МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**Московский авиационный институт**
**(национальный исследовательский университет)**

Институт № 8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

# ОТЧЁТ

По дисциплине: «Введение в авиационную и ракетно-космическую технику»
На тему: «Доставка марсохода»

Оценка:
Подпись преподавателя:

Выполнили:
Группа М8О-101БВ-24

Литвинова В. А.
Рахматуллин А. Р.
Муханов Г. И.
Гаврилов В. А.

Москва 2024

# СПИСОК ИСПОЛНИТЕЛЕЙ

Группа М8О-101БВ-24

| | | |
|---|---|---|
| Тимлид, математик, составитель документа | _____ | Литвинова Виктория Александровна |
| Главный программист, ksp-инженер | _____ | Рахматуллин Айдар Рамильевич |
| Составитель видеоматериала и презентации | _____ | Муханов Глеб Ильич |
| Главный физик | _____ | Гаврилов Владислав Алексеевич |

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Космические исследования Марса представляют собой одну из самых захватывающих и сложных задач современной науки и техники. Доставка марсохода на поверхность Марса требует глубокого понимания как физических, так и математических аспектов, связанных с космическими полетами. В данной работе будет рассмотрен процесс доставки марсохода на Марс, начиная с теоретических основ, включая физическую и математическую модели, и заканчивая практическим применением этих знаний в симуляторе Kerbal Space Program.

В теоретической части работы будет представлена физическая модель, описывающая основные силы, действующие на космический аппарат во время полета, такие как гравитация, сила аэродинамического сопротивления и управляющая сила. Будут приведены соответствующие физические формулы, позволяющие рассчитать параметры полета, такие как скорость, высота и время. Математическая модель будет включать численные решения этих уравнений, что позволит более точно смоделировать траекторию полета и оптимизировать параметры запуска.

Практическая часть работы будет сосредоточена на моделировании полета ракеты-носителя марсохода в Kerbal Space Program, где будет создан автопилот для автоматизации процесса доставки. В ходе экспериментов будут собраны данные о траекториях, времени полета и других ключевых параметрах, что позволит визуализировать и проанализировать результаты.

В конце работы будет проведено сравнение теоретических и практических данных, что даст возможность оценить точность разработанных моделей и выявить возможные направления для дальнейших улучшений проекта.

# 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1. 1 Реальные миссии

### Миссия Mars Pathfinder

Mars Pathfinder, запущенная NASA в 1996 году, стала одной из первых успешных миссий по доставке марсохода на поверхность Марса. Основной целью миссии было демонстрировать технологии, необходимые для будущих исследований. Марсоход Sojourner, который весил всего 10,6 кг, стал первым мобильным исследователем на Марсе. Он успешно провел 83 дня на поверхности планеты, исследуя состав камней и почвы, а также передавая данные о климате и атмосфере. Pathfinder продемонстрировала эффективность использования малых марсоходов и новых технологий, таких как автоматическая навигация.

### Миссия Mars Exploration Rover

В 2003 году NASA запустила две миссии Mars Exploration Rover: Spirit и Opportunity. Оба марсохода были предназначены для поиска признаков воды и изучения геологии Марса. Spirit приземлился в кратере Гусев, а Opportunity — в районе Меридани Планум. Оба марсохода превзошли все ожидания: Spirit работал до 2010 года, а Opportunity — до 2018 года, что значительно превышало их первоначальный срок службы в 90 дней. Они обнаружили доказательства существования воды в прошлом Марса, а также собрали обширные данные о минералогии и климате планеты.

### Миссия Mars Science Laboratory (Curiosity)

Запущенная в 2011 году, миссия Mars Science Laboratory с марсоходом Curiosity стала одной из самых амбициозных. Curiosity весил около 900 кг и был оснащен сложными научными инструментами для анализа химического состава почвы и атмосферы. Он приземлился в кратере Гейла в 2012 году и продолжает свою работу по сей день. Curiosity подтвердил наличие органических молекул и метана в атмосфере, что открывает новые горизонты для поиска жизни на Марсе. Миссия также предоставляет данные о климате и геологии планеты, что важно для будущих пилотируемых миссий.

### Миссия Mars 2020 (Perseverance)

Запущенная в 2020 году, миссия Mars 2020 с марсоходом Perseverance направлена на поиск признаков древней жизни и сбор образцов для будущей

возвращаемой миссии. Perseverance приземлился в кратере Джезеро в феврале 2021 года. Он оснащен новыми инструментами, включая камеру для записи звука и эксперимент по производству кислорода из углекислого газа в атмосфере Марса. Perseverance также проводит испытания технологии, которая может быть использована для будущих миссий с людьми. Важной частью миссии является сбор образцов, которые будут возвращены на Землю в рамках совместной миссии NASA и ESA.

