```

double delta=opt[2]-opt[3];
//Если искомое значение входит в окрестность оптимального решения то решение найдено
if (Math.abs(delta)<=opt[4]) return true;
//если нет
else{
    //если предыдущая разность отличается знаком от текущей , значит мы перешагнули оптимальное
    решение, делим шаг пополам
    if (M.sign(delta)!=M.sign(opt[5])) opt[0]=opt[0]/2;
    //Если разность положительна то увеличиваем параметр на шаг
    if (delta>0) {
        opt[1]=opt[1]+opt[0];
        opt[5]=delta;
    //Если разность отрицательная то уменьшаем параметр на шаг
    }else{
        opt[1]=opt[1]-opt[0];
        opt[5]=delta;
    }
    return false;
}
}
//метод для поиска наибольшего значения значения
//opt[0]-шаг варьирования
//opt[1]-параметр
//opt[2]-получившееся значение
//opt[3]-предыдущие значение
//opt[4]-допустимая погрешность
//opt[5]-предыдущий знак
//Если результат получен то возвращает true
public static boolean Max(double[] opt){
    double dv=opt[2]-opt[3];
    //если разность меньше одного градуса
    if (Math.abs(opt[0])<(0.1/57.3)) return true;
    else{
        if (dv<0) {
            opt[0]=opt[0]/2;
            opt[5]=opt[5]*(-1);
        }
        if (opt[5]>0) {
            opt[1]=opt[1]+opt[0];
            opt[3]=opt[2];
        }else{
            opt[1]=opt[1]-opt[0];
            opt[3]=opt[2];
        }
    }
    return false;
}
}
package methods.rungekutta;

```

```

//Класс для решения системы диф. уравнений методом РунгеКутта 4 порядка
public abstract class RungeKutta {
    //Правые части диф. уравнений
    //i-номер уравнения
    //x-значение переменной по которой интегрируем
    //y-массив начальных данных
    public double F(int i,double x, double[] y ){
        return 0;
    }
}

```

/******

Один шаг метода Рунге-Кутты четвертого порядка для решения системы дифференциальных уравнений.

Алгоритм совершает один шаг метода для системы диффузов $y[i]'=F(i,x,y)$ для $i=1..n$

Начальная точка имеет координаты $(x,y[1], ..., y[n])$

После выполнения алгоритма в переменной y содержится состояние системы в точке $x+h$

*****/

```
public void SystemRungeKuttaStep(double x,double h, double[] y ){
```

```
    int n=y.length-1;
```

```
    double yt[]= new double[n+1];
    double k1[]= new double[n+1];
    double k2[]= new double[n+1];
    double k3[]= new double[n+1];
    double k4[]= new double[n+1];
```

```
    int i =1;
```

```
    while (i<=n) {
        k1[i] = h*this.F(i, x, y);
        i++;
    }
```

```
    i=1;
```

```
    while (i<=n) {
        yt[i] = y[i]+0.5*k1[i];
        i++;
    }
```

```
    i=1;
```

```
    while (i<=n) {
        k2[i] = h*this.F(i, x+h*0.5, yt);
        i++;
    }
```

```
    i=1;
```

```
    while (i<=n) {
        yt[i] = y[i]+0.5*k2[i];
        i++;
    }
```

```
    i=1;
```

```
    while (i<=n) {
        k3[i] = h*this.F(i, x+h*0.5, yt);
        i++;
    }
```

```
    i=1;
```

```
    while (i<=n){
        yt[i] = y[i]+k3[i];
        i++;
    }
```

```
    i=1;
```

```
    while (i<=n) {
        k4[i] = h*this.F(i, x+h, yt);
        i++;
    }
```

```
    i=1;
```

```
    while (i<=n) {
        y[i] = y[i]+(k1[i]+2.0*k2[i]+2.0*k3[i]+k4[i])/6;
        i++;
    }
```

```
}
```

```

}

package base.matrix;
import base.earth.Earth;
import base.math.M;
//Класс предназначен для перехода между системами координат
public class CK {
    //Пересчет из инерциальной ск в гринвичскую ск
    //ck-{x,y,z,vx,vy,vz}
    //result-{x,y,z,vx,vy,vz}
    //s-звездное время,рад
    public static double[] InertcToGrinvich(double[] ck,double s){
        double[] result = {0,0,0,0,0,0};
        result[0]=Math.cos(s)*ck[0]+Math.sin(s)*ck[1];
        result[1]=(-1)*Math.sin(s)*ck[0]+Math.cos(s)*ck[1];
        result[2]=ck[2];
        result[3]=Math.cos(s)*ck[3]+Math.sin(s)*ck[4]+(Earth.w)*ck[1];
        result[4]=(-1)*Math.sin(s)*ck[3]+Math.cos(s)*ck[4]-(Earth.w)*ck[0];
        result[5]=ck[5];
        return result;
    }
    //Пересчет из гринвичскую ск в земную
    //ck-{x,y,z,vx,vy,vz}
    //result -{longitude,latitude}
    public static double[] GrinvichToEarth(double[] ck){
        double[] result = {0,0};

        if ((ck[0]>0) & (ck[1]>=0)) { result[0]=Math.atan(ck[1]/ck[0]);} // Первая четверть
        if ((ck[0]<=0) & (ck[1]>0)) { result[0]=Math.PI/2+Math.atan(Math.abs(ck[0])/ck[1]);} // Вторая
четверть
        if ((ck[0]>0) & (ck[1]<=0)) { result[0]=(-1)*Math.atan(Math.abs(ck[1])/ck[0]);} // Третья четаерть
        if ((ck[0]<=0) & (ck[1]<0)) { result[0]= -Math.PI/2-Math.atan(Math.abs(ck[0])/Math.abs(ck[1]));} // Четвертая
четверть

        if (ck[2]>=0) { result[1]= Math.atan(ck[2]/M.vec(ck[0],ck[1],0));}
        if (ck[2]<0) { result[1]= (-1)*Math.atan(Math.abs(ck[2])/M.vec(ck[0],ck[1],0));}

        return result;
    }
    //Пересчет из инерциальной ск в земную
    //ck-{x,y,z,vx,vy,vz}
    //result -{longitude,latitude}
    //s-звездное время,рад
    public static double[] InertcToEarth(double[] ck,double s){
        return GrinvichToEarth(InertcToGrinvich(ck,s));
    }
}

package base.earth;

import base.earth.Earth;
//В этом классе описаны методы, моделирующие воздействие окружающей среды на движущейся ЛА
public class Environment {
    //Возвращает величину набегающего потока,кг / м*с2
    //г-плотность, кг/м3
    //v-скорость, м/с
    public static double incidentflow(double r,double v){
        return r*v*v/2;
    }
    //Возвращает величину набегающего потока,кг / м*с2
    //h-высота, м
    //v-скорость, м/с
    public static double incidentflowH(double h,double v){

```

```

        return Earth.densityExp(h)*v*v/2;
    }
    //Возвращает величину лобового сопротивления,Н
    //q-величина набегающего потока,кг / м*с2
    //s-площадь миделя, м2
    //сха-безразмерный коэффициент, вычисленный по оси x скоростной системы координат
    public static double drag(double сха,double q,double s){
        return сха*q*s;
    }
    //Возвращает величину лобового сопротивления,Н
    //v-скорость, м/с
    //h-высота, м
    //s-площадь миделя, м2
    //сха-безразмерный коэффициент, вычисленный по оси x скоростной системы координат
    public static double dragHV(double сха,double h,double v,double s){
        return сха*incidentflowH(h,v)*s;
    }
    //Возвращает величину подъемной силы,Н
    //q-величина набегающего потока,кг / м*с2
    //s-площадь миделя, м2
    //суа-безразмерный коэффициент, вычисленный по оси y скоростной системы координат
    public static double lift(double суа,double q,double s){
        return суа*q*s;
    }
    //Возвращает величину подъемной силы,Н
    //v-скорость, м/с
    //h-высота, м
    //s-площадь миделя, м2
    //суа-безразмерный коэффициент, вычисленный по оси y скоростной системы координат
    public static double liftHV(double суа,double h,double v,double s){
        return суа*incidentflowH(h,v)*s;
    }
    //Возвращает перегрузку по оси
    //f-проекция всех сил на ось,Н
    //m-масса ЛА, кг
    //g-величина гравитационного ускорения,м/с2
    public static double axleload(double f,double m,double g){
        return f/(m*g);
    }
}

package base.earth;
//Класс описывающий модель сферической Земли
public class EarthSphere extends Earth{
    public static final double r=6371000;//средний радиус Земли,м

    //Возвращает величину гравитационного ускорения,м/с2
    //h-высота над поверхностью Земли,м
    public static double g(double h){
        return mu/((h+r)*(h+r));
    }

    //Возвращает высоту над уровнем моря,м
    //x-координата x в стартовой системе координат,м
    //y-координата y в стартовой системе координат,м
    public static double HCKStart(double x,double y){
        return Math.sqrt((r+y)*(r+y)+x*x)-r;
    }
    //угол,рад
    //x-координата x в стартовой системе координат,м
    //y-координата y в стартовой системе координат,м
    public static double AngleCKStart(double x,double y){
        return Math.atan(x/(r+y));
    }
}

```



```

    }

} package base.vector;

import base.bean.BeanStageAtm;

import java.util.Vector;

//Класс для удобного представления результатов интегрирования атмосферной ступени
public class VectorAtm {
    //Таблица с полным результатом
    private Vector vector;
    //Таблица с нужными строками для показа
    private Vector vectorview;

    private int indexmaxq=0; //номер строки максимального скоростного напора
    private int indexmaxa=0; //номер строки максимального угла атаки
    private int indexvstart=0; //номер строки начала аэродинамического разворота
    private int indexvend=0; //номер строки конца аэродинамического разворота
    private int indexendstage1=0; //номер строки конца работы второй ступени
    private int indexend=0; //индекс последней строки

    private double maxq=0; //максимальный скоростной напор
    private double maxa=0; //максимальный угол атаки
    private boolean flagvstart=true; //флаг прохода начала аэродинамического участка
    private boolean flagvend=true; //флаг прохода конца аэродинамического участка

    private boolean flagtwostage=true; //флаг наличия второй ступени

    public VectorAtm() {
        this.vector= new Vector();
        this.vectorview= new Vector();
    }

    public void clear(){
        indexmaxq=0; //номер строки максимального скоростного напора
        indexmaxa=0; //номер строки максимального угла атаки
        indexvstart=0; //номер строки начала аэродинамического разворота
        indexvend=0; //номер строки конца аэродинамического разворота
        indexendstage1=0; //номер строки конца работы второй ступени
        indexend=0; //индекс последней строки

        maxq=0; //максимальный скоростной напор
        maxa=0; //максимальный угол атаки
        flagvstart=true; //флаг прохода начала аэродинамического участка
        flagvend=true; //флаг прохода конца аэродинамического участка

        flagtwostage=true; //флаг наличия второй ступени

        vector.clear();
        vectorview.clear();
    }

    public void add(double time, double x, double y, double v, double teta, double a, double nx, double ny, double p,
        double xa, double ya, double m, double q, double vstart, double vend){
        if (q>maxq){
            maxq=q;
            indexmaxq=vector.size();
        }
        if (Math.abs(a)>Math.abs(maxa)){
            maxa=a;
            indexmaxa=vector.size();
        }
    }
}

```

```

        if ((flagvstart) && (v >= vstart)) {
            indexvstart=vector.size();
            flagvstart=false;
        }
        if ((flagvend) && (v >= vend)) {
            indexvend=vector.size();
            flagvend=false;
        }
        if (time==0) {
            flagtwostage=! flagtwostage;
            if (flagtwostage) indexendstage1=vector.size()-1;
        }
        indexend=vector.size();
        vector.add(new BeanStageAtm(time,x,y,v,teta,a,nx,ny,p,xa,ya,m,q));
    }

    public Vector getVector() {
        return vector;
    }

    public Vector getVectorview() {
        vectorview.add(vector.get(0));
        vectorview.add(vector.get(indexvstart));
        vectorview.add(vector.get(indexmaxa));
        vectorview.add(vector.get(indexvend));
        vectorview.add(vector.get(indexmaxq));
        if (flagtwostage){
            vectorview.add(vector.get(indexendstage1));
            vectorview.add(vector.get(indexendstage1+1));
            vectorview.add(vector.get(vector.size()-1));
        }else vectorview.add(vector.get(vector.size()-1));
        return vectorview;
    }

    public int getIndexmaxq() {
        return indexmaxq;
    }

    public int getIndexmaxa() {
        return indexmaxa;
    }

    public int getIndexvstart() {
        return indexvstart;
    }

    public int getIndexvend() {
        return indexvend;
    }

    public int getIndexendstage1() {
        return indexendstage1;
    }

    public int getIndexend() {
        return indexend;
    }

    public double getVxend(){
        return
        ((BeanStageAtm)vector.get(indexend)).getV()*Math.cos(((BeanStageAtm)vector.get(indexend)).getTeta());
    }

```

```

        public double getVyend(){
            return
            ((BeanStageAtm)vector.get(indexend)).getV()*Math.sin(((BeanStageAtm)vector.get(indexend)).getTeta());
        }

        public double getXend(){
            return ((BeanStageAtm)vector.get(indexend)).getX();
        }

        public double getYend(){
            return ((BeanStageAtm)vector.get(indexend)).getY();
        }
    }

package base.vector;

import base.bean.BeanStageVacuum;
import base.math.M;

import java.util.Vector;

public class VectorVacuum {
    private Vector vector;
    private int indexend=0; //индекс последней строки
    public void add(double time,double x, double y, double vx, double vy,double h, double teta ){
        indexend=vector.size();
        vector.add(new BeanStageVacuum(time,x,y,vx,vy,h,teta));
    }

    public VectorVacuum() {
        this.vector = new Vector();
    }

    public Vector getVector() {
        return vector;
    }

    public int getIndexend() {
        return indexend;
    }

    public double getXend(){
        return ((BeanStageVacuum)vector.get(indexend)).getX();
    }
    public double getYend(){
        return ((BeanStageVacuum)vector.get(indexend)).getY();
    }
    public double getVxend(){
        return ((BeanStageVacuum)vector.get(indexend)).getVx();
    }
    public double getVyend(){
        return ((BeanStageVacuum)vector.get(indexend)).getVy();
    }
    public double getHend(){
        return ((BeanStageVacuum)vector.get(indexend)).getH();
    }
    public double getTetaend(){
        return ((BeanStageVacuum)vector.get(indexend)).getTeta();
    }
    public double getVend(){
        return M.vec(getVxend(),getVyend(),0);
    }
}

```

```

package base.stage;
//Класс позволяет преобразовать одни проектные параметры ступени в другие
public class StageParam {
    //Возвращает тягу двигателя в пустоте(для вне атмосферной ступени)
    //m0-масса ступени,кг
    //t-начальная тяговооруженность
    public static double getP(double m0,double t){
        return t*m0*9.81;
    }
    //Возвращает тягу двигателя в пустоте (для атмосферной ступени)
    //m0-масса ступени,кг
    //t-начальная тяговооруженность
    public static double getP(double m0,double t,double h){
        return t*m0*9.81*h;
    }
    //Возвращает секундный расход топлива
    //p-тяга двигателя,Н
    //pud-удельный импульс,м/с
    public static double getDm(double p,double pud){
        return p/pud;
    }
    //Возвращает площадь миделя
    //m0-масса ступени,кг
    //n-нагрузка на мидель,кг/м2
    public static double getS(double m0,double n){
        return m0/n;
    }
}

package base.stage;
//Описывает полностью проектные параметры ступени в вакууме (и сброс обтекателя)
public class StageVacuum extends Stage{

    protected double timefaultcowl=0;//время сброса головного обтекателя,с
    protected double massfaultcowl=0;//масса головного обтекателя,кг

    public StageVacuum(double p, double dm, double m0, double mf) {
        super(p,dm,m0,mf);
        this.timefaultcowl=0;
        this.massfaultcowl=0;
    }

    public StageVacuum(double p, double dm, double m0, double mf, double timefaultcowl, double massfaultcowl) {
        super(p,dm,m0,mf);
        this.timefaultcowl=timefaultcowl;
        this.massfaultcowl=massfaultcowl;
    }

    // возвращает массу ступени в определенные момент времени, кг
    // time- время в секундах
    public double mass(double time){
        if (time<timefaultcowl) return super.mass(time);
        else return m0-massfaultcowl-dm*time;
    }

    public double getTimefaultcowl() {
        return timefaultcowl;
    }

    public double getMassfaultcowl() {
        return massfaultcowl;
    }
}

```

```
}
```

```
package base.earth;
```

```
//Класс в котором содержатся табличные данные параметров Земли взятых с метеозондов
```

```
public class EarthTable {
```

```
    //Таблица для давления, 1 столбец - высота,м; 2 столбец - давление,Н/м2
```

```
    public static final double[][] pressure={ {0,101324.72},
```

```
        {500,95453.22},
```

```
        {1000,89875.03},
```

```
        {1500,84566.14},
```

```
        {2000,79497.24},
```

```
        {2500,74692.32},
```

```
        {3000,70124.70},
```

```
        {3500,65774.41},
```

```
        {4000,61656.09},
```

```
        {4500,57748.42},
```

```
        {5000,54044.74},
```

```
        {5500,50534.37},
```

```
        {6000,47213.32},
```

```
        {6500,44068.25},
```

```
        {7000,41097.84},
```

```
        {7500,38290.08},
```

```
        {8000,35647.64},
```

```
        {8500,33146.52},
```

```
        {9000,30790.72},
```

```
        {9500,28578.90},
```

```
        {10000,26491.08},
```

```
        {11000,22690.07},
```

```
        {12000,19390.35},
```

```
        {13000,16571.92},
```

```
        {14000,14164.13},
```

```
        {15000,12106.97},
```

```
        {16000,10347.92},
```

```
        {17000,8845.91},
```

```
        {18000,7561},
```

```
        {19000,6464},
```

```
        {20000,5526},
```

```
        {21000,4725},
```

```
        {22000,4040},
```

```
        {23000,3454},
```

```
        {24000,2954},
```

```
        {25000,2526},
```

```
        {26000,2162},
```

```
        {27000,1854},
```

```
        {28000,1594},
```

```
        {29000,1372},
```

```
        {30000,1183},
```

```
        {31000,1022},
```

```
        {32000,885},
```

```
        {33000,767},
```

```
        {34000,666},
```

```
        {35000,580},
```

```
        {36000,505},
```

```
        {37000,441},
```

```
        {38000,385},
```

```
        {39000,337},
```

```
        {40000,295},
```

```
        {41000,259},
```

```
        {42000,228},
```

```
        {43000,201},
```

```
        {44000,177},
```

```
        {45000,156},
```

```
        {46000,138},
```

```

        {47000,122},
        {48000,108},
        {49000,95},
        {50000,84},
        {55000,45},
        {60000,24},
        {65000,12},
        {70000,5},
        {75000,2},
        {80000,1},
        {85000,0.45},
        {90000,0.1},
        {95000,0.07},
        {100000,0.0324}};

//Возвращает давление на заданной высоте(возвращает нижнюю границу), Н/м2
//h - высота,м
public static double pressure(double h){
    for (int i=1;i<pressure.length;i++){
        if (h<=pressure[i][0]) return pressure[i-1][1];
    }
    return 0;
}
}

package rocket.filters;

import javax.servlet.http.HttpServletRequest;

public class AirParam {
    static double[][] Request(HttpServletRequest request){
        try{
            double mah0=Double.parseDouble(request.getParameter("mah0"));
            double mah1=Double.parseDouble(request.getParameter("mah1"));
            double mah2=Double.parseDouble(request.getParameter("mah2"));
            double mah3=Double.parseDouble(request.getParameter("mah3"));
            double mah4=Double.parseDouble(request.getParameter("mah4"));
            double mah5=Double.parseDouble(request.getParameter("mah5"));
            double mah6=Double.parseDouble(request.getParameter("mah6"));
            double mah7=Double.parseDouble(request.getParameter("mah7"));
            double mah8=Double.parseDouble(request.getParameter("mah8"));
            double mah9=Double.parseDouble(request.getParameter("mah9"));
            double mah10=Double.parseDouble(request.getParameter("mah10"));
            double mah11=Double.parseDouble(request.getParameter("mah11"));
            double mah12=Double.parseDouble(request.getParameter("mah12"));
            double mah13=Double.parseDouble(request.getParameter("mah13"));
            double mah14=Double.parseDouble(request.getParameter("mah14"));
            double mah15=Double.parseDouble(request.getParameter("mah15"));
            double mah16=Double.parseDouble(request.getParameter("mah16"));
            double mah17=Double.parseDouble(request.getParameter("mah17"));
            double mah18=Double.parseDouble(request.getParameter("mah18"));
            double mah19=Double.parseDouble(request.getParameter("mah19"));
            double mah20=Double.parseDouble(request.getParameter("mah20"));
            double mah21=Double.parseDouble(request.getParameter("mah21"));
            double mah22=Double.parseDouble(request.getParameter("mah22"));
            double mah23=Double.parseDouble(request.getParameter("mah23"));
            double mah24=Double.parseDouble(request.getParameter("mah24"));
            double mah25=Double.parseDouble(request.getParameter("mah25"));

            double cxa0=Double.parseDouble(request.getParameter("cxa0"));
            double cxa1=Double.parseDouble(request.getParameter("cxa1"));
            double cxa2=Double.parseDouble(request.getParameter("cxa2"));
            double cxa3=Double.parseDouble(request.getParameter("cxa3"));
            double cxa4=Double.parseDouble(request.getParameter("cxa4"));

```

```

double cxa5=Double.parseDouble(request.getParameter("cxa5"));
double cxa6=Double.parseDouble(request.getParameter("cxa6"));
double cxa7=Double.parseDouble(request.getParameter("cxa7"));
double cxa8=Double.parseDouble(request.getParameter("cxa8"));
double cxa9=Double.parseDouble(request.getParameter("cxa9"));
double cxa10=Double.parseDouble(request.getParameter("cxa10"));
double cxa11=Double.parseDouble(request.getParameter("cxa11"));
double cxa12=Double.parseDouble(request.getParameter("cxa12"));
double cxa13=Double.parseDouble(request.getParameter("cxa13"));
double cxa14=Double.parseDouble(request.getParameter("cxa14"));
double cxa15=Double.parseDouble(request.getParameter("cxa15"));
double cxa16=Double.parseDouble(request.getParameter("cxa16"));
double cxa17=Double.parseDouble(request.getParameter("cxa17"));
double cxa18=Double.parseDouble(request.getParameter("cxa18"));
double cxa19=Double.parseDouble(request.getParameter("cxa19"));
double cxa20=Double.parseDouble(request.getParameter("cxa20"));
double cxa21=Double.parseDouble(request.getParameter("cxa21"));
double cxa22=Double.parseDouble(request.getParameter("cxa22"));
double cxa23=Double.parseDouble(request.getParameter("cxa23"));
double cxa24=Double.parseDouble(request.getParameter("cxa24"));
double cxa25=Double.parseDouble(request.getParameter("cxa25"));
//Зависимость коэффициента аэродинамического сопротивления от числа Маха
double[][] cxa=
{{mah0,cxa0},{mah1,cxa1},{mah2,cxa2},{mah3,cxa3},{mah4,cxa4},{mah5,cxa5},{mah6,cxa6},{mah7,cxa7},
{mah8,cxa8},{mah9,cxa9},{mah10,cxa10},{mah11,cxa11},{mah12,cxa12},{mah13,cxa13},{mah14,cxa14},{mah1
5,cxa15},
{mah16,cxa16},{mah17,cxa17},{mah18,cxa18},{mah19,cxa19},{mah20,cxa20},{mah21,cxa21},{mah22,cxa22},
{mah23,cxa23},{mah24,cxa24},{mah25,cxa25}};
return cxa;
}catch (Exception e){

}
return new double[0][0];
}
}

package rocket.filters;

import rocket.bean.*;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

public class FilterParam implements Filter {
    public void destroy() {
    }

    public void doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) throws ServletException,
    IOException {

        ParamStageAtm paramStageAtm1= new
        ParamStageAtm(req.getParameter("m0atm1"),req.getParameter("mfatm1"),req.getParameter("patm1"),req.getParam
eter("pudatm1"),req.getParameter("hatm1"),req.getParameter("satm1"),req.getParameter("vstartatm1"),req.getParam
eter("vendatm1"),req.getParameter("cyaatm"),AirParam.Request((HttpServletRequest)req));

        ParamStageAtm paramStageAtm2= new
        ParamStageAtm(req.getParameter("m0atm2"),req.getParameter("mfatm2"),req.getParameter("patm2"),req.getParam
eter("pudatm2"),req.getParameter("hatm2"),req.getParameter("satm2"),req.getParameter("vstartatm2"),req.getParam
eter("vendatm2"),req.getParameter("cyaatm"),AirParam.Request((HttpServletRequest)req));

```

```

        ParamStageVacuum paramStageInt= new
ParamStageVacuum(req.getParameter("m0int"),req.getParameter("mfint"),req.getParameter("pint"),req.getParamete
r("pudint"),req.getParameter("timefaultcowlint"),req.getParameter("massfaultcowlint"));
        ParamStageVacuum paramStageEnd= new
ParamStageVacuum(req.getParameter("m0end"),req.getParameter("mfend"),req.getParameter("pend"),req.getParam
eter("pudend"),req.getParameter("timefaultcowlend"),req.getParameter("massfaultcowlend"));

        ParamIntegration paramIntegrationatm = new ParamIntegration(req.getParameter("stepintegrationatm"));
        ParamIntegration paramIntegrationint = new ParamIntegration(req.getParameter("stepintegrationint"));
        ParamIntegration paramIntegrationend = new ParamIntegration(req.getParameter("stepintegrationend"));

        ParamOrbit paramorbit = new ParamOrbit(req.getParameter("latitudestart"),req.getParameter("i"));

        ParamBoundary paramBoundaryStageAtm = new
ParamBoundary(req.getParameter("attakmax"),req.getParameter("attakstep"),req.getParameter("tetaepselonatm"),re
q.getParameter("tetaatm"));
        ParamBoundary paramBoundaryStageInt = new
ParamBoundary("0",req.getParameter("phistepint"),req.getParameter("tetaepselonint"),req.getParameter("tetaint"));
        ParamBoundary paramBoundaryStageIntPassiv = new
ParamBoundary("0",req.getParameter("phistepint"),req.getParameter("altitudeepselonint"),req.getParameter("altitud
eint"));
        ParamBoundary paramBoundaryStageEndTeta = new
ParamBoundary("0",req.getParameter("phistepend"),req.getParameter("tetaepselonend"),req.getParameter("tetaend"
));
        ParamBoundary paramBoundaryStageEndAltitude = new
ParamBoundary("0",req.getParameter("bstepend"),req.getParameter("altitudeepselonend"),req.getParameter("altitud
eend"));

        ParamInit paramInitInt = new
ParamInit(req.getParameter("xint"),req.getParameter("yint"),req.getParameter("vxint"),req.getParameter("vyint"));
        ParamInit paramInitEnd = new
ParamInit(req.getParameter("xend"),req.getParameter("yend"),req.getParameter("vxend"),req.getParameter("vyend"
));

        if (!paramStageAtm1.isError()) req.setAttribute("paramStageAtm1",paramStageAtm1);
        if (!paramStageAtm2.isError()) req.setAttribute("paramStageAtm2",paramStageAtm2);
        if (!paramStageInt.isError()) req.setAttribute("paramStageInt",paramStageInt);
        if (!paramStageEnd.isError()) req.setAttribute("paramStageEnd",paramStageEnd);
        if (!paramIntegrationatm.isError()) req.setAttribute("paramIntegrationatm",paramIntegrationatm);
        if (!paramIntegrationint.isError()) req.setAttribute("paramIntegrationint",paramIntegrationint);
        if (!paramIntegrationend.isError()) req.setAttribute("paramIntegrationend",paramIntegrationend);
        if (!paramorbit.isError()) req.setAttribute("paramorbit",paramorbit);
        if (!paramBoundaryStageAtm.isError())
req.setAttribute("paramBoundaryStageAtm",paramBoundaryStageAtm);
        if (!paramBoundaryStageInt.isError()) req.setAttribute("paramBoundaryStageInt",paramBoundaryStageInt);
        if (!paramBoundaryStageIntPassiv.isError())
req.setAttribute("paramBoundaryStageIntPassiv",paramBoundaryStageIntPassiv);
        if (!paramBoundaryStageEndTeta.isError())
req.setAttribute("paramBoundaryStageEndTeta",paramBoundaryStageEndTeta);
        if (!paramBoundaryStageEndAltitude.isError())
req.setAttribute("paramBoundaryStageEndAltitude",paramBoundaryStageEndAltitude);
        if (!paramInitInt.isError()) req.setAttribute("paramInitInt",paramInitInt);
        if (!paramInitEnd.isError()) req.setAttribute("paramInitEnd",paramInitEnd);

        chain.doFilter(req, resp);
    }

    public void init(FilterConfig config) throws ServletException {

    }

}

```



```

package rocket.charts;

import de.laures.cewolf.DatasetProducer;
import de.laures.cewolf.DatasetProduceException;

import java.io.Serializable;
import java.util.Vector;
import java.util.Map;
import java.util.Date;

import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import base.bean.BeanStageAtm;

public class ChartStageAtm implements DatasetProducer,Serializable{

    private String nameseries; //наименование серии
    private Vector vector; //данные для графика
    private String valuex; //какое значение взять из бина StepStage1Bean для оси x
    private String valuey; //какое значение взять из бина StepStage1Bean для оси y

    public void setNameseries(String nameseries) {
        this.nameseries = nameseries;
    }

    public void setVector(Vector vector) {
        this.vector = vector;
    }

    public void setValuex(String valuex) {
        this.valuex = valuex;
    }

    public void setValuey(String valuey) {
        this.valuey = valuey;
    }

    public Object produceDataset(Map params) throws DatasetProduceException {
        XYSeries series1 = new XYSeries(nameseries);
        double x=0,y=0;
        for (int i=1;i<=vector.size();i++){
            BeanStageAtm step = (BeanStageAtm) vector.elementAt(i-1);
            if (valuex.equals("time")) x=step.getTime();
            if (valuex.equals("x")) x=step.getX()/1000;
            if (valuex.equals("y")) x=step.getY()/1000;
            if (valuex.equals("v")) x=step.getV()/1000;
            if (valuex.equals("teta")) x=Math.toDegrees(step.getTeta());
            if (valuex.equals("a")) x=Math.toDegrees(step.getA());
            if (valuex.equals("nx")) x=step.getNx();
            if (valuex.equals("ny")) x=step.getNy();
            if (valuex.equals("p")) x=step.getP();
            if (valuex.equals("xa")) x=step.getXa();
            if (valuex.equals("ya")) x=step.getYa();
            if (valuex.equals("m")) x=step.getM();

            if (valuey.equals("time")) y=step.getTime();
            if (valuey.equals("x")) y=step.getX()/1000;
            if (valuey.equals("y")) y=step.getY()/1000;
            if (valuey.equals("v")) y=step.getV()/1000;
            if (valuey.equals("teta")) y=Math.toDegrees(step.getTeta());
            if (valuey.equals("a")) y=Math.toDegrees(step.getA());
            if (valuey.equals("nx")) y=step.getNx();
            if (valuey.equals("ny")) y=step.getNy();
        }
    }
}

```

```

        if (valuey.equals("p")) y=step.getP();
        if (valuey.equals("xa")) y=step.getXa();
        if (valuey.equals("ya")) y=step.getYa();
        if (valuey.equals("m")) y=step.getM();

        series1.add(x,y);
    }

    XYSeriesCollection dataset = new XYSeriesCollection();
    dataset.addSeries(series1);

    return dataset;
}

public boolean hasExpired(Map params, Date since) {

        return (System.currentTimeMillis() - since.getTime()) > 5000;

    }

    public String getProducerId() {
        return "ChartStageAtm";
    }
}

package rocket.charts;

import de.laures.cewolf.DatasetProducer;
import de.laures.cewolf.DatasetProduceException;

import java.io.Serializable;
import java.util.Vector;
import java.util.Map;
import java.util.Date;

import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import base.math.M;
import base.bean.BeanStageVacuum;

public class ChartStageVacuum implements DatasetProducer,Serializable{

    private String nameseries; //наименование серии
    private Vector vector; //данные для графика
    private String valuex; //какое значение взять из бина StepStage1Bean для оси x
    private String valuey; //какое значение взять из бина StepStage1Bean для оси y

    public void setNameseries(String nameseries) {
        this.nameseries = nameseries;
    }

    public void setVector(Vector vector) {
        this.vector = vector;
    }

    public void setValuex(String valuex) {
        this.valuex = valuex;
    }

    public void setValuey(String valuey) {
        this.valuey = valuey;
    }

    public Object produceDataset(Map params) throws DatasetProduceException {

```

```

XYSeries series1 = new XYSeries(nameseries);
double x=0,y=0;
for (int i=1;i<=vector.size();i++){
    BeanStageVacuum step = (BeanStageVacuum) vector.elementAt(i-1);
    if (valuex.equals("time")) x=step.getTime();
    if (valuex.equals("x")) x=step.getX()/1000;
    if (valuex.equals("y")) x=step.getY()/1000;
    if (valuex.equals("vx")) x=step.getVx()/1000;
    if (valuex.equals("vy")) x=step.getVy()/1000;
    if (valuex.equals("teta")) x=Math.toDegrees(step.getTeta());
    if (valuex.equals("h")) x=step.getH()/1000;
    if (valuex.equals("v")) x=M.vec(step.getVx(),step.getVy(),0)/1000;

    if (valuey.equals("time")) y=step.getTime();
    if (valuey.equals("x")) y=step.getX()/1000;
    if (valuey.equals("y")) y=step.getY()/1000;
    if (valuey.equals("vx")) y=step.getVx()/1000;
    if (valuey.equals("vy")) y=step.getVy()/1000;
    if (valuey.equals("teta")) y=Math.toDegrees(step.getTeta());
    if (valuey.equals("h")) y=step.getH()/1000;
    if (valuey.equals("v")) y=M.vec(step.getVx(),step.getVy(),0)/1000;

    series1.add(x,y);
}

XYSeriesCollection dataset = new XYSeriesCollection();
dataset.addSeries(series1);

return dataset;
}

public boolean hasExpired(Map params, Date since) {

    return (System.currentTimeMillis() - since.getTime()) > 5000;

}

public String getProducerId() {
    return "ChartStage1";
}

}

package rocket.charts;

import de.laures.cewolf.DatasetProducer;
import de.laures.cewolf.DatasetProduceException;

import java.io.Serializable;
import java.util.Vector;
import java.util.Map;
import java.util.Date;

import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import base.math.M;
import base.bean.BeanStageAtm;
import base.bean.BeanStageVacuum;

public class ChartFullTrajectory implements DatasetProducer,Serializable{

    private String nameseries1; //наименование 1 серии
    private String nameseries2; //наименование 2 серии
    private String nameseries3; //наименование 3 серии
    private Vector vector1; //данные для графика 1

```

```

private Vector vector2;    //данные для графика 2
private Vector vector3;    //данные для графика 3

private String valuex;     //какое значение взять из бина StepStage1Bean для оси x
private String valuey;     //какое значение взять из бина StepStage1Bean для оси y

public void setNameseries(String nameseries1,String nameseries2,String nameseries3) {
    this.nameseries1 = nameseries1;
    this.nameseries2 = nameseries2;
    this.nameseries3 = nameseries3;
}

public void setVector(Vector vector1,Vector vector2,Vector vector3) {
    this.vector1 = vector1;
    this.vector2 = vector2;
    this.vector3 = vector3;
}

public void setValuex(String valuex) {
    this.valuex = valuex;
}

public void setValuey(String valuey) {
    this.valuey = valuey;
}

public Object produceDataset(Map params) throws DatasetProduceException {
    XYSeriesCollection dataset = new XYSeriesCollection();
    if (vector1 !=null ){
        XYSeries series1 = new XYSeries(nameseries1);
        double x=0,y=0;
        for (int i=1;i<=vector1.size();i++){
            BeanStageAtm step = (BeanStageAtm) vector1.elementAt(i-1);
            if (valuex.equals("time")) x=step.getTime();
            if (valuex.equals("x")) x=step.getX()/1000;
            if (valuex.equals("y")) x=step.getY()/1000;
            if (valuex.equals("v")) x=step.getV()/1000;
            if (valuex.equals("teta")) x=step.getTeta();
            if (valuex.equals("a")) x=step.getA();
            if (valuex.equals("nx")) x=step.getNx();
            if (valuex.equals("ny")) x=step.getNy();
            if (valuex.equals("p")) x=step.getP();
            if (valuex.equals("xa")) x=step.getXa();
            if (valuex.equals("ya")) x=step.getYa();
            if (valuex.equals("m")) x=step.getM();

            if (valuey.equals("time")) y=step.getTime();
            if (valuey.equals("x")) y=step.getX()/1000;
            if (valuey.equals("y")) y=step.getY()/1000;
            if (valuey.equals("v")) y=step.getV()/1000;
            if (valuey.equals("teta")) y=step.getTeta();
            if (valuey.equals("a")) y=step.getA();
            if (valuey.equals("nx")) y=step.getNx();
            if (valuey.equals("ny")) y=step.getNy();
            if (valuey.equals("p")) y=step.getP();
            if (valuey.equals("xa")) y=step.getXa();
            if (valuey.equals("ya")) y=step.getYa();
            if (valuey.equals("m")) y=step.getM();

            series1.add(x,y);
        }
        dataset.addSeries(series1);
    }
}

```

```

if (vector2 !=null ){
    XYSeries series2 = new XYSeries(nameseries2);
    double x=0,y=0;
    for (int i=1;i<=vector2.size();i++){
        BeanStageVacuum step = (BeanStageVacuum) vector2.elementAt(i-1);
        if (valuex.equals("time")) x=step.getTime();
        if (valuex.equals("x")) x=step.getX()/1000;
        if (valuex.equals("y")) x=step.getY()/1000;
        if (valuex.equals("vx")) x=step.getVx()/1000;
        if (valuex.equals("vy")) x=step.getVy()/1000;
        if (valuex.equals("teta")) x=step.getTeta();
        if (valuex.equals("h")) x=step.getH();
        if (valuex.equals("v")) x=M.vec(step.getVx(),step.getVy(),0);

        if (valuey.equals("time")) y=step.getTime();
        if (valuey.equals("x")) y=step.getX()/1000;
        if (valuey.equals("y")) y=step.getY()/1000;
        if (valuey.equals("vx")) y=step.getVx()/1000;
        if (valuey.equals("vy")) y=step.getVy()/1000;
        if (valuey.equals("teta")) y=step.getTeta();
        if (valuey.equals("h")) y=step.getH();
        if (valuey.equals("v")) y=M.vec(step.getVx(),step.getVy(),0);

        series2.add(x,y);
    }
    dataset.addSeries(series2);
}
if (vector3 !=null ){
    XYSeries series3 = new XYSeries(nameseries3);
    double x=0,y=0;
    for (int i=1;i<=vector3.size();i++){
        BeanStageVacuum step = (BeanStageVacuum) vector3.elementAt(i-1);
        if (valuex.equals("time")) x=step.getTime();
        if (valuex.equals("x")) x=step.getX()/1000;
        if (valuex.equals("y")) x=step.getY()/1000;
        if (valuex.equals("vx")) x=step.getVx()/1000;
        if (valuex.equals("vy")) x=step.getVy()/1000;
        if (valuex.equals("teta")) x=step.getTeta();
        if (valuex.equals("h")) x=step.getH();
        if (valuex.equals("v")) x=M.vec(step.getVx(),step.getVy(),0);

        if (valuey.equals("time")) y=step.getTime();
        if (valuey.equals("x")) y=step.getX()/1000;
        if (valuey.equals("y")) y=step.getY()/1000;
        if (valuey.equals("vx")) y=step.getVx()/1000;
        if (valuey.equals("vy")) y=step.getVy()/1000;
        if (valuey.equals("teta")) y=step.getTeta();
        if (valuey.equals("h")) y=step.getH();
        if (valuey.equals("v")) y=M.vec(step.getVx(),step.getVy(),0);

        series3.add(x,y);
    }
    dataset.addSeries(series3);
}
return dataset;
}

public boolean hasExpired(Map params, Date since) {

    return (System.currentTimeMillis() - since.getTime()) > 5000;

}

public String getProducerId() {

```

```

        return "ChartFullTrajectory";
    }
}

package rocket.bean;

public class ParamBoundary {
    protected double init;
    protected double step;
    protected double epselon;
    protected double valueend;

    protected boolean error=false;

    public ParamBoundary(String init, String step, String epselon, String valueend) {
        try{
            this.init = Double.parseDouble(init);
            this.step = Double.parseDouble(step);
            this.epselon = Double.parseDouble(epselon);
            this.valueend= Double.parseDouble(valueend);
        }catch(Exception e){
            error=true;
        }
    }

    public double getInit() {
        return init;
    }

    public double getInitRad() {
        return init/(57.295779513);
    }

    public double getStep() {
        return step;
    }

    public double getStepRad() {
        return step/(57.295779513);
    }

    public double getEpselon() {
        return epselon;
    }

    public double getEpselonRad() {
        return epselon/(57.295779513);
    }

    public double getValueend() {
        return valueend;
    }

    public double getValueendRad() {
        return valueend/(57.295779513);
    }

    public boolean isError() {
        return error;
    }
}

package rocket.bean;

```

```

public class ParamInit {
    protected double x;
    protected double y;
    protected double vx;
    protected double vy;

    protected boolean error=false;

    public ParamInit(String x, String y, String vx, String vy) {
        try{
            this.x = Double.parseDouble(x);
            this.y = Double.parseDouble(y);
            this.vx = Double.parseDouble(vx);
            this.vy = Double.parseDouble(vy);
        }catch(Exception e){
            error=true;
        }
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public double getVx() {
        return vx;
    }

    public double getVy() {
        return vy;
    }

    public boolean isError() {
        return error;
    }
}
package rocket.bean;

public class ParamStageAtm extends ParamStageVacuum{
    //Коэффициент высотности сопла
    protected double h;
    //Площадь миделя
    protected double s;
    //Средняя производная коэффициента подъемной силы по углу атаки
    protected double cya;
    //Зависимость коэффициента аэродинамического сопротивления от числа Маха
    protected double[][] cxa;
    //Скорость начала аэродинамического разворота
    protected double vstart;
    //Скорость конца аэродинамического разворота
    protected double vend;

    public ParamStageAtm(String m0,String mf, String p, String pud, String h, String s, String vstart, String vend,
String cya, double[][] cxa ) {
        super(m0, mf, p, pud,"0", "0");
        try{
            this.h= Double.parseDouble(h);
            this.s= Double.parseDouble(s);
            this.vstart=Double.parseDouble(vstart);

```

```

        this.vend=Double.parseDouble(vend);
        this.cya =57.3*Double.parseDouble(cya);
        this.cxa=cxa;
    }catch(Exception e){
        error=true;
    }
}

public double getH() {
    return h;
}

public double getS() {
    return s;
}

public double getCya() {
    return cya;
}

public double[][] getCxa() {
    return cxa;
}

public double getVstart() {
    return vstart;
}

public double getVend() {
    return vend;
}

}
package base.bean;

import base.stage.StageVacuum;
import base.stage.StageAtm;

public class BeanRocket {
    private StageAtm stageatm1;
    private StageAtm stageatm2;
    private StageVacuum stageinterim;
    private StageVacuum stageend;
    private String name;
    private String fullname;

    public BeanRocket(StageAtm stageatm1, StageAtm stageatm2, StageVacuum stageinterim1, StageVacuum
stageend, String name, String fullname) {
        this.stageatm1 = stageatm1;
        this.stageatm2 = stageatm2;
        this.stageinterim = stageinterim1;
        this.stageend = stageend;
        this.name = name;
        this.fullname = fullname;
    }

    public BeanRocket() {
        this.stageatm1 = null;
        this.stageatm2 = null;
        this.stageinterim = null;
        this.stageend = null;
        this.name = null;
    }

```



```

        this.fullname = null;
    }

    public void setName(String name) {
        this.stageatm1 = null;
        this.stageatm2 = null;
        this.stageinterim = null;
        this.stageend = null;
        this.name = null;
        this.fullname = null;
        if (name.equals("samara")){
            double[][] cxa=
{{0,0.215},{0.2,0.214},{0.4,0.212},{0.6,0.218},{0.8,0.269},{1.0,0.466},{1.2,0.524},{1.4,0.516},
{1.6,0.476},{1.8,0.461},{2.0,0.429},{2.2,0.408},{2.4,0.367},{2.6,0.376},{2.8,0.354},{3.0,0.339},
{3.2,0.330},{3.4,0.321},{3.6,0.313},{3.8,0.306},{4.0,0.299},{4.2,0.292},{4.4,0.285},
{4.6,0.279},{4.8,0.274},{5.0,0.269}};
            this.stageatm1=new StageAtm(985993,985993/3145,65000,49540.02,1.244,6.473,cxa,0.173,50,270);
            this.stageend=new StageVacuum(82480,84280/3453,10218.76,7664.80,190-
this.stageatm1.getTimework(),300);
        }
        if (name.equals("soyuz")){
            double[][] cxa=
{{0,0.215},{0.2,0.214},{0.4,0.212},{0.6,0.218},{0.8,0.269},{1.0,0.466},{1.2,0.524},{1.4,0.516},
{1.6,0.476},{1.8,0.461},{2.0,0.429},{2.2,0.408},{2.4,0.367},{2.6,0.376},{2.8,0.354},{3.0,0.339},
{3.2,0.330},{3.4,0.321},{3.6,0.313},{3.8,0.306},{4.0,0.299},{4.2,0.292},{4.4,0.285},
{4.6,0.279},{4.8,0.274},{5.0,0.269}};
            this.stageatm1=new StageAtm(4774020,4774020/3100,310000,181800,1.175,21.5,cxa,0.173,50,270);
            this.stageinterim=new StageVacuum(997000,997000/3100,113000,89900,0,0);
            this.stageend=new StageVacuum(297300,297300/3300,17100,8500,0,0);
        }
    }

    public StageAtm getStageatm1() {
        return stageatm1;
    }

    public StageAtm getStageatm2() {
        return stageatm2;
    }

    public StageVacuum getStageinterim() {
        return stageinterim;
    }

    public StageVacuum getStageend() {
        return stageend;
    }

    public String getName() {
        return name;
    }

    public String getFullname() {
        return fullname;
    }
}

```

3 Инструкция по применению программного комплекса STK при системном анализе космических миссий

Настоящая инструкция позволяет пользователям приобрести навыки, навыки для создания и управления объектами в ПК STK. Учащийся познакомится со структурой статических и динамических элементов базы данных ПК STK и научится сохранять и извлекать объекты из этой базы. После создания различных объектов в ПК STK учащийся научится «оживлять» проект для того, чтобы получить информацию, зависящую от времени в окне «Map». В добавление описываются формы получения отчетов в статической и динамической формах.