**Миссия Tianwen-1**

Китайская миссия Tianwen-1, запущенная в 2020 году, стала первой независимой миссией Китая по исследованию Марса. Она включает в себя орбитальный аппарат, посадочный модуль и марсоход Zhurong. Zhurong приземлился на поверхность Марса в мае 2021 года и стал первым китайским марсоходом на планете. Он проводит исследования геологии, климата и поиски воды. Tianwen-1 продемонстрировала растущие возможности Китая в области космических исследований и стала важным шагом в международной космической программе.

**Миссия Emirates Mars Mission (Hope)**

Запущенная Объединенными Арабскими Эмиратами в 2020 году, миссия Emirates Mars Mission с орбитальным аппаратом Hope стала первой арабской миссией на Марс. Она не включает марсоход, но предоставляет важные данные о атмосфере Марса. Hope приземлился на орбиту планеты в феврале 2021 года и начал передавать данные о климате и погодных условиях на Марсе. Миссия направлена на изучение динамики атмосферы и сезонных изменений, что важно для понимания климатических процессов на планете.

Эти миссии продемонстрировали значительный прогресс в области космических исследований и расширили наши знания о Марсе, открывая новые горизонты для будущих исследований и возможных пилотируемых миссий.

## 1. 2 Подробный план полёта

1. Взлёт и выход на орбиту планеты А

1.1. Запуск и начальный разгон

Описание процесса:

Космический аппарат запускается с поверхности планеты, начинает ускоряться для преодоления гравитационного притяжения.

Задачи:

Определить минимальную силу тяги, необходимую для взлёта.

Установить оптимальный угол наклона траектории для эффективного выхода на орбиту, чтобы минимизировать потери топлива из-за сопротивления атмосферы (если есть).

1.1.1 Коррекция траектории:

Постоянный мониторинг азимутального угла $\phi$ для выведения на правильный курс.

Определение момента начала коррекции курса в зависимости от текущей скорости и высоты аппарата.

1.2. Выход на заданную орбиту

Описание процесса:

Аппарат достигает низкой орбиты планеты (около 200-300 км над поверхностью).

Задачи:

Вычислить скорость, необходимую для поддержания орбиты на данной высоте.

Оценить потери топлива на этом этапе, учитывая изменение массы аппарата (стадия первой ступени).

1.3. Отделение первых ступеней

Описание процесса:

После достижения орбитальной скорости происходит отделение первых ступеней аппарата.

Задачи:

Оценить изменение массы и скорости аппарата после отделения первой ступени.

Рассчитать погрешности, связанные с изменением массы и центром масс аппарата.

2. Межпланетный перелёт

2.1. Манёвр перехода с орбиты планеты A

Описание процесса:

Аппарат выполняет манёвр Хоумана для выхода с орбиты планеты A на эллиптическую траекторию, ведущую к планете B.

Задачи:

Определить момент выполнения манёвра для минимизации затрат топлива.

Рассчитать изменение скорости (импульс), необходимое для перехода на траекторию межпланетного перелёта.

Учёт внешних факторов:

Оценить влияние гравитационных возмущений со стороны других планет и Солнца.

Вычислить влияние солнечного ветра и других космических факторов на траекторию.

2.2. Коррекция траектории в межпланетном пространстве

Описание процесса:

На пути к планете B космический аппарат может потребовать выполнения корректирующих манёвров.

Задачи:

Осуществлять регулярный мониторинг текущих координат аппарата в сферической системе координат.

Определить моменты выполнения корректирующих импульсов для точного подлёта к планете B.

2.2.1 Оценка погрешностей:

Выявить основные источники погрешностей при расчётах траектории.

Разработать методики для минимизации отклонений траектории.

3. Подлёт и торможение у планеты B

3.1. Вход в гравитационное поле планеты B

Описание процесса:

Космический аппарат входит в сферу гравитационного влияния планеты B.

Задачи:

Вычислить параметры траектории подлёта с учётом гравитации планеты B.

Рассчитать угол вхождения в гравитационное поле для минимизации необходимого торможения.

3.1.1 Коррекция траектории:

Вычислить необходимые корректирующие манёвры для выхода на орбиту планеты.

Учесть возможные гравитационные манёвры с использованием спутников или луны планеты B (если таковые имеются).