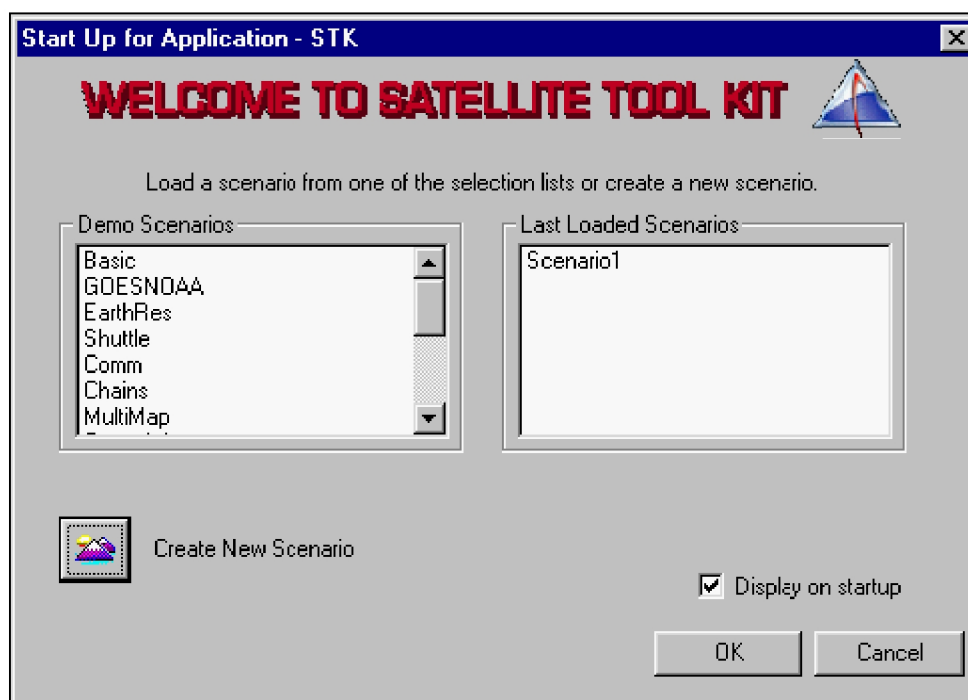
3.1 Начальные навыки работы с ПК

Папка, в которую устанавливается STK, содержит поддиректорию «Tutorial», в которой сохраняется всю информацию, полученную во время самостоятельной работы учащегося при овладении ПК.

Первым основным шагом является создание учебного сценария.

Сценарием называется объект высшего уровня в STK. Он включает в себя карту и содержит в себе все остальные объекты (спутники, навигационную аппаратуру и т.д.). Этот раздел программы обучения знакомит с процессом создания и насыщением информацией сценария.

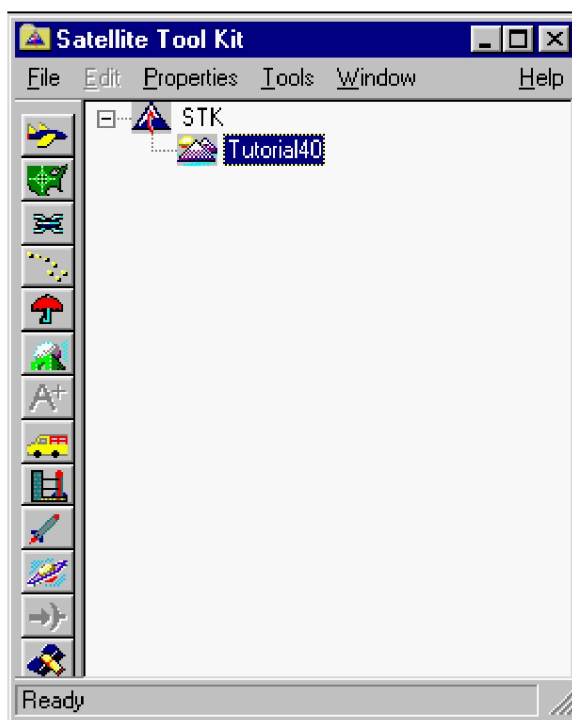
Когда впервые запускается STK, в отдельном окне открывается «Startup Wizard»



Для создания нового сценария требуется нажать кнопку, находящуюся слева от поля «Create a New Scenario» в «Startup Wizard» в результате открывается окно карты.

Для переименования сценария следует нажать на иконку слева от имени, присвоенного сценарию по умолчанию, потом на имя Scenario# в главном окне. После этого набирается «Tutorial40» в выделенном тексте и нажимается кнопка Enter.

Главное окно обновляется для отображения нового имени.

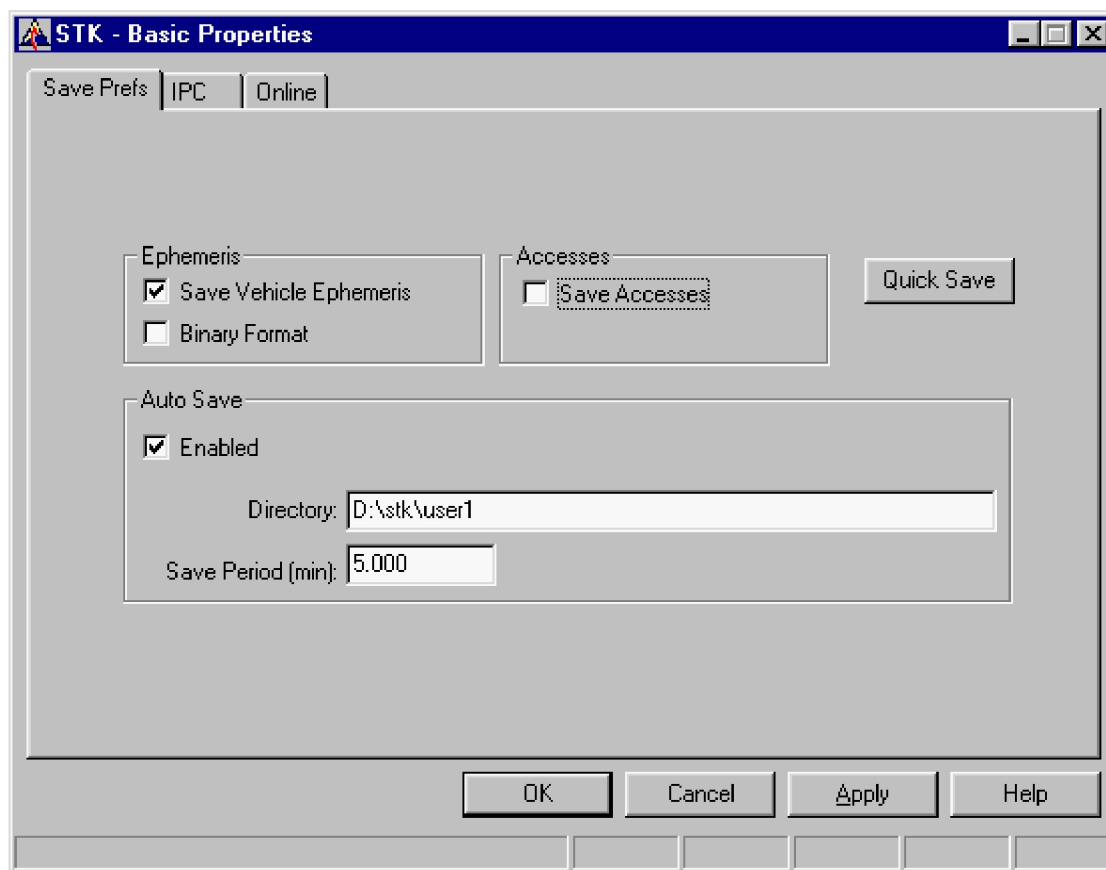


Перед тем, как выполнить какие-либо задачи в STK необходимо установить параметры Tutorial40, которые будут влиять на все аспекты сценария, после его создания. Для выполнения задач, поставленных на начальном этапе, целесообразно использовать параметры, описанные в дальнейшем.

Существует возможность изменять параметры сценария в период работы программы.

В первую очередь необходимо установить параметры приложения STK в целом. Эти параметры высшего уровня влияют на любой объект внутри приложения, независимо от открытого сценария. Для установки параметров приложения следует убедиться, что главное окно программы активно. Щелкните на меню «Properties» и выберите «Basic» во всплывшем окне. В результате появится окно базовых параметров.

Важно часто сохранять свой проект для предотвращения потери информации. Можно использовать опцию Авто сохранения.



Для удобства следует убедиться, что период автосохранения равен 5 минутам, включены пункты «Auto save» и «Save Vehicle Ephemeris», а пункты «Binary Format» и «Save Accesses» выключены. Нажмите клавишу OK для принятия изменений и закрытия окна «Basic Properties»

Важно установить параметры не только на уровне приложения, но и в самом сценарии. Необходимо установить единицы измерения, используемые на протяжении всей работы и так же установить необходимые параметры графики для удобства отображения и восприятия объектов и графики на экране.

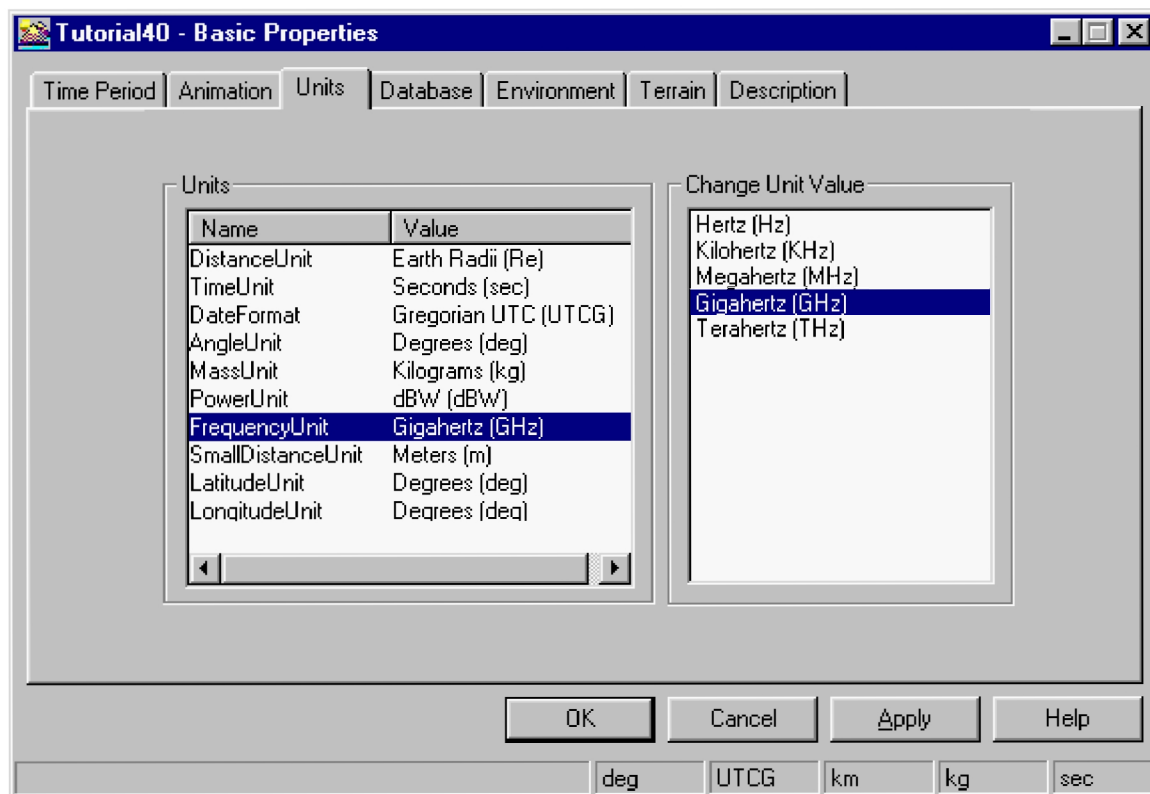
Единицы измерения выбираются путем выделения мышкой сценария Tutorial40 в главном окне и нажатием вкладки Basic в появившемся окне свойств.

Для пробы измените время начала сценария на 1 Nov 1997 00:00:00.00 и время его окончания на 1 Nov 1997 04:00:00.00. Затем нажмите на вкладку Animation.

The screenshot shows the 'Tutorial40 - Basic Properties' dialog box with the 'Time Period' tab selected. The 'Period' section contains three text input fields: 'Start' with the value '1 Nov 1997 00:00:00.00', 'Stop' with the value '1 Nov 1997 04:00:00.00', and 'Epoch' with the value '1 Nov 1997 00:00:00.00'. At the bottom, there are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'. Below these buttons is a row of unit selection boxes containing 'deg', 'UTC', 'km', 'kg', and 'sec'.

The screenshot shows the 'Tutorial40 - Basic Properties' dialog box with the 'Animation' tab selected. The 'Start Time' field is highlighted and contains the value '1 Nov 1997 00:00:00.00'. Below it, there is a checkbox for 'End Time' which is unchecked, followed by a dropdown menu and a text field containing '1 Nov 1997 04:00:00.00'. Further down, there is a 'Time Step' dropdown menu set to '60.000 sec' and a 'High Speed' dropdown menu set to '1.000 sec'. At the bottom, there are four buttons: 'OK', 'Cancel', 'Apply', and 'Help'. Below these buttons is a row of unit selection boxes containing 'deg', 'UTC', 'km', 'kg', and 'sec'.

Для того чтобы удостовериться, что период задан правильно, проверьте поле Start Time на вкладке Animation 1 Nov 1997 00:00:00.00, затем нажмите на вкладку Units.



Для удобства, установите все единицы измерений так, как показано в таблице. Чтобы изменить значения по умолчанию, выделите интересующий вас элемент, затем выберите правильное значение в окне «Change Unit Value». Чтобы выбрать элемент из списка, выделите мышкой этот элемент. Чтобы снять выделение, еще раз кликните по нему мышкой.

Field	Setting
Distance Unit (расстояние)	Nautical Miles (морские мили)
Time Unit (время)	Hours (часы)
Date Format (формат даты)	UTC Gregorian (Грегорианский календарь)
Angle Unit (угол)	Degrees (градусы)
Mass Unit (масса)	Kilograms (килограммы)
Power Unit (сила)	dBW
Frequency Unit (частота)	GHertz (ГигаГерцы)
SmallDistanceUnit ()	Meters (метры)
LatitudeUnit (широта)	Degrees (градусы)
LongitudeUnit (долгота)	Degrees (градусы)

После окончания установки следует нажать кнопку ОК.

Параметры графики управляют отображением информации на экране и возможностями, доступными в окне Map. Для того, чтобы установить параметры графики для собственного сценария, нажмите на кнопку панели инструментов окна Map.

Свойства карты отвечают за отображение информации: орбиты, названия объектов, трассы КА и другой информации о спутнике.

В таблице, показанной здесь, опция «Show Tool Bar» включена, а опция «Show Elevation Regions» отключена.

Для удобства, убедитесь, что опции «Show Tool Bar», «Show Status Bar» и «Show Scroll Bars» включены, а опция «Show Elevation Regions option» отключена. После окончания, нажмите на вкладку Details.

Далее следует наполнить сценарий различными объектами. Целесообразно начинать с выбора и задания размещения наземных станций, пусковых площадок и станций слежения.

Нажмите на иконку «Facility icon» в нижней части главного окна (когда курсор наводится на иконку, то в нижней части окна высвечивается ее значение). Измените название объекта, например, на «Baikonur».

Сначала следует выделить двойным щелчком мыши старое название, а затем вписать новое название в выделенном месте.

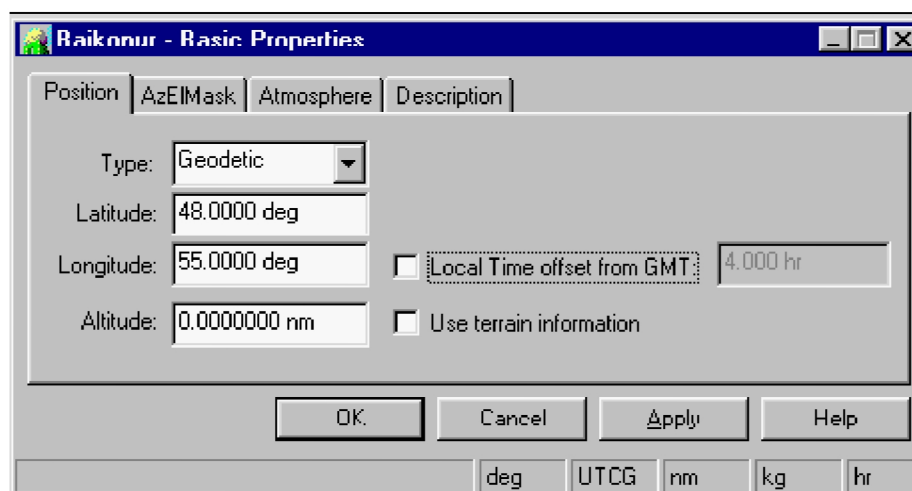
Нажатием Enter осуществляется установка нового имени.

В результате Главное окно обновится, чтобы отобразить новый объект.

В окне Карты он появится в месте с координатами 0 градусов широты и 0 градусов долготы.

Далее необходимо выбрать «Basic» из меню «Properties» (или нажать на правую кнопку мыши для появления меню, в котором можно выбрать свойства и инструменты для объекта).

Установка координат объекта осуществляется в первой вкладке Position окна Basic Properties. Координаты задают положение объекта на карте.

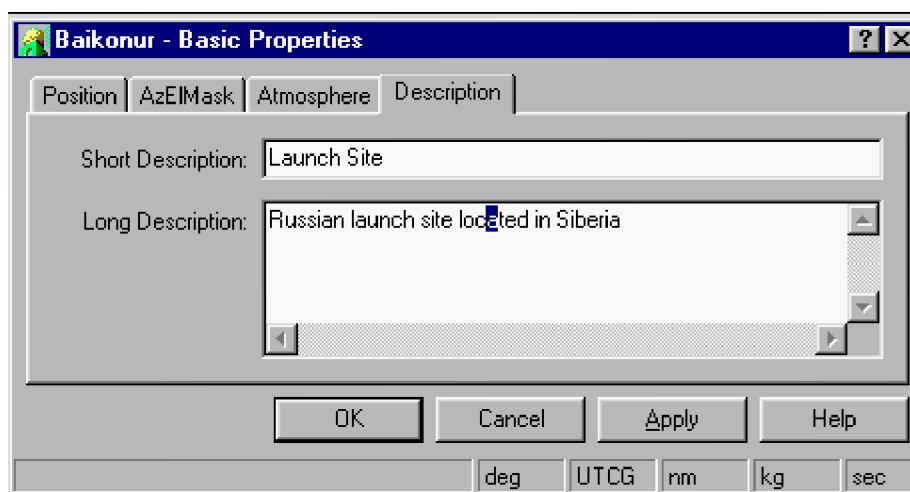


Для проверки усвоения установите значения на вкладке Position так, как показано ниже, и убедитесь, что опции Local Time Offset from GMT и Use Terrain Information выключены.

Field	Setting
Position type	Geodetic
Latitude	48.0
Longitude	55.0
Altitude	0.0

Далее следует нажать вкладку Description.

Чтобы в любой момент получить информацию об объекте следует использовать вкладку Description, где можно самостоятельно описать объект.

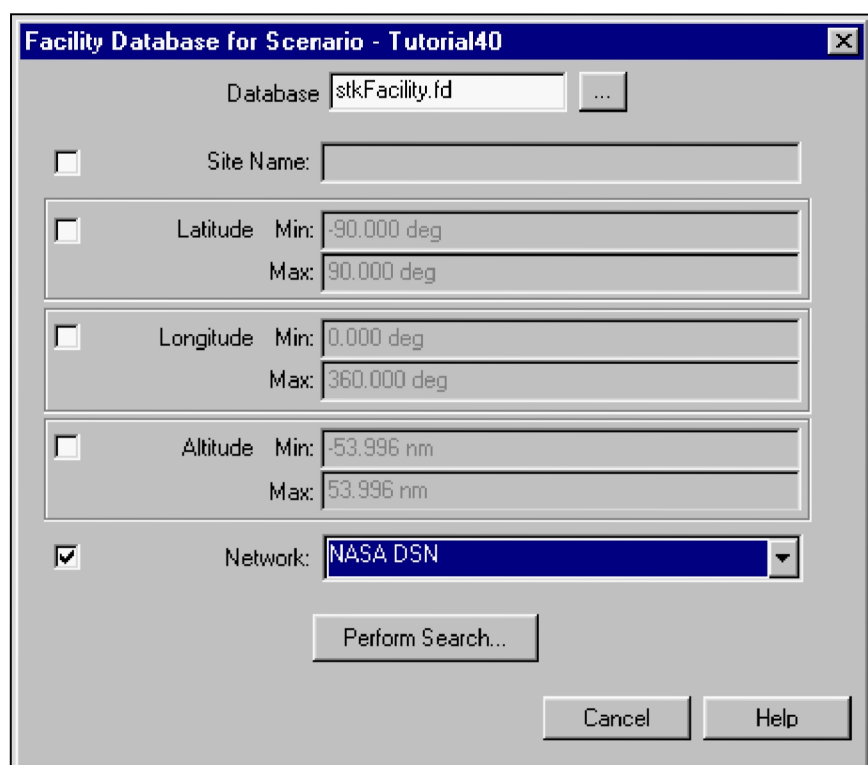


Следующим этапом освоения возможностей ПК является добавление новых объектов.

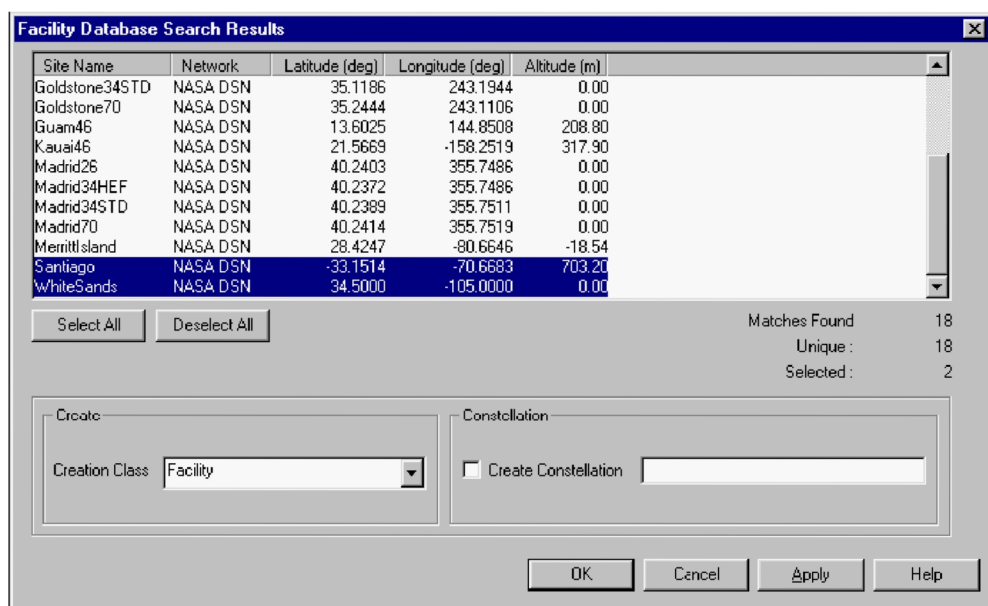
Для примера можно добавить еще четыре новых объекта, указанных в таблице. Для этого необходимо выполнить шаги, описанные выше для добавления новых объектов, Детально описывать объекты необязательно.

Name	Latitude	Longitude	Altitude	Short
Perth	-31.0	116.0	0.0	Australian Tracking Station
Wallops	37.8602	-75.5095	-0.0127878	NASA Launch Site/Tracking Station

После определения параметров каждого объекта следует нажать кнопку ОК. Для добавления еще двух объектов в сценарий воспользуемся Базой данных. Для этого в главном окне следует выделить создаваемый сценарий, а затем в меню Tools выбрать Facility Database.

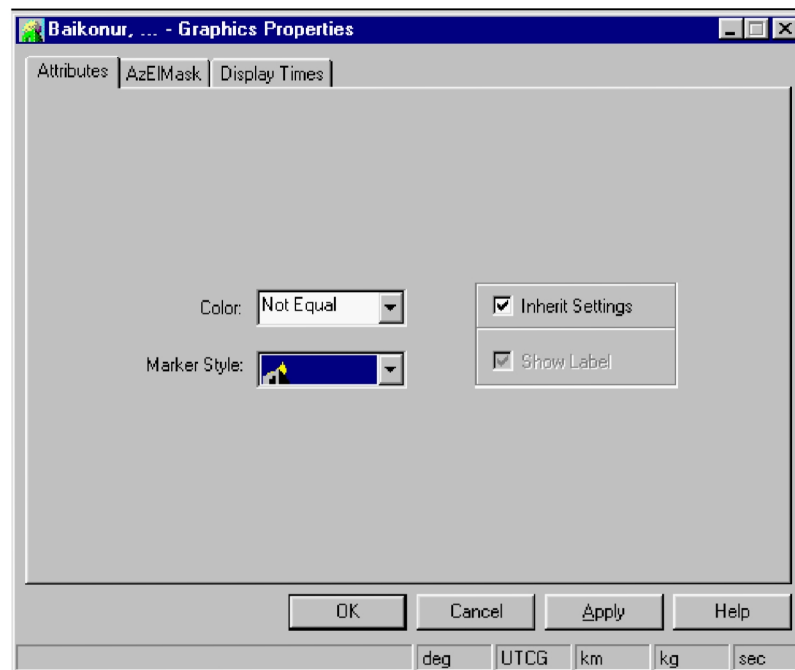


Необходимо включить опцию Network и выбрать, например, NASA DSN, а далее нажать кнопку Perform Search...

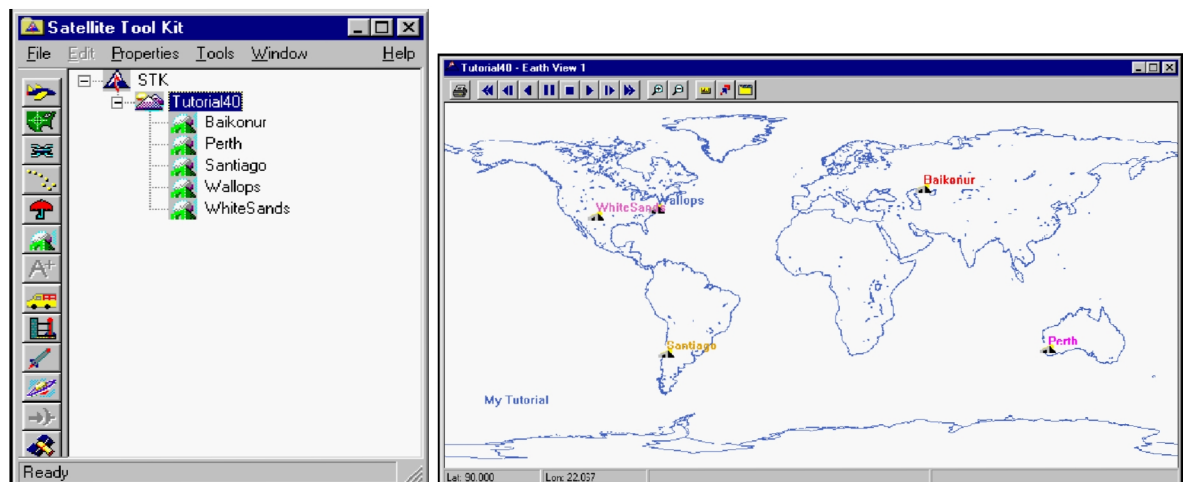


После появления в окне результатов поиска, найдите в конце списка Santiago и WhiteSands, затем нажмите OK. В окне Facility Database нажмите Cancel.

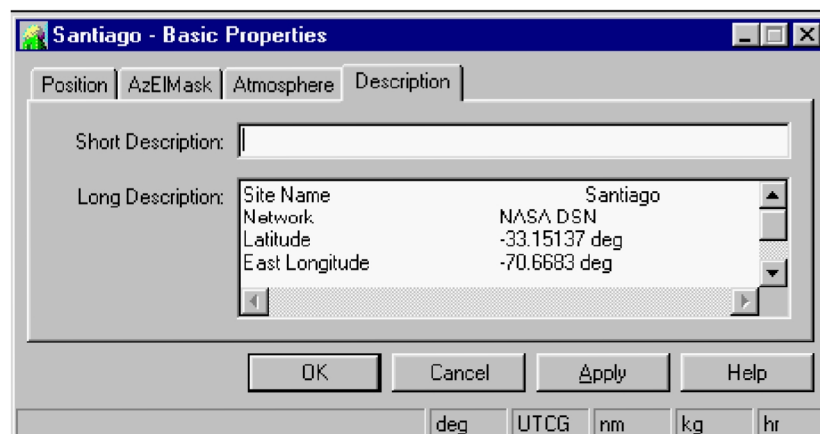
В главном окне выделите Baikonur, затем нажмите и держите кнопку Shift, попутно выделяя все оставшиеся четыре объекта, потом нажмите на правую кнопку мыши и выберите Graphics в появившемся меню.



На вкладке Attributes измените Marker Style так, чтобы он отобразил иконку объекта. Нажмите OK.



Когда объекты добавляются в сценарий из базы данных, для них автоматически создается описание. Для просмотра описания, например, объекта Santiago следует открыть меню Basic Properties для этого объекта и выбрать опцию Description.

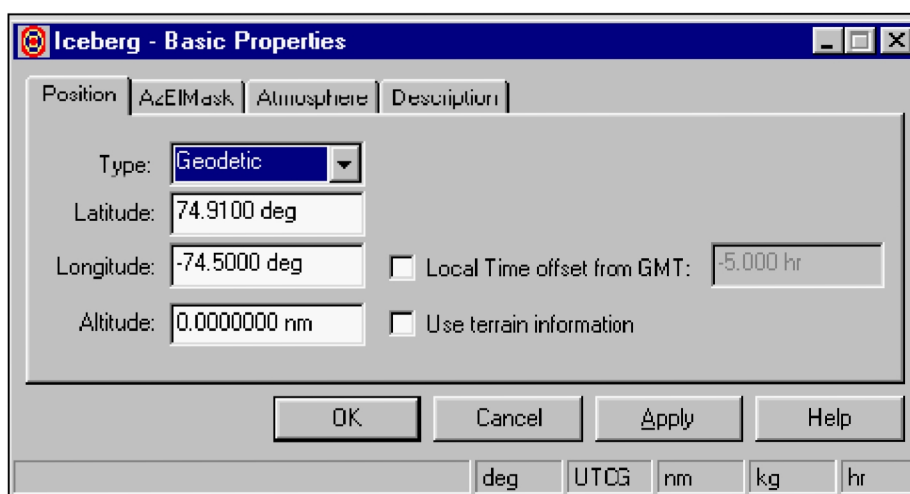


Поле Long Description содержит кроме информации о местонахождении и другие данные, касающиеся выделенного объекта.

Создание целей наблюдения является важным этапом написания сценария. Пусть целью наблюдения является ледник в Северной Америке. Следует нажать иконку Target в нижней части главного окна, затем обозначить цель Iceberg и нажать Enter.

Далее следует нажать на вкладку Basic в меню Properties с целью задания координат цели и ее описания.

Первая вкладка в окне Basic Properties называется Position.



Необходимо определить долготу, широту и высоту над уровнем моря для новой цели. На вкладке Description вводится короткое описание. Следует изменить Position Type на Geodetic.

Name	Latitude	Longitude	Altitude	Short
Iceberg	74.91	-74.50	0.0	Only the tip

После окончания нажать кнопку ОК. Для того, чтобы лучше видеть объект на Карте, следует открыть его графические свойства и поменять Marker Style на иконку объекта.

Создание сценария для подвижного объекта демонстрируется для корабля.

Вначале следует добавить корабль в сценарий Tutorial40. Нажать на иконку корабля и изменить название нового объекта на Cruise. Затем следует нажать Enter, чтобы принять новое имя.

Далее открывается окно Basic Properties для корабля.

Cruise - Basic Properties

Route | Attitude | Atmosphere | Description

Start Time: 1 Nov 1997 00:00:00.00 Step Size: 0.000 hr

Stop Time: 26 Nov 1997 19:03:13.55 Propagator: GreatArc

Latitude	Longitude	Altitude	Rate	Turn Radius
44.100	-8.500	0.000	4.900	0.000
51.000	-26.600	0.000	4.900	0.000
52.100	-40.100	0.000	4.900	0.000
60.200	-55.000	0.000	4.900	0.000
68.200	-65.000	0.000	4.900	0.000
72.200	-70.100	0.000	4.900	0.000
74.900	-74.500	0.000	4.900	0.000

☒ Update Map Graphics

Edit Mode:

☒ Add New Point

☐ Change Current Point

Insert Point

Delete Point

0.000 deg 0.000 deg 0.000 nm 30.000 nm/ 0.000 nm

OK Cancel Apply Help

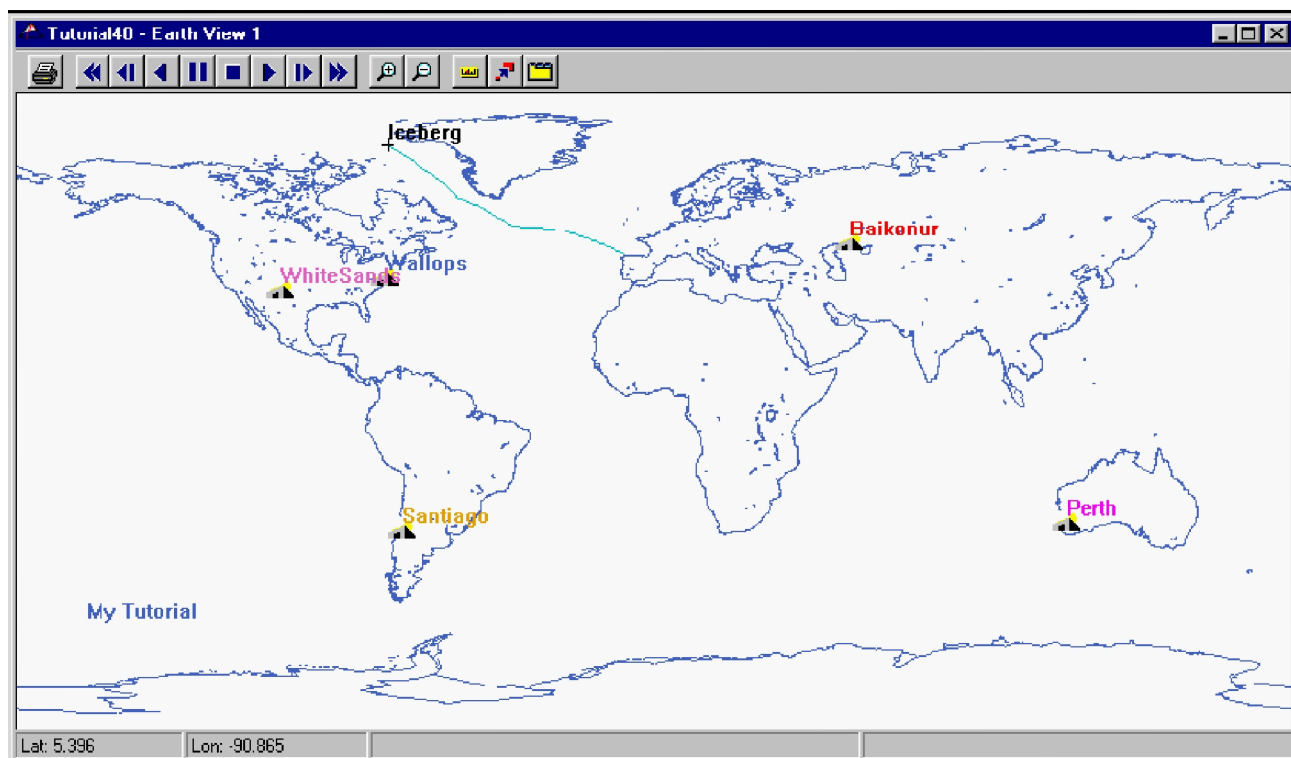
deg UTCG nm kg hr

Следует убедиться, что время старта на вкладке Route установлено на 1 Nov 1997 00:00:00 и Propagator на Great Arc.