3.2. Торможение и выход на орбиту

Описание процесса:

Аппарат замедляется до орбитальной скорости планеты B для выхода на её орбиту.

Задачи:

Определить импульс для торможения и его продолжительность.

3.2.1 Оценка погрешностей:

Определить факторы, влияющие на изменение скорости при торможении (погрешности датчиков, влияние атмосферы).

Минимизировать ошибки в оценке скорости и траектории.

4. Посадка на планету $B$

4.1. Подготовка к посадке

Описание процесса:

Перед началом спуска космический аппарат стабилизируется на орбите, корректируется угол захода на посадку.

Задачи:

Определить оптимальную точку начала спуска для мягкой посадки.

Установить последовательность действий для торможения в атмосфере.

4.1.1 Оценка потенциальных рисков:

Оценить погодные и атмосферные условия и влияние на посадку.

Учитывать возможные отклонения в траектории из-за плотности атмосферы или рельефа поверхности.

4.2. Спуск и торможение

Описание процесса:

Аппарат начинает снижение, используя парашюты для торможения.

Задачи:

Вычислить параметры торможения.

Осуществить мониторинг угла наклона, азимутального и полярного углов, чтобы удерживать заданную траекторию спуска.

4.2.1 Коррекция в процессе спуска:

В случае отклонений корректировать траекторию с помощью двигателей малой тяги.

Определить момент активации посадочных систем амортизаторов.

4.3. Посадка и стабилизация

Описание процесса:

Завершающая фаза спуска, плавное касание поверхности и стабилизация аппарата.

Задачи:

Оценить параметры посадки (скорость касания, угол наклона).

Произвести автоматическую стабилизацию аппарата на поверхности.

4.3.1 Анализ погрешностей:

Определить точность выполнения всех манёвров на финальном этапе.

Оценить влияние погрешностей измерений на итоговую точность посадки.

5. Итоговый анализ полёта

5.1. Анализ каждого этапа

Провести анализ данных по каждому этапу полёта: взлёт, межпланетный перелёт, подлёт к планете, посадка.

Выявить отклонения от запланированных параметров и причины ошибок.

5.2. Рекомендации для улучшения

Разработать меры по снижению погрешностей на всех этапах будущих миссий.

Оптимизировать процесс коррекции траектории для уменьшения расхода топлива и повышения точности посадки.

## 1. 3 Физическая модель

Рассмотрим формулы, необходимые для дальнейших вычислений в математической модели.

Закон сохранения импульса:

$$\sum_{i=0}^{n} m_i \cdot V_i = const,$$

$m_i$ – масса i-й точечной массы тела (системы тел)

$V_i$ – относительная скорость i-й точечной массы тела

n – количество точектела

Скорость движения ракеты из закона сохранения импульса для фиксированного момента реактивного движения:

$$V_p = \frac{m_г}{m_p} \cdot V_г,$$

$V_p$ — скорость ракеты

$V_г$ — скорость выхлопа газов (реактивного движения)

$m_p$ — масса ракеты

$m_г$ — масса газов

Формула Циолковского:

$$V_p = \omega \cdot \ln\left(\frac{m_p + m_г}{m_p}\right),$$

$V_p$ — скорость, которую ракета приобретает по истечению всех газов из сопла реактивного двигателя

ω – скорость истечения газов из двигателя ракеты

$m_p$ – масса конструкции ракеты

$m_г$ – масса газов, отброшенных ракетой

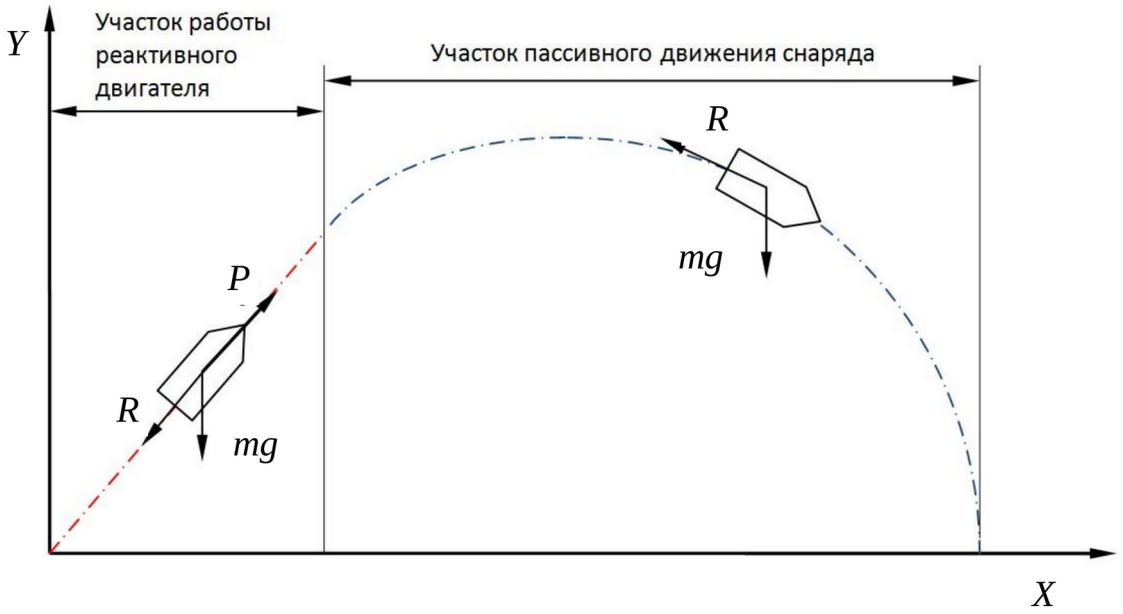



*Рис. 1 — Траэктория движения реактивного снаряда*

На Рисунке 1 графически изображена траэктория движения реактивного снаряда. Здесь:

mg — вес ракеты

P — сила тяги от реактивного двигателя

R — сила атмосферного сопротивления

Характеристическая скорость ракеты:

$$V_x = \omega \cdot \ln\left(\frac{m_0}{m_0 - m_r}\right) + V_0,$$

$V_0$ – скорость ракеты в момент запуска реактивного двигателя (обычно для ракеты космического назначения $V_0 = 0$ м/с)

ω – скорость истечения продуктов горения из сопла реактивного двигателя.

Реальная конечная скорость:

$$V_K = V_X - \Delta V$$

где $\Delta V_X$– потери характеристической скорости ракеты (к примеру, из-за сопротивления среды)

Масса топлива для вывода на орбиту:

$$m_r = \frac{\left(e^{\frac{V_x}{\omega}} - 1\right) \cdot (s-1)}{s - e^{\frac{V_x}{\omega}}} \cdot m_{ПН}$$

$$s = \frac{m_T}{m_K} + 1$$

$m_{ПН}$ – масса полезного груза, которую требуется вывести на орбиту

$V_X$ – скорость, которую должен приобрести полезный груз на орбите

ω – скорость истечения продуктов горения определяется характеристиками реактивного двигателя

s – конструктивная характеристика ракетного блока

В таблице 1 приведены некоторые абсолютные массовые характеристики многоступенчатой ракеты.

*Таблица 1. Абсолютные массовые характеристики многоступенчатой ракеты.*

| **Наименование характеристики** | **Выражение** | **Примечания** |
|---|---|---|
| Масса *i*-й ступени | $m_i = m_{ПН}^i + m_{РБ}^i$ | $m_{ПН}^i$ – масса полезной нагрузки *i*-й ступени; $m_{РБ}^i$ – масса ракетного блока *i*-й ступени |
| Масса полезной нагрузки *i*-й ступени | $m_{ПН}^i = m_{i+1}$ | $m_{i+1}$ – масса следующей ступени |
| Масса ракетного блока *i*-й ступени ракеты | $m_{РБ}^i = m_T^i + m_K^i$ | $m_T^i$ – масса топлива ракетного блока *i*-й ступени; $m_K^i$ – масса конструкции ракетного блока *i*-й ступени |
| Стартовая масса ракеты | $m_0 = m_I$ | $m_I$ – масса первой ступени |

В таблице 2 некоторые относительные массовые характеристики многоступенчатой ракеты.

*Таблица 2. Относительные массовые характеристики многоступенчатой ракеты.*

| **Наименование характеристики** | **Выражение** | **Примечания** |
|---|---|---|
| Число Циолковского *i*-й ступени ракеты | $z_i = \frac{m_i}{m_i - m_T^i}$ | где $m_i$ – масса *i*-й ступени; $m_T^i$ – масса топлива ракетного блока *i*-й ступени |
| Конструктивная характеристика ракетного блока *i*-й ступени ракеты | $s_i = \frac{m_{РБ}^i}{m_K^i} = \frac{m_{РБ}^i}{m_{РБ}^i - m_T^i}$ | где $m_{РБ}^i$ – масса ракетного блока *i*-й ступени; $m_K^i$ – масса конструкции ракетного блока *i*-й ступени; $m_T^i$ – масса топлива ракетного блока *i*-й ступени |
| Относительная масса *i*-й ступени ракеты | $p_i = \frac{m_i}{m_{ПН}^i}$ | где $m_{ПН}^i$ – масса полезной нагрузки *i*-й ступени |