Координаты движения корабля вводятся построчно. Когда завершается ввод строки нажимается кнопка Insert Point (ставится точка, через которую будет проходить корабль).

Latitude	Longitude	Altitude	Rate	Turn radius
44.1	-8.5	0.0	4.9	0.0
51.0	-26.6	0.0	4.9	0.0
52.1	-40.1	0.0	4.9	0.0
60.2	-55.0	0.0	4.9	0.0
68.2	-65.0	0.0	4.9	0.0
72.5	-70.1	0.0	4.9	0.0
74.9	-74.5	0.0	4.9	0.0

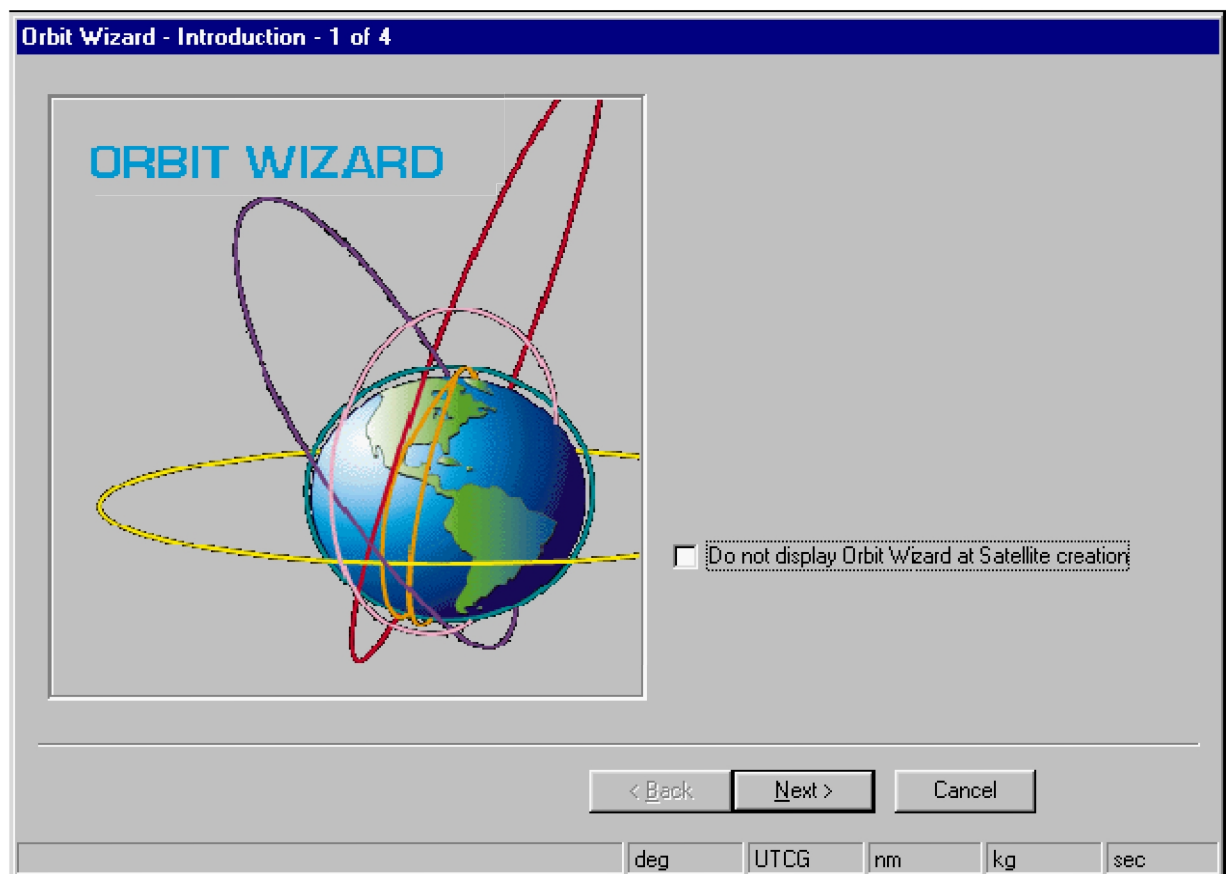
Далее осуществляется определение местонахождения корабля. Для этого нажимается вкладка Attitude в окне Basic Properties и проверяется, что в пункте Attitude Type установлено ECI velocity alignment with nadir constraint.



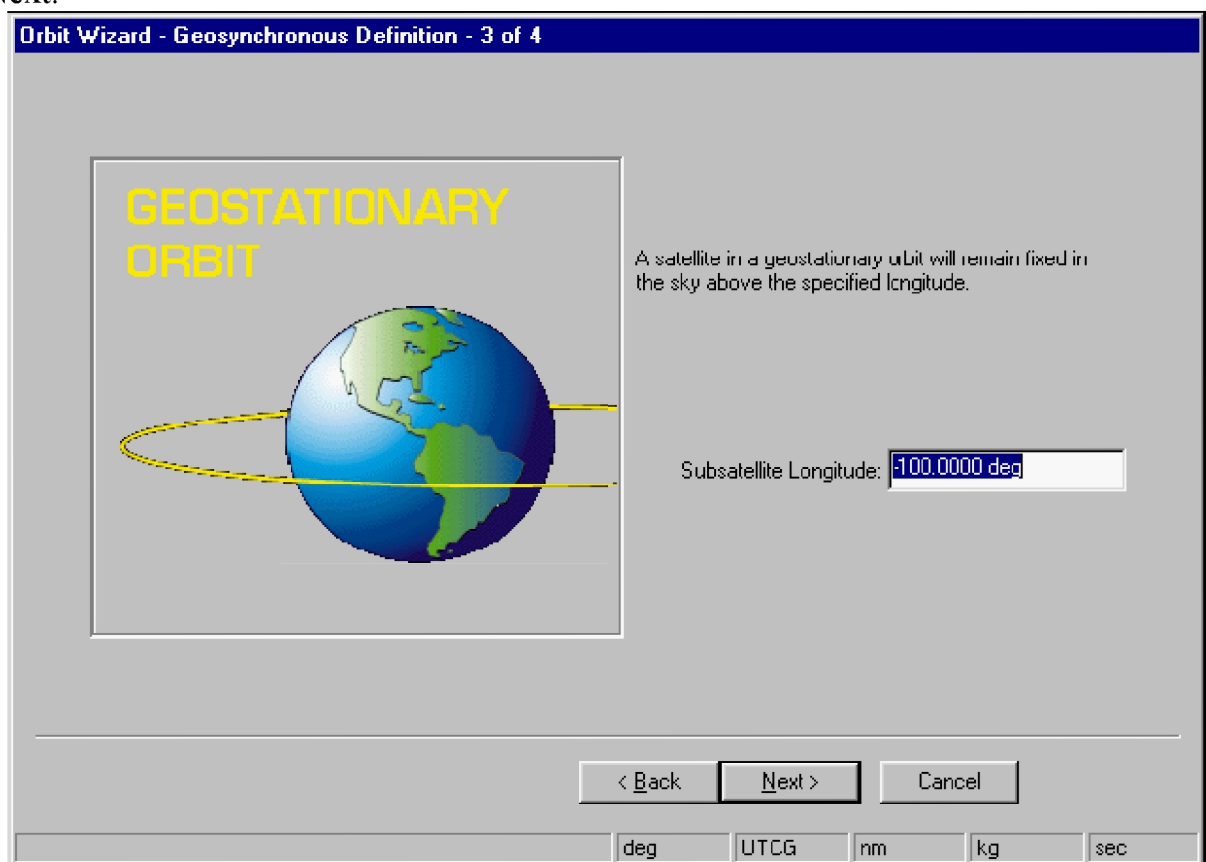
Далее при составлении сценария следует добавить спутники и космические аппараты. Например, добавляются ERS1, Shuttle, TDRS_East и TDRS_West.

На протяжении всего упражнения время будет отображаться в секундах. В окне Basic Properties выберите вкладку Units, измените измерение времени на секунды и нажмите ОК.

Теперь добавляются в сценарий спутники. Например, сначала в сценарий добавляются два передающих геостационарных спутника (TDRS). В главном окне следует нажать на изображение спутника. В результате откроется Orbit Wizard (мастер выбора орбиты).



Нажимается клавиша Next. Выбирается опция Geostationary и снова нажимается Next.

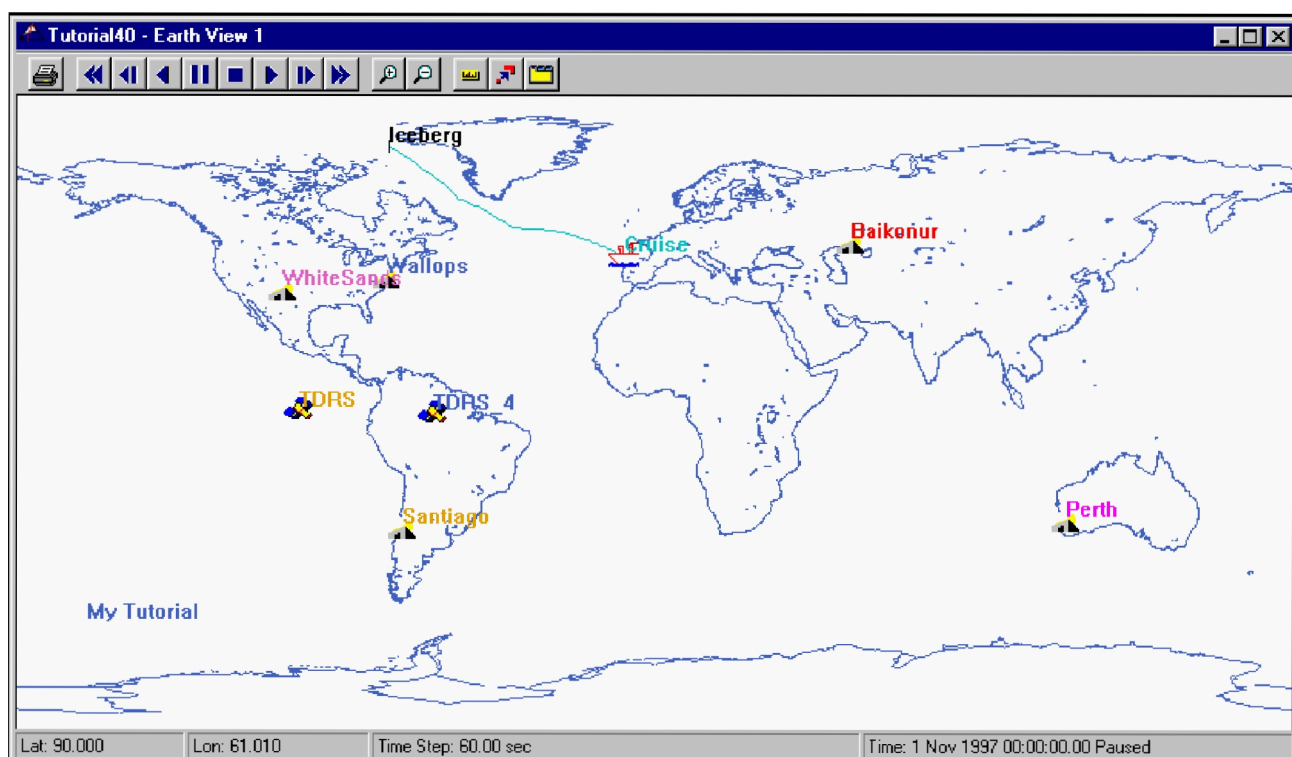


Следует убедиться, что в окне Мастера геостационарных орбит долгота (Longitude) имеет значение -100 (долгота стояния спутника на экваторе), затем следует нажать Next.

Далее измените время старта просмотра сценария и окончания просмотра на 1 Nov 1997 00:00:00.00 и 2 Nov 1997 00:00:00.00, соответственно. Далее следует поменять название спутника на TDRS.

Теперь следует ввести в сценарий еще один спутник TDRS. Для этого необходимо сделать активным окно сценария, выбрать в меню Files пункт Insert. После открытия вкладки Insert следует выбрать из списка доступных расширений конкретный спутник. Далее нажимается клавиша ОК, чтобы добавить спутник.

Трассы обоих спутников отображаются в виде линий. Чтобы лучше видеть спутники, следует кликнуть на спутники поочередно и выбрать в появившемся меню Graphics. Далее выбирается иконка спутника в качестве Marker Style.

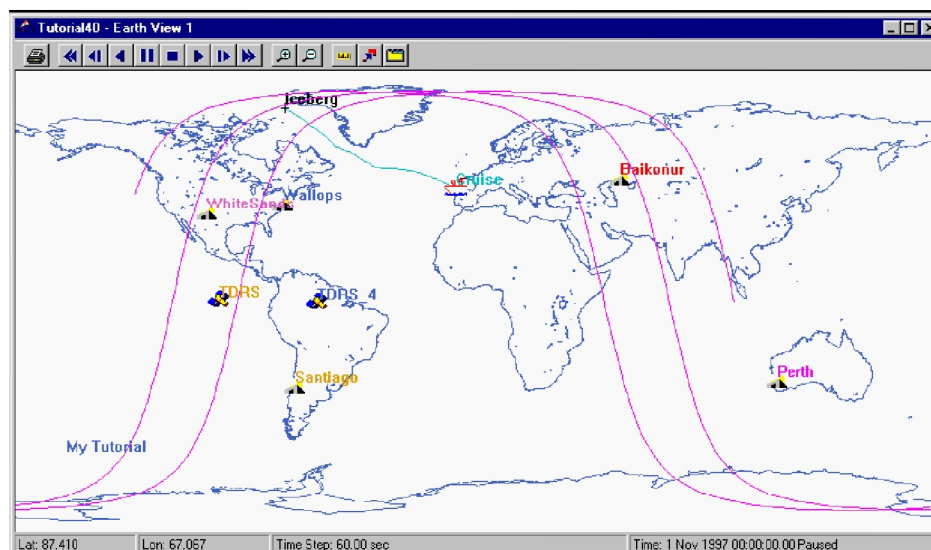


Добавление спутников осуществляется нажатием на иконку с изображением спутника. Когда появится окно мастера выбора орбиты нажимается Cancel и изменяется имя спутника на ERS1.

Далее вводятся параметры орбиты для ERS1. Перед тем, как вводить значения, приведенные в таблице, следует нажать кнопку со стрелочкой, чтобы поменять значение RAAN на Lon Ascen Node (долгота восходящего узла).

Orbital Element	Setting
Start Time	1 Nov 1997 00:00:00.00
Stop Time	1 Nov 1997 04:00:00.00
Step Size	60.00
Propagator	J4 Perturbation
Orbit Epoch	1 Nov 1997 00:00:00.00
Coordinate type	Classical
Coordinate System	J2000
Semimajor Axis	3867.7846
Eccentricity	0.0
Inclination	98.50
Argument of Perigee	0.0
Lon Ascen Node	99.38
True Anomaly	0.0

После окончания ввода следует нажать кнопку ОК.



По реализуемому учебному сценарию следует добавить космический корабль. Для этого следует вновь нажать на иконку с изображением спутника. Когда появится мастер выбора орбиты, следует нажать Cancel и ввести название Shuttle. Необходимо открыть окно базовых параметров для Shuttle и ввести параметры, приведенные в таблице, аналогично как было сделано для ERS1.

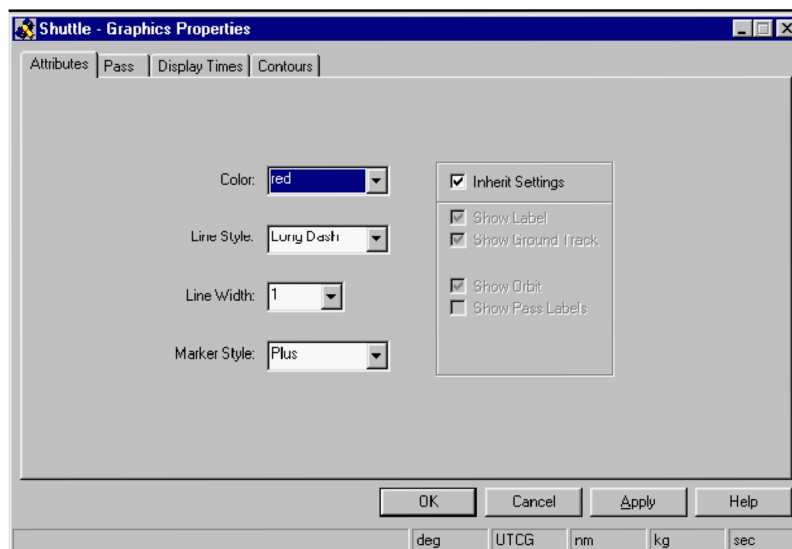
Перед тем, как вводить значения, приведенные в таблице, следует нажать на кнопку со стрелочкой, чтобы поменять RAAN на Lon Asc. Node и Semimajor Axis (главная ось) на Apogee Altitude (высота апогея).

Orbital Element	Setting
Start Time	1 Nov 1997 00:00:00.00
Stop Time	1 Nov 1997 03:00:00.00

Step Size	60.00
Propagator	J4 Perturbation
Orbit Epoch	1 Nov 1997 00:00:00.00
Coordinate type	Classical
Coordinate System	J2000
Apogee Altitude	200.0 nm
Perigee Altitude	200.0 nm
Inclination	28.5
Argument of Perigee	0.0
Lon Asc Node	-151.0
True Anomaly	0.0

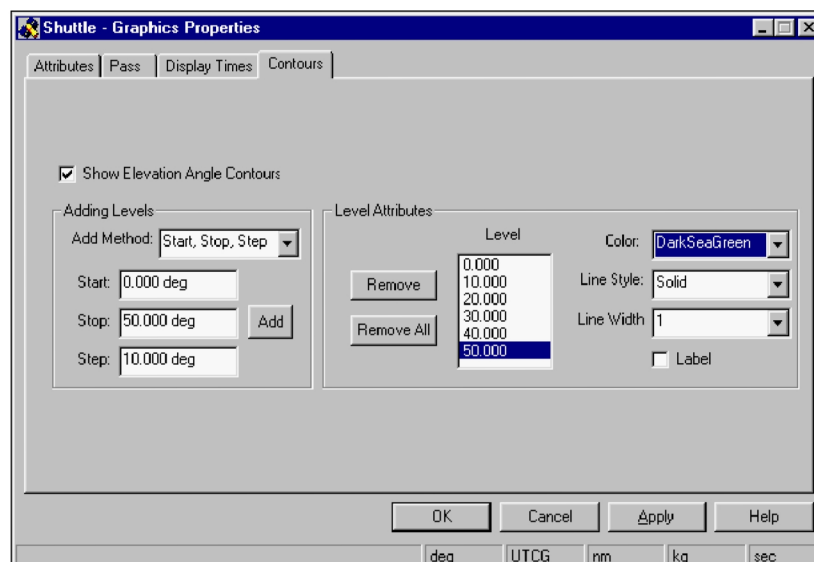
После окончания следует нажать кнопку ОК.

После того, как вы ввели параметры Shuttle, можно поменять свойства объекта, например, вид и цвет линии, обозначающие его трассу на поверхности Земли, чтобы отличить трассу от других движущихся объектов. Для этого следует выделить Shuttle в главном окне, нажать на него правой кнопкой мыши, выбрать пункт Graphics в появившемся меню,

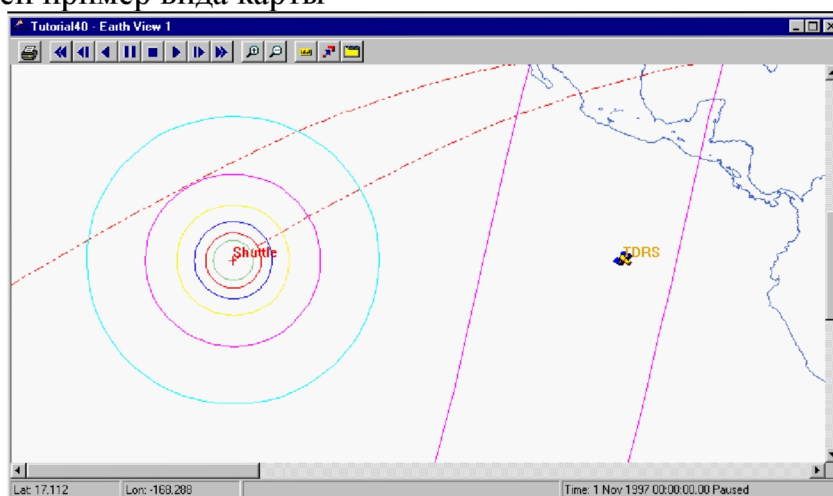


на вкладке Attributes изменить Line Style (стиль линии) на Long Dash (длинный пунктир), Marker Style (стиль маркера) на Plus и нажать Apply.

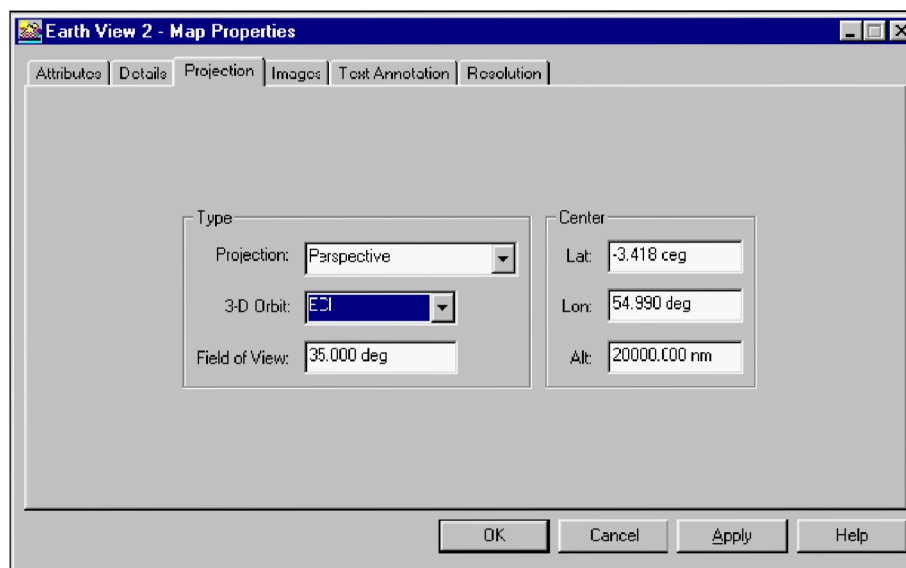
Далее следует выбрать вкладку Contours.



Теперь следует установить предельный угол места при котором возможно наблюдение Shuttle. Необходимо убедиться, что на вкладке Contours Add Method курсор установлен на Start, Stop, Step. Затем следует ввести значения 0, 50 и 10 для них, соответственно, и нажать Add. В списке Level выбрать первый уровень (0.00) и выключить опцию Label. Далее проделать то же самое для других уровней. Включите опцию Show Elevation Angle Contours и нажать кнопку ОК. Ниже приведен пример вида карты



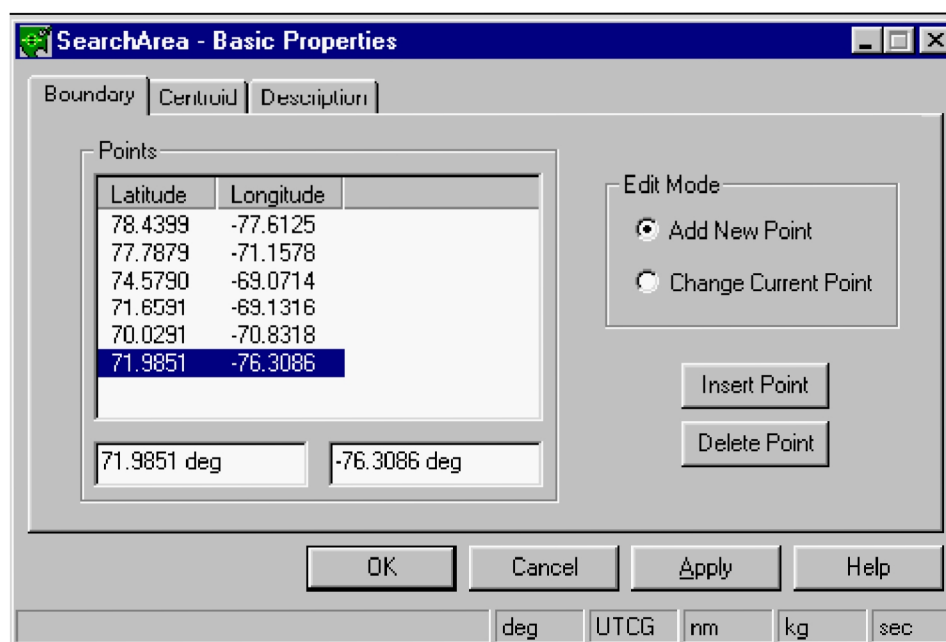
Можно наблюдать орбиты спутников в 3-D. Для этого в главном окне следует выделить Tutorial40 и выбрать в меню Tools пункт New Window. Когда появится окно карты, следует нажать на последнюю кнопку в верхнем ряду - появится окно Graphics Properties. Выбрать вкладку Projection. Поставить значения, показанные ниже на рисунке.



При реализации сценария важно выделять области наблюдения поверхности (видимости) со спутников. Они наглядно представляют границы интересных регионов на поверхности Земли.

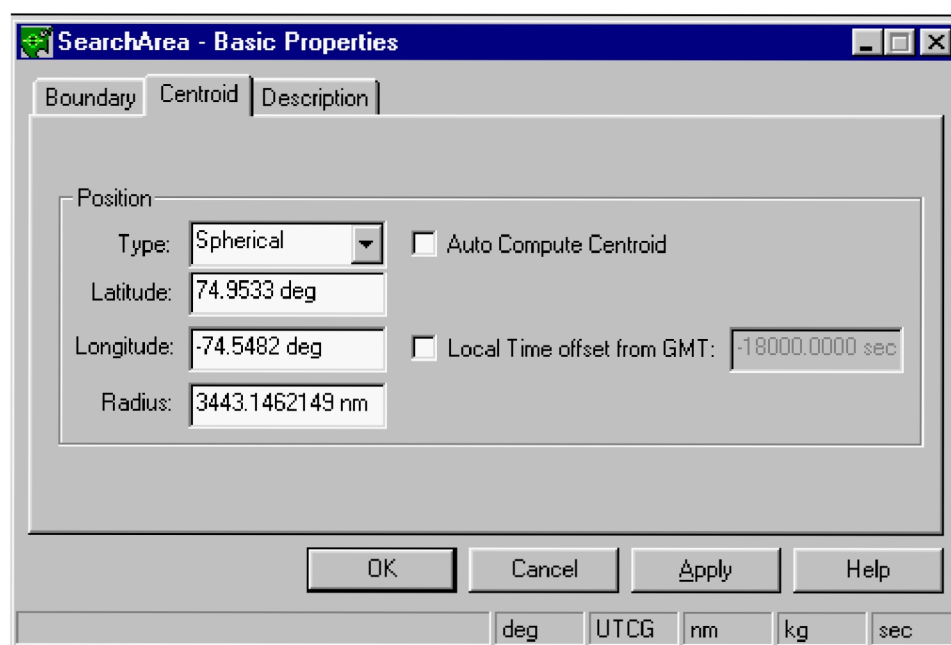
Рассмотрим ситуацию, когда корабль наткнулся на айсберг. Создадим область охвата, которая определяет границы поиска выживших пассажиров и оставшегося целым оборудования. Следует нажать на иконку Area Target в нижней части главного окна, а затем назвать зону – SearchArea, открыть окно Graphics Properties для SearchArea. Далее на вкладке Attributes изменить толщину линий (Line Width) на 3, выбрать None для Marker Type и выключить опции Inherit Settings и Show Label. Далее нажать OK.

Теперь следует задать цель наблюдения. Необходимо открыть окно Basic Properties для задания области охвата,



ввести долготу и широту, как показано на рисунке. После того как были введены значения долгота/широта, нажимайте Insert Point.

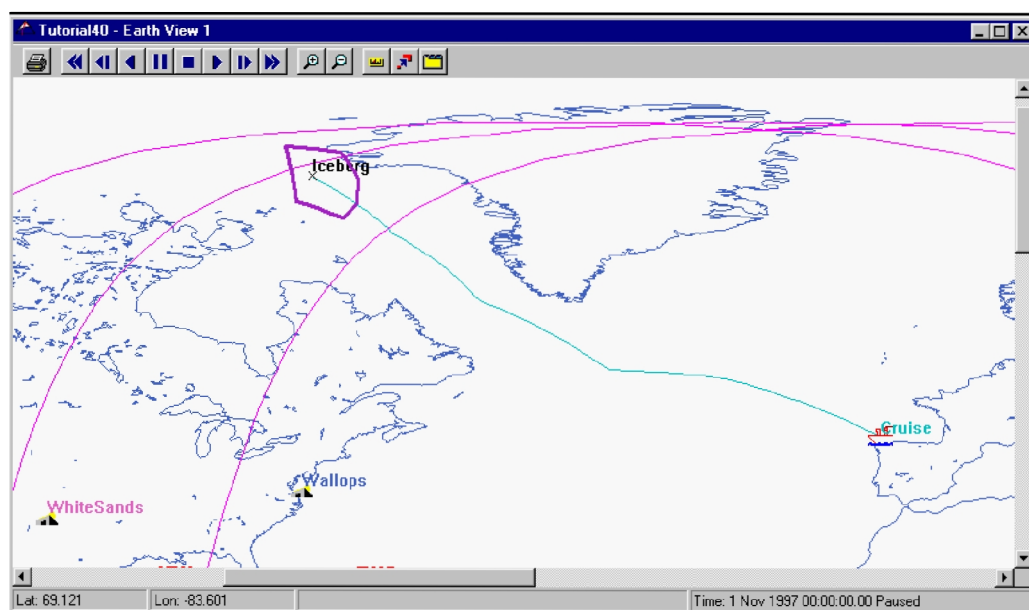
Теперь следует перейти на вкладку Centriod.



Следует выключить опцию Auto Centroid Compute, изменить Position Type на Spherical и ввести широту 74.9533 и долготу -74.5482, а также радиус 3433.1462149.

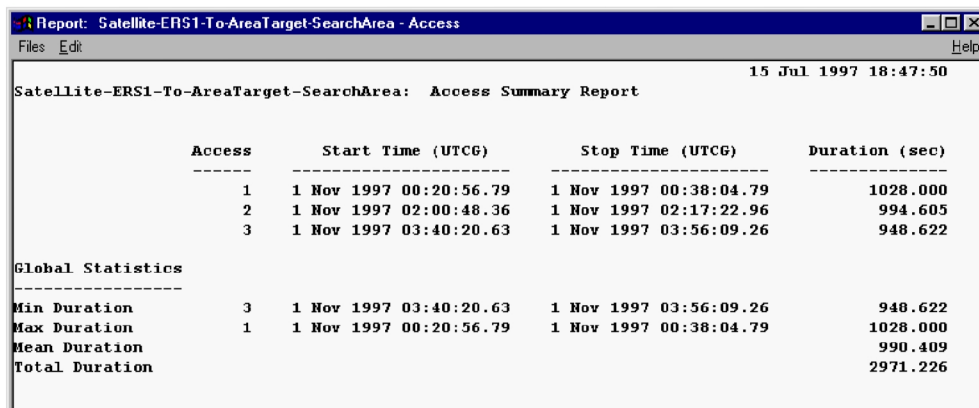
После окончания нажать OK.

Далее следует открыть окно Graphics Properties для айсберга, изменить Marker Style на X и нажать OK.



Пусть требуется выяснить, может ли спутник ERS1 видеть крушение и может ли он помочь в поиске и спасении людей. В главном окне следует выделить ERS1, кликнуть правой кнопкой мыши и выбрать Access из появившегося меню. Далее

выбрать SearchArea из списка доступных объектов, нажать последовательно Compute, Access... в поле Reports чтобы посмотреть отчет об области обзора спутника.



Report: Satellite-ERS1-To-AreaTarget-SearchArea - Access

15 Jul 1997 18:47:50

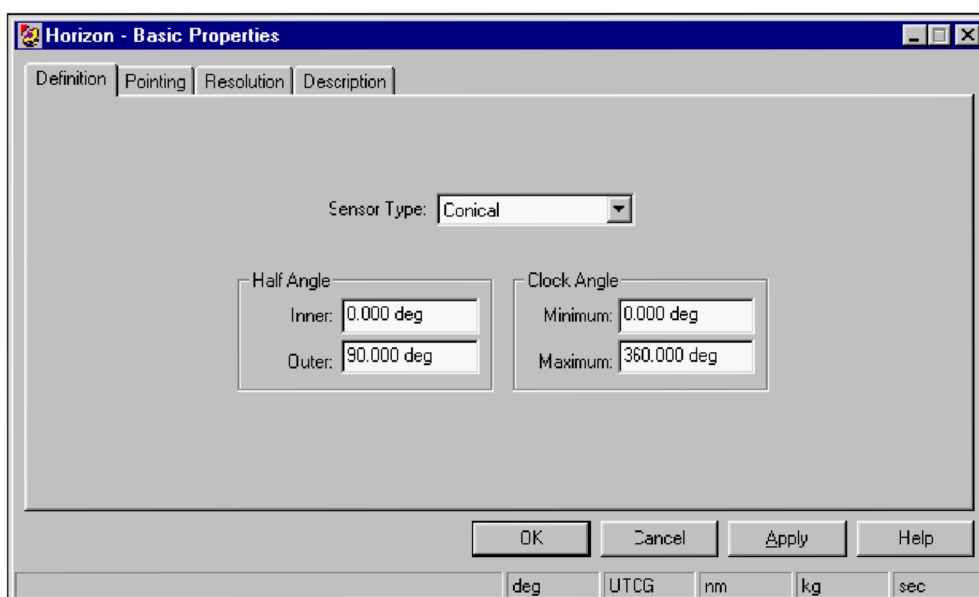
Satellite-ERS1-To-AreaTarget-SearchArea: Access Summary Report

Access	Start Time (UTC)	Stop Time (UTC)	Duration (sec)
1	1 Nov 1997 00:20:56.79	1 Nov 1997 00:38:04.79	1028.000
2	1 Nov 1997 02:00:48.36	1 Nov 1997 02:17:22.96	994.605
3	1 Nov 1997 03:40:20.63	1 Nov 1997 03:56:09.26	948.622

Global Statistics

Min Duration	3	1 Nov 1997 03:40:20.63	1 Nov 1997 03:56:09.26	948.622
Max Duration	1	1 Nov 1997 00:20:56.79	1 Nov 1997 00:38:04.79	1028.000
Mean Duration				990.409
Total Duration				2971.226

Для использования спутника ERS1 в указанных целях, он должен иметь на борту сканер наблюдения и антенну, для передачи информации со спутника на Землю. В главном окне следует выделить ERS1, нажать на иконку с изображением сенсора и назвать его Horizon. В классе спутника ERS1 появится подкласс сенсора. Теперь необходимо задать свойства определения объектов для сенсора на вкладке Basic Properties.



Horizon - Basic Properties

Definition | Pointing | Resolution | Description

Sensor Type: Conical

Half Angle

Inner: 0.000 deg

Outer: 90.000 deg

Clock Angle

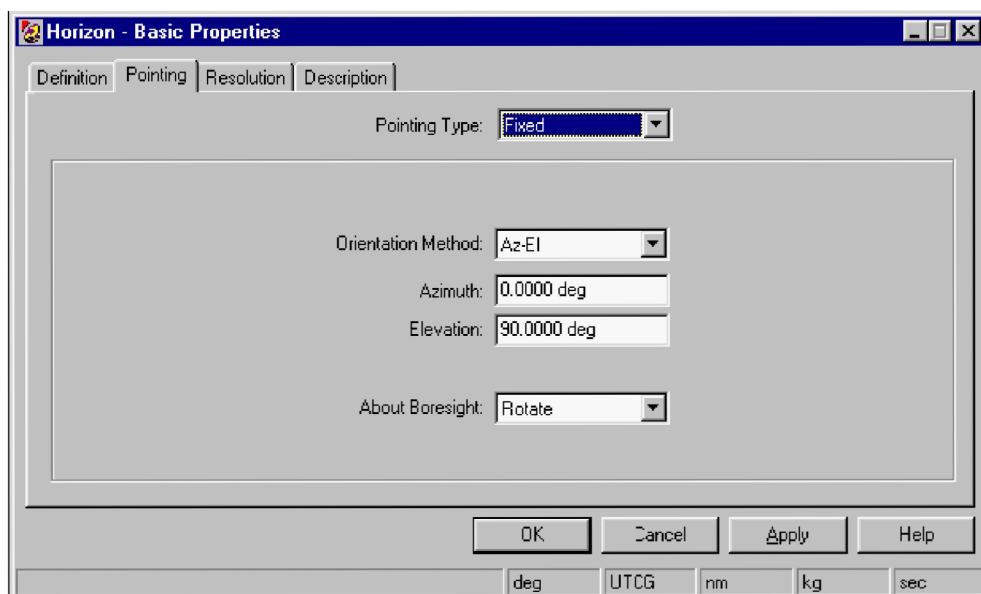
Minimum: 0.000 deg

Maximum: 360.000 deg

OK Cancel Apply Help

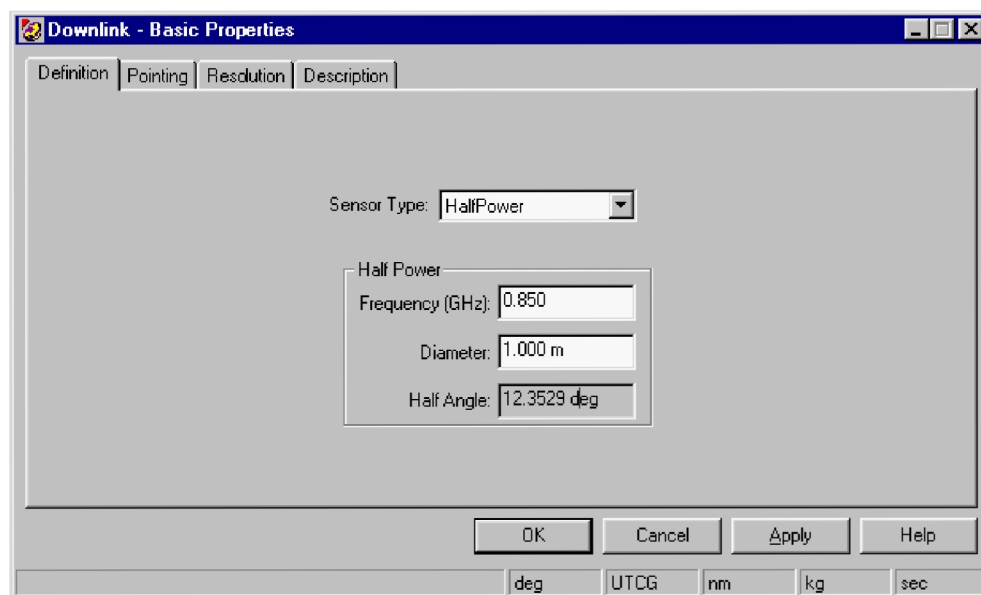
deg UTC nm kg sec

Введите значения, как показано на рисунке, а затем перейдите на вкладку Pointing.

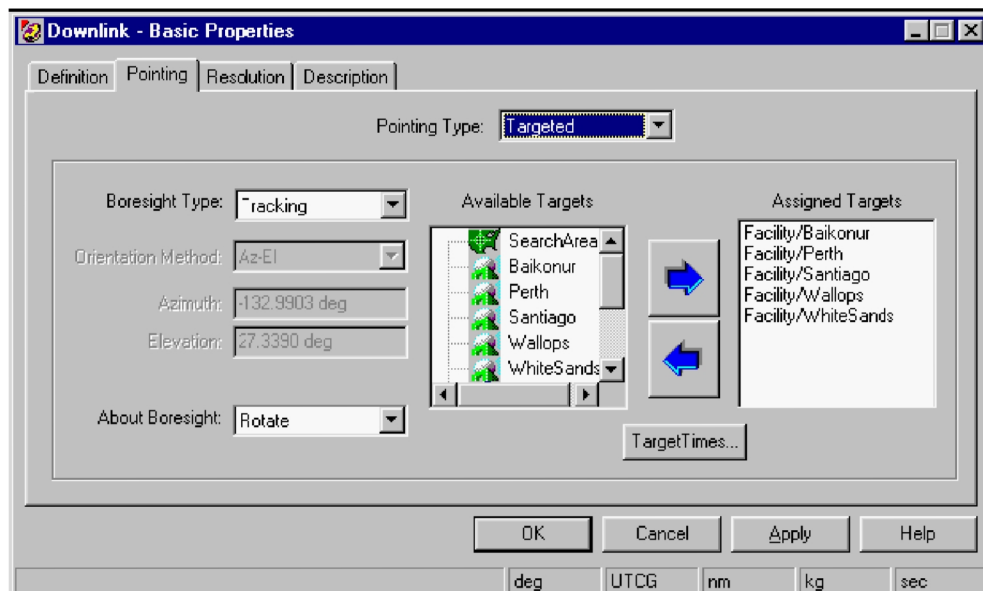


Следует направить сенсор нашего спутника строго на Землю. Чтобы это сделать, следует проверить, чтобы в опции Pointing Type был нажата клавиша Fixed и значения Elevation равнялись 90 градусам. После этого нажать OK.