Формула Циолковского для многоступенчатой ракеты:

$$V_{\Sigma X} = V^{-1}x + \ldots + V^N x = \sum_{i=1}^{N} V^i x = \sum_{i=1}^{N} \omega_i \cdot \ln z_i,$$

$$V_X^i = \omega_i \cdot \ln\left(\frac{m_i}{m_i - m_r^i}\right),$$

$V_{\Sigma X}$ – общая характеристическая скорость ракеты

$V^i_X$– идеальная характеристическая скорость i-й ступени

$\omega_i$ – скорость истечения продуктов сгорания топлива реактивного двигателя i-й ступени

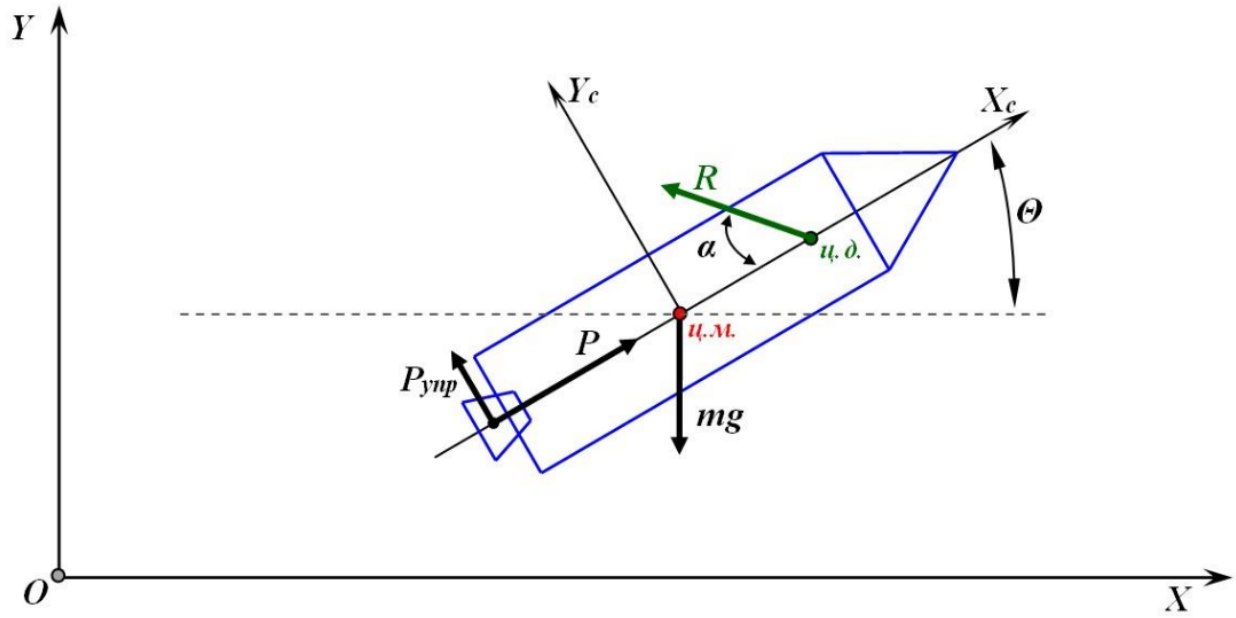



*Рис. 2 — Внешние силы, действующие на ракету в полёте*

На Рисунке 2 графически показаны внешние силы, действующие на ракету в полёте. Здесь:

$X_c$, $Y_c$ — СК связанная с ракетой

P – сила тяги от реактивного двигателя

$P_{упр}$ – реактивная управляющая сила

mg – сила тяжести

R – сила аэродинамического сопротивления

α – угол атаки воздушного потока

θ – угол между осью Ox и $X_c$

Сила тяги:

$$P = P_0 + S_\alpha \cdot p_{H0}[1 - \pi(y)],$$

$P_0$ – тяга на уровне моря или стендовое значение тяги

$S_\alpha$ – площадь выходного сечения сопла

$\pi(y) = p_0 / p_{H0}$ – функция высоты, представляющая отношение атмосферного давления на произвольной высоте $p_H$ к давлению на уровне моря $p_{H0}$

Расход топлива ракеты при полёте:

$$m(t)=m_0-mt$$

$$m=\frac{P}{I}$$

$m_0$ — начальная масса ракеты

m — секундный расход топлива

P – сила тяги

I – удельный импульс ракетного двигателя

t – текущее время полета ракеты.