Далее следует описать свойства передающей антенны, установленной на спутнике ERS1, используя действия, описанные выше. Чтобы задать параметры для антенны, следует задать Basic Properties для нее. Следует установить значения как показано ниже.



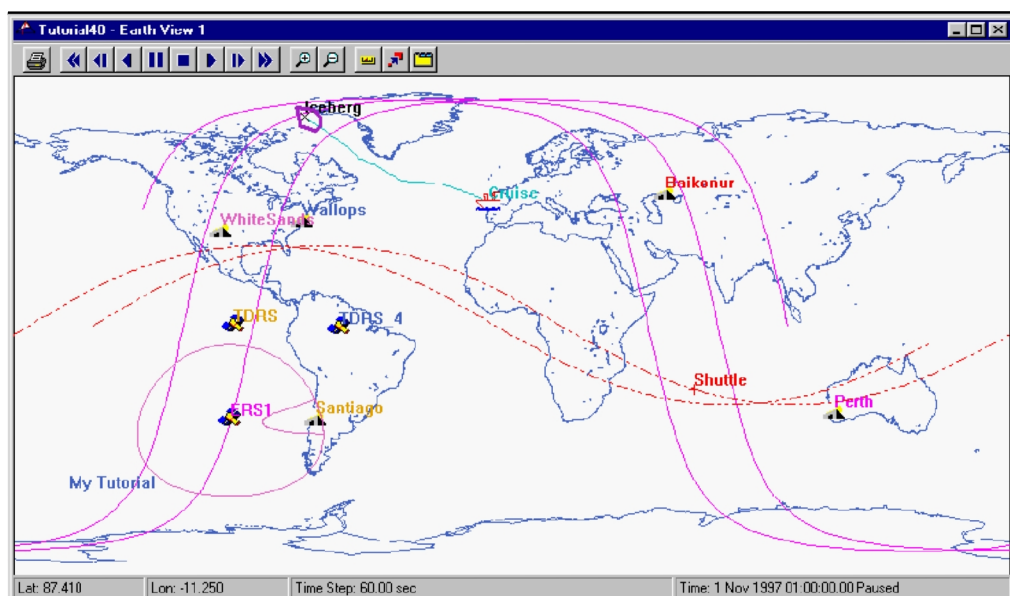
Затем необходимо перейти на вкладку Pointing.



Нужно обеспечить ориентацию антенны на заданные наземные станции. Для этого следует изменить Pointing Type на Targeted и Boresight (опорное направление) Type на Tracking (Слежение).

Далее нужно выбрать Baikonur в списке доступных объектов и нажать на стрелочку «вправо», чтобы добавить его в список заданных объектов. Таким же способом следует добавить оставшиеся объекты. Нажать OK.

Чтобы отобразить новую информацию необходимо «оживить» сценарий. Для этого нажать на кнопку Play. Ниже на картинке показан результат.



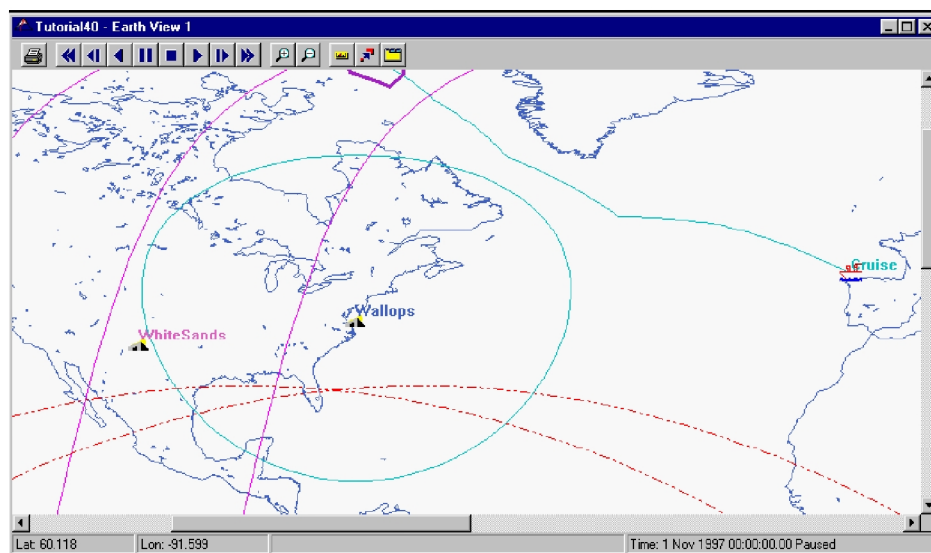
По собственному выбору моделирование и визуализацию сценария можно прервать.

Сенсоры целей и объектов можно добавлять. Например, если необходимо добавить объекту Wallops сенсор с названием FiveDegElev необходимо открыть его Basic Properties для установления параметров. На вкладке Definition tab следует выбрать Conical и установить значение Inner Half-angle равное 0.0 и значение Outer Half-

angle 85.0, оставив значения для Clock Angles по умолчанию. Затем перейти на вкладку Pointing.

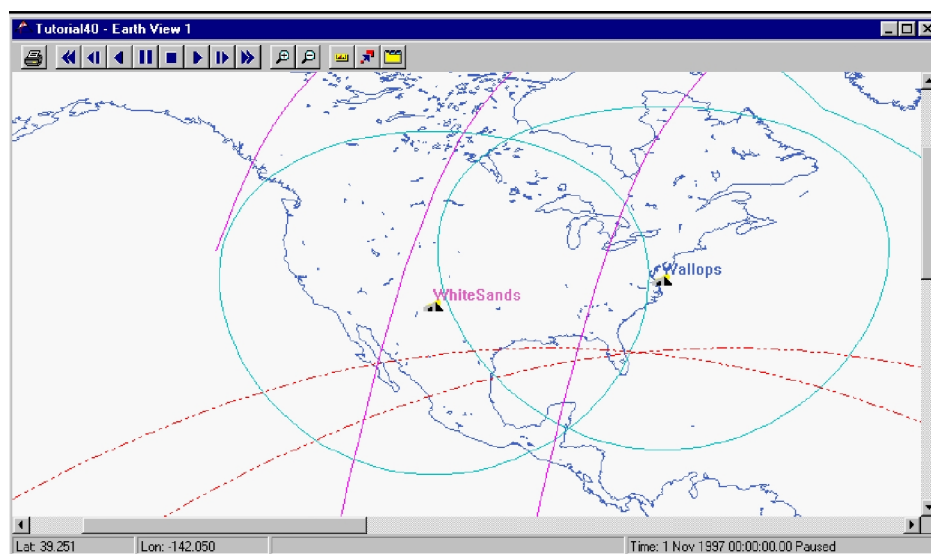
На этой вкладке следует убедиться, что в опции Pointing Type установлено Fixed и значение Elevation равно 90.0. Нажать OK.

Теперь следует открыть окно Graphics Properties для сенсора и выбрать вкладку Projection. На этой вкладке установить Maximum Altitude на 424.0 nm и Step Count равный 1, нажать OK.



Для того, чтобы этот сенсор можно было использовать для другого объекта WhiteSands, следует выделить сенсор FiveDegElev в главном окне, затем выбрать пункт Save в меню Files.

Теперь следует добавить такой же сенсор в объект WhiteSands, выделив в главном окне WhiteSands, а затем нажать Insert... в меню Files. Следует изменить File Type на Sensor и выбрать FiveDegElev.sn.



3.2. Анализ орбитальных миссий

В этом разделе учащиеся приобретают навыки анализа орбитальных миссий (орбит спутников), осваивают управление компонентами сценария для получения дополнительной информации и ее графического отображения. При этом в процессе выполнения работы учащийся приобретает навык создания орбит, вывода на экран трасс КА, визуализации сценариев, изменения способов отображения карты и документирования результатов.

1. Изменение параметров орбиты.

Если необходимо получить информацию о высоте апогея и перигея вместо значений главной оси и эксцентриситета, то следует нажать на иконку с изображением стрелочки «вниз» справа от поля Semimajor Axis и выбрать Apogee Altitude в появившемся списке.

ERS1 - Basic Properties

Orbit | Attitude | Pass Break | Mass | Description

Start Time: 1 Nov 1997 00:00:00.00 Step Size: 60.000 sec

Stop Time: 2 Nov 1997 00:00:00.00 Propagator: J4Perturbation

Orbit Epoch: 1 Nov 1997 00:00:00.00 Apogee Altitude: 423.8661335 nm

Coord Epoch: 1 Jan 2000 11:58:56.82 Perigee Altitude: 423.8661335 nm

Coord Type: Classical Inclination: 98.5000 deg

Coord System: J2000 Argument of Perigee: 0.0000 deg

Prop Specific: Special Options... Lon. Asc. Node: 99.3800 deg

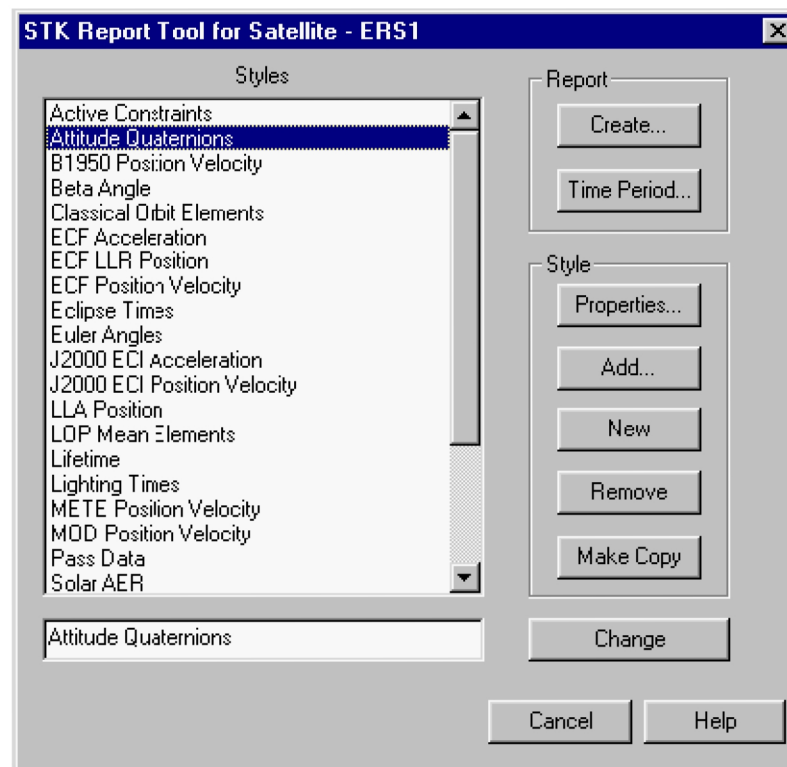
True Anomaly: 0.0000 deg

OK Cancel Apply Help

deg UTCG nm kg sec

2. Создание отчета.

Возможность создавать отчеты в STK облегчает получение и анализ информации. В STK реализовано несколько стандартных шаблонов отчетов. Для создания отчета в главном окне следует выделить ERS1, кликнуть правой кнопкой мыши и выбрать Report из появившегося меню.



Если необходимо задокументировать информацию об ориентации спутника ERS1, например, через кватернионы, следует в списке шаблонов выбрать Attitude Quaternions и нажать Create... для получения информации.

Report: ERS1 - Attitude Quaternions

Files Edit Help

07 Jul 1997 18:38:51

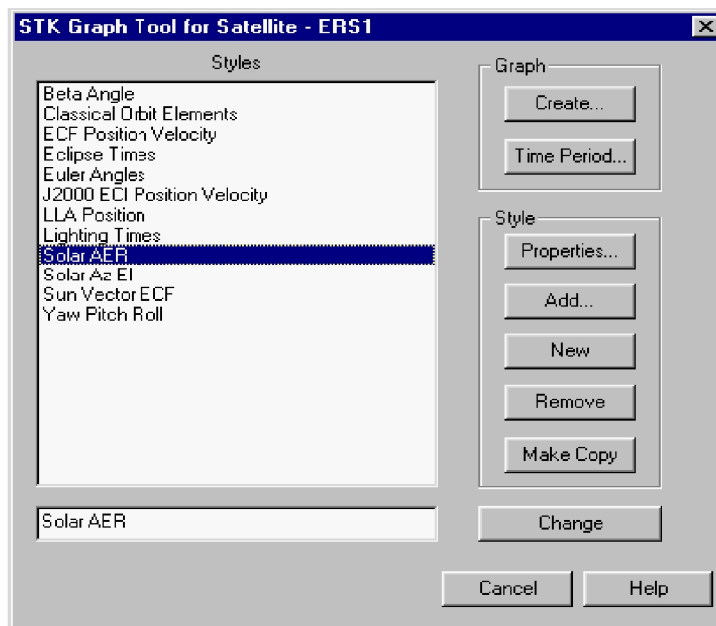
Satellite-ERS1: Attitude Quaternions

Time (UTC)	q1	q2	q3	q4
1 Nov 1997 00:00:00.00	0.686327	-0.170161	0.633662	0.313803
1 Nov 1997 00:01:00.00	0.706376	-0.178537	0.612376	0.306845
1 Nov 1997 00:02:00.00	0.725711	-0.186838	0.590532	0.299513
1 Nov 1997 00:03:00.00	0.744310	-0.195062	0.568148	0.291823
1 Nov 1997 00:04:00.00	0.762153	-0.203207	0.545243	0.283794
1 Nov 1997 00:05:00.00	0.779220	-0.211272	0.521836	0.275442
1 Nov 1997 00:06:00.00	0.795491	-0.219250	0.497944	0.266788
1 Nov 1997 00:07:00.00	0.810948	-0.227139	0.473586	0.257851
1 Nov 1997 00:08:00.00	0.825576	-0.234931	0.448783	0.248650
1 Nov 1997 00:09:00.00	0.839357	-0.242619	0.423553	0.239206
1 Nov 1997 00:10:00.00	0.852277	-0.250197	0.397916	0.229540
1 Nov 1997 00:11:00.00	0.864323	-0.257655	0.371893	0.219672
1 Nov 1997 00:12:00.00	0.875482	-0.264983	0.345504	0.209623
1 Nov 1997 00:13:00.00	0.885744	-0.272170	0.318770	0.199415
1 Nov 1997 00:14:00.00	0.895099	-0.279204	0.291714	0.189067
1 Nov 1997 00:15:00.00	0.903538	-0.286073	0.264356	0.178599
1 Nov 1997 00:16:00.00	0.911053	-0.292764	0.236719	0.168033
1 Nov 1997 00:17:00.00	0.917640	-0.299262	0.208827	0.157385
1 Nov 1997 00:18:00.00	0.923293	-0.305552	0.180702	0.146677
1 Nov 1997 00:19:00.00	0.928009	-0.311619	0.152369	0.135924

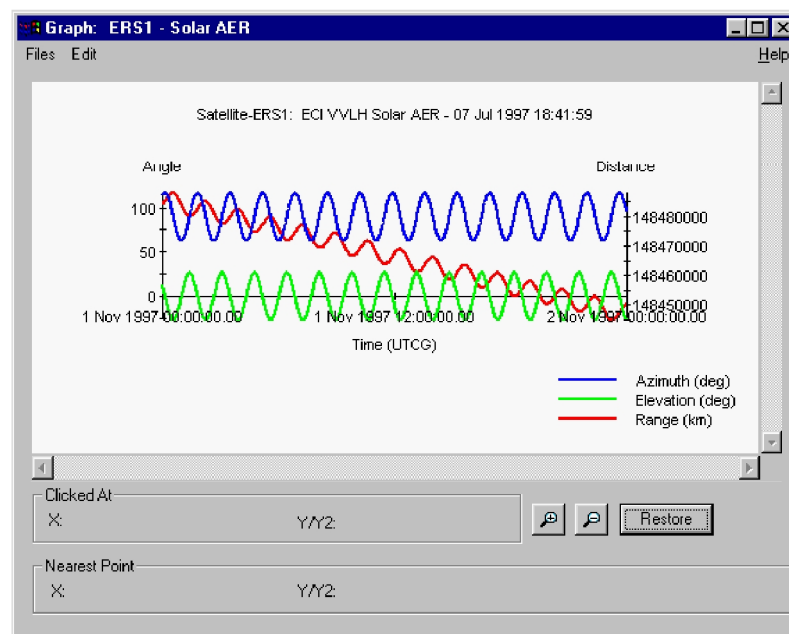
Чтобы закрыть это окно следует выбрать в меню Files пункт Close, чтобы распечатать отчет в меню Files следует выбрать пункт Print.

3. Получение информации в графическом виде.

Для просмотра той же информации в графическом виде в главном окне следует выделить спутник ERS1 и нажать Graph в меню Tools.



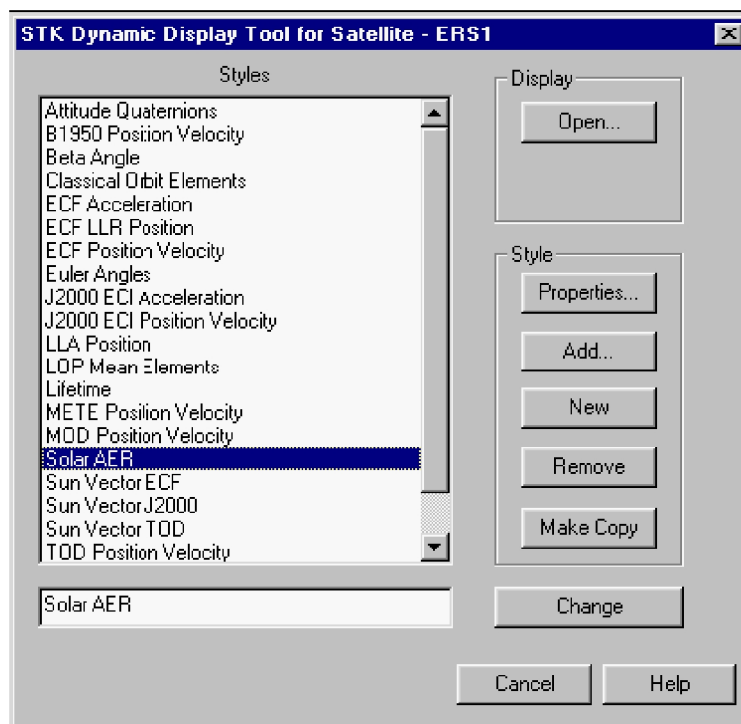
Далее в появившемся окне выбрать Solar AER в списке стилей отчета и нажать Create...



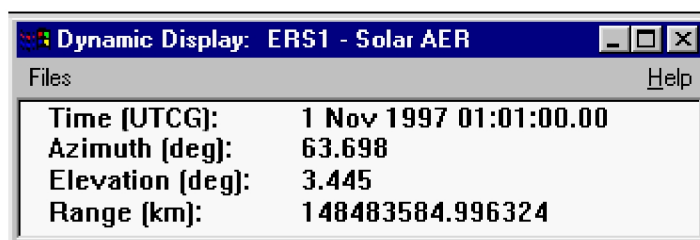
Можно кликнуть на любую точку графика, чтобы получить числовую информацию о ее координатах.

4. Динамическое отображение информации.

Можно следить за изменением любых параметров, например, азимута, долготы восходящего узла и расстояния до объекта в динамике. Для этого следует выделить ERS1 в главном окне и выбрать пункт Dynamic Display в меню Tools.

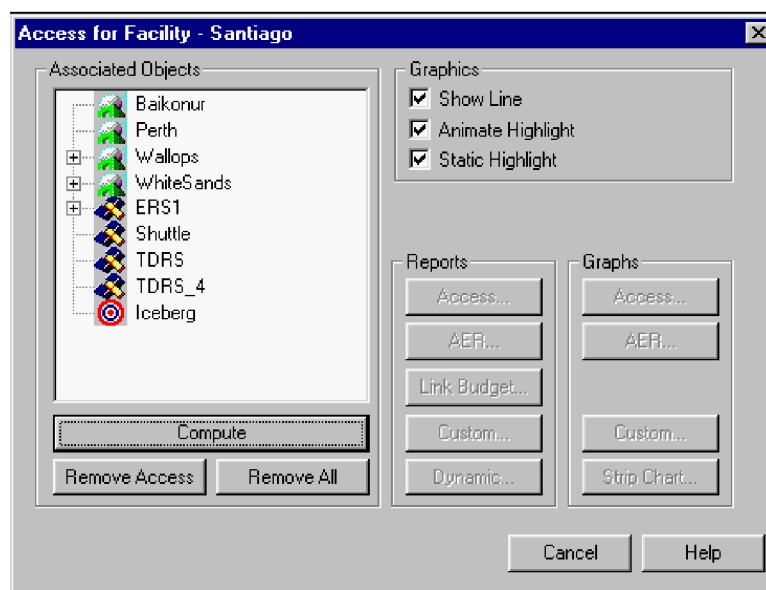


В появившемся окне выбрать Solar AER из списка доступных шаблонов и нажать Open... После этого следует запустить анимацию сценария.



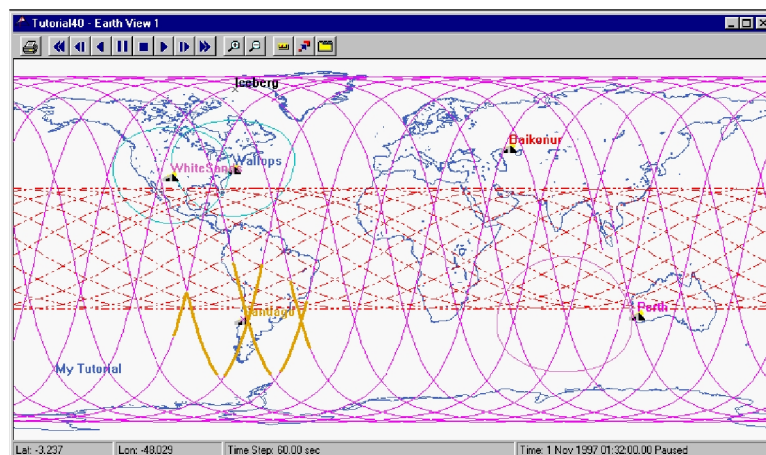
5. Анализ условий связь спутника с объектом или целью.

Для отображения этих процессов следует в главном окне выбрать объект Santiago и пункт Access в меню Tools.



В списке Associated Objects следует выбрать ERS1 и убедиться, что опции Show

Line, Animate Highlight и Static Highlight включены. Далее нажать Compute.

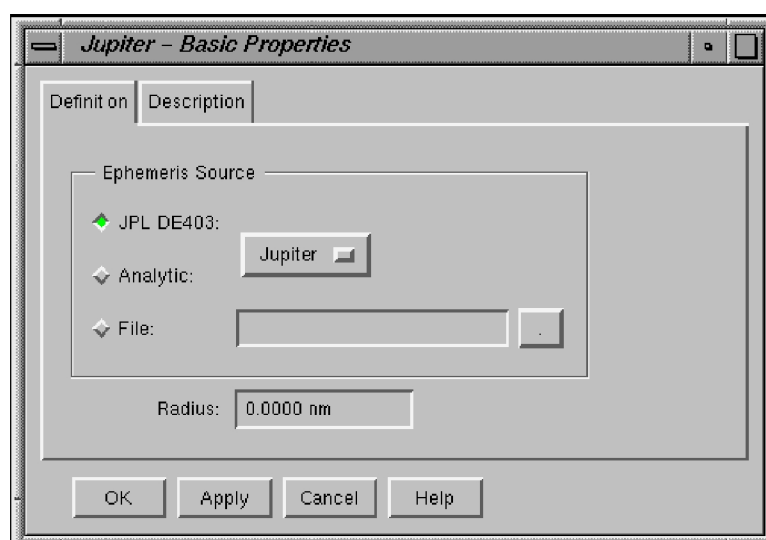


6. Анализ условий связи между спутниками.

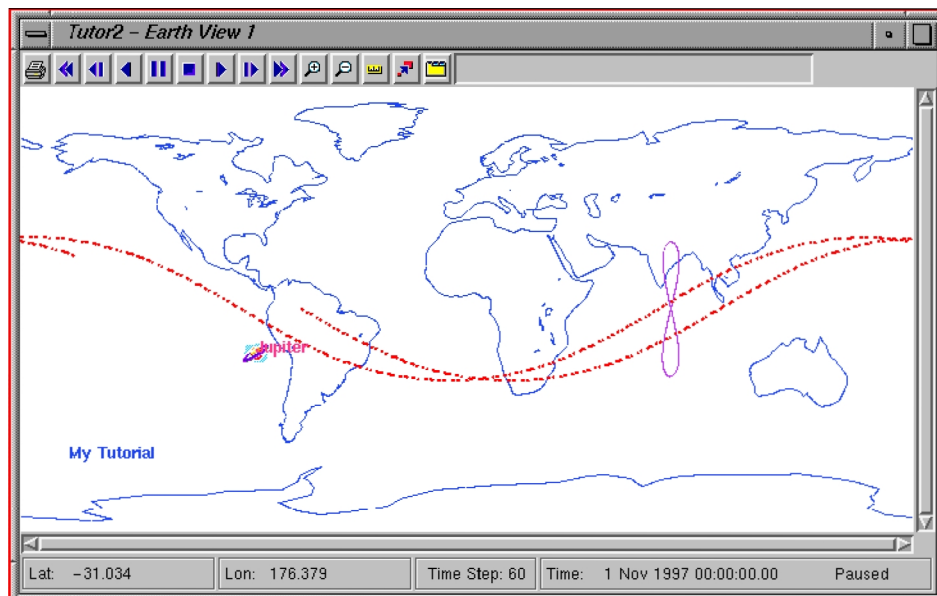
Рассматривается пример анализа возможности и планирования связи между двумя спутниками: Shuttle и TDRS_4. Для этого следует открыть окно Access для Shuttle, выделить TDRS_4 в списке Associated Objects и нажать Compute.

7. Для анализа возможности поддержки межпланетных миссий существует возможность анализа возможности связи между спутником и планетой/межпланетным КА.

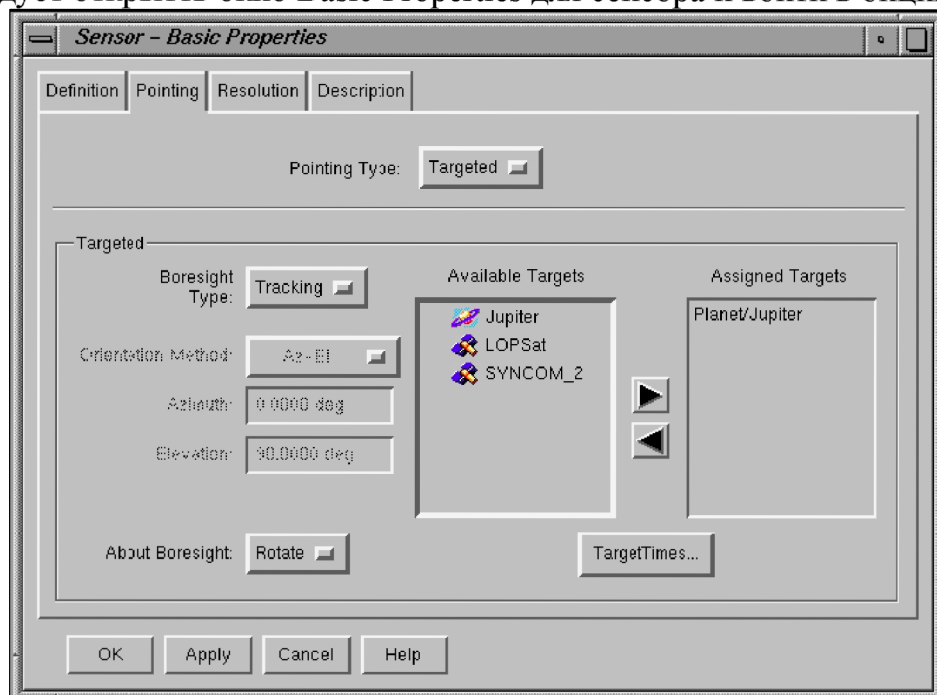
Чтобы добавить планету или межпланетный КА в сценарий следует нажать на иконку с изображением планеты и назвать новую планету, например Jupiter. Для этого следует открыть окно Basic Properties для нового объекта.



На вкладке Definition выбрать JPL DE-403, потом выделить Jupiter в меню и нажать OK. Теперь следует добавить в сценарий, например, телескоп Hubble. Для этого необходимо выбрать Insert... в меню Files, изменить File Type на Satellite и выбрать Hubble.sa.



Далее следует открыть окно Basic Properties для сенсора и войти в опцию Pointing.



Так как сенсор телескопа должен быть направлен на Юпитер, следует изменить Pointing Type на Targeted и Boresight Type на Tracking. Далее выбрать Jupiter в списке Available Targets, нажать стрелочку «вправо», чтобы добавить его в список Assigned Targets. Нажать OK.

Теперь все подготовлено для того, чтобы установить связь между телескопом Hubble и планетой Юпитер. Следует открыть окно Access для телескопа, выбрать Jupiter в списке Associated Objects и нажать Access... в поле Reports для просмотра времени связи.

Report: Satellite-Hubble-Sensor-Sensor-To-Planet-Jupiter - Access

Files Edit Help

1 Jul 1997 20:34:03

Satellite-Hubble-Sensor-Sensor-To-Planet-Jupiter: Access Summary Report

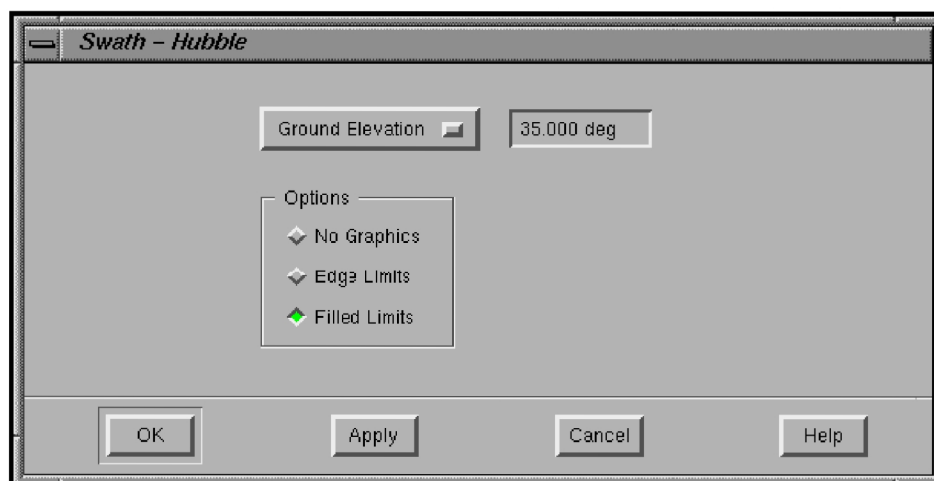
Access	Start Time (UTC0)	Stop Time (UTC0)	Duration (sec)
1	1 Nov 1997 00:00:00.00	1 Nov 1997 00:29:23.09	1763.089
2	1 Nov 1997 01:03:27.97	1 Nov 1997 02:05:54.74	3746.766
3	1 Nov 1997 02:40:01.12	1 Nov 1997 03:00:00.00	1198.880

Global Statistics

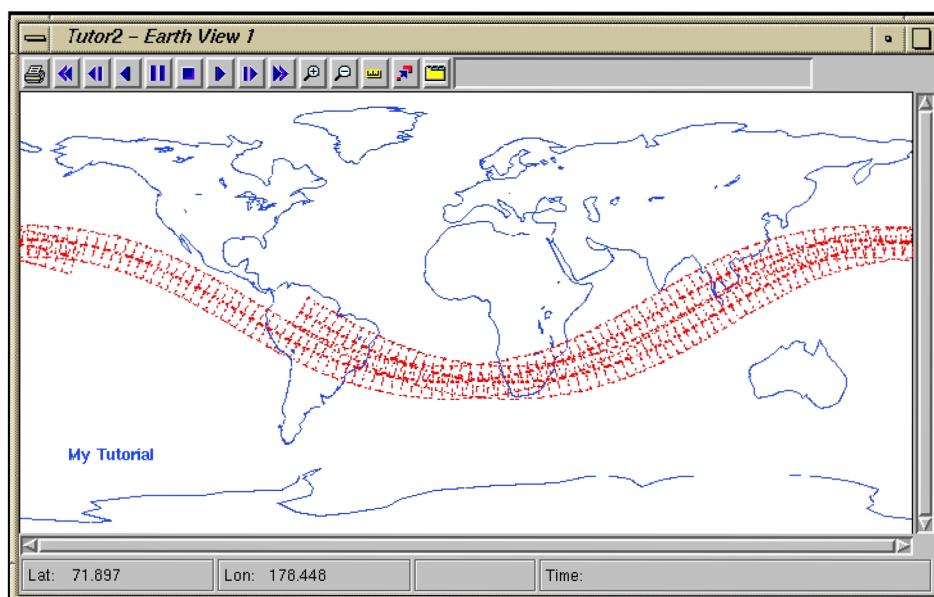
Min Duration	3	1 Nov 1997 02:40:01.12	1 Nov 1997 03:00:00.00	1198.880
Max Duration	2	1 Nov 1997 01:03:27.97	1 Nov 1997 02:05:54.74	3746.766
Mean Duration				2236.245
Total Duration				6708.735

8. Анализ обзора земной поверхности.

Для отображения полосы обзора со спутника в качестве примера выбирается Hubble. Следует в главном окне кликнуть правой кнопкой мыши и выбрать Swath в появившемся меню.



Далее необходимо изменить Ground Elevation на 35.00° и включить опцию Filled Limits, нажать Apply.



9. Анализ движения на длительном интервале времени (Long-term Orbit Predictor)

Чтобы проанализировать изменения орбиты спутника на протяжении, например, двух лет, следует выполнить долгосрочный прогноз. Для этого в главном окне следует выбрать иконку с изображением спутника. После появления мастер выбора орбиты, следует нажать Cancel. Далее задайте имя нового спутника, например, LOPSat и откройте окно Basic Properties для задания данных для этого спутника.

LOPSat - Basic Properties

Orbit | Attitude | Pass Break | Mass | Description

Start Time: 1 Nov 1997 00:00:00.00 Step Size: 1.000 day
 Stop Time: 1 Nov 1999 00:00:00.00 Propagator: LOP

Orbit Epoch: 1 Nov 1997 00:00:00.00 Semimajor Axis: 22766.8218462 nm
 Coord Epoch: 1 Jan 2000 11:59:56.92 Eccentricity: 0.00100000
 Coord Type: Classical Inclination: 28.5000 deg
 Coord System: J2000 Argument of Perigee: 360.0000 deg
 Prop Specific: Force Models... Lon. Asc. Node: 260.0 deg
 True Anomaly: 300.35 deg

OK Apply Cancel Help

Введите значения, показанные на картинке.

После того, как значения установлены, следует нажать кнопку Force Models на этой же вкладке.

LOP Force Model - LOPSat

Earth Gravity

Maximum Degree: 6
 Maximum Order: 6

Drag

☐ Use Cd: 2.300000
 Advanced Settings...

Third Body Gravity

☒ Use Solar Gravity
☒ Use Lunar Gravity

Solar Radiation Pressure

☒ Use Cp: 1.500000
 Atmosphere Height: 90.000 km

Physical Data

Drag Cross-Sectional Area: 20.000 m²
 SRP Cross-Sectional Area: 20.000 m²
 Vehicle Mass: 1000.000 kg

OK Apply Cancel Help

В появившемся окне измените Earth Gravity Maximum Degree и Maximum Order на 6. Убедитесь, что опция Use Drag выключена, а опция Use Solar Radiation Pressure включена. Далее нажмите кнопку ОК, а затем кнопку нажмите ОК на вкладке Orbit.

Приложение 1 Расчетные методики и модели для анализа выполнимости выведения заданной полезной нагрузки выбранной РКН

П1.1 Проектные характеристики ракет космического назначения

Многоступенчатая ракета состоит из полезной нагрузки, выводимой на орбиту, и ускорителей. *Ускорителем* (ракетным блоком) составной ракеты называется каждая отделяемая часть ракеты, содержащая в своем составе топливо и двигательную установку. *Ступенью* называется соединение ускорителя (ракетного блока) с полезной нагрузкой, которую разгоняет ускоритель рассматриваемой ступени. Полезной нагрузкой каждой ступени является следующая по порядку работы ступень носителя. Так, ракета на старте является *первой ступенью* носителя; часть ракеты после отделения ускорителя первой ступени называется *второй ступенью* и т.д.

Составные ракеты могут иметь последовательное, параллельное и смешанное соединение ускорителей (рис. 1).

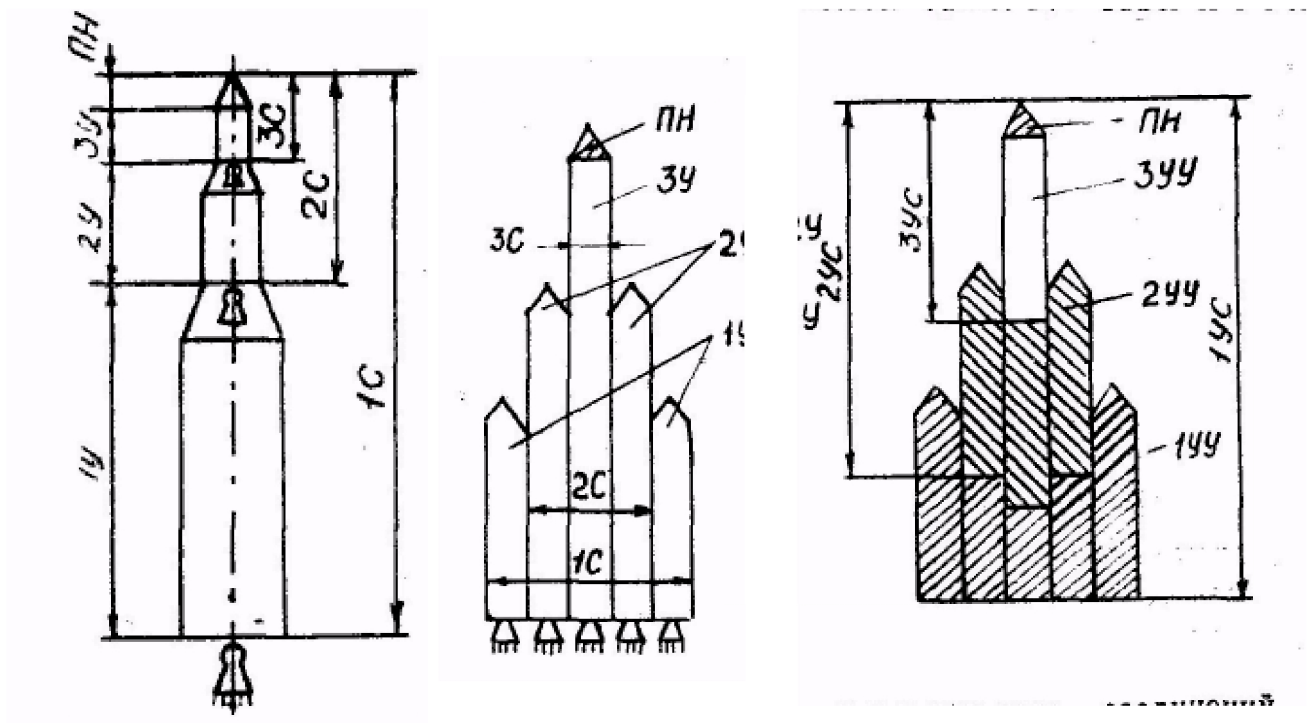


Рис.1 Схемы соединения ускорителей

При *последовательном соединении* (поперечном делении) ускорителей работа двигателей следующей ступени происходит после окончания работы двигателей и отбрасывания ускорителя предыдущей ступени.

При *параллельном соединении* (продольном делении) ускорителей одновременно работают двигатели всех ускорителей. После выгорания горючего и отделения предыдущего ускорителя продолжают работу двигатели всех остальных ускорителей.

Последовательно связанные ускорители в комбинации с параллельно присоединенными ускорителями образуют *смешанную схему ракеты* (например, ракеты – носители «Союз», «Сатурн-1 В-7»).

Расчет энергетики многоступенчатой ракеты с параллельным и смешанным соединением ускорителей приводится к расчету энергетики ракеты с последовательным соединением ускорителей введением понятия об условных ускорителях ступеней. Условный ускоритель ступени включает отбрасываемый блок и часть топлива,

выгоревшего из оставшихся блоков за время работы отбрасываемого ускорителя (см.рис.1).

При выполнении баллистических расчетов применяют следующие массовые характеристики:

- m_i — начальная масса i -ой ступени;
- m_{Ti} — масса топлива ($m_{Ti} + m_{окi}$) i -го ускорителя,
- m_{Ki} — сухая масса i -го ускорителя (масса конструкции);
- $m_{пн}$ — масса полезной нагрузки многоступенчатой ракеты.

Используются также ряд безразмерных массовых характеристик.

Относительной массой ступени называется отношение начальной массы ступени к массе ее полезной нагрузки (следующей ступени):

$$p_i = \frac{m_i}{m_{i+1}}.$$

Относительной массой многоступенчатой ракеты называется отношение ее стартовой массы к массе полезной нагрузки, выводимой на орбиту:

$$p_{\Sigma} = \frac{m_1}{m_{пн}}, p_{\Sigma} = p_1 \cdot p_2 \dots p_n = \prod_{i=1}^n p_i.$$

Этот параметр непосредственно связан со стоимостью выведения на орбиту полезной нагрузки. Для ракет-носителей космического назначения (РКН) $p_{\Sigma} = 30 \dots 100$.

Конструктивной характеристикой ускорителя называется отношение массы ускорителя к его сухой массе:

$$s_i = \frac{m_{Ki} + m_{Ti}}{m_{Ki}}.$$

Эта характеристика определяет степень совершенства конструкции ускорителей (ракетных блоков) и зависит от типа двигателей: для жидкостных ракетных двигателей (ЖРД) $s_i = 8 \dots 16$, для ракетных двигателей твердого топлива (РДТТ) $s_i = 5 \dots 11$.

Числом Циолковского (отношением масс) ступени называется отношение начальной массы ступени к ее массе после выгорания горючего работающего ускорителя:

$$z_i = \frac{m_i}{m_i - m_{\dot{O}i}}, z_i = 3 \dots 6.$$

Эти безразмерные массовые параметры связаны между собой следующим соотношением:

$$\frac{z_i - 1}{z_i} = \frac{p_i - 1}{p_i} = \frac{s_i - 1}{s_i}.$$

Для анализа возможности реализации космической миссии проводится поверочно-проектировочный расчет программной траектории запуска космического аппарата на опорную орбиту. При этом должны быть известны следующие данные.

1. Тактико-технические требования к опорной орбите КА:

— радиусы перигея r_{π} и апогея r_{α} орбиты выведения; угол истинной аномалии, ϑ_0 , определяющий положение перигея относительно точки выведения на орбиту, угол наклона орбиты i .

2. Географические условия старта: долгота λ_c и широта φ_c пункта старта.

3. Проектные характеристики носителя и его двигательных установок. Общие характеристики: стартовая масса m_0 , масса полезной нагрузки $m_{пн}$, площадь миделевого (наибольшего) сечения носителя S_M , общая длина носителя L , размах стабилизаторов l .

Проектные характеристики носителя и его двигательных установок

В таблице 1 приняты обозначения: n — число ступеней носителя, m — число ступеней, работающих в плотных слоях атмосферы.

4. Аэродинамические характеристики носителя в стартовой конфигурации, представленные в виде графических или табличных зависимостей в указанных диапазонах аргументов:

$$C_{\text{xmin}}(M, H), M = 0 \div 5, H = H_{\text{dàn}} = 10 \text{ êi}$$

Высота Н, км	0	11	25	46	54	80	95
--------------	---	----	----	----	----	----	----

Подготовка исходных данных для анализа возможности этапа выведения завершается определением системы баллистических проектных параметров и характеристик

двигательной установки. *Баллистическими проектными параметрами ракеты* называются параметры, которые при заданных конструктивной схеме ракеты, характеристиках двигательных установок и программе полета однозначно определяют конечную скорость выводимой на орбиту полезной нагрузки.

Используется следующая система проектных параметров.

1. Числа Циолковского ступеней

$$z_i = \frac{m_i}{m_i - m_{\dot{o}i}}, i = \overline{1, \dots, n}.$$

Вместо числа Циолковского могут быть использованы также одни из следующих параметров:

$$a_{ki} = \frac{m_{\dot{o}i}}{m_i} \text{ — коэффициент заполнения топливом ускорителя } i\text{-й ступени [}$$

$$\mu_{ki} = \frac{m_i - m_{\dot{o}i}}{m_i} \text{ — относительная конечная масса ступени, которые связаны между}$$

собой и с z_i соотношениями:

$$\mu_{ki} = \frac{1}{z_i}; \mu_{ki} = 1 - a_{ki}; a_{ki} = 1 - \frac{1}{z_i}; a_{ki} = 1 - \mu_{ki}; z_i = \frac{1}{\mu_{ki}}; z_i = \frac{1}{1 - a_{ki}}.$$

2. Удельные тяги двигателей на уровне моря и в пустоте:

$$P_{\dot{o}\ddot{a}0i} = \frac{P_{0i}}{\dot{m}_i}, i = \overline{1, m}, \quad P_{\dot{o}\ddot{a}\ddot{i}i} = \frac{P_{ii}}{\dot{m}_i}, i = \overline{1, n}.$$

3. Отношения удельных тяг в пустоте и на Земле (коэффициенты высотности сопел двигателя) для ступеней, работающих в плотных слоях атмосферы:

$$\lambda_i = \frac{P_{\dot{o}\ddot{a}\ddot{i}i}}{P_{\dot{o}\ddot{a}0i}} = \frac{P_{ii}}{P_{0i}}, i = \overline{1, m}.$$

4. Начальные тяговооруженности ступеней:

$$n_{0i} = \frac{P_{0i}}{m_i \cdot g_0}, i = \overline{1, m}, \quad n_{0i} = \frac{P_{ii}}{m_i \cdot g_0}, i = \overline{m+1, n}.$$

5. Начальные нагрузки на мидель для ступеней, работающих в плотных слоях атмосферы:

$$p_{Mi} = \frac{m_i}{S_{Mi}}, i = \overline{1, m}.$$

Для каждой ступени должны быть определены также следующие характеристики двигательных установок:

эффективная скорость истечения газов в пустоте $u_i = P_{\dot{o}\ddot{a}\ddot{i}i}$;

секундный расход топлива $m_i = \frac{P_{ii}}{P_{\dot{o}\ddot{a}\ddot{i}i}}$;

время работы ступени $t_{ki} = \frac{m_{\dot{o}i}}{\dot{m}_i}$.

П р и м е ч а н и е. Для носителей с параллельным соединением ускорителей 1 и 2 ступеней сначала осуществляется приведение носителя к последовательному соединению условных ускорителей:

- определяется время работы 1-й ступени, которое равно времени работы отделяемого ускорителя 1-й ступени,

$$m_{\dot{o}1} = \frac{P_{i\dot{o}1}}{P_{\dot{o}\ddot{a}\ddot{i}\dot{o}1}}; \quad t_{k1} = \frac{m_{\dot{o}\dot{a}}}{\dot{m}_{\dot{o}1}}$$

- рассчитывается масса топлива Δm_2 которая расходуется вторым ускорителем за время работы первого,

$$m_{\delta 2} = \frac{F_{\delta 2}}{P_{\delta 2}}; \quad \Delta m_2 = \dot{m}_{\delta 2} t_{\delta 2};$$

- определяются массы топлива 1 и 2 ступеней и масса второй ступени:

$$m_{T1} = m_{Ty1} + \Delta m_2; \quad m_{T2} = m_{Ty2} + \Delta m_2; \quad m_2 = m_1 - m_{y1} - \Delta m_2.$$

- определяются эффективные значения проектных параметров 1-й ступени: удельные тяги на Земле и в пустоте:

$$P_{\delta 01}^{\Sigma} = \frac{P_{01}^{\Sigma}}{\dot{m}_1} = \frac{P_{01} + P_{02}}{\dot{m}_{\delta 1} + \dot{m}_{\delta 2}}; \quad P_{\delta r1}^{\Sigma} = \frac{P_{r1}^{\Sigma}}{\dot{m}_1} = \frac{P_{r1} + P_{r2}}{\dot{m}_{\delta 1} + \dot{m}_{\delta 2}}$$

- степень высотности сопла

$$\lambda_{\delta 1}^{\Sigma} = \frac{P_{\delta r1}^{\Sigma}}{P_{\delta 01}^{\Sigma}} = \frac{P_{r1}^{\Sigma}}{P_{01}^{\Sigma}}$$

- начальная тяговооруженность 1-й ступени

$$n_{01} = \frac{P_{01}^{\Sigma}}{m_1 g_0} = \frac{P_{01} + P_{02}}{m_1 g_0}.$$

П1.2 Приближенная оценка возможности запуска на орбиту

Конечная скорость запуска на опорную орбиту является одновременно начальной скоростью орбитального полета, поэтому она определяется через заданные параметры орбиты выведения. Расчет орбитальных скоростей движения КА отражен в разделе невозмущенное движение. Обычно вывод КА производится в перигей орбиты и в направлении, совпадающем с местной горизонталью.

Начальные орбитальные скорости V_0 при запуске в перигее на круговую, эллиптическую, параболическую или гиперболическую орбиты рассчитываются по формулам

$$V_{\delta 0} = \sqrt{\frac{\mathfrak{R}}{R + H_0}}; \quad V_{\delta \infty} = V_{\delta 0} \cdot \sqrt{2 - \frac{r_{\pi}}{a}}; \quad V_{\delta \infty} = \sqrt{2} \cdot V_{\delta 0}; \quad V_{\delta \infty} = \sqrt{V_{\delta 0}^2 + V_{\infty}^2},$$

где $\mathfrak{R} = 3.98602 \cdot 10^5 \frac{\text{м}^3}{\text{с}^2}$ гравитационный параметр Земли; $R=6371$ км — средний радиус Земли; H_0 — заданная высота круговой орбиты или ее перигея; $r_{\pi} = R + H_0$ — радиус перигея орбиты выведения; $a = 0.5(r_{\pi} + r_{\alpha})$ — большая полуось эллиптической орбиты; r_{α} — радиус апогея орбиты выведения; V_{∞} — гиперболический избыток скорости при выходе из гравитационного поля Земли

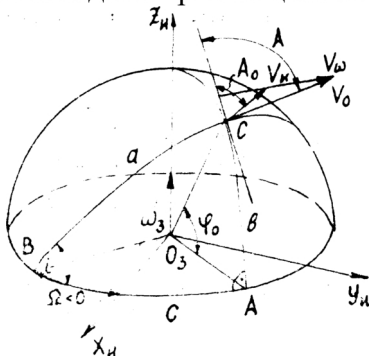


Рис. 2. Определение азимута стрельбы

Для определения азимута стрельбы A_0 необходимо вычислить угол между плоскостью опорной орбиты и местным меридианом пункта старта (рис. 2) из формулы косинусов углов сферического прямоугольного треугольника ABC:

$$A = \arcsin \left(\frac{\cos i}{\cos \varphi_0} \right).$$

Скорость полета в конце участка выведения на промежуточную орбиту относительно стартовой системы координат определяется из формулы

$$V_k = \sqrt{V_0^2 + V_\omega^2 - 2 \cdot V_0 \cdot V_\omega \cdot \sin A},$$

где V_0 — орбитальная скорость на промежуточной орбите; $V_\omega = \omega_3 \cdot R \cdot \cos \varphi_0$ — скорость стартового стола, обусловленная вращением Земли; $\omega_3 = 7.2921 \cdot 10^{-5} \text{ 1/c}$ — угловая скорость вращения Земли; $R = 6371 \text{ км}$ — средний радиус Земли.

Азимут стрельбы вычисляется по формуле

$$A_0 = \arccos \left(\frac{V_0}{V_k} \cdot \cos A \right).$$

Оценка энергетических возможностей носителя производится с целью проверки выполнимости задачи выведения носителем номинальной полезной нагрузки на опорную орбиту. Запасы топлива в ускорителях, конструктивно-компоновочная схема ракеты и удельные характеристики двигательной установки определяют располагаемую характеристическую скорость.

Располагаемой характеристической скоростью РКН называют скорость, которую теоретически он может развить под действием реактивных сил, двигаясь прямолинейно в вакууме при отсутствии гравитации. Эта скорость определяется известной формулой Циолковского:

$$V_{\text{Х} \delta \alpha \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}} = \sum_{i=1}^n V_{\text{е} \delta \alpha \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}} = \sum_{i=1}^n u_i \ln z_i.$$

где $u_i = P_{\delta \alpha \tilde{\alpha} \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}}$ — эффективная скорость истечения газов; n — число ступеней ракеты.

При вычислении $V_{\delta \alpha \tilde{\alpha} \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}}$ следует зафиксировать характеристические скорости каждой ступени в отдельности, которые могут служить верхней оценкой развиваемых каждой ступенью приращений скорости.

Поставленная задача выведения КА на опорную орбиту определяет требуемые запасы топлива носителя и, следовательно, требуемую для запуска характеристическую скорость. Сначала определяется идеальная требуемая характеристическая скорость запуска. *Идеальной требуемой характеристической скоростью запуска* называется скорость, которая должна быть сообщена ЛА на поверхности Земли мгновенным импульсом, чтобы КА вышел без сопротивления атмосферы в точку, соответствующую концу активного участка, с заданной орбитальной скоростью V_0 . Эта скорость определяется с помощью интеграла энергии:

$$V_{\text{х} \delta \alpha \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}} = \sqrt{V_0^2 + \frac{2\Re}{R} \left(1 - \frac{R}{R + H_0} \right)};$$

Идеальные характеристические требуемые скорости запуска КА на круговую, эллиптическую, параболическую и гиперболическую орбиты вычисляются соответственно последующим формулам:

$$V_{\text{х} \delta \alpha \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}} = \sqrt{\frac{\Re}{R + H_0}} \cdot \sqrt{\frac{2 \cdot (R + H_0)}{R} - 1} = V_{\text{е} \delta \alpha \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}} \cdot \sqrt{\frac{2 \cdot (R + H_0)}{R} - 1};$$

$$V_{\text{х} \gamma \text{е} \delta \tilde{\alpha} \tilde{\alpha} \tilde{\alpha}} = \sqrt{\frac{\Re}{R} \cdot \left(2 - \frac{R}{a} \right)} = V_1 \cdot \sqrt{2 - \frac{R}{a}};$$

$$V_{\tilde{o}i\dot{\alpha}\dot{o}} = \sqrt{\frac{2\Re}{R}} = V_{\Pi}$$

$$V_{\tilde{o}\tilde{\alpha}\dot{e}i} = \sqrt{\frac{2\Re}{R} + V_{\infty}^2} = \sqrt{V_{\Pi}^2 + V_{\infty}^2};$$

где V_I и V_{Π} — первая и вторая космические скорости.

П1.3 Расчет и анализ программной траектории движения первой ступени ракеты-носителя

При поверочно-проектировочном расчете траектории первой ступени рассматривается движение относительно стартовой системы координат в вертикальной плоскости, и принимаются следующие допущения: пренебрегают переносной и кориолисовой силами инерции, поле силы тяжести принимается однородным и плоскопараллельным, секундный расход массы принимается постоянным.

Система дифференциальных уравнений в проекциях на оси траекторной системы координат с учетом принятых допущений имеет следующий вид:

$$\dot{V} = g_0 \cdot (n_{xk} - \sin \Theta), \quad V_0 = 0;$$

$$\dot{\Theta} = \frac{g_0}{V} \cdot (n_{yk} - \cos \Theta), \quad \Theta_0 = \pi/2;$$

$$\dot{x} = V \cdot \cos \Theta, \quad x_0 = 0;$$

$$\dot{y} = V \cdot \sin \Theta, \quad y_0 = 0;$$

где $n_{xk} = \frac{P - X_a}{mg_0}; \quad n_{yk} = \alpha^0 \cdot \frac{(P/57,296) + Y_a^{\alpha^0}}{mg_0};$

$$m = m_0 - |\dot{m}| \cdot t; \quad P = P_i - (P_i - P_0) \cdot \frac{p(y)}{P_0};$$

$$X_a = C_{xa} q S; \quad Y_a^{\alpha^0} = C_{ya}^{\alpha^0} / S.$$

Параметры атмосферы, используемые при вычислении подъемной силы и лобового сопротивления, заданы таблицами стандартной атмосферы в виде зависимостей плотности ρ , температуры T , давления p и скорости звука a от высоты.

Через проектные параметры первой ступени величины, стоящие и правых частях дифференциальных уравнений, а также время работы двигателя определяются по формулам:

$$n_x = \frac{P - X_a}{mg_0} = \frac{n_{01} \cdot \left[\lambda - (\lambda - 1) \cdot \frac{p}{p_0} \right] - \frac{1}{g_0 p_M} \cdot C_{xa} q}{1 - \frac{n_{01} g_0}{P_{\dot{o}\tilde{\alpha}01}} t};$$

$$n_y = \alpha^0 \cdot \frac{(P/57,296) + Y_a^{\alpha^0}}{mg_0} = \alpha^0 \cdot \frac{\frac{n_{01}}{57,296} \left[\lambda - (\lambda - 1) \cdot \frac{p}{p_0} \right] + \frac{1}{g_0 p_M} \cdot C_{xa} q}{1 - \frac{n_{01} g_0}{P_{\dot{o}\tilde{\alpha}01}} t};$$

$$t_{e1} = \frac{z_1 - 1}{z_1} \cdot \frac{P_{\dot{o}\tilde{\alpha}01}}{n_{01} g_0}.$$

При выборе программы полета носителя на активном участке необходимо обеспечить полезной нагрузке максимально возможную конечную скорость для заданных конечных значений угла наклона траектории и высоты с учетом необходимых ограничений, наложенных условиями старта, аэродинамикой, прочностью конструкции и возможностями системы управления.

Большинство ограничений наложены на движение первой ступени, поэтому оптимальная программа движения первой ступени строится по ограничениям. Единственная возможность оптимизации заключается в варьировании участка ненулевых углов атаки.

В соответствии с ограничениями траектория первой ступени разделяется на характерные участки (рис.3).

Стартовый вертикальный участок возможно короткой длительности, продолжительность которого определяется временем, необходимым, чтобы развить достаточную для эффективного действия органов управления скорость $V_1 \cong 50 \text{ м/с}$. Этот момент времени t_1 может быть приближенно определен из второй формулы Циолковского, которая является трансцендентным уравнением относительно t_1 :

$$V_1 = P_{\text{дв}} \ln \frac{1}{1 - \frac{n_{01} g_0}{P_{\text{дв}}} t_1} - g_0 t_1 \quad 1$$

Программа на вертикальном участке: $\alpha = 0 (\varphi = \Theta = \pi/2)$.

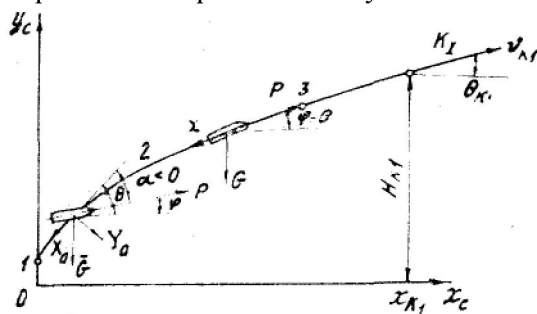


Рис. 3. Основные участки траектории первой ступени

Начальный участок разворота, на котором система управления быстро отклоняет продольную ось носителя до заданного максимального отрицательного угла атаки α_m затем постепенно уменьшает угол атаки до нулевого или пренебрежимо малого значения.

Длительность этого участка определяется моментом времени t_2 , когда достигается скорость $V_2 = 270 \text{ м/с}$, соответствующая $M_2 = 0,8$. Этот момент времени приближенно рассчитывается также из уравнения Циолковского:

$$V_2 = P_{\text{дв}} \ln \frac{1}{1 - \frac{n_{01} g_0}{P_{\text{дв}}} t_2} - g_0 t_2$$

Программа изменения угла атаки может быть взята в виде непрерывной функции (рис.4):

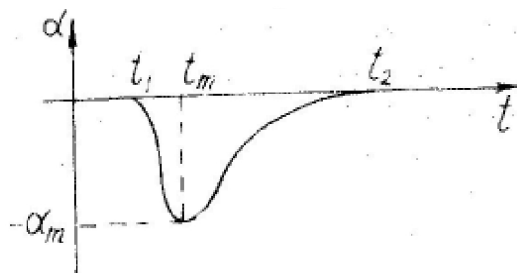


Рис.4. Программа изменения угла атаки

$\alpha = -\alpha_m \sin^2 \frac{\pi(t-t_1)}{(t-t_1) + \Re(t_2-t)}$, где $\Re = \frac{t_m - t_1}{t_2 - t_m} = \frac{1}{4}$; t_m — момент времени, когда достигается минимум угла атаки.

Такая зависимость определяет семейство программ, зависящих от одного параметра α_m . Варьированием этого параметра α_m определяется программа, обеспечивающая в конце работы первой ступени нужный угол наклона траектории Θ_{k1} . Угол α_m ориентировочно можно определить по графикам рис.5-6 в зависимости от заданного Θ_{k1} и от тяговооруженности ступени n_{01} . Выбрав шаг варьирования $\Delta\alpha_m$ в окрестности α_m по результатам расчетов траектории следует построить для заданного носителя график $\Theta_{k1}(\alpha_m)$, по которому определяется уточненное значение α_m . Для найденного значения α_m окончательно выполняется расчет траектории первой ступени.

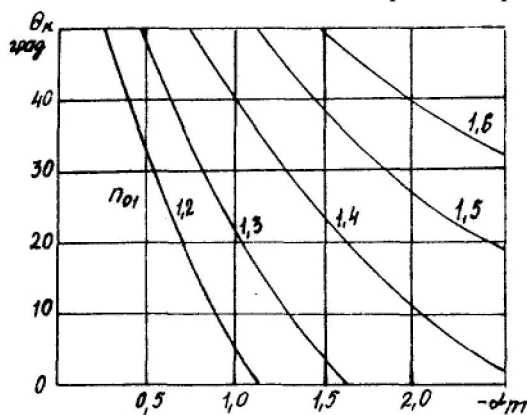


Рис.5. Зависимость Θ_{k1} от α_m

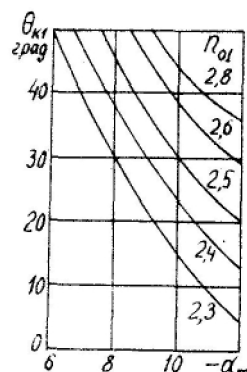


Рис. 6. Зависимость Θ_{k1} от α_m для больших n_{01}

На участке гравитационного разворота, на котором угол атаки равен нулю, искривление траектории носителя происходит только под действием силы тяжести. За это время носитель проходит околозвуковой диапазон скоростей и совершает разгон до гиперзвуковых скоростей, одновременно преодолевая плотные слои атмосферы с минимальным лобовым сопротивлением.

Примечание. При запуске КА ориентировочно можно принимать значения угла наклона траектории Θ_{k1} , в конце работы первой ступени в зависимости от высоты выведения H_0 :

Высота орбиты, км	185	200	250	300	400	500
Угол наклона траектории в конце первой ступени, град	20	25	28	30	35	40

Если на 1-й ступени установлены твердотопливные ускорители с коротким временем работы, то следует ориентировочно принимать значение угла наклона траектории в конце работы второй ступени $\Theta_{k2} = 0.5 \cdot \Theta_{k1}$, где Θ_{k1} — берется в зависимости от высоты орбиты.

Для многоступенчатых РКН этим участком заканчивается программа первой ступени. В конце участка гравитационного разворота РКН выйдет на определенную высоту $y_{k1} = H_{k1}$, разовьет скорость V_{k1} и будет иметь заданный угол наклона траектории Θ_{k1} , которые являются начальными условиями для построения программы и расчета траектории второй ступени.

П1.4 Расчет и анализ программных траекторий внеатмосферных ступеней ракеты космического назначения

Выбор схемы выведения

Перед началом расчета движения верхних ступеней носителя, работающих в разреженных слоях атмосферы ($H > 50$ км), необходимо произвести выбор схемы выведения. В зависимости от структуры и назначения носителя могут встретиться следующие варианты схем выведения на опорную орбиту.

В а р и а н т 1. Вывод на орбиту обеспечивает одна верхняя (вторая) ступень двухступенчатого носителя с ЖРД. При выборе программы движения решается двухпараметрическая краевая задача выведения ($H_k = H_{ид\acute{a}}$, $\Theta_k = \Theta_{ид\acute{a}}$).

В а р и а н т 2. Вывод на орбиту завершают две верхние ступени трехступенчатого носителя с ЖРД. Промежуточная ступень обеспечивает постепенное уменьшение угла наклона траектории до $\Theta_{k2} = (1/3 \dots 1/4)\Theta_{k1}$. При выборе программы движения этой ступени решается однопараметрическая краевая задача. Верхняя ступень завершает вывод на орбиту. При выборе программы полета верхней ступени решается двухпараметрическая краевая задача ($H_{ид\acute{a}}$, $\Theta_{ид\acute{a}}$).

В а р и а н т 3. Вывод на орбиту завершают две верхние ступени с РДТТ. Двигатели на твердом топливе работают короткое время, за которое при непрерывной работе двигателей полезная нагрузка не успевает подняться на заданную высоту. В этом случае вводится пассивный участок полета между промежуточной и верхней ступенями носителя. Промежуточная ступень обеспечивает выведение под таким углом наклона траектории в конце работы ступени, чтобы при пассивном полете по баллистической траектории в апогее была достигнута заданная высота орбиты $H_{ид\acute{a}}$. В апогее включается двигатель верхней ступени, обеспечивающий разгон полезной нагрузки до орбитальной скорости при соблюдении условия $\Theta = 0$.

При определении траекторий верхних ступеней носителя необходимо учитывать кривизну поверхности Земли и неоднородность поля силы тяжести. Аэродинамическими и инерционными силами, обусловленными вращением Земли, пренебрегаются. Система дифференциальных уравнений движения в проекциях на оси стартовой системы координат (рис. 7) с учетом принятых допущений после линеаризации проекций гравитационного ускорения имеет вид :

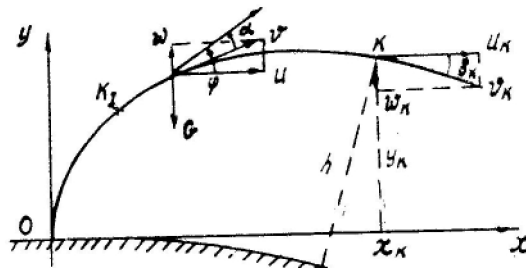


Рис.7. Схема движения верхней, ступени носителя

$$\left. \begin{aligned} \dot{u} &= p \cos \varphi - v^2 x, \\ \dot{\omega} &= p \sin \varphi - g_0 + 2v^2 y, \\ x &= u, \dot{y} = \omega, \end{aligned} \right\}$$

где $p \frac{P}{m} = \frac{n_0 g_0}{1 - \beta \cdot t} = \frac{n_0 g_0}{1 - a \cdot \xi}$ – ускорение силы тяги;

$\beta = \frac{\dot{m}}{m} = \frac{n_0 g_0}{P_{ид\acute{a}}}$ – относительный секундный расход топлива;

$a = m_T / m$ — коэффициент наполнения топливом ступени;

$\xi = t/t_k$ — безразмерное время;

m_T, m — масса топлива и начальная масса ступени;

t_k — время работы ступени; $v^2 = g_0/R$.

Начальные условия выражаются через параметры, полученные в конце траектории предыдущей ступени:

$$u_0 = V_{k1} \cos \Theta_{k1}, \quad \omega_0 = V_{k1} \sin \Theta_{k1}, \quad x_0 = x_{k1}, \quad y_0 = y_{k1}.$$

Конечные условия зависят от варианта схемы выведения.

В качестве приближенно-оптимальной программы угла тангажа принимается программа, полученная из решения вариационной задачи движения верхней ступени в плоскопараллельном гравитационном поле вне атмосферы (рис.8):

$$tg \varphi = tg \varphi_0 + Bt, \text{ где } B = b/t_k$$

После подбора параметров оптимальной программы угла тангажа по найденным параметрам окончательно рассчитываются конечные параметры движения верхней ступени относительно стартовой системы координат по формулам. Затем определяются ошибки конечных параметров и осуществляется их сравнение с заданными допустимыми отклонениями.

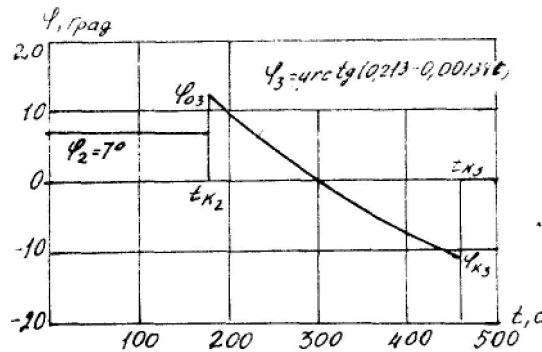


Рис. 8. Оптимальные программы движения верхних ступеней

Примечание. Для варианта запуска КА после завершения расчета конечных параметров движения с табличным значением коэффициента наполнения топливом ступени a' следует сделать пересчет конечных параметров движения на действительное значение коэффициента a ступени по следующему алгоритму:

по первой формуле Циолковского определяются приращение скорости и конечная скорость за полное время работы ступени:

$$\Delta V = u_i (\ln z - \ln z'), \quad z' = 1/(1 - a'), \quad V_k = V'_k + \Delta V;$$

по второй формуле Циолковского определяется приращение пути, проходимого последней ступенью по дуге орбиты за время $(t_k - t'_k)$:

$$\Delta S = \left[V_0 t_k + u_i t_k \cdot \left(1 - \frac{\ln z}{z-1} \right) \right] - \left[V_0 t'_k + u_i t'_k \cdot \left(1 - \frac{\ln z'}{z'-1} \right) \right]$$

где V_0 — начальная скорость ступени;

находится приращение угловой дальности ступени и полная угловая дальность:

$$\Delta \delta_k = \Delta S / (R + H_{i\partial a}), \quad \delta_k = \delta'_k + \Delta \delta_k;$$

рассчитываются конечные параметры движения ступени.

$$x_k = (R + H_{i\partial a}) \cdot \sin \delta_k, \quad y_k = (R + H_{i\partial a}) \cdot \cos \delta_k - R,$$

$$u_k = V_k \cos \delta_k, \quad \omega_k = -V_k \sin \delta_k.$$

В результате решения двухпараметрической краевой задачи выведения па орбиту при

полном выгорании топлива последней ступени достигается максимально возможная конечная скорость, которая может не совпадать со скоростью, необходимой для движения по заданной орбите. Для обеспечения вывода полезной нагрузки с нужной скоростью необходимо сделать пересчет времени движения последней ступени, что эквивалентно изменению запаса топлива. Принимая, что в конце работы последней ступени движение происходит без изменения угла наклона траектории (для КЛА $\Theta_{i\dot{\alpha}} = 0$) и без сопротивления атмосферы, для пересчета скорости допустимо применять формулу Циолковского.

Находится избыток (недостаток) скорости с учетом добавки от вращения Земли: $\Delta V = V_k + \Delta V_{\dot{\alpha}\dot{\alpha}} - V_{i\dot{\alpha}\dot{\alpha}}$. Этот избыток (недостаток) скорости возникает из-за излишнего (недостаточного) запаса топлива последней ступени.

Определяется потребный запас топлива и время работы ступени по формулам:

$$\Delta V = u_n \cdot (\ln z_n - \ln z'_n) + g_i (t_{kn} - t'_{kn}) \sin \Theta_{i\dot{\alpha}\dot{\alpha}},$$

$$z'_n = z_n \exp(-\Delta V/u_i);$$

$$m'_{\dot{\alpha}\dot{\alpha}} = \frac{z'_n - 1}{z'_n} m_n;$$

$$t'_{kn} = m'_{Tn} / \dot{m}_n,$$

Здесь n номер последней ступени, величины со штрихами соответствуют потребным значениям.

С новым значением времени работы t'_{kn} последней ступени носителя необходимо повторить расчет двухпараметрической (для КА) краевой задачи выведения по ранее приведенным алгоритмам.

Как видно из изложенного выше, выведение на заданную орбиту требует решения трехпараметрической для КА краевой задачи, которая решается методом последовательных приближений.

После пересчета следует найти оценку массы полезной нагрузки, которая может быть выведена на заданную орбиту, и если есть избыток (недостаток) топлива Δm_T , то его необходимо учесть в полезной нагрузке:

$$\Delta m_T = m_{Tn} - m'_{Tn}; \quad m'_{ii} = m_{ii} + \Delta m_{\dot{\alpha}\dot{\alpha}}.$$

Пересчет конечных параметров из стартовой системы координат в инерциальную геоцентрическую систему координат

В результате расчета траектории выведения найдены координаты и проекции скорости движения в конце активного участка относительно стартовой системы координат $Ox_c y_c z_c$. Для определения характеристик орбитального движения необходимо для этого же момента времени вычислить координаты и проекции скорости относительно инерциальной системы отсчета.

В качестве инерциальной системы отсчета возьмем геоцентрическую экваториальную систему координат, ось $O_c x_e$ которой проходит через меридиан точки старта в момент окончания активного участка. Заметим, что введенная таким образом инерциальная система координат повернута относительно звездной геоцентрической инерциальной системы на угол S , где S — местное звездное время в точке старта в момент выхода КА на опорную орбиту.

Положение стартовой системы координат $O_3 x_c y_c z_c$

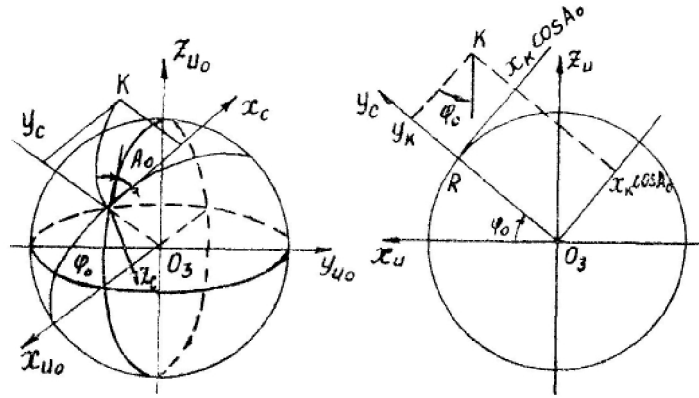


Рис. 9 Переход от стартовой к инерциальной системе координат.

относительно принятой инерциальной $O_3x_ey_ez_e$ определяется широтой пункта старта φ_0 и азимутом запуска A_0 (рис. 9).

Переход от координат конца активного участка x_k, y_k, z_k в стартовой системе к начальным координатам x_0, y_0, z_0 орбитального движения в геоцентрической инерциальной системе (рис. 9)) выполняется по формулам :

$$x_0 = -x_k \cos A_0 \sin \varphi_0 + (R + y_k) \cos \varphi_0;$$

$$y_0 = x_k \sin A_0;$$

$$z_0 = x_k \cos A_0 \cos \varphi_0 + (R + y_k) \sin \varphi_0.$$

Величина радиуса - вектора начальной точки орбитального движения

$$r_0 = \sqrt{x_0^2 + y_0^2 + z_0^2}.$$

Проекции относительной скорости V_k на оси геоцентрической системы $O_3x_ey_ez_e$ выражаются через проекции относительной скорости u_k, ω_k на стартовые оси аналогичными формулами.

Абсолютная скорость в начале орбитального движения складывается из относительной скорости $V^r = V_k$, и переносной скорости, которая определяется формулой:

$$\vec{V}^e = \vec{\omega}_3 \times \vec{r}_0 = \begin{vmatrix} \vec{i}_e & \vec{j}_e & \vec{\mathfrak{R}}_e \\ 0 & 0 & \omega_3 \\ x_0 & y_0 & z_0 \end{vmatrix} = -\omega_3 y_0 \vec{i}_e + \omega_3 x_0 \vec{j}_e,$$

где $\vec{i}_e, \vec{j}_e, \vec{\mathfrak{R}}_e$ – единичные векторы геоцентрической системы координат.

Таким образом, проекции абсолютной скорости на геоцентрические оси координат и начальной точке орбиты определяются формулами:

$$V_{x0} = \dot{x}_0 = -u_k \cos A_0 \sin \varphi_0 + \omega_k \cos \varphi_0 - \omega_3 y_0;$$

$$V_{y0} = \dot{y}_0 = u_k \sin A_0 + \omega_3 x_0;$$

$$V_{z0} = \dot{z}_0 = u_k \cos A_0 \cos \varphi_0 + \omega_k \sin \varphi_0.$$

Величина начальной скорости орбитального движения и угол наклона ее к местному горизонту соответственно

$$V_0 = \sqrt{\dot{x}_0^2 + \dot{y}_0^2 + \dot{z}_0^2}, \quad \Theta_0 = \arcsin \frac{x_0 \dot{x}_0 + y_0 \dot{y}_0 + z_0 \dot{z}_0}{r_0 V_0}.$$

Приложение 2. Расчетные методики и справочные данные для анализа орбитальных этапов космической миссии

Формулировка задачи.

В результате расчета траектории выведения для ракеты-носителя найдены координаты $x_{\text{СК}}, y_{\text{СК}}, (z_{\text{СК}} = 0)$ и проекции скорости движения $V_{x\text{СК}}, V_{y\text{СК}}, (V_{z\text{СК}} = 0)$ в конце активного участка относительно стартовой системы координат $Ox_c Y_c Z_c$. Для определения характеристик реализовавшейся орбиты выведения космического аппарата необходимо для этого же момента времени вычислить координаты x_0, y_0, z_0 и проекции скорости V_{x0}, V_{y0}, V_{z0} относительно инерциальной системы отсчета $O_z X_u Y_u Z_u$ (начальные условия пассивного движения) и орбитальные элементы (долготу восходящего узла $\dot{\Omega}$ относительно принятой системы отсчета, наклонение орбиты относительно плоскости экватора i , эксцентриситет e и большую полуось орбиты a , аргумент перигея ω , момент времени прохождения через перигей τ).

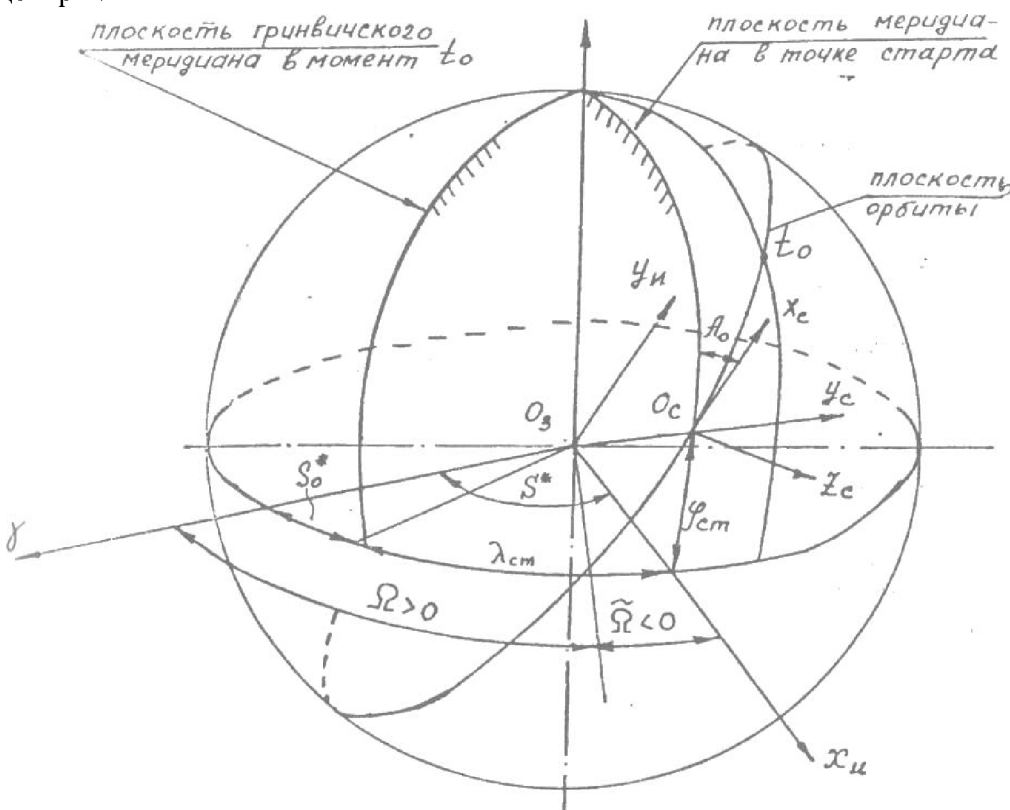


Рис. 1 Основные системы координат и их ориентация в пространстве

координат

В качестве инерциальной системы отсчета выбрана геоцентрическая экваториальная система координат, ось $O_3 X_u$ которой проходит через меридиан точки старта в момент окончания активного участка. Введенная таким образом инерциальная система координат повернута относительно звездной геоцентрической инерциальной системы на угол S^* , где S^* - местное звездное время в момент выхода космического аппарата на опорную орбиту.