Сила аэродинамического сопротивления R.

Уравнение Бернулли:

$$\frac{\rho V^2}{2}+p=const$$

ρ — плотность воздуха

p — давление

Считаем, что газ несжимаем, значит температура постоянна.

$$R=\frac{\rho V^2}{2}\cdot S,$$

$$\rho_2=\rho_1=\rho,\ V_1=V,\ V_2=0,\ R=c\cdot q\cdot S,$$

$\rho_2$ — встречное сопротивление

S — коэффициент, учитывающий отклонение эксперементальной аэродинамической силы от теоретической (обычно 1.11)

$q=\frac{\rho V^2}{2}$ - скоростной набор

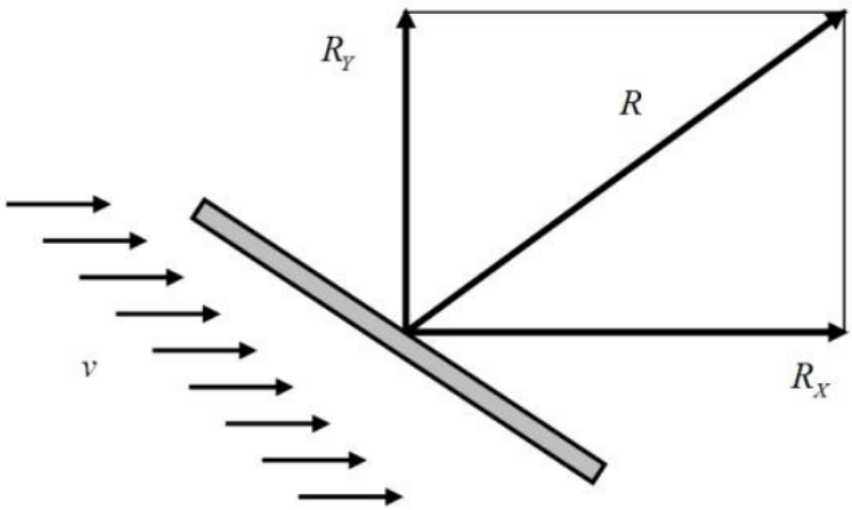



*Рис. 3 — Схема обтекания наклонной пластинки воздушным потоком*

На Рисунке 3 изображена схема обтекания наклонной пластинки воздушным потоком. Здесь:

$R_x = c_x \cdot \frac{pV^2}{2} \cdot S$, $R_y = c_y \cdot \frac{pV^2}{2} \cdot S$ —

величины силы лобового сопротивления и подъёмной силы, где $c_x$ и $c_y$ — их коэффициенты — зависят от угла.

Проанализируем влияние аэродинамических сил на ракету на примитивном примере (модель схожа со многими реальными ракетами). На Рисунке 4 приведено схематически влияние подъёмной силы и силы лобового сопротивления на ракету

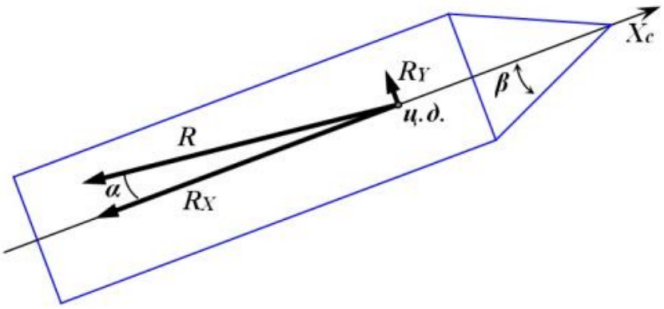



*Рис. 4 — Влияние подъёмной силы и силы лобового сопротивления на ракету*

$C_y = 3\alpha$, $C_x = 2\beta^2$,

β — угол полураствора конуса

α — угол атаки

S — площадь миделя

При проэктировании стремимся уменьшить влияние $C_x$ и $C_y$.