Положение стартовой системы координат $Ox_c Y_c Z_c$ относительно принятой инерциальной $O_x X_u Y_u Z_u$ определяется широтой пункта старта $\varphi_{ст}$ и азимутом запуска A_0 (рис.1).

Переход от координат конца активного участка $x_{\text{ск}}, y_{\text{ск}}, z_{\text{ск}}$ стартовой системе к начальным координатам x_0, y_0, z_0 орбитального движения в геоцентрической инерциальной системе (рис. 1) выполняется по формулам:

$$\begin{aligned} x_0 &= -x_{\text{ск}} \cos A_0 \sin \varphi_{\text{ст}} + (R_{\text{з}} + y_{\text{ск}}) \cos \varphi_{\text{ст}} \\ y_0 &= x_{\text{ск}} \sin A_0 \\ z_0 &= x_{\text{ск}} \cos A_0 \sin \varphi_{\text{ст}} + (R_{\text{з}} + y_{\text{ск}}) \cos \varphi_{\text{ст}} \end{aligned} \quad (1)$$

Величина радиуса-вектора начальной точки орбитального движения

$$r_0 = \sqrt{x_0^2 + y_0^2 + z_0^2}. \quad (2)$$

Проекции относительной скорости $V_{\text{ск}}$ на оси геоцентрической $O_{\text{з}}X_u Y_u Z_u$ выражаются через проекции относительной скорости $V_{x\text{ск}}, V_{y\text{ск}}, V_{z\text{ск}} = 0$ на стартовые оси аналогичными формулами.

Абсолютная скорость в начале орбитального движения складывается из относительной скорости, $\bar{V}^r = \bar{V}_{\text{ск}}$ и переносной скорости, которая определяется формулой

$$\bar{V}^e = \bar{\omega}_{\text{з}} \times \bar{r}_0 = \begin{vmatrix} \bar{i}_u & \bar{j}_u & \bar{k}_u \\ 0 & 0 & \omega_{\text{з}} \\ x_0 & y_0 & z_0 \end{vmatrix} = -\omega_{\text{з}} y_0 \bar{i}_u + \omega_{\text{з}} x_0 \bar{j}_u$$

где $\bar{i}_u, \bar{j}_u, \bar{k}_u$ единичные векторы геоцентрической системы координат.

Таким образом, проекции абсолютной скорости на геоцентрические оси координат в начальной точке орбиты определяются формулами

$$\begin{aligned} V_{x0} &= -V_{x\text{ск}} \cos A_0 \sin \varphi_{\text{ст}} + V_{y\text{ск}} \cos \varphi_{\text{ст}} - \omega_{\text{з}} y_0 \\ V_{y0} &= V_{x\text{ск}} \sin A_0 + \omega_{\text{з}} x_0 \\ V_{z0} &= V_{x\text{ск}} \cos A_0 \sin \varphi_{\text{ст}} + V_{y\text{ск}} \cos \varphi_{\text{ст}} \end{aligned} \quad (3)$$

Величина начальной скорости орбитального движения и угол наклона ее к местному горизонту соответственно:

$$\begin{aligned} V_0 &= \sqrt{V_{x0}^2 + V_{y0}^2 + V_{z0}^2} \\ \theta_0 &= \arcsin \frac{x_0 V_{x0} + y_0 V_{y0} + z_0 V_{z0}}{r_0 V_0} \end{aligned} \quad (4)$$

Определение элементов орбиты выведения

При запуске космических аппаратов определяются элементы орбиты выведения. Сначала находятся компоненты и модуль векторной константы интеграла площадей (секториальной скорости):

$$\begin{aligned} C_1 &= y_0 V_{y0} - z_0 V_{y0} \\ C_2 &= z_0 V_{x0} - x_0 V_{z0} \\ C_3 &= x_0 V_{y0} - y_0 V_{x0} \\ C &= \sqrt{C_1^2 + C_2^2 + C_3^2} \end{aligned} \quad (5)$$

Расчет элементов орбиты производится в следующей последовательности:

-долгота восходящего узла $\dot{\Omega}$ относительно принятой инерциальной системы отсчета:

$$\text{tg} \dot{\Omega} = -\frac{C_1}{C_2}$$

$$\begin{aligned}\operatorname{sig} n(\sin \tilde{\Omega}) &= \operatorname{sig} n C_1 \\ \operatorname{sig} n(\cos \tilde{\Omega}) &= \operatorname{sig} n C_2\end{aligned}\quad (6)$$

-наклонение орбиты относительно плоскости экватора

$$\cos i = \frac{C_3}{c} \quad (7)$$

-эксцентриситет e , большая полуось орбиты a , период обращения T :

$$\begin{aligned}e &= \sqrt{1 + (v_0 - 2)V_0 \cos^2 \theta_0} \\ v_0 &= \frac{V_0^2 r_0}{\mu} = \left(\frac{V_0}{V_{\text{кр}}} \right)^2 \\ a &= \frac{r_0}{2 - v_0} \\ p &= a(1 - e^2) \\ T &= 2\pi \frac{a^{3/2}}{\sqrt{\mu}}\end{aligned}\quad (8)$$

- аргумент перицентра ω :

$$\begin{aligned}tg \vartheta_0 &= \frac{v_0 \sin \theta_0 \cos \theta_0}{v_0 \cos^2 \theta_0 - 1} \\ \operatorname{sig} n(\sin \vartheta_0) &= \operatorname{sig} g(\sin \theta_0) \\ \operatorname{sig} n(\cos v_0) &= \operatorname{sign} n(v_0 \cos^2 \theta_0 - 1) \\ \cos \theta_0 &= \frac{\sqrt{c_1^2 + c_2^2 + c_3^2}}{r_0 V_0}\end{aligned}\quad (9)$$

$$\begin{aligned}\sin \theta_0 &= \frac{x_0 V_{x0} + y_0 V_{y0} + z_0 V_{z0}}{r_0 V_0} \\ \sin U_0 &= \frac{z_0}{r_0 \sin i}\end{aligned}\quad (10)$$

$$\begin{aligned}\cos u_0 &= \frac{x_0 \cos \tilde{\Omega}_0 + y_0 \sin \tilde{\Omega}_0}{r_0} \\ \omega &= u_0 - \vartheta_0\end{aligned}\quad (11)$$

- момент времени прохождения через перицентр:

$$\begin{aligned}tg \frac{E_0}{2} &= \sqrt{\frac{1 - e}{1 + e}} tg \frac{\vartheta_0}{2} \\ \tau &= t_0 - \frac{a^{3/2}}{\sqrt{\mu}} (E_0 - e \sin E_0)\end{aligned}\quad (12)$$

П2.2 Возмущенное движение космического аппарата

Движение космического аппарата происходит в сложном силовом поле, характеризуемом большим числом сил различной физической природы. Основной силой является ньютоновская сила гравитационного притяжения, соответствующая сферической модели Земли с равномерным по радиусу распределением массы.

Движение под действием этой силы называется невозмущенным движением, характеризуемым постоянством элементов орбиты. При проектировании космических аппаратов, движущихся по низким высотам до 1000 км, необходимо использовать более сложные модели движения, учитывающие дополнительные

силы. К их числу относятся сила, которая появляется в случае использования более сложной, модели Земли - эллипсоида вращения (обычно ограничиваются учетом второй зональной гармоники разложения потенциала поля притяжения), а также аэродинамическая сила, возникающая за счет действия атмосферы Земли. Такое движение космического аппарата называется *возмущенным* движением.

Учет дополнительных сил приводит к появлению периодических и вековых возмущений в движении космического аппарата.

При проектировании космических аппаратов учитывают обычно искомые возмущения, которые нарастают монотонно от витка к витку.

Вековые возмущения вычисляются по приближенной методике за один виток космического аппарата вокруг Земли (на интервале изменения аргумента широты u и от 0 до 2π).

Ф о р м у л и р о в к а з а д а ч и . Для известных начальных значений элементов орбиты определить вековые возмущения элементов орбиты за один и за N витков движения.

Эволюция орбиты, вызванная нецентральностью поля притяжения

Вековые возмущения наклона i , фокального параметра p , эксцентриситета e орбиты равны нулю ($\delta i = \delta p = \delta e = 0$). Вековое возмущение (прецессия) относительно звезд (в абсолютной системе координат) долготы восходящего узла орбиты за 1 виток определяется по соотношению

$$\delta \Omega = -\frac{2\pi}{p^2} \frac{\varepsilon}{\mu} \cos i \quad (13)$$

Под влиянием сжатия Земли восходящий узел орбиты перемещается в сторону, противоположную вращению Земли (к Западу), для прямых орбит ($i < 90^\circ$) и по направлению вращения Земли (к Востоку) для обратных орбит ($i > 90^\circ$). Для полярных орбит ($i = 90^\circ$) прецессия плоскости орбиты отсутствует ($\delta \Omega = 0$).

Вековое возмущение (прецессия) аргумента перигея орбиты в абсолютной системе координат за 1 виток определяется по соотношению

$$\delta \omega = \frac{2\pi}{p^2} \frac{\varepsilon}{\mu} (5 \cos^2 i - 1)$$

Под влиянием сжатия Земли аргумент перигея орбиты при $i < 63^\circ 26'$ и $i > 116^\circ 34'$ аргумент смещается в направлении движения космического аппарата, при $63^\circ 26' < i < 116^\circ 34'$ аргумент перигея орбиты смещается в направлении, противоположном движению космического аппарата. При $i = 63^\circ 26'$ и $i = 116^\circ 34'$ смещение перигея отсутствует ($\delta \omega = 0$).

Вековые возмущения за N витков $\Delta \Omega, \Delta \omega$:

$$\Delta \Omega = N \delta \Omega$$

$$\Delta \omega = N \delta \omega \quad (15)$$

В формулах (13), (14) p - фокальный параметр и e - эксцентриситет орбиты в начальный момент времени.

Эволюция орбиты, вызванная влиянием атмосферы

Вековые возмущения долготы восходящего узла Ω , наклона орбиты i (если не учитывается захват атмосферы вращением Земли), аргумента перигея ω (для случая экспоненциальной модели плотности атмосферы) равны нулю ($\delta \Omega = \delta i = \delta \omega = 0$). Вековые возмущения фокального параметра p и эксцентриситета e за 1 виток определяются по нижеприведенным формулам для модели изотермической атмосферы $p = p_1 \exp\left(-\frac{H-H_1}{H}\right)$:

-эллиптические орбиты с малым начальным эксцентриситетом

$$0 < e < \frac{2\tilde{H}}{a}$$

$$\delta p = -4\pi\sigma\rho_{\text{cp}}p^2\left(1 + \frac{v^2}{8} + \frac{v^4}{192} + \dots\right) \quad (16)$$

$$\delta e = -2\pi\sigma\rho_{\text{cp}}p\left[v\left(1 + \frac{v^2}{8} + \frac{v^4}{192} + \dots\right) + e\left(1 + \frac{3v^2}{8} + \frac{5v^4}{192} + \dots\right)\right]$$

-эллиптические орбиты со средним начальным эксцентриситетом

$$\frac{1.5\tilde{H}}{a} < e < 0.5$$

$$\delta p = -\frac{2\sigma\rho_n p^2}{1-e^2} \sqrt{\frac{2\pi}{v}} (f_0 - 0.5e^2 f_2 - 0.125e^4 f_4 - \dots)$$

$$\delta e = -2\sigma\rho_n p \sqrt{\frac{2\pi}{v}} (f_1 + e f_2 - 0.5e^2 f_4 - \dots)$$

$$f_0 = 1 + \frac{1}{8v} + \frac{9}{128v^2} + \dots \quad (17)$$

$$f_1 = 1 - \frac{3}{8v} - \frac{15}{128v^2} - \dots$$

$$f_2 = 1 - \frac{7}{8v} + \frac{57}{128v^2} - \dots$$

$$f_3 = 1 - \frac{11}{8v} + \frac{225}{128v^2} - \dots$$

$$f_4 = 1 - \frac{15}{8v} + \frac{489}{128v^2} - \dots$$

-эллиптические орбиты с большим начальным эксцентриситетом

$$0.5 < e < 1$$

$$\delta p = -2f_0\sigma p_n \sqrt{\frac{2\pi}{e}} p^3 \tilde{H}$$

$$\delta e = -2f_1\sigma p_n (1+e) \sqrt{\frac{2\pi}{e}} p \tilde{H} \quad (18)$$

Здесь $v = \frac{ae}{\tilde{H}}$,

$\tilde{H} = \frac{R_0 T}{g_0 M}$ - высота однородной атмосферы (равна высоте некоторого фиктивного

столба однородной атмосферы, плотность которой равна ρ_1 , и который имеет на высоте $H = H_1$ { то же давление, что и рассматриваемая атмосфера),

p_n - плотность атмосферы в перигее,

$p_{\text{cp}} = p_n \exp(-v)$ - средняя плотность атмосферы для околокруговой орбиты.

Вековые возмущения фокального параметра p и эксцентриситета e за N витков рассчитываются по формулам.

Под влиянием атмосферы эллиптическая орбита космического аппарата с течением времени все более приближается к круговой.

Период обращения монотонно убывает, а средняя скорость полета возрастает. Максимальная скорость понижения высоты орбиты приходится на район апогея, минимальная - на район перигея орбиты.

Для круговой (или околокруговой $e < 0,02$) орбиты вековые возмущения обычно рассчитываются в полярных координатах.

Формулы расчета изменения траекторных параметров за 1 виток:

-изменение модуля радиуса – вектора

$$\delta r = -4\pi\sigma\rho r^2 \quad (19)$$

-смещение вдоль орбиты

$$\delta r = 12\pi^2\sigma\rho r^2 \quad (20)$$

-изменение периода обращения

$$\delta T = -12\pi^2\sigma\rho\sqrt{\frac{r^5}{\mu}} \quad (21)$$

-изменение радиальной составляющей скорости полета

$$\delta V_r = -2\sigma\rho\sqrt{\mu r} \quad (22)$$

-изменение трансверсальной составляющей скорости полета

$$\delta V_n = 2\pi\sigma\rho\sqrt{\mu r} \quad (23)$$

где $r = R_3 + H$

Формулы расчета изменений траекторных параметров за N витков

$$\Delta l = \frac{N^2}{2} \delta l$$

$$\Delta r = N \delta r$$

$$\Delta V_n = N \delta V_n$$

$$\Delta T = \frac{2N-1}{2} \delta T \text{ (возмущение периода обращения за N витков).}$$

При проектировании космических аппаратов, движущихся по низким орбитам, важнейшей характеристикой является время существования на орбите, т. е. время пассивного движения с момента выхода космического аппарата на орбиту до момента входа в плотные слои атмосферы и прекращения его существования. Время существования можно приближенно рассчитать только для круговых орбит по формуле

$$t_{\text{сущ}} = \frac{F(H_0)}{\sigma}$$

где $F(H_0)$ - функция от начальной высоты полета отыскивается по табл. 4 для соответствующего индекса солнечной активности,

σ - баллистический коэффициент космического аппарата.

Для получения зависимости изменения высоты полета от времени движения $H(\Delta t_i)$ необходимо выбрать шаг расчета по высоте ΔH и воспользоваться формулой (25):

$$\begin{aligned} H_t &= (H \Delta t_i) = H_0 - \Delta H_i \\ \Delta t_i &= t_i - t_0 = \frac{F(H_i) - F(H_0)}{\sigma} \end{aligned} \quad (26)$$

где Δt_i - время снижения с высоты H_0 до высоты H_i :

$\Delta H = \frac{H_K - H_0}{K}$, K - количество шагов, на который разбит рассматриваемый интервал высот ($H_0 \div H_K$).

В качестве конечной высоты, по достижении которой расчет необходимо прекращать, рекомендуется выбирать критическую высоту круговой орбиты H^* (высота круговой орбиты, для которой время существования равно периоду обращения космического аппарата по орбите).

Критическая высота круговой орбиты находится в результате решения уравнения

$$\frac{2\pi(R_3 + H^*)^{3/2}}{\sqrt{\mu}} = \frac{F(H^*)}{\sigma}$$

Для большинства космических аппаратов H^* лежит в диапазоне от 90 км до 110 км.

При проведении расчетов по вышеприведенным формулам рекомендуется использовать табл. 4, в которой в зависимости от высоты полета H приводятся величины функции $F(H)$ для минимального $(65 * 10^{-22} \frac{\text{Вт}}{\text{м}^2 \text{Гц}})$, среднего $(175 * 10^{-22} \frac{\text{Вт}}{\text{м}^2 \text{Гц}})$ и максимального $(275 * 10^{-22} \frac{\text{Вт}}{\text{м}^2 \text{Гц}})$ значений индекса солнечной активности F_0 , высоты однородной атмосферы $\tilde{H}$ модельной плотности воздуха ρ для среднего значения индекса солнечной активности. Эти данные соответствуют Государственному стандарту СССР "Атмосфера Земли верхняя. Модель плотности для проектных баллистических расчетов искусственных спутников Земли" (ГОСТ 25645.101 - 83).

Таблица 1

$H, \text{ км}$	$\tilde{H}, \text{ км}$	$\rho \text{ кг/м}^3$	$F(H), \text{ м}^2 \cdot \text{сут/км}$		
			$F_{0\text{мин}}$	$F_{0\text{ср}}$	$F_{0\text{макс}}$
120	6	2,440-8	3,365-5	3,365-5	3,365-5
130	12	8,357-9	1,255-4	1,221-4	1,192-4
140	16	4,201-9	3,514-4	3,209-4	3,046-4
150	19	2,425-9	8,029-4	6,842-4	6,342-4
160	22	1,514-9	1,612-3	1,285-3	1,167-3
170	25	9,954-10	2,966-3	2,219-3	1,979-3
180	27	6,766-10	5,823-3	3,609-3	3,166-3
190	32	4,916-10	8,695-3	5,585-3	4,774-3
200	34	3,645-10	1,380-2	8,270-3	6,945-3
210	36	2,748-10	2,113-2	1,187-2	9,728-3
220	38	2,102-10	3,659-2	1,659-2	1,333-2
230	40	1,628-10	4,484-2	2,272-2	1,787-2
240	42	1,274-10	6,357-2	3,058-2	2,353-2
250	43	1,007-10	8,885-2	4,056-2	3,050-2
260	45	8,023-10	1,227-1	5,313-2	3,900-2
270	46	6,442-11	1,676-1	6,883-2	4,930-2
280	48	5,209-11	2,269-1	8,829-2	6,168-2
290	49	4,239-11	3,044-1	1,123-1	7,646-2
300	51	3,469-11	4,054-1	1,416-1	9,401-2
310	52	2,854-11	5,361-1	1,774-1	1,147-1
320	53	2,360-11	7,042-1	2,207-1	1,391-1
330	55	1,960-11	9,195-1	2,729-1	1,675-1
340	56	1,635-11	1,194-1	3,356-1	2,007-1
350	57	1,369-11	1,542	4,105-1	2,392-1
360	58	1,151-11	1,981	4,997-1	2,838-1
370	59	9,704-12	2,534	6,056-1	3,351-1
380	60	8,212-12	3,226	7,309-1	3,941-1
390	62	6,970-12	4,091	8,786-1	4,617-1
400	63	5,934-12	5,166	1,052	5,389-1

Продолжение табл. 1

$H, \text{ км}$	$\tilde{H}, \text{ км}$	$\rho \text{ кг/м}^3$	$F(H), \text{ м}^2 \cdot \text{сут/км}$		
			$F_{0\text{мин}}$	$F_{0\text{ср}}$	$F_{0\text{макс}}$
410	64	5,066-12	6,500	1,256	6,270-1
420	65	4,337-12	8,147	1,493	7,271-1
430	66	3,722-12	1,018	1,771	8,406-1
440	67	3,201-12	1,267	2,093	9,692-1
450	68	2,760-12	1,573+1	2,467	1,114
460	69	2,365-12	1,946+1	2,900	1,278
470	70	2,065-12	2,401+1	3,401	1,462
480	71	1,792-12	2,954+1	3,978	1,669
490	72	1,557-12	3,624+1	4,641	1,902
500	73	1,356-12	4,435+1	5,404	2,161
510	74	1,183-12	5,415+1	6,277	2,452
520	75	1,034-12	6,594+1	7,278	2,776
530	76	9,053-13	8,011+1	8,420	3,138
540	76	7,937-12	9,712+1	9,723	3,540
550	77	6,970-13	1,175+2	1,121+1	3,987
560	78	6,129-13	1,418+2	1,289+1	4,483
570	79	5,398-13	1,708+2	1,481+1	5,033
580	80	4,760-13	2,054+2	1,698+1	5,642
590	81	4,204-13	2,464+2	1,944+1	6,316
600	82	3,717-13	2,916+2	2,223+1	7,060
610	96	3,377-13	3,597+2	2,531+1	7,788
620	97	3,044-13	4,329+2	2,880+1	8,666
630	98	2,747-13	5,156+2	3,252+1	9,319
640	99	2,482-13	6,084+2	3,670+1	1,031+11
650	101	2,246-13	7,112+2	4,133+1	1,109+1
660	102	2,035-13	8,249+2	4,647+1	1,222+1
670	103	1,846-13	1,088+3	5,213+1	1,344+1
680	104	1,676-13	1,239+3	5,832+1	1,462+1
690	105	1,524-13	1,404+3	6,517+1	1,606+1
700	107	1,387-13		7,270+1	1,762+1

Продолжение табл. 1

$H, \text{ км}$	$\tilde{H}, \text{ км}$	$\rho, \text{ кг/м}^3$	$F(H), \text{ м}^2 \cdot \text{ сут/км}$		
			$F_{0\text{мин}}$	$F_{0\text{ср}}$	$F_{0\text{макс}}$
710	108	1,583-13	1,583+3	8,084+1	1,933+1
720	109	1,152-13	1,774+3	8,988+1	2,074+1
730	110	1,051-13	1,988+3	9,980+1	2,272+1
740	111	9,605-14	2,215+3	1,106+2	2,486+1
750	112	8,783-14	2,460+3	1,225+2	2,721+1
760	113	8,038-14	2,724+3	1,354+2	2,975+1
770	115	7,363-14	3,008+3	1,495+2	3,255+1
780	116	6,750-14	3,312+3	1,649+2	3,557+1
790	117	6,194-14	3,638+3	1,817+2	3,887+1
800	118	5,687-14	3,987+3	1,999+2	4,245+1
810	119	1,183-12	4,359+3	2,198+2	4,635+1
820	120	1,034-12	4,757+3	2,414+2	5,030+1
830	121	9,053-13	5,181+3	2,648+2	5,489+1
840	122	7,937-12	5,632+3	2,903+2	5,989+1
850	123	6,970-13	6,111+3	3,179+2	6,533+1
860	124	6,129-13	6,621+3	3,478+2	7,124+1
870	125	5,398-13	7,162+3	3,802+2	7,767+1
880	126	4,760-13	7,736+3	4,153+2	8,476+1
890	127	4,204-13	8,344+3	4,533+2	9,237+1
900	128	3,717-13	8,987+3	4,944+2	1,006+2
910	129	3,377-13	9,668+3	5,388+2	1,092+2
920	130	3,044-13	1,039+4	5,867+2	1,189+2
930	131	2,747-13	1,115+4	6,384+2	1,295+2
940	132	2,482-13	1,195+4	6,942+2	1,410+2
950	133	2,246-13	1,279+4	7,543+2	1,535+2
960	134	2,035-13	1,369+4	8,192+2	1,671+2
970	135	1,846-13	1,463+4	8,890+2	1,818+2
980	135	1,676-13	1,561+4	9,641+2	1,978+2
990	136	1,524-13	1,665+4	1,045+3	2,152+2
1000	137	1,387-13	1,775+4	1,132+3	2,341+2

В табл. 1 число со знаком, стоящее после значения параметра, является показателем степени десяти - сомножителя значения параметра, например:

$$1,115 + 3 = 1,115 \cdot 10^{+3}; 2,441 - 11 = 2,441 \cdot 10^{-11}.$$

Оценка времени существования космического аппарата на эллиптической орбите является сложной задачей, которая решается и я путем численного интегрирования уравнений, описывающих потухшее движение космического аппарата.

Для оценочных расчетов $t_{\text{сущ}}$ можно использовать следующие формулы:

$$t_{\text{сущ}} = \begin{cases} \frac{F(H_{\text{ср0}})F_1(v_0)}{\sigma}, & \text{при } e_0 < \frac{\tilde{H}}{a_0} \\ \frac{\Phi(H_{\text{п0}})\psi_1(e_0)F_2(v_0)}{\sigma}, & \text{при } \frac{\tilde{H}}{a_0} < e_0 < e_{\text{кр}} \\ \frac{\Phi(H_{\text{п0}})\psi(e_0)}{\sigma}, & \text{при } e_0 > e_{\text{кр}} \end{cases}$$

$$\text{Здесь } v_0 = \frac{a_0 e_0}{\tilde{H}}$$

$$e_{\text{кр}} \sim 0.66 \sqrt{\frac{\tilde{H}}{R_3 + H_{\text{п0}}}} - 0.22 \frac{\tilde{H}}{R_3 + H_{\text{п0}}}$$

$$\Phi(H_{\text{п0}}) = \frac{R_3 + H_{\text{п0}}}{p(H_{\text{п0}})} \sqrt{\frac{\pi}{2\mu\tilde{H}}}$$

$$\psi_1(e_0) = \frac{e_0^{3/2}}{2(1 - e_0)}$$

$$\psi(e_0) = \frac{e_0^{\frac{3}{2}}}{2} \left(1 + \frac{1}{6}e_0 + \frac{31}{48}e_0^2 + \frac{71}{480}e_0^3 + \dots \right)$$

$$\text{при } v_0 \leq 2 F_1(v_0) = \left(1 + \frac{v_0^2}{8} + \frac{v_0^4}{192} + \dots \right)^{-1}$$

$$F_2(v_0) = 2 \exp(v_0) \left[\sqrt{2\pi v_0^3} \left[1 - \frac{v_0^2}{8} - \frac{v_0^4}{192} + \dots \right] \right]$$

$$\text{при } v_0 \geq 1,5$$

$$F_2(v_0) = \left(1 - \frac{3}{8v_0} - \frac{15}{128v_0^2} - \frac{105}{1024v_0^3} - \dots \right)^{-1}$$

Баллистический коэффициент σ является основным проектным параметром космического аппарата и определяется по формуле:

$$\sigma = \frac{C_x S}{m}$$

где C_x - коэффициент аэродинамического сопротивления, который для большинства современных космических аппаратов лежит в пределах 2-2,5 (обычно принимается 2,2),

m - масса космического аппарата,

S - площадь максимального сечения космического аппарата, перпендикулярная вектору скорости полета (площадь миделевого сечения).

Для ориентированного космического аппарата определение величины S не представляет труда, для неориентированного космического аппарата в форме цилиндра при его беспорядочном вращении

$$S = Ld(0.818^* + 0.25 \frac{L}{d})$$

где L и d - соответственно длина и диаметр цилиндра.

Для космических аппаратов с выпуклой поверхностью при равновероятных положениях осей $S \sim S_{\text{полн}}/4$, где $S_{\text{полн}}$ – площадь всей поверхности.

П2.3 Расчет вспомогательных характеристик движения космического аппарата

Вспомогательные характеристики движения определяют привязку траектории космического аппарата к поверхности Земли, что необходимо для выполнения целевых задач полета и для нормального функционирования наземных служб контроля и управления движением.

Расчет трассы полета космического аппарата

Трасса полета представляет собой геометрическое место точек на поверхности вращающейся Земли, через зенит которых проходит космический аппарат (проекция траектории движения на поверхности Земли).

Формулировка задачи: При известных начальных условиях движения рассчитать географическую широту φ и географическую долготу λ точек проекции положения космического аппарата на земную поверхность в зависимости от времени полета.

Уравнения для расчета трассы космического аппарата с учетом не центральности поля притяжения, но без учета влияния сопротивления атмосферы, имеют вид

$$\varphi = \arcsin(\sin i \sin u) \quad (28)$$

$$\lambda = \Omega - S_0^* + \arctg(\tg u \cos i) + \delta\Omega \frac{t}{T} \quad (29)$$

Здесь u - аргумент широты текущей точки полета, связанный со временем полета t , T - период обращения по орбите, $\delta\Omega$ - вековое смещение долготы восходящего узла орбиты за 1 виток, S_0^* - звездное время на гринвичском меридиане, определяющее положение гринвичского меридиана в текущий момент времени относительно направления на точку весеннего равноденствия.

Если орбита является круговой (или околосферической), то аргумент широты u и текущей точки полета определяется по формуле

$$u = \omega \Delta t + u_0 = u_0 + \sqrt{\frac{\mu}{r^3}} (t - t_0) \quad (30)$$

где r - модуль радиуса-вектора (или средний радиус) круговой орбиты, t_0 - момент времени выхода космического аппарата на орбиту выведения или начала движения по орбите.

Если орбита полета является эллиптической, то аргумент широты определяется по следующим соотношениям:

$$E - e \sin E = \sqrt{\frac{\mu}{a^3}} (t - \tau) = M \quad (31)$$

$$\tg \frac{\vartheta}{2} = \sqrt{\frac{1+e}{1-e}} \tg \frac{E}{2} \quad (32)$$

$$\delta\omega = \pi p^{-3} \epsilon \mu^{-1} (5 \cos^2 i - 1) \quad (33)$$

$$\omega = \omega_0 + \frac{\delta\omega(t-t_0)}{T} \quad (34)$$

$$u = \omega + \vartheta \quad (35)$$

Основной трудностью при расчете μ является решение уравнения Кеплера (31), которое выполняется с использованием численного метода, например, метода Ньютона.

Для орбите эксцентриситетами не превосходящими 0,3 можно использовать решение уравнения (31) в виде ряда

$$E = M + e \sin M + \frac{e^2}{2} \sin 2M + \frac{e^3}{8} (3 \sin 3M - M) + \frac{e^4}{16} (2 \sin 4M - \sin 2M) + \dots$$

При этом ряд быстро сходится и в большинстве случаев можно ограничиться рассмотрением нескольких первых членов. Первый неучитываемый член может быть использован для оценки точности получаемых результатов.

Если трасса привязывается к конкретному моменту времени (т. с. отражает характер прохождения космического аппарата над поверхностью Земли), то она строится в зависимости от текущего момента времени. В курсовой работе такая трасса строится для орбиты выведения.

Текущее звездное время на гринвичском меридиане определяется соотношением

$$S_0^* = S_0 + \omega_3 (t - t_{SO}) \quad (36)$$

где S_0 - звездное время на гринвичском меридиане в некоторую гринвичскую полночь (берется из Астрономического ежегодника), t_{SO} - время наступления полночи.

В разд. 2.1.1.2 рассчитывается долгота восходящего узла орбиты $\tilde{\Omega}$ в специально выбранной инерциальной системе координат, повернутой относительно звездной инерциальной системы координат на угол S^* , где S^* - местное звездное время на меридиане точки старта в момент выхода космического аппарата на орбиту выведения (связь между углами Ω и $\tilde{\Omega}$ иллюстрируется рис. 1). Это сделано, чтобы не привязываться к истинному звездному времени во всех расчетах.

Тогда уравнение (20) для расчета трассы примет вид

$$\lambda = \tilde{\Omega} + \lambda_{ст} + \arctg(tg u \cos i) - \omega_3 \tilde{t} + \delta \Omega \frac{\tilde{t}}{T} \quad (37)$$

Здесь $\tilde{\Omega} + \lambda_{ст} = \Omega - S_0^*$

$\lambda_{ст}$ - географическая долгота точки старта,

$\tilde{t} = t - t_0$ - время, отсчитываемое от момента выведения на орбиту (для простоты можно положить равным нулю).

Географические координаты точки начала трассы определяются при $t = t_0$:

$$\begin{aligned} \varphi_0 &= \arcsin(\sin u_0 \sin i) \\ \lambda_0 &= \lambda_{ст} + \tilde{\Omega} + \arctg(tg u_0 \cos i) \end{aligned} \quad (38)$$

Если расчет трассы носит чисто иллюстративный характер, то трасса может строиться в зависимости от аргумента широты μ . В курсовой работе такая трасса строится для рабочей орбиты, начиная с условной точки $\varphi_0 = 0$, $\lambda_0 = 0$ для момента времени $t_0 = 0$. При этом $u_0 = 0$, прецессией аргумента перигея пренебрегаем.

Расчет проводится по следующим соотношениям:

$$\varphi = \arcsin(\sin i \cdot \sin u)$$

$$\lambda = \arctg(tg \ u \cos i) - \omega_3(t - t_0) + \delta \Omega \frac{(t-t_0)}{T} \quad (39)$$

$$t = t_0 + \frac{a^{3/2}}{\sqrt{\mu}} [(E - e \sin E) - (E_0 - e \sin E_0)]$$

$$tg \frac{E}{2} = \sqrt{\frac{1+e}{1-e}} tg \frac{v}{2}$$

$$tg \frac{E_0}{2} = \sqrt{\frac{1+e}{1-e}} tg \frac{v_0}{2}$$

$$v = u - \omega$$

$$v_0 = u - \omega_0$$

Примечание. Для временных привязок при баллистических расчетах обычно принимается 0-е января 1900 г., при этом $S_0 = 0.2769194$ рад, $t_{so} = 0$.

Для перехода на 0-е января текущего года N , используется соотношение

$$t_0(N_r) = 365(N_r - 1900) + \left[\frac{N_0 - 1}{4} \right].$$

где $\left[\frac{N_0 - 1}{4} \right]$ - целая часть числа.

Для перехода на заданную дату старта ракеты - носителя используется соотношение

$$t_{ст} = t_0(N_r) + d.$$

где d - число дней от 0-го января текущего года до даты старта. Для перехода на момент выхода космического аппарата на орбиту выведения t_0 используется соотношение

$$t_0 = t_{ст} + \Delta t_{ст},$$

где $\Delta t_{ст}$ - продолжительность времени активного участка выведения.

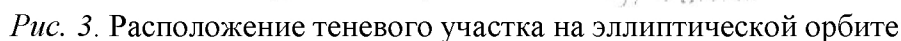
Расчет условий освещенности в орбитальном полете

При проектировании космических аппаратов важным вопросом является определение условий их освещенности. Обычно в качестве источника тока используются бортовые солнечные преобразователи. Кроме того тепловые режимы конструкций и бортовых систем космических аппаратов (например, датчиков системы ориентации) также определяются условиями освещенности. На начальных этапах проектирования достаточно ограничиться приближенным определением условий входа в тень и выхода из неё. В этом случае задача сводится к оценке максимального времени теневого участка.

Формулировка задачи . Для заданных элементов орбиты космического аппарата и угле склонения Солнца δ_c найти угловые координаты точки входа в тень $v_{вх}$ и выхода из неё $v_{вых}$, а также оценить максимальную продолжительность теневого участка.

Тень от Земли принимается цилиндрической (см. рис. 2 для круговой орбиты, рис. 3 для эллиптической орбиты). Максимальное время нахождения космического аппарата в тени соответствует варианту, когда Солнце будет находиться в плоскости орбиты космического аппарата.

Для круговых орбит радиуса r максимальное время нахождения космического аппарата в тени определяется из соотношения (рис. 2)



Для эллиптических орбит (см. рис. 3) необходимо вычислить угловое расстояние Солнца до плоскости экватора в плоскости орбиты δ_c :

где δ_c - склонение Солнца, которое может быть взято из Астрономического ежегодника или определено использованием координат Солнца. Истинная аномалия точки входа в тень определяется в результате решения уравнения

Истинная аномалия точки выхода $u_{\text{вых}}$ из тени определяется в результате решения уравнения

Уравнения (41), (42) целесообразно решать методом последовательных приближений.

Длительность пребывания в тени определяется по соотношению

$$\Delta t_{\text{OCB}} = \sqrt{\frac{a^3}{\mu}} [(E_{\text{BbK}} - e \sin E_{\text{BbK}}) - (E_{\text{BK}} - e \sin E_{\text{BK}})].$$

$$\operatorname{tg} \frac{E_{\text{вых}}}{2} = \sqrt{\frac{1-e}{1+e}} \operatorname{tg} \frac{v_{\text{вых}}}{2}, \operatorname{tg} \frac{E_{\text{вх}}}{2} = \sqrt{\frac{1-e}{1+e}} \operatorname{tg} \frac{v_{\text{вх}}}{2}. \quad (43)$$

Расчет зоны взаимной видимости космического аппарата и наземного пункта

В практике баллистического проектирования космических аппаратов различного назначения одной из первоочередных задач является определение условий прямой видимости космического аппарата как с пункта управления, так и с обслуживаемых наземных объектов.

Формулировка задачи. Для известных элементов орбиты космического аппарата, максимального значения угла $Y_{\min}$ (угла возвышения космического аппарата над наземным пунктом), заданных географических координатах наземного пункта $(\varphi_{\text{нп}}, \lambda_{\text{нп}})$ найти точки входа в зону радио-видимости и выхода из неё, а также продолжительность пребывания в зоне взаимной видимости $\Delta t_{\text{ЗВВ}}$.

Целесообразно расчет зоны взаимной видимости проводить одновременно с расчетом трассы космического аппарата. В первом приближении достаточно принять при записи геометрических условий, что Земля — сфера. Условием наличия взаимной видимости является выполнение неравенства

$$Y \geq Y_{\min}$$

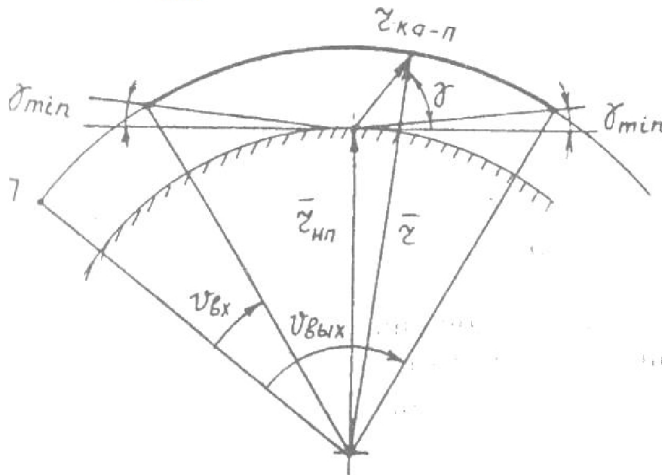


Рис. 4 Иллюстрация определения точек входа и выхода КА на зоны видимости с наземного пункта

Из геометрических условий (рис. 4) следует, что

$$\sin Y = \frac{\bar{r}_{\text{ка-нп}} \bar{r}_{\text{нп}}}{|\bar{r}_{\text{ка-нп}}| |\bar{r}_{\text{нп}}|} \quad (44)$$

где $\bar{r}_{\text{нп}}$ — радиус-вектор положения наземного пункта в выбранной ранее инерциальной системе отсчета. $\bar{r}$ — радиус-вектор текущего положения космического аппарата, $\bar{r}_{\text{ка-нп}} = \bar{r} - \bar{r}_{\text{нп}}$ — радиус-вектор положения космического аппарата относительно наземного пункта.

Уравнение для определения моментов входа в зону видимости и выхода из неё

$$\Phi = \bar{r}_{\text{ка-нп}} \bar{r}_{\text{нп}} - |\bar{r}_{\text{ка-нп}}| R_{\text{З}} \sin Y_{\min} = 0, \quad (45)$$

$$\text{где } \bar{r}_{\text{ка-нп}} \bar{r}_{\text{нп}} = (x - x_{\text{нп}})x_{\text{нп}} + (y - y_{\text{нп}})y_{\text{нп}} + (z - z_{\text{нп}})z_{\text{нп}}, \quad (46)$$

$$\bar{r}_{\text{ка-нп}} = \sqrt{(x - x_{\text{нп}})^2 + (y - y_{\text{нп}})^2 + (z - z_{\text{нп}})^2}. \quad (47)$$

Если $\Phi > 0$, то имеется взаимная видимость, если $\Phi < 0$, то взаимная видимость отсутствует. Изменение функции Φ от отрицательных значений на положительные соответствует восходу космического аппарата над горизонтом по отношению к наземному пункту.

Изменение функции Φ от положительных значений на отрицательное соответствует заходу космического аппарата за горизонт по отношению к наземному пункту.

Здесь x, y, z — проекции радиуса-вектора космического аппарата на оси выбранной инерциальной геоцентрической системы координат (определяются по формулам разд. 2.1.1.1.).

$x_{\text{нп}}, y_{\text{нп}}, z_{\text{нп}}$ — координаты наземного пункта в инерциальной геоцентрической системе координат:

$$\begin{aligned} x_{\text{нп}} &= R_3 \cos \varphi_{\text{нп}} \cos[\lambda_{\text{нп}} - \lambda_{\text{ст}} + \omega_3(t - t_0)] \\ y_{\text{нп}} &= R_3 \cos \varphi_{\text{нп}} \sin[\lambda_{\text{нп}} - \lambda_{\text{ст}} + \omega_3(t - t_0)] \\ z_{\text{нп}} &= R_3 \sin \varphi_{\text{нп}}. \end{aligned} \quad (48)$$

Расчет геометрических характеристик обзора земной поверхности

В качестве геометрических характеристик обзора земной поверхности используются зона обзора и ширина полосы обзора.

Зона обзора представляет собой сферический круг, граница которого видна со спутника под углом обзора аппаратуры спутника $\delta_{\text{КА}}$ или спутник виден из каждой точки границы под углом места Y и определяется центральным углом ζ (рис. 5).

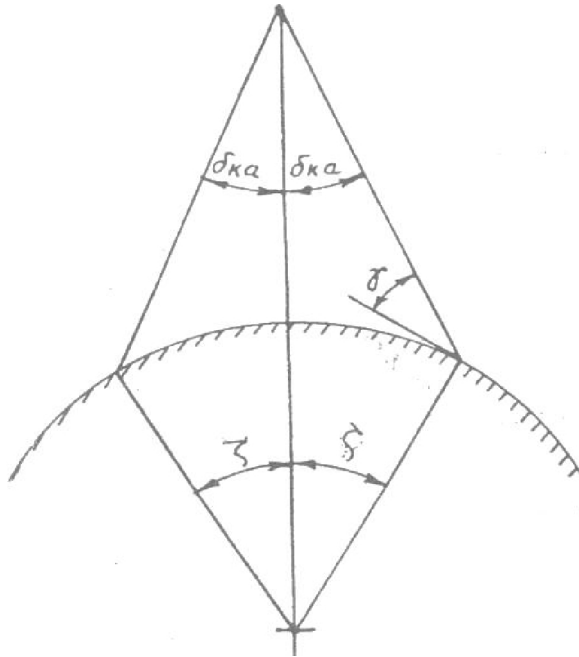


Рис.5 Иллюстрация границ видимости земной поверхности с КА

Полоса обзора на земной поверхности определяется как геометрическое место мгновенных зон обзора на некотором интервале времени. Ширина полосы обзора I — это расстояние между огибающими мгновенных зон обзора в плоскости большого круга.

Формулировка задачи. Зная элементы орбиты и угол обзора бортовой аппаратуры космического аппарата $\delta_{\text{КА}}$, найти величину зоны обзора, ширину полосы обзора и минимальный угол места Y , под которым виден космический аппарат с полосы обзора.

Величина зоны обзора определяется по формуле

$$\zeta = \frac{\pi}{2} - \delta_{\text{КА}} - \arccos\left(\frac{r \sin \delta_{\text{КА}}}{R_3}\right), \quad (49)$$

где r — модуль радиуса вектора, R — средний радиус Земли. Величина угла места

$$Y = \arccos\left(\frac{r \sin \delta_{\text{КА}}}{R_3}\right). \quad (50)$$

Величина ширины полосы обзора

$$l = 2R_3\zeta. \quad (51)$$

П2.4 Рекомендации по выбору рабочей орбиты для космического аппарата

Выбор орбит запускаемых космических аппаратов осуществляется с учетом большого числа факторов. К ним относятся условия старта и прежде всего географические координаты космодромов, возможные азимуты старта, положение на земной поверхности пунктов контроля и наблюдения и т. д.

Но основным фактором, во многом определяющим выбор орбиты, является назначение космического аппарата, устойчивость орбиты по отношению к действующим основным возмущениям. Ниже приводятся рекомендации по выбору рабочей орбиты для космических аппаратов, имеющих специализированное назначение.

1. Спутники связи, целевой задачей которых является ретрансляция информации от передающего пункта, находящегося на поверхности Земли, к пунктам ее приема на поверхности Земли.

При выборе орбиты спутника связи должна быть обеспечена его радиовидимость в течение продолжительного времени. Для этого целесообразно использовать высоко эллиптические орбиты, например, с периодом ~ 12 ч. как для спутников типа "Молния".

Высота перигея выбирается малой, но в то же время обеспечивающей малые аэродинамические возмущения, ~ 500 км.

Высота апогея при этом составляет ~ 40000 км.

Наклонение орбиты целесообразно выбирать равным $\sim 63.4^\circ$, чтобы избежать векового ухода аргумента перигея $\delta\omega = 0$), вызываемого нецентральностью поля тяготения.

Спутник запускается так, чтобы аргумент перицентра его орбиты был равен 270° , при этом перигей орбиты оказывается в каждом полушарии, а апогей - в северном полушарии. То, что апогей находится в северном полушарии и не имеет векового ухода обеспечивает длительное пребывание спутника в северном полушарии. Так, например, при периоде 12 ч. спутник находится в районе апогея в северном полушарии около 11 ч. Положение орбиты в пространстве (долгота восходящего узла) выбирается так, чтобы в районе апогея спутник находился в зоне радиовидимости передающего пункта и приемных пунктов. Большая высота апогея обеспечивает при этом радиовидимость спутника с очень значительной части земной поверхности, включая полярные районы. Продолжительность сеансов между пунктами зоны видимости одного спутника за сутки может достигать 8 ч.

Кратность периода обращения продолжительности суток даёт возможность использовать определенный спутник в конкретное время суток, так как через два витка траектории спутник оказывается над той же точкой поверхности Земли, над которой Находится 24 ч. назад. Если создать систему спутников, то возможность обеспечить постоянную в течение суток связь на значительной территории страны.

В настоящее время для спутников связи также используется геостационарная орбита с элементами $i = 0, e = 0, T = 24$ ч. В этом случае спутник, выведенный в определенную точку орбиты «зависает» над поверхностью Земли. Для создания почти глобальной спутниковой системы связи достаточно поместить три спутника на стационарную орбиту со смещением один относительно другого на 120° , однако с их помощью невозможно реализовать связь с пунктами, расположенными в приполярных областях Земли (широта более 70°).

Возмущающие факторы приводят к сдвигу геостационарного спутника по долготе, а также к изменению наклонения орбиты на 0.9 в год.

Изменение наклона орбиты приводит к появлению трассы спутника в виде восьмерки, размах которой с севера на юг равен наклону орбиты.

Так как геостационарная орбита становится очень «населённой», то возможно в будущем использование орбит с периодами обращения 12; 8 и 6 часов, т. е. кратными суткам.

2. Спутники исследования природных ресурсов Земли, контроля окружающей среды и наблюдения метеорологическими процессами, целевой задачей которых является получение информации об атмосфере и всей поверхности Земли. Так как спутники должны иметь возможность обозревать всю Землю, включая полярные области, то целесообразно выбирать наклон орбиты около 90° . При этом отсутствует смещение долготы восходящего узла ($\delta\Omega = 0$) орбиты, что позволяет создать устойчивую спутниковую систему, требующую минимальных затрат топлива на синхронизацию их движения.

Важным условием является одномасштабность снимков, что упрощает их обработку, увеличивает оперативность анализа. Для выполнения этого требования достаточно сделать орбиту спутника круговой или околокруговой.

Выбор высоты орбиты определяется разрешающей способностью бортовой аппаратуры. В то же время, чем больше высота орбиты, тем больше полоса обзора поверхности Земли.

В последнее время для рассматриваемого типа спутников начали использоваться солнечно-синхронные (гелиостационарные) орбиты. Наклон орбиты этих спутников выбран так, чтобы угловая скорость векового ухода восходящего узла орбиты была равна угловой скорости поворота радиуса-вектора Земли относительно Солнца:

$$2\pi \frac{\varepsilon \cos i}{\mu p^2} \frac{1}{T_\partial} = \frac{2\pi}{T_\odot}, \quad (52)$$

где T_∂ - драконический период обращения спутника (интервал времени между двумя последовательными прохождениями спутником экваториальной плоскости при движении с юга на север), найденный при учете полярного сжатия земного эллипсоида.

Для круговых орбит высотой H

$$T_\partial = \frac{2\pi(R_\oplus + H)^{3/2}}{\sqrt{\mu}} \left[1 + \frac{\varepsilon}{\mu} (R_\oplus + H)^{-2} (1 - 4\cos^2 i) \right]. \quad (53)$$

Наклон солнечно-синхронной круговой орбиты высотой H находится в результате решения уравнения

$$\frac{\varepsilon}{\sqrt{\mu}} \frac{\cos i}{(R_\oplus + H)^3} T_\odot = 2\pi(R_\oplus + H)^{3/2} \left[1 + \frac{\varepsilon}{\mu(R_\oplus + H)^2} (1 - 4\cos^2 i) \right]. \quad (54)$$

Среди множества солнечно-синхронных орбит имеются такие орбиты, у которых трассы орбит последующих суток проходят по трассам предыдущих суток, т. е. имеется суточная повторяемость движения спутника относительно поверхности Земли или геосинхронность.

Для таких солнечно-синхронных геосинхронных орбит выполняется соотношение

$$kT_\partial = T_{\text{сол}},$$

где k - целое число, $T_{\text{сол}}$ — продолжительность солнечных суток.

В табл. 5 приведены элементы солнечно-синхронных геосинхронных орбит с высотами до 36000 км.

Таблица 5

H, км	272	568	895	1264	1684	2166	2727	3391	4191	5180
i, град	97	98	99	101	103	106	110	116	125	142

Изменение высоты солнечно-синхронной орбиты ведет к нарушению геосинхронности: при ее увеличении трассы последующих суток будут смещаться в западном направлении относительно трасс предыдущих суток, при ее уменьшении смещение будет происходить в восточном направлении.

Равенство (55) нарушается и взаимосвязь периода обращения спутника и солнечных суток представляется в виде

$$kT_{\vartheta} = T_{\text{сол}} \left(1 + \frac{\Delta b}{2\pi R_{\text{З}}} \right), \quad (56)$$

где Δb - суточный сдвиг трассы на экваторе (положительное значение соответствует сдвигу в западном направлении, отрицательное значение - в восточном направлении).

Совместное решение уравнений (54), (56) позволяет определить высоту H и наклонение солнечно-синхронной орбиты, обеспечивающей суточный сдвиг трассы на экваторе на величину Δb с целью обеспечения обзора земной поверхности.

Для обеспечения непрерывного обзора земной поверхности должно быть выполнено условие $l \geq 2\Delta b$, при этом полный обзор будет осуществлен за n суток:

$$n = \frac{\pi R_{\text{З}}}{\Delta b}. \quad (57)$$

При выборе высоты орбиты необходимо учесть требуемое качество получения информации, задаваемое предельно допустимым разрешением на местности $\delta R_{\text{зон}}$ в метрах:

$$\delta R_{\text{зон}} = \frac{H}{2R_{\text{зон}}f}, \quad (58)$$

где H - высота орбиты в метрах,

$R_{\text{зон}}$ — разрешающая способность системы «объектив- пленка» в лин/мм (для современной аппаратуры от 50 до 100 л/мм).

f - фокусное расстояние объектива в мм (обычно находится в диапазоне от 80 до 200 мм).

П2.5 Маневрирование космического аппарата

Под маневром понимается управляемое движение космического аппарата, в результате которого происходит изменение орбиты под действием силы тяги реактивного двигателя.

Активные участки полета при баллистическом проектировании космических аппаратов, работающих на химических источниках энергии, аппроксимируются точками приложения мгновенных импульсов скорости, действие тяги двигателя сводится к скачкообразному изменению скорости полета без изменения координат космического аппарата за время работы двигателя.

Орбита, по которой совершает полет космический аппарат до проведения маневра называется начальной орбитой. В результате проведения одного или нескольких маневров космический аппарат переводится на конечную (требуемую) орбиту. В промежутках между маневрами полет происходит по переходным орбитам.

В зависимости от поставленной задачи различаются следующие основные виды маневров:

- маневры перехода с одной орбиты на другую (межорбитальный переход);
- корректирующие маневры;
- маневры сближения на орбите;
- маневры схода с орбиты при возвращении на Землю.

В общем случае расчет маневров космического аппарата сводится к решению краевой задачи.

После определения полных затрат характеристической скорости маневр (сумма импульсов скорости ΔV_{Σ}) определяется потребный запас топлива на его совершение по формуле Циолковского

$$m_T = m_0 \left[1 - \exp \left(-\frac{\Delta V_{\Sigma}}{P_{уд}} \right) \right], \quad (59)$$

где m_0 - начальная масса космического аппарат, $P_{уд}$ - удельный импульс тяги двигателя.

Продолжительность работы двигателей приближенно оценивается по соотношению

$$\Delta t = \frac{m_T}{P} P_{уд}, \quad (60)$$

где P - тяга двигателя космического аппарата.

Расчет маневров перехода между орбитами

В курсовой работе под этими маневрами понимаются маневр перехода с орбиты выведения на рабочую орбиту (например, на геостационарную или высокоэллиптическую), маневр перехода с рабочей орбиты на предпусковую.

Если маневры совершаются в плоскости начальной орбиты, то они называются компланарными, в случае изменения плоскости орбиты в пространственно-некомпланарными.

Формулировка задачи : Выбрать схему маневра, число и величины импульсов скорости, временные характеристики перелета (моменты включения и время работы двигателя, длительность всего перелета), затраты топлива и массу полезной нагрузки, выводимой на конечную орбиту $(\Omega^{(k)}, i^{(k)}, \omega^{(k)}, p^{(k)}, e^{(k)})$ при старте космического аппарата с начальной орбиты $(\Omega^{(0)}, i^{(0)}, \omega^{(0)}, p^{(0)}, e^{(0)})$ и нефиксированном времени перелета.

Компланарные маневры $(i^{(k)} = i^{(0)}, \Omega^{(k)} = \Omega^{(0)})$

Переход с круговой орбиты на эллиптическую или с эллиптической орбиты на круговую орбиту, касающиеся друг друга (рис. 6).

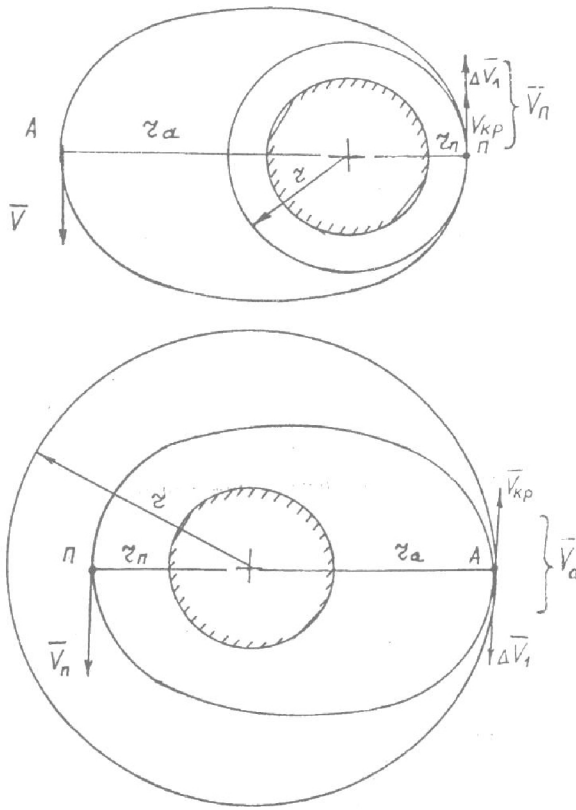


Рис. 6 Схема одноимпульсного перехода между компланарными касающимися орбитами

Маневр осуществляется путем приложения одного импульса в точке касания орбит (т. е. в перигее или апогее эллиптической орбиты). Скорости в перигее и апоцентре эллиптической орбиты определяются по формулам

$$v = \sqrt{\frac{2\mu r_a}{r_a(r_a+r_n)}} V_a = \sqrt{\frac{2\mu r_n}{r_a(r_a+r_n)}},$$

где $r_a = \frac{p}{1-p}$, $r_n = \frac{p}{1+p}$ - радиусы апоцентра и перигея орбиты.

Величина разгонного импульса скорости ΔV_1 перехода с круговой орбиты радиуса r на внешнюю эллиптическую орбиту $r_n = r, r_a > r$ (или тормозного импульса скорости при переходе с внешней эллиптической орбиты на круговую орбиту)

$$\Delta V_1 = V_n - V_{kp} = \sqrt{\frac{\mu}{r_n}} \left(\sqrt{\frac{2r_a}{r_a+r_n}} - 1 \right). \quad (61)$$

Величина тормозного импульса скорости ΔV_1 перехода с круговой орбиты радиуса r на внутреннюю эллиптическую орбиту $r_a = r, r_n < r$ (или разгонного импульса скорости при переходе с внутренней эллиптической орбиты на круговую орбиту):

$$\Delta V_1 = V_{kp} - V_a = \sqrt{\frac{\mu}{r_a}} \left(1 - \sqrt{\frac{2r_n}{r_a+r_n}} \right), \quad (62)$$

Переход с круговой орбиты на гиперболическую орбиту, касающиеся друг друга, при уходе на траекторию межпланетного перелёта (или наоборот с гиперболической орбиты на круговую при возврате с траектории межпланетного перелёта) (рис. 7). Маневр осуществляется путем приложения одного импульса в точке касания орбит (в перигее гиперболической орбиты).

Величина импульса скорости определяется по формуле

$$\Delta V_1 = V_n - V_{kp} = \sqrt{V_{\infty}^2 + \frac{2\mu}{r_n}} - \sqrt{\frac{\mu}{r_g}}, \quad (63)$$

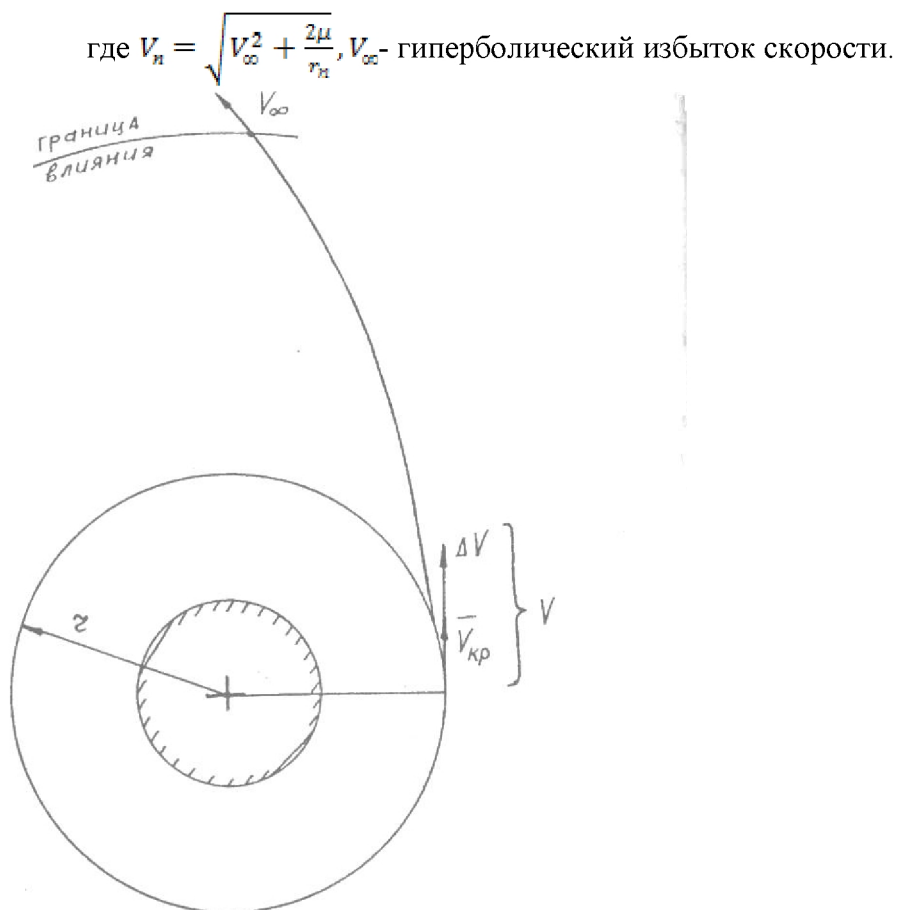


Рис. 7 Схема одноимпульсного перехода на гиперболическую орбиту

Переход между круговыми орбитами. Минимальное количество импульсов скорости для перехода равно двум. Для случая двух импульсного перехода минимальные затраты характеристической скорости достигаются, если в качестве переходной орбиты выбрана эллиптическая орбита, которая в перигее и апогее касается начальной и конечной круговых. Первый импульс скорости дастся для перехода с начальной орбиты на переходную орбиту, второй импульс скорости — для перехода с переходной орбиты на конечную орбиту.

Время перелета определяется по соотношению

$$\Delta t = \frac{\pi}{\sqrt{\mu}} \left(\frac{r^{(a)} + r^{(k)}}{2} \right)^{3/2} \quad (64)$$

На рис. 8 приведена иллюстрация перехода с внутренней орбиты на внешнюю орбиту (например, переход с орбиты выведения на рабочую орбиту).

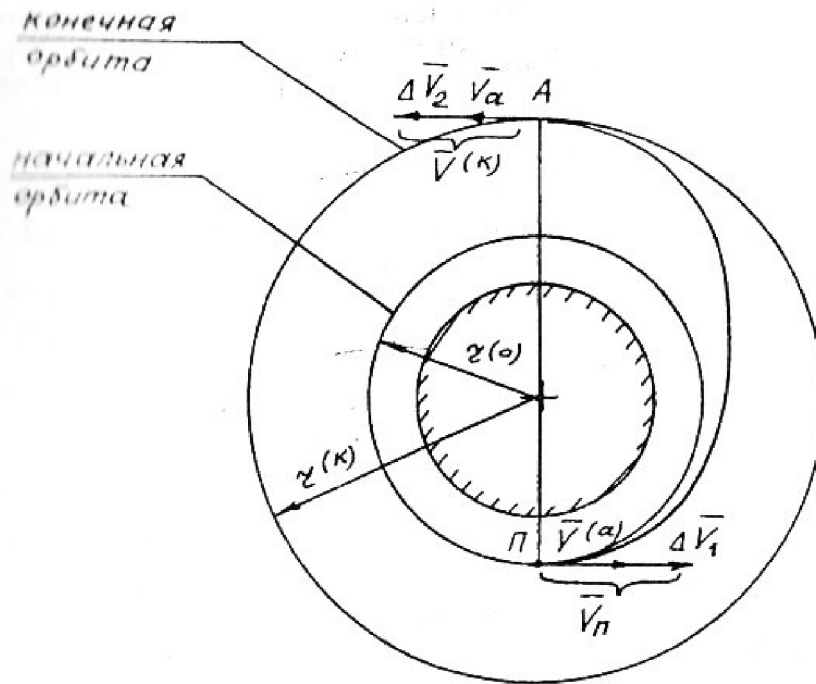


Рис. 8 Схема двухимпульсного перелета между компланарными круговыми орбитами по эллипсу Цандера-Гомана

Оба импульса скорости – разгонные и определяются по формулам

$$\begin{aligned}\Delta V_1 &= \sqrt{\frac{\mu}{r^{(0)}}} \left(\sqrt{\frac{2r^{(k)}}{r^{(0)} + r^{(k)}}} - 1 \right), \\ \Delta V_2 &= \sqrt{\frac{\mu}{r^{(k)}}} \left(1 - \sqrt{\frac{2r^{(0)}}{r^{(0)} + r^{(k)}}} \right).\end{aligned}\quad (65)$$

Аналогично рассчитывается переход с внешней орбиты на внутреннюю орбиту (например, переход с рабочей орбиты на предпусковую орбиту). Оба импульса скорости – тормозные и находятся по соотношениям

$$\begin{aligned}\Delta V_1 &= \sqrt{\frac{\mu}{r^{(0)}}} \left(1 - \sqrt{\frac{2r^{(k)}}{r^{(0)} + r^{(k)}}} \right), \\ \Delta V_2 &= \sqrt{\frac{\mu}{r^{(k)}}} \left(\sqrt{\frac{2r^{(0)}}{r^{(0)} + r^{(k)}}} - 1 \right).\end{aligned}\quad (66)$$

При переходах с внутренней орбиты на внешнюю орбиту может быть использован трехимпульсный баллистический перелет (рис. 9), в котором присутствуют две переходные эллиптические орбиты.

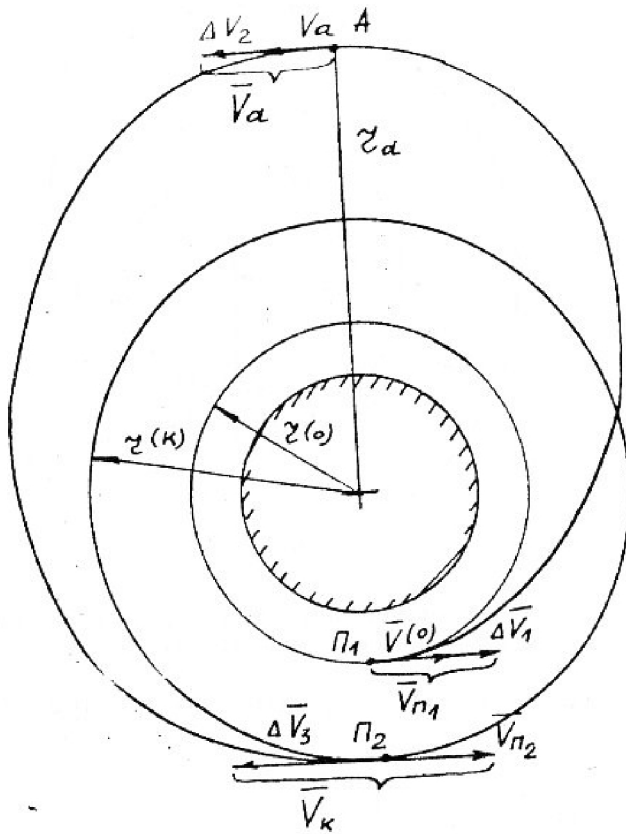


Рис. 9 Схема трехимпульсного биэллиптического перелета между компланарными круговыми орбитами

Первый и второй импульсы в этом случае будут разгонными, а третий – тормозным:

$$\begin{aligned}\Delta V_1 &= \sqrt{\frac{\mu}{r^{(0)}}} \left(\sqrt{\frac{2r_a}{r_a + r^{(0)}}} - 1 \right), \\ \Delta V_2 &= \sqrt{\frac{\mu}{r_a}} \left(\sqrt{\frac{2r^{(k)}}{r^{(k)} + r_a}} - \sqrt{\frac{2r^{(0)}}{r^{(0)} + r_a}} \right), \\ \Delta V_3 &= \sqrt{\frac{\mu}{r^{(k)}}} \left(\sqrt{\frac{2r_a}{r_a + r^{(k)}}} - 1 \right).\end{aligned}\quad (67)$$

Время перелета определяется по соотношению

$$\Delta t = \frac{\pi}{\sqrt{\mu}} \left[\left(\frac{r^{(0)} + r_a}{2} \right)^{3/2} + \left(\frac{r^{(k)} + r_a}{2} \right)^{3/2} \right]. \quad (68)$$

Если время на перелет ограничено $\Delta t \leq \Delta t^*$, то по (68) можно найти предельно допустимое значение r_a^* .

Сравнение двух и трехимпульсных переходов приводит к выводу, что если $r^{(0)}/r_a < r^{(0)}/r_k$, то трехимпульсная программа более экономична, чем оптимальная двухимпульсная программа при $r^{(0)}/r^{(k)} < 0.064$, и менее экономична при $\frac{r^{(0)}}{r^{(k)}} > 0.084$; если $0.064 < r^{(0)}/r^{(k)} < 0.084$, то в зависимости от величины отношения $r^{(0)}/r_a$ экономичным может быть как двухимпульсный, так и трехимпульсный перелет.

Аналогично можно рассмотреть трехимпульсный биэллиптический переход с внешней орбиты на внутреннюю орбиту. В этом случае первый импульс будет разгонным, а второй и третий – тормозными.

Формулы для расчета перелета:

$$\begin{aligned}
\Delta V_1 &= \sqrt{\frac{\mu}{r^{(0)}}} \left(\sqrt{\frac{2r_a}{r^{(0)} + r_a}} - 1 \right), \\
\Delta V_2 &= \sqrt{\frac{\mu}{r_a}} \left(\sqrt{\frac{2r^{(0)}}{r^{(0)} + r_a}} - \sqrt{\frac{2r^{(k)}}{r^{(k)} + r_a}} \right), \\
\Delta V_3 &= \sqrt{\frac{\mu}{r^{(k)}}} \left(\sqrt{\frac{2r_a}{r_a + r^{(k)}}} - 1 \right).
\end{aligned} \tag{69}$$

Переход между околосредними орбитами. Рассматривается случай двухимпульсного перехода. Известно, что энергетически оптимальным является перелет, в котором импульсы скорости прикладываются перпендикулярно текущему радиусу (трансверсально). На рис. 10 показана схема перехода между околосредними орбитами.

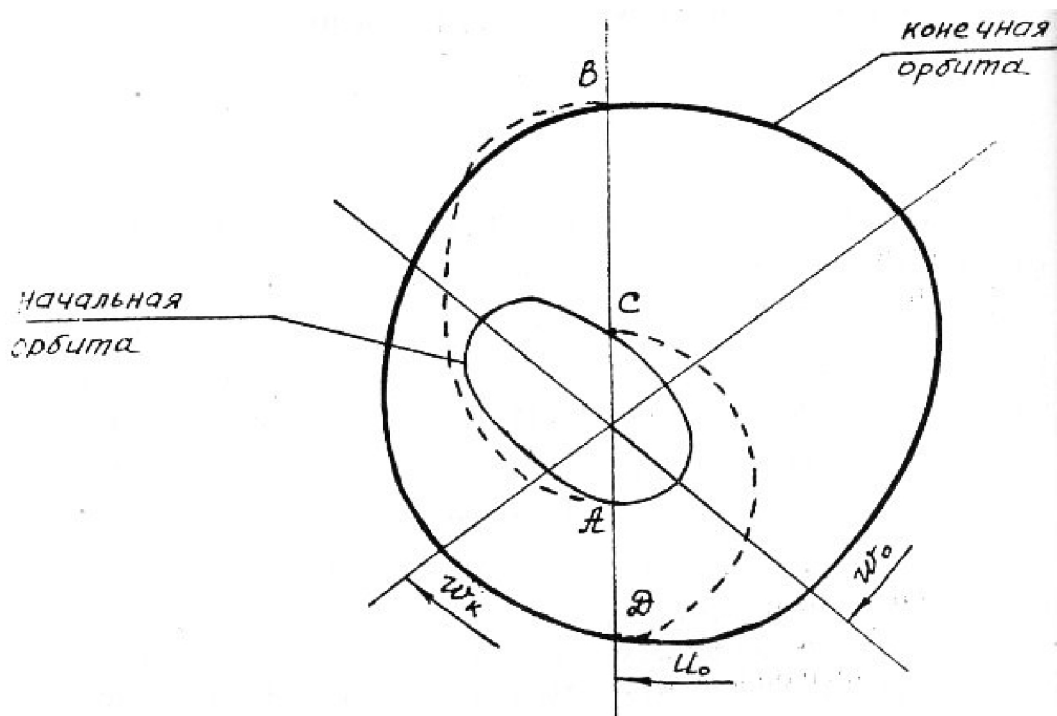


Рис. 10 Схема двухимпульсного перелета между компланарными околосредними орбитами

Точки приложения импульса разнесены вдоль орбиты на угол 180° , и лежат на прямой, проходящей через притягивающий центр (такая прямая называется линией переключения).

Существуют две траектории двухимпульсного перехода с начальной орбиты на конечную, эквивалентные по энергетическим затратам. Для первой траектории (дуга АВ) аргументы широты точек приложения трансверсальных импульсов $u_1 = u_0$, $u_2 = u_0 + \pi$, а величины импульсов определяются по соотношениям

$$\begin{aligned}
\Delta V_1 &= \frac{\omega_{cp}}{4} (A_1 - A_2 \cos u_0 - A_3 \sin u_0), \\
\Delta V_2 &= \frac{\omega_{cp}}{4} (A_1 + A_2 \cos u_0 + A_3 \sin u_0).
\end{aligned} \tag{70}$$

Для второй траектории (дуга СД) аргументы широты тяги приложения трансверсальных импульсов $u_1 = u_0 + \pi$, $u_2 = u_0$, а величины импульсов определяются по соотношениям

$$\Delta V_1 = \frac{\omega_{cp}}{4} (A_1 + A_2 \cos u_0 - A_3 \sin u_0),$$

$$\Delta V_2 = \frac{\omega_{cp}}{4} (A_1 - A_2 \cos u_0 - A_3 \sin u_0). \quad (71)$$

Здесь $\omega_{cp} = \frac{1}{\bar{r}^{(0)}} \sqrt{\frac{\mu}{\bar{r}^{(0)}}}$ - средняя угловая скорость движения по начальной орбите, $\bar{r}^{(0)} = 0.5(r_a^{(0)} + r_n^{(0)})$ - средний радиус начальной орбиты, $u_0 = \arctg \frac{A_3}{A_2}$ - аргумент широты линии переключения

$$A_1 = p^{(k)} - p^{(0)},$$

$$A_2 = p^{(k)} e^{(k)} \cos \omega^{(k)} - p^{(0)} e^{(0)} \cos \omega^{(0)},$$

$$A_3 = p^{(k)} e^{(k)} \sin \omega^{(k)} - p^{(0)} e^{(0)} \sin \omega^{(0)}.$$

Для аппаратов дистанционного зондирования Земли в момент прохождения минимального расстояния от поверхности (в перигее) производится фотографирование. Если в процессе полета требуется производить фотографирование различных участков земной поверхности, то необходимо смещать перигей без изменения ее формы (рис. 11).

Такой двухимпульсный маневр является частным случаем выше рассмотренного маневра и рассчитывается по соотношениям

$$\Delta V_1 = -\frac{\omega_{cp}}{4} (r_a - r_n) \sin \frac{\alpha}{2},$$

$$\Delta V_2 = -\frac{\omega_{cp}}{4} (r_a - r_n) \sin \frac{\alpha}{2},$$

$$u_1 = -\frac{\pi}{2}, u_2 = \frac{\pi}{2}. \quad (72)$$

Здесь α - угол поворота линии апсид, аргументы широты u_1, u_2 отсчитываются от биссектрисы угла α (линия переключения перпендикулярна биссектрисе угла α).

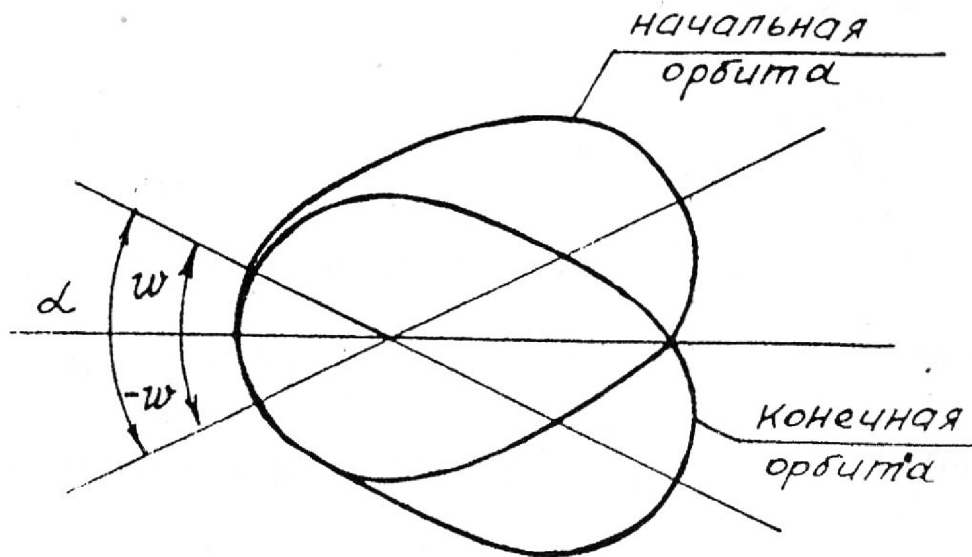


Рис. 11 Иллюстрация к маневру поворота линии апсид без изменения геометрических размеров орбиты

Вышеприведенные формулы дают достаточно точный результат в случае, если средние радиусы начальной и конечной круговых орбит отличаются не более чем на 3000 км.

Некомпланарные маневры ($i^{(k)} \neq i^{(н)}, \Omega^{(k)} = \Omega^{(н)}$)

При выборе схемы некомпланарного маневра необходимо руководствоваться следующим правилом: изменение плоскости орбиты следует проводить в точке, где скорость космического аппарата минимальна.

Переход между некомпланарными круговыми орбитами равного радиуса. Для случая одноимпульсного перехода между круговыми орбитами импульс скорости прикладывается в точке пересечения орбит (рис. 12) и определяется по соотношению

$$\Delta V = 2V_{кр} \sin \frac{\Delta i}{2}, \quad (73)$$

где $\Delta i = |i^{(к)} - i^{(н)}|$, $V_{кр} = \sqrt{\frac{\mu}{r}}$ - скорость движения по орбите.

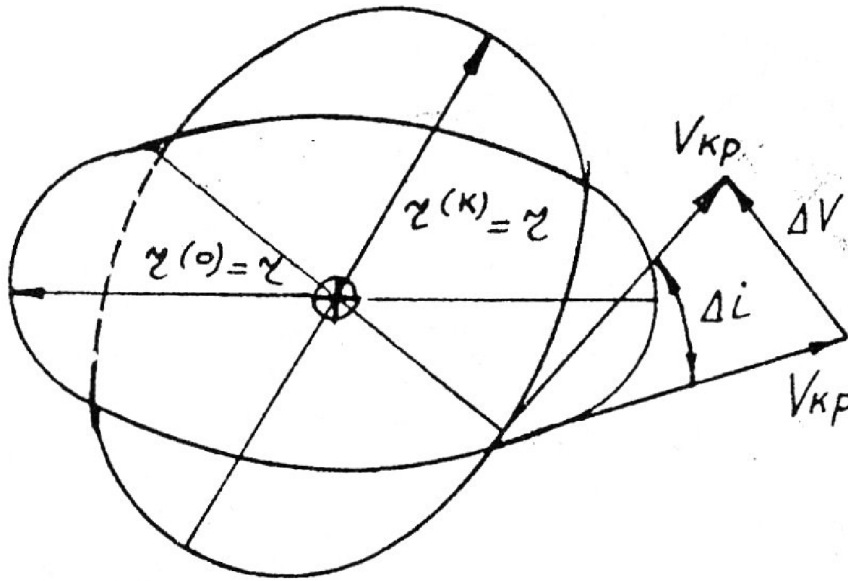


Рис. 12 Схема одноимпульсного поворота плоскости круговой орбиты

Более оптимальным с точки зрения энергетики является трехимпульсный маневр (рис. 13):

- первый импульс ΔV_1 обеспечивает перевод космического аппарата на переходную орбиту, компланарную начальной, с перицентром равным $r^{(0)}$ и апоцентром r_a ;

- второй импульс ΔV_2 прикладывается в апоцентре переходной орбиты и поворачивает ее плоскость на угол некомпланарности Δi без изменения геометрических характеристик орбиты;

- третий импульс ΔV_3 прикладывается в перицентре переходной орбиты, компланарной конечной орбите, и реализует переход на круговую орбиту радиуса $r^{(0)}$

$$\begin{aligned} \Delta V_1 &= \sqrt{\frac{\mu}{r^{(0)}}} \left(\sqrt{\frac{2r_a}{r^{(0)} + r_a}} - 1 \right), \\ \Delta V_2 &= \sqrt{\frac{\mu}{r^{(0)}}} \left(\sqrt{\frac{4r^{(0)}}{r^{(0)} + r_a} (1 - \cos \Delta i)} \right), \\ \Delta V_3 &= \sqrt{\frac{\mu}{r^{(0)}}} \left(\sqrt{\frac{2r_a}{r^{(0)} + r_a}} - 1 \right). \end{aligned} \quad (74)$$

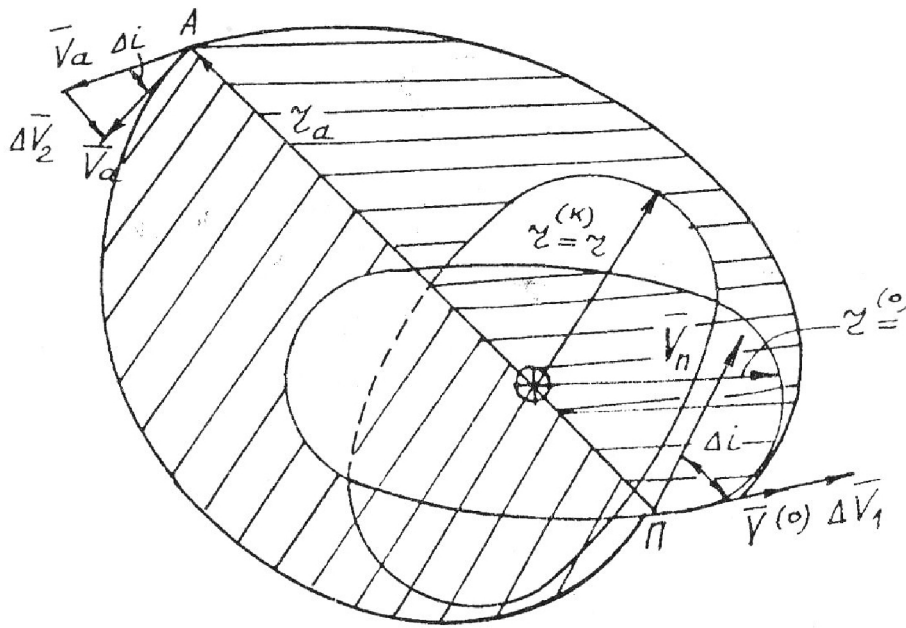


Рис. 13 Схема трехимпульсного поворота плоскости круговой орбиты

Время перелета определяется формулой

$$\Delta t = \frac{2\pi}{\sqrt{\mu}} \left(\frac{r^{(o)} + r_a}{2} \right)^{\frac{3}{2}} \quad (75)$$

Здесь может быть решена однопараметрическая задача оптимизации по r_a с учетом возможного ограничения на допустимую продолжительность перелета из условия минимальных энергетических затрат на переход.