$R_x \sim \frac{1}{n}$, $R_y \sim \frac{1}{n}$, т. к. $\beta \sim \frac{1}{n}$

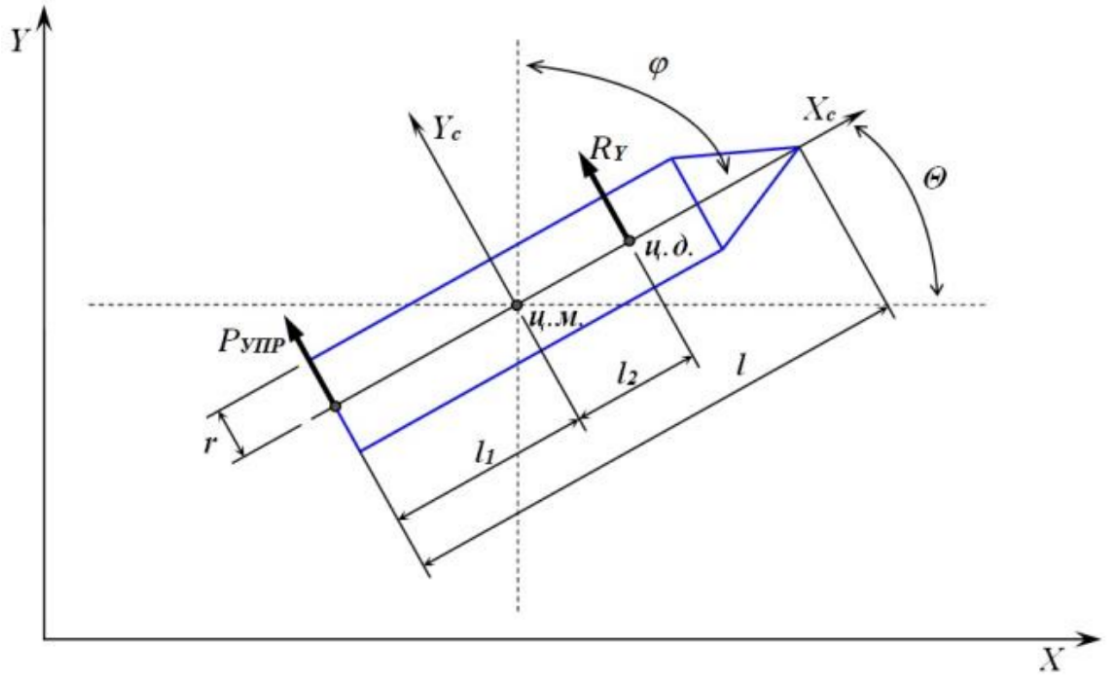



*Рис. 5 — Управляемый поворот ракеты относительно центра масс*

Управляющая сила:

Управляющая сила предназначена для управления поворотом силы тяги ракеты в процессе полёта.

Закон изменения угла поворота:

$M_{упр} = \varepsilon \cdot I_U,$

где ε — угловое ускорение

$I_U$ — момент инерции вращения ракеты относительно центра масс

$M_{упр} = P_{упр} l_1 - R_y l_2,$

$\varepsilon = \frac{d\omega}{dt} = \omega',$

$\omega = \frac{d\varphi}{dt} = \varphi',$

$\varepsilon = \varphi'' = \frac{1}{I_U (P_{упр} l_1 - R_y l_2)},$

$I_U = \frac{1}{4} \cdot m \cdot r^2 + \frac{1}{12} \cdot m \cdot l^2$ — момент инерции вращения относительно поперечных осей (приблизительный).

Расчёт траэктории движения ракеты в полёте:

Запишем II закон Ньютона в проэкциях на оси x и y:

$$\begin{cases} (P - R_X) \cdot \cos\Theta - (P_{УПР} + R_Y) \cdot \sin\Theta = m_{РН} \cdot a_x \\ (P - R_X) \cdot \sin\Theta + (P_{УПР} + R_Y) \cdot \cos\Theta - m_{РН} \cdot g = m_{РН} \cdot a_y \end{cases}.$$

Тогда получим:

$$\begin{cases} m_{РН} \cdot \ddot{x} = (P - R_X) \cdot \cos\Theta - (P_{УПР} + R_Y) \cdot \sin\Theta \\ m_{РН} \cdot \ddot{y} = (P - R_X) \cdot \sin\Theta + (P_{УПР} + R_Y) \cdot \cos\Theta - m_{РН} \cdot g \\ \Theta = \frac{\pi}{2} - \varphi \\ \ddot{\varphi} = \frac{1}{I_И} \cdot (P_{УПР} \cdot l_1 - R_Y \cdot l_2) \end{cases}.$$