Переход между некомпланарными круговыми орбитами разного радиуса (например, перевод космического аппарата на геостационарную орбиту спутника Земли). Существует несколько схем двухимпульсных переходов с орбиты радиуса $r^{(o)}$ на орбиту радиуса $r^{(k)} > r^{(o)}$, из которых для сравнения выбираются две.

Схема 1 (рис. 14): первый импульс ΔV_1 увеличивает скорость до такой величины, чтобы апоцентр переходной орбиты равнялся $r^{(k)}$, второй импульс скорости прикладывается в апоцентре переходной орбиты и осуществляет поворот плоскости орбиты на угол некомпланарности Δi и доразгон для выхода на конечную орбиту.

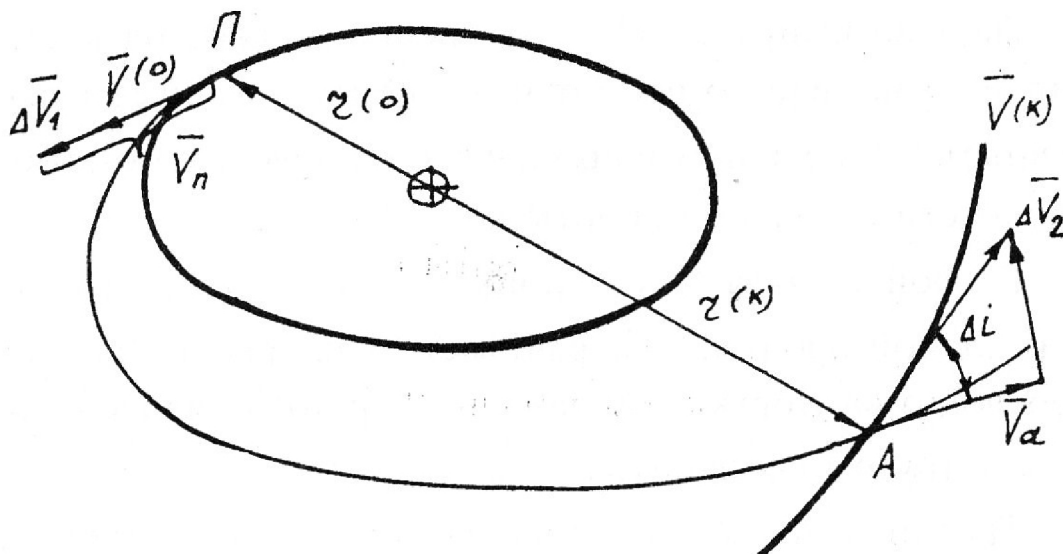


Рис. 14 Схема двухимпульсного перелета между некомпланарными круговыми орбитами разного радиуса

Схема 2: первый импульс ΔV_1 одновременно увеличивает скорость до достижения в апогее переходной орбиты расстояния $r^{(K)}$ и поворачивает плоскость переходной орбиты на угол Δi_1 ($\Delta i_1 < \Delta i$), второй импульс прикладывается в апоцентре переходной орбиты, поворачивает плоскость орбиты на угол $\Delta i_2 = \Delta i - \Delta i_1$ и увеличивает скорость до круговой, соответствующей радиусу орбиты $r^{(K)}$.

Для второго случая импульсы скорости рассчитываются по формулам

$$\begin{aligned}\Delta V_1 &= \sqrt{\frac{\mu}{r^{(0)}}} \sqrt{\frac{r^{(0)} + 3r^{(K)}}{r^{(0)} + r^{(K)}}} - 2\sqrt{\frac{2r^{(K)}}{r^{(0)} + r^{(K)}}} \cos \Delta i_1, \\ \Delta V_2 &= \sqrt{\frac{\mu}{r^{(K)}}} \sqrt{\frac{3r^{(0)} + r^{(K)}}{r^{(0)} + r^{(K)}}} - 2\sqrt{\frac{2r^{(0)}}{r^{(0)} + r^{(K)}}} \cos (\Delta i - \Delta i_1). \quad (76)\end{aligned}$$

Для первого случая импульсы скорости рассчитываются по формулам (76), в которых $\Delta i_1 = 0$. Здесь может быть решена однопараметрическая задача оптимизации по Δi_1 из условия обеспечения минимальных энергетических затрат на переход.

Заслуживает внимания трехимпульсный маневр перехода с тремя поворотами плоскости переходных орбит.

Первый импульс ΔV_1 обеспечивает перевод космического аппарата на первую переходную орбиту с радиусом перицентра равным $r^{(0)}$ и радиусом апоцентра r_a , повернутую относительно плоскости начальной орбиты на угол Δi_1 .

Второй импульс ΔV_2 прикладывается в апоцентре первой переходной орбиты, поворачивает ее плоскость на угол Δi_2 и переводит на вторую компланарную орбиту с радиусом апоцентра r_a и радиусом перицентра $r^{(K)}$.

Третий импульс ΔV_3 прикладывается в перицентре второй переходной орбиты, доворачивает ее плоскость на угол $\Delta i_3 = \Delta i - \Delta i_1 - \Delta i_2$ и переводит на конечную круговую орбиту радиуса $r^{(K)}$.

Формулы для расчета величины импульсов скорости

$$\begin{aligned}\Delta V_1 &= \sqrt{\frac{\mu}{r^{(0)}}} \sqrt{\frac{r^{(0)} + 3r_a}{r^{(0)} + r_a}} - 2\sqrt{\frac{2r_a}{r^{(0)} + r_a}} \cos \Delta i_1, \\ \Delta V_2 &= \sqrt{\frac{\mu}{r_a}} \sqrt{\frac{2r^{(0)}}{r^{(0)} + r_a} + \frac{2r^{(K)}}{r^{(K)} + r_a}} - 4\sqrt{\frac{r^{(K)}}{(r^{(0)} + r_a)(r^{(K)} + r_a)}} \cos \Delta i_2, \\ \Delta V_3 &= \sqrt{\frac{\mu}{r^{(K)}}} \sqrt{\frac{r^{(K)} + 3r_a}{r^{(K)} + r_a}} - 2\sqrt{\frac{2r_a}{r^{(K)} + r_a}} \cos (\Delta i - \Delta i_1 - \Delta i_2). \quad (77)\end{aligned}$$

Время перелета оценивается по формуле

$$\Delta t_{\text{пер}} = \frac{\pi}{\sqrt{\mu}} \left[\left(\frac{r^{(0)} + r_a}{2} \right)^{3/2} + \left(\frac{r^{(K)} + r_a}{2} \right)^{3/2} \right]. \quad (78)$$

Оптимизируя полные затраты скорости, находятся оптимальные значения $r_a, \Delta i_1, \Delta i_2$.

Расчет корректирующих маневров

Корректирующие маневры предназначены для устранения ошибок действительной траектории полета, вызванных действием возмущающих факторов или другими причинами. Коррекция, как правило, подвергаются отдельные элементы орбиты. Величины импульсов скорости при этом относительно невелики, что позволяет использовать линеаризованные модели.

Формулировка задачи: Задана величина, на которую необходимо изменить элемент орбиты или положение космического аппарата в пространстве. Выбрать тип корректирующего импульса, найти точку приложения, оценить минимальную величину импульса скорости и полные затраты характеристической скорости, если известны элементы орбиты космического аппарата до момента проведения коррекции.

Коррекция околокруговых орбит

Расчет коррекции на поддержание заданной высоты полета H^ или движения космического аппарата в заданном диапазоне высот $(H_{max} - \Delta H, H_{max})$.* Методика расчета основывается на соотношениях разд. 1.2.2 и Государственном стандарте СССР "Расчеты баллистические искусственных спутников Земли. Методика расчета затрат топлива на маневрирование" (ГОСТ 25645.301-83).

Учитываются только возмущения, вызванные действием атмосферы (аэродинамическое торможение космического аппарата). Корректирующие импульсы полагаются трансверсальными.

Затраты характеристической скорости на поддержание заданной высоты полета H^* определяются по следующим формулам:

$$- \text{за виток } \Delta V_{\text{хар}}^{(в)} = 2\pi\sigma\rho(H^*)\sqrt{\mu(R_z + H^*)}, \quad (79)$$

$$- \text{за сутки } \Delta V_{\text{хар}}^{(с)} = \frac{\mu\sigma\rho(H^*)}{R_z + H^*} 86400. \quad (80)$$

Поддержание высоты орбиты в заданном диапазоне высот $(H_{max} - \Delta H, H_{max})$ осуществляется с помощью двухимпульсной коррекции (с использованием эллипса Цандера-Гомана), выполняемых циклично по мере падения высоты полета. Величины корректирующих импульсов находятся по соотношениям

$$\Delta V_1 = \sqrt{\frac{\mu}{R_z + H_{max} - \Delta H}} \left(\sqrt{\frac{R_z + H_{max}}{R_z + H_{max} - \frac{\Delta H}{2}}} - 1 \right), \quad (81)$$

$$\Delta V_2 = \sqrt{\frac{\mu}{R_z + H_{max}}} \left(1 - \sqrt{\frac{R_z + H_{max} - \Delta H}{R_z + H_{max} - \frac{\Delta H}{2}}} \right), \quad (82)$$

$$\Delta V_{\text{хар}} = \Delta V_1 + \Delta V_2. \quad (83)$$

Интервал времени между двумя последовательными коррекциями (цикл коррекции) вычисляется по формуле из разд. 2.1.2.2

$$\Delta t = \frac{F(H_{max}) - F(H_{max} - \Delta H)}{\sigma}. \quad (84)$$

Коррекция положения космического аппарата в малой окрестности траектории движения. Пусть $\Delta r, \Delta n, \Delta b$ - отклонения относительно круговой орбиты радиуса r координат космического аппарата вдоль радиуса-вектора, трансверсали и по нормали к плоскости орбиты, вызванные действием аналогично спроектированных компонент корректирующего импульса $\Delta V_r, \Delta V_n, \Delta V_b$:

$$\Delta r = \frac{\sin \tilde{u}}{\omega_{cp}} \Delta V_r + \frac{2(1 - \cos \tilde{u})}{\omega_{cp}} \Delta V_n, \quad (85)$$

$$\Delta n = -\frac{2(1 - \cos \tilde{u})}{\omega_{cp}} \Delta V_r - \frac{3\tilde{u} - 4\sin \tilde{u}}{\omega_{cp}} \Delta V_n, \quad (86)$$

$$\Delta b = \frac{\sin \tilde{u}}{\omega_{cp}} \Delta V_b. \quad (87)$$

Здесь $\omega_{cp} = \sqrt{\frac{\mu}{r_{cp}^3}}$ – средняя угловая скорость движения космического аппарата по орбите, $\tilde{u}$ – угловая дальность полета, отсчитываемая от точки приложения импульса.

Зная требуемые значения $\Delta r, \Delta n, \Delta b$, которые характеризуют требуемое изменение положения космического аппарата на орбите в момент времени t , отсчитываемый от момента проведения коррекции, из соотношений (85) – (87) можно найти необходимые проекции корректирующего импульса.

Коррекция эллиптических орбит

Связь между проекциями корректирующего импульса $\Delta V_r, \Delta V_n, \Delta V_b$ и вызываемыми ими изменениями элементов орбиты определяется следующими формулами:

$$\Delta \rho = \sqrt{\frac{\rho}{\mu}} \frac{2\rho}{1 + e \cos \vartheta} \Delta V_n, \quad (88)$$

$$\Delta e = \sqrt{\frac{\rho}{\mu}} \left(\sin \vartheta \Delta V_r + \frac{e \cos^2 \vartheta + 2 \cos \vartheta + e}{1 + e \cos \vartheta} \Delta V_n \right), \quad (89)$$

$$\Delta \omega = \frac{1}{e} \sqrt{\frac{\rho}{\mu}} \left[-\cos \vartheta \Delta V_r + \frac{\sin \vartheta (2 + e \cos \vartheta)}{1 + e \cos \vartheta} \Delta V_n - \frac{e \operatorname{ctg} i \sin u}{1 + e \cos \vartheta} \Delta V_b \right], \quad (90)$$

$$\Delta \Omega = \sqrt{\frac{\rho}{\mu}} \frac{\cos u}{1 + e \cos \vartheta} \Delta V_b, \quad (91)$$

$$\Delta r \approx \frac{1}{1+e} \Delta \rho - \frac{\rho}{(1+e)^2} \Delta e, \quad (92)$$

Здесь $\rho, e, \omega, \Omega, i$ – элементы орбиты в момент времени приложения импульса, аргумент широты u и угол истинной аномалии относятся к точке орбиты, в которой импульс прикладывается.

На основании соотношений (88) – (93) выбирается тип корректирующего импульса (трансверсальный ΔV_n , радиальный ΔV_r , бинормальный ΔV_b), точка приложения (угол истинной аномалии ϑ или аргумент широты u) и минимальная величина импульса для требуемого изменения элемента орбиты.

Например, для компенсации накопившихся возмущений долготы восходящего угла орбиты $\Delta \Omega$ из соотношения (91) следует, что необходимо выбрать бинормальный импульс, приложить в точке с аргументом широты около 90° или 270° для орбит с малыми значениями эксцентриситета и определить минимальное значение корректирующего импульса по соотношению

$$\Delta V_{bmin} \approx \sqrt{\frac{\mu}{\rho}} \sin i \Delta \Omega. \quad (94)$$

Для компенсации накопившихся возмущений аргумента перицентра ω может быть использован любой из типов корректирующих маневров, например, радиальный ΔV_r , который целесообразно приложить в перигеуме или апоцентре орбиты.

Минимальное значение корректирующего импульса определяется по формуле

$$\Delta V_{rmin} \approx \sqrt{\frac{\mu}{\rho}} e \Delta \omega. \quad (95)$$

П2.7 Расчет маневра сближения на орбите

Маневр сближения предполагает синхронизация движения двух космических аппаратов, один из которых активный (1), совершающий маневрирование, второй –

пассивный (2). Обычно перед этим обеспечивается компланарность орбит обоих космических аппаратов.

Сближение космических аппаратов состоит из двух основных этапов: дальней сближение – рассчитывается по формулам межорбитальных переходов (разд. 1.2.2.1) и сближение на конечном участке – обычно рассчитывается в системе координат, связанной с пассивным космическим аппаратом.

Маневр дальнего сближения

Для того, чтобы в результате совершения маневра два космических аппарата оказались в малой окрестности друг от друга необходимо обеспечить между ними в начальный момент времени (момент времени приложения первого импульса скорости) определенный угол, называемый углом фазирования.

Если рассматривается двухимпульсный перелет, совершаемый по эллипсу Цандера-Гомана между круговыми орбитами (рис. 15), то угол фазирования вычисляется по следующим формулам:

- для случая перехода на внешнюю орбиту

$$v^* = \pi \left[1 - \left(\frac{r^{(1)} + r^{(2)}}{2r^{(2)}} \right)^{3/2} \right], \quad (96)$$

- для случая перехода на внутреннюю орбиту

$$v^* = \pi \left[\left(\frac{r^{(1)} + r^{(2)}}{2r^{(2)}} \right)^{3/2} - 1 \right]. \quad (97)$$

Если в начальный момент времени угловое расстояние между активным (1) и пассивным (2) космическим аппаратом $\tilde{v}$, не равно углу фазирования v^* , то необходимо рассчитать время ожидания угла фазирования (называется временем фазирования), которое определяется по формулам:

- для первого случая (96)

$$T_{\text{ож}} = \frac{r^{(1)3/2}}{\sqrt{\mu} \left[1 - \left(\frac{r^{(1)}}{r^{(2)}} \right)^{3/2} \right]} \Delta v, \quad (98)$$

- для второго случая (97)

$$T_{\text{ож}} = \frac{r^{(1)3/2}}{\sqrt{\mu} \left[\left(\frac{r^{(1)}}{r^{(2)}} \right)^{3/2} - 1 \right]} \Delta v, \quad (99)$$

где

$$\Delta v = \tilde{v} - v^*, 0 \leq \Delta v < 2\pi.$$

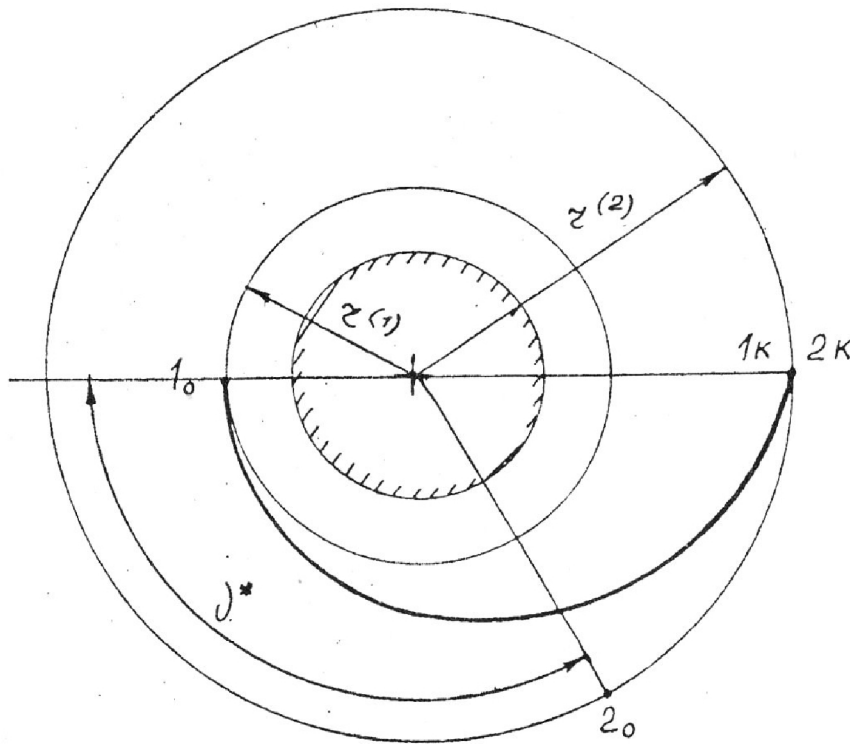


Рис. 15 Схема перелета с фазированием

Чем меньше разница в периодах обращения космических аппаратов, тем больше время фазирования для заданного угла рассогласования. Поэтому при малых разницах в периодах для уменьшения времени фазирования целесообразно использовать биэллиптические перелеты. В этом случае угол фазирования зависит от радиуса апоцентра переходной орбиты r_a и можно подобрать такой радиус апоцентра, чтобы в начальный момент времени $\Delta v = 0$.

В случае биэллиптического перелета на внешнюю орбиту при заданном начальном угловом рассогласовании v между космическими аппаратами радиус апоцентра переходной орбиты, обеспечивающей условие фазирования, находится из уравнения:

$$\left(\frac{r(1)+r_a}{2}\right)^{3/2} + \left(\frac{r(2)+r_a}{2}\right)^{3/2} = \frac{v r_a^{3/2}}{\pi} \quad (100)$$

Маневр сближения на конечном участке

Маневр сближения на конечном участке считается двух импульсным и рассчитывается в рамках метода свободных траектории. Полагается, что оба космических аппарата движутся по близким около круговым орбитам. Движение активного космического аппарата рассматривается в орбитальной системе координат, связанной с пассивным космическим аппаратом (ось $O\tilde{X}$

направлена вдоль орбитальной скорости, ось $O\tilde{Y}$ направлена по геоцентрическому радиусу-вектору, ось $O\tilde{Z}$ дополняет систему координат до правой).

После выполнения этапа дальнего сближения имеются ошибки, приведенные в исходных данных к курсовой работе в орбитальной системе координат $\tilde{x}_0, \tilde{y}_0, \tilde{z}_0, \tilde{v}_{x0}, \tilde{v}_{y0}, \tilde{v}_{z0}$.

Невозмущенное относительное движение приближенно описывается следующими соотношениями:

$$\begin{aligned}
\tilde{x} &= \tilde{x}_0 + 6(\sin(\omega_{cp}\Delta t) - \omega_{cp}\Delta t)\tilde{y}_0 + \left(\frac{4}{\omega_{cp}}\sin(\omega_{cp}\Delta t) - 3\Delta t\right)\tilde{v}_{x0} - \\
&- \frac{2}{\omega_{cp}}(1 - \cos(\omega_{cp}\Delta t))\tilde{v}_{y0} \\
\tilde{y} &= (4 - 3\cos(\omega_{cp}\Delta t))\tilde{y}_0 + \frac{2}{\omega_{cp}}(1 - \cos(\omega_{cp}\Delta t))\tilde{v}_{x0} + \\
&+ \frac{1}{\omega_{cp}}\sin(\omega_{cp}\Delta t)\tilde{v}_{y0} \\
\tilde{z} &= \cos(\omega_{cp}\Delta t)\tilde{z}_0 + \frac{1}{\omega_{cp}}\sin(\omega_{cp}\Delta t)\tilde{v}_{z0} \\
\dot{\tilde{v}}_x &= -6\omega_{cp}(1 - \cos(\omega_{cp}\Delta t))\tilde{y}_0 - 2\sin(\omega_{cp}\Delta t)\tilde{v}_{y0} + 4\cos(\omega_{cp}\Delta t)\tilde{v}_{x0} \\
\dot{\tilde{v}}_y &= 3\omega_{cp}\sin(\omega_{cp}\Delta t)\tilde{y}_0 + 2\sin(\omega_{cp}\Delta t)\tilde{v}_{x0} + \cos(\omega_{cp}\Delta t)\tilde{v}_{y0} \\
\dot{\tilde{v}}_z &= -\omega_{cp}\sin(\omega_{cp}\Delta t)\tilde{z}_0 + 2\cos(\omega_{cp}\Delta t)\tilde{v}_{z0} \quad (101)
\end{aligned}$$

где ω_{cp} — средняя угловая скорость движения орбитальной станции по около круговой орбите, Δt — интервал времени движения космического аппарата от момента завершения этапа дальнего сближения. Полагается, что первый импульс скорости прикладывается сразу после завершения этапа дальнего сближения и обеспечивает совпадение координат обоих космических аппаратов через интервал времени Δt_k (продолжительность этапа ближнего наведения):

$$\tilde{x}(\Delta t_k) = \tilde{y}(\Delta t_k) = \tilde{z}(\Delta t_k) = 0.$$

Проекции первого импульса скорости на осп орбитальной системы координат определяются по соотношениями:

$$\begin{aligned}
\Delta V_{x1} &= \tilde{v}_{x0}^* - \tilde{v}_{x0} \\
\Delta V_{y1} &= \tilde{v}_{y0}^* - \tilde{v}_{y0} \\
\Delta V_{z1} &= \tilde{v}_{z0}^* - \tilde{v}_{z0} \quad (102)
\end{aligned}$$

где $\tilde{v}_{x0}^*$, $\tilde{v}_{y0}^*$ — находятся из решения системы линейных уравнений

$$\begin{cases}
\left(\frac{4}{\omega_{cp}}\sin(\omega_{cp}\Delta t_k) - 3\Delta t_k \right) \tilde{v}_{x0}^* - \frac{2}{\omega_{cp}}(1 - \cos(\omega_{cp}\Delta t_k)) \tilde{v}_{y0}^* = - \\
\quad = -\tilde{x}_0 - 6(\sin(\omega_{cp}\Delta t) - \omega_{cp}\Delta t)\tilde{y}_0, \\
\frac{2}{\omega_{cp}}(1 - \cos(\omega_{cp}\Delta t)) \tilde{v}_{x0}^* + \frac{1}{\omega_{cp}}\sin(\omega_{cp}\Delta t) \tilde{v}_{y0}^* = - \\
\quad = -(4 - 3\cos(\omega_{cp}\Delta t))\tilde{y}_0, \\
\tilde{v}_{z0}^* = -\omega_{cp}\Delta t \tilde{v}_{z0}
\end{cases} \quad (103)$$

Второй импульс в момент Δt_k компенсирует рассогласование по скорости. Проекция второго импульса скорости на оси орбитальной системы координат определяются по соотношениями:

$$\begin{aligned}
\Delta V_{x2} &= -\{-6\omega_{cp}(1 - \cos(\omega_{cp}\Delta t_k))\tilde{y}_0 + (4\cos(\omega_{cp}\Delta t) - 3)\tilde{v}_{x0}^* - \\
&\quad - 2\sin(\omega_{cp}\Delta t)\tilde{v}_{y0}^*\}, \\
\Delta V_{y2} &= -\{-3\omega_{cp}\sin(\omega_{cp}\Delta t)\tilde{y}_0 + 2\sin(\omega_{cp}\Delta t)\tilde{v}_{x0}^* + \cos(\omega_{cp}\Delta t)\tilde{v}_{y0}^*\}, \\
\Delta V_{z2} &= -\{\omega_{cp}\sin(\omega_{cp}\Delta t)\tilde{z}_0 + \cos(\omega_{cp}\Delta t)\tilde{v}_{z0}^*\}. \quad (104)
\end{aligned}$$

Таким образом, полные затраты характеристической скорости на сближение находятся по формуле:

$$\Delta V_{xp} = \Delta V_1 + \Delta V_2 = \sqrt{\Delta V_{x1}^2 + \Delta V_{y1}^2 + \Delta V_{z1}^2} + \sqrt{\Delta V_{x2}^2 + \Delta V_{y2}^2 + \Delta V_{z2}^2}.$$

Предполагая, что маневр сближения на конечном участке производится в пределах одного витка, можно найти оптимальное время маневра $\Delta t_{\text{в}}$ и рассчитать параметры маневра из условия минимальной величины характеристической скорости.

П2.8 Расчет маневра схода с орбиты при возвращении на Землю

Маневр схода с орбиты при возвращении на Землю должен обеспечить как 15 штатном, так и в аварийном варианте спуска не превышение допустимых перегрузок и термодинамического нагрева при полете в плотных слоях атмосферы, что обуславливается в первую очередь значением угла входа в плотные слои $\theta_{\text{вх}}$. Рассматривается случай спуска с круговой орбиты.

Формулировка задачи: Найти оптимальный угол ориентации α и минимальную величину тормозного импульса скорости $\Delta V_{\text{т}}$ из условия обеспечения заданного угла входа в плотные слои атмосферы $\theta_{\text{вх}}^*$.

Угол ориентации тормозного импульса α откладывается от направления, противоположного вектору скорости полета и лежит в диапазоне $0 < \alpha < \frac{\pi}{2}$. Методика расчета основывается на ГОСТ 25645.301 - 83 "Расчеты баллистические искусственных спутников Земли. Методика расчета затрат топлива на маневрирование".

Минимальное значение тормозного импульса определяется по формуле:

$$\Delta V_{\text{т}} = V_{\text{орб}} \left[1 - \sqrt{\frac{2(\tilde{r} - 1)}{(\tilde{r} \sec \theta_{\text{вх}}^*)^2 - 1}} \right]$$

где $V_{\text{орб}} = \sqrt{\frac{\mu}{r^{(0)}}}$ - скорость движения по круговой предпусковой орбите радиуса $r^{(0)}$, $\tilde{r} = \frac{r^{(0)}}{R_{\text{атм}}}$ - радиус условной границы атмосферы ($R_{\text{атм}} = R_{\text{з}} + 100 \text{ км}$).

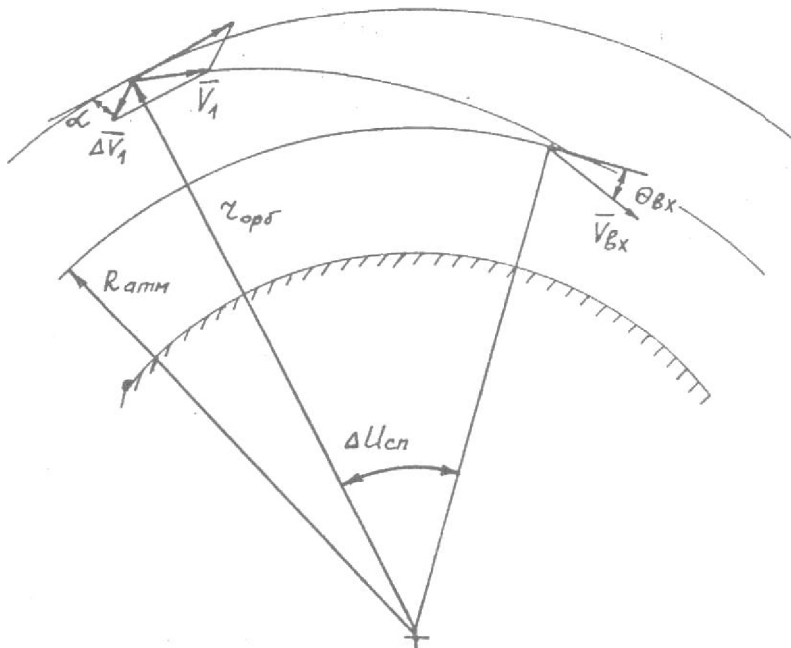


Рис. 16. Маневр схода с предпусковой орбиты

Оптимальный угол ориентации тормозного импульса равен:

$$\alpha_{opt} \begin{cases} 0, \text{ при } |b| > 1; \\ \arccos b, \text{ при } |b| \leq 1; \end{cases} \quad (107)$$

Где

$$b = \frac{\Delta \tilde{V}_\tau^2 + 2(\tilde{r} - 1)}{\Delta \tilde{V}_\tau}, \quad \Delta \tilde{V}_\tau = \frac{\Delta V_\tau}{V_{орб}}.$$

П2.9 Спуск космического аппарата на Землю

Расчет внеатмосферного участка движения

Формулировка задачи: По известным величине ΔV_τ и направлению ориентации α тормозного импульса, радиусу $r^{(0)}$ и наклонению i круговой предпусковой орбиты, географическим координатам (широта φ_τ и долгота λ_τ) точки включения тормозной двигательной установки найти скорость $V_{вх}$ и угол входа $\theta_{вх}$ в плотные слои атмосферы, а также угловую дальность $\Delta u_{сп}$, время полета $\Delta t_{сп}$; до входа в плотные слои атмосферы и географические, координаты входа в плотные слои атмосферы.

Скорость входа и угол входа определяются по соотношениям:

$$V_{вх} = \sqrt{V_{орб}^2 + \Delta V_\tau^2 - 2V_{орб}\Delta V_\tau \cos \alpha} \quad (108)$$

$$\theta_1 = -\arctg [\Delta V_\tau \sin \alpha (V_{орб} - \Delta V_\tau \cos \alpha)^{-1}] \quad (109)$$

Здесь $V_{вх} = \sqrt{V_{орб}^2 + \Delta V_\tau^2 - 2V_{орб}\Delta V_\tau \cos \alpha}$ — скорость космического аппарата после подачи тормозного импульса;

$\theta_1 = -\arctg [\Delta V_\tau \sin \alpha (V_{орб} - \Delta V_\tau \cos \alpha)^{-1}]$ — угол наклона траектории космического аппарата после подачи тормозного импульса;

$$V_{орб} = \sqrt{\frac{\mu}{r^{(0)}}} - \text{скорость на предпусковой круговой орбите радиуса } r^{(0)}.$$

Продолжительность и угловая дальность полета на внеатмосферном участке вычисляются по формулам:

$$\Delta u_{сп} = \arccos \left[\frac{1}{e} \left(1 - \frac{p}{r^{(0)}} \right) \right] - \arccos \left[\frac{1}{e} \left(1 - \frac{p}{R_s + H_{атм}} \right) \right] \quad (110)$$

$$\Delta t_{сп} = \frac{a^{3/2}}{\sqrt{\mu}} [E_{вх} - E_1 - (\sin E_{вх} - \sin E_1)] \quad (111)$$

Здесь p , a , e — фокальный параметр, большая полуось и эксцентриситет орбиты внеатмосферного участка движения:

$$p = \frac{(V_1 r^{(0)} \cos \theta_1)^2}{\mu},$$

$$a = \frac{p}{1 - e^2},$$

$$e = \sqrt{1 + v_1(v_1 - 2) \cos^2 \theta_1}, \quad v_1 = \frac{V_1^2 r^{(0)}}{\mu} \quad (112)$$

$E_1, E_{вх}$ — углы эксцентрической аномалии в точках начала и конца внеатмосферного участка движения, соответствующие углам истинной аномалии $\vartheta_1, \vartheta_{вх}$:

$$\operatorname{tg} \frac{E_1}{2} = \sqrt{\frac{1-e}{1+e}} \operatorname{tg} \frac{\vartheta_1}{2}, \quad \operatorname{tg} \frac{E_{вх}}{2} = \sqrt{\frac{1-e}{1+e}} \operatorname{tg} \frac{\vartheta_{вх}}{2},$$

$$\cos \vartheta_1 = \frac{1}{\varepsilon} \left(1 - \frac{p}{r_{(0)}} \right), \cos \vartheta_{\text{вх}} = \frac{1}{\varepsilon} \left(1 - \frac{p}{R_s + H_{\text{атм}}} \right). \quad (113)$$

Аргумент широты точки включения тормозной двигательной установки и (в предположении его нахождения на восходящем участке траектории) определяется соотношением:

$$u_{\tau} = \arcsin \left(\frac{\sin \varphi_{\tau}}{\sin i} \right). \quad (114)$$

Аргумент широты точки входа в плотные слои атмосферы:

$$u_{\text{вх}} = u_{\tau} + \Delta u_{\text{сп}}, \quad (115)$$

географическая широта, на которой произошел вход в плотные слои атмосферы:

$$\varphi_{\text{вх}} = \arcsin(\sin u_{\text{вх}} \cdot \sin i), \quad (116)$$

географическая долгота точки входа в плотные слои атмосферы:

$$\lambda_{\text{вх}} = \lambda_m + \arctg(\operatorname{tg} \Delta u_{\text{сп}} \cos i) - \omega_3 \Delta t_{\text{сп}} \quad (117)$$

Расчет атмосферного участка движения

Расчет траекторных параметров полета возможен только путем численного интегрирования уравнений математической модели движения. В настоящем разделе приведены приближенные аналитические соотношения, позволяющие сформировать требования к условиям входа в плотные слои атмосферы (в первую очередь к углу входа), исходя из необходимости обеспечить допустимые перегрузки и термодинамические параметры на всей траектории полета.

Приближенно географические координаты точки достижения спускаемым аппаратом поверхности Земли (без учета движения на парашюте) рассчитываются по нижеприведенным соотношениям:

угловая дальность полета на атмосферном участке спуска

$$\Delta U_{\text{атм}} = \frac{L}{R_s}, \quad (118)$$

географическая широта точки достижения поверхности Земли

$$\varphi_{\text{вх}} = \arcsin[\sin(u_{\tau} + \Delta u_{\text{сп}} + \Delta u_{\text{атм}}) \sin i], \quad (119)$$

географическая долгота точки достижения поверхности Земли

$$\lambda_{\text{вх}} = \lambda_{\text{вх}} + \arctg(\operatorname{tg} \Delta u_{\text{атм}} \cos i) - \omega_3 \Delta t_{\text{сп}} \quad (120)$$

Здесь L и $\Delta t_{\text{атм}}$ — дальность и время полета на атмосферном участке спуска, найденные численным интегрированием; $-u_{\tau}, \Delta u_{\text{сп}}, \lambda_{\text{вх}}$ характеризуют внеатмосферный участок полета (см. разд. 2.3.1)

Формулировка задачи: На основании известных условий входа в плотные слои атмосферы (угол $\vartheta_{\text{вх}}$ и скорость $V_{\text{вх}}$) оценить максимальные значения контролируемых характеристик полета - перегрузки, удельного теплового потока ($n_{\text{max}}, Q_{\text{т max}}$) и температуры конструкции в критической точке ($T_{\text{кр}}^{\circ}$).

Случай движения с нулевым аэродинамическим качеством

Этот случай рассматривается для спускаемых аппаратов сферической формы ($K=0$) и для спускаемых аппаратов несферической формы, совершающих аварийное движение, связанное с выходом из строя системы управления движением. При этом спускаемый аппарат закручивается относительно продольной оси и происходит осреднение влияния на траекторию движения возникающей подъемной силы.

Величина максимальной перегрузки n_{max} оценивается по соотношению:

$$n_{\text{max}} = \frac{\beta \sin \vartheta_{\text{вх}}}{2 g_0 \exp(1)} V_{\text{вх}}^2, \quad (121)$$

при этом скорость полета в этот момент составляет:

$$V_{n \text{ max}} \approx 0,605 V_{\text{вх}}. \quad (122)$$

Максимальная величина удельного теплового потока (учитывается только конвективный тепловой поток, обтекание считается ламинарным), подведенного в критической точке, оценивается по формуле:

$$\dot{Q}_{max} \approx \tilde{k} V_{вк}^3 \sqrt{-\frac{\beta \sin \theta_{вк}}{6\sigma \exp(1)}}, \quad (123)$$

при этом скорость полета в этот момент составляет:

$$V_{Q\ max} \approx 0,847 V_{вк}. \quad (124)$$

Здесь $\tilde{k} = c R_{кр}^{4,2} \rho_0^{1,2} V_1^3$, ρ_0 - плотность атмосферы на поверхности Земли, β - параметр модели атмосферы, $R_{кр}$ радиус кривизны носка спускаемого аппарата, V_1 - первая космическая скорость для Земли,
 $c \approx (38 \div 45) \cdot 10^{10} \frac{\text{Дж}}{\text{м}^{3/2} \cdot \text{час}}$

Максимальная температура конструкции в критической точке:

$$T_{кр}^{\circ} = \left(\frac{\dot{Q}_{max}}{\tilde{\sigma} \varepsilon} \right)^{\frac{1}{4}}, \quad (125)$$

где $\tilde{\sigma}$ - постоянная Стефана-Больцмана, ε - коэффициент черноты обшивки спускаемого аппарата (обычно берется 0,9). Исходя из условий:

$$n_{max} \leq n_{доп}, \dot{Q}_{max} \leq \dot{Q}_{доп}, \quad (126)$$

формируются требования на угол входа $\theta_{вк}$ при заданных конструктивных характеристиках спускаемого аппарата.

Случай движения с ненулевым аэродинамическим качеством

Для случая полета с максимальным значением аэродинамического качества K_{max} максимальное значение перегрузки достигается в точке возникновения рикошета:

$$n_{max} \approx \frac{\beta}{2g} V_{вк}^2 \theta_{вк}^2 \sqrt{1 + \frac{1}{K_{max}^2} \exp\left(\frac{2\theta_{вк}}{K_{max}}\right)}. \quad (127)$$

В той же точке достигает максимального значения удельный тепловой поток:

$$\dot{Q}_{max} \approx \tilde{k} V_{вк}^3 \theta_{вк} \left(\frac{\beta}{2\sigma K_{max}} \right)^{\frac{1}{2}} \exp\left(\frac{3\theta_{вк}}{K_{max}}\right).$$

П2.10 Список основных констант

$g_0 = 9,82 \frac{\text{м}}{\text{с}^2}$ — гравитационное ускорение на удалении от центра Земли, равном среднему радиусу;

$\mu = 398600 \frac{\text{км}^3}{\text{с}^2}$ - гравитационный параметр Земли;

$\varepsilon = 2,634 \cdot 10^{10} \frac{\text{км}^5}{\text{с}^2}$ — постоянная сжатия Земли;

$R_3 = 6371 \text{ км}$ - средний радиус сферической модели Земли;

$T_{сл} = 86400 \text{ с}$ — продолжительность солнечных суток;

$T_3 = 3,1536 \cdot 10^8 \text{ с}$ — период обращения Земли вокруг Солнца;

$V_1 = 7,91 \frac{\text{км}}{\text{с}}$ - первая космическая скорость у Земли;

$Q = 5,6704 \cdot 10^{-8} \frac{\text{кДж}}{\text{м}^2 \cdot \text{с} \cdot \text{К}^4}$ — постоянная Стефана-Больцмана;

$\beta = 0,15 \cdot 10^3 \text{ м}^{-1}$ — логарифмический градиент плотности атмосферы;

$H_{атм} = 100 \text{ км}$ — условная граница плотной атмосферы Земли;

$\exp(1) = 2,7183$ — основание натурального логарифма;

$\pi = 3,14159$.

Список рекомендуемой дополнительной литературы

1. Аппазов Р. Ф., Сытин О. Г. Методы проектирования траекторий носителей и спутников Земли. М.: Наука. 1987. 440 с.
2. Механика космического полета: Учебник для вузов /Л/. С. Константиновы др.; Под ред. П. П. Мишина. М.: Машиностроение, 1989. 408 с.
3. Полет космических аппаратов. Примеры и задачи /"Справочник под ред. Г. С. Титова. 2-е изд. переработ, и доп. М.. Машиностроение, 1990. 272 с.
4. Белоконов В. Л/. Динамика полета космических аппаратов: Конспект лекций. Куйбышев. Авиац. ин-т. Куйбышев. 1985. 53 с.
5. ГОСТ 25645.301-83. Расчеты баллистических искусственных спутников Земли. Методика расчета затрат топлива на маневрирование. 1984г.
6. ГОСТ 25645.101-83. Атмосфера Земли. Модель плотности для проектных баллистических расчетов искусственных спутников Земли. 1984 г.