Физика торможения:

$$\frac{dV}{dt} = -\frac{1}{2} pV^2 cS \frac{1}{m} - g,$$

c — коэффициент аэродинамического сопротивления

$$\rho \sim \frac{1}{n},\ g \sim \frac{1}{n^2},$$

$\rho = \rho_0 e^{\frac{-n}{H}}$ — плотность атмосферы

$\rho_0$ — плотность на поверхности

H — высота масштабирования атмосферы

$$g = g_0 \left( \frac{R}{R+h} \right)$$

R — радиус планеты

$g_0$ — ускорение свободного падения на поверхности

h — высота над поверхностью

Торможение с помощью парашюта:

$V_{nap} = \sqrt{\frac{2mg}{\rho C_{nap} S_{nap}}}$ - скорость стабилизации, из-за большого сопротивления, аппарат достигает равновесной скорости, при которой силы аэродинамического сопротивления и тяжести уравновешены

В итоге получаем динамические параметры в течение полёта:

$m_T(t) = m_{T0} - (P + P_{упр}) \cdot \frac{t}{I}$ - масса топлива

Если $m_T(t) > 0$:

$m_{PH} = m_0 - m_{T0} + m_T(t)$
$P(t) = P$
$P_{упр}(t) = P_{упр}$

Если $m_T(t) < 0$:

$m_{PH} = m_0 - m_{T0}$
$P(t) = 0$
$P_{упр}(t) = 0$

$$M_{упр}(t) = P_{упр} \cdot \frac{1}{2}$$

$$I_И = \frac{1}{4} \cdot m \cdot r^2 + \frac{1}{12} \cdot m \cdot l^2 = \frac{1}{16} \cdot m \cdot D^2 + \frac{1}{12} \cdot m \cdot l^2$$

$$\omega(t_{i+1}) = \omega(t_i) + \frac{M_{упр}}{I_И} \cdot dt$$

$$\varphi(t_{i+1}) = \varphi + \omega_i \cdot dt$$

$$V_X(t_{i+1}) = V_X(t_i) + \frac{(P(t) - R_X) \cdot \cos\left(\frac{\pi}{2} - \varphi\right) - P_{упр} \cdot \sin\left(\frac{\pi}{2} - \varphi\right)}{m_{РН}(t)} \cdot dt$$

$$V_Y(t_{i+1}) = V_Y(t_i) + \frac{(P(t) - R_Y) \cdot \cos\left(\frac{\pi}{2} - \varphi\right) + P_{упр} \cdot \sin\left(\frac{\pi}{2} - \varphi\right)}{m_{РН}(t)} \cdot dt$$

$$x(t_{i+1}) = x(t_i) + V_X(t_{i+1}) \cdot dt$$

$$y(t_{i+1}) = y(t_i) + V_Y(t_{i+1}) \cdot dt$$

Гомановские переходные траектории — экономичные межпланетные перелёты.

Гоманов доказал, что в орбитальной механике эллиптическая орбита, образующая касательную к исходной и целевой орбитам, с наименьшими затратами энергии.

## 1. 4 Математическая модель

**Сначала разберём некоторые константы, используемые в дальнейшем решении:**

$m_0$ = 438665 кг —  общая начальная масса ракеты

$m_{T0}$ = 348400 кг — начальная масса топлива ракеты

Вместо $P_i$ подставляем значения $P_y$, $P_{1,2}$, $P_3$ , $P_4$:

$P_y$ = 1515217 Н *6 — сила тяги ускорителей (1 атм)

$P_{1,2}$ = 1319032 Н — сила тяги 1, 2 двигателя (1 атм)

$P_3$ = 13875 Н — сила тяги 3 двигателя (1 атм)

$P_4$ = 64286 Н — сила тяги 4 двигателя (1 атм)

$P_{упр}$ = 4 Н — управляющая сила

Вместо $\omega_0$ подставляем значения $\omega_y$, $\omega_{1,2}$, $\omega_3$ , $\omega_4$:

$\omega_y$ = 100,494/с *6 — скорость истечения продуктов сгорания топлива ускорителей (1 атм)

$\omega_{1,2}$ = 44,407/с — скорость истечения продуктов сгорания топлива 1, 2 двигателя (1 атм)

$\omega_3$ = 1,53/с  — скорость истечения продуктов сгорания топлива 3 двигателя (1 атм)

$\omega_4$ = 6,555/с — скорость истечения продуктов сгорания топлива 4 двигателя (1 атм)

**В итоге после подставления получаем динамические параметры в течение полёта:**

$P = P_i + S_\alpha \cdot p_{H0}[1 - \pi(y)]$, где $P_i = \{ P_y, P_{1,2}, P_3, P_4 \}$.

$m_T(t) = 348400\,кг - (P + 4H) \cdot \frac{t}{I}$,

| Если $m_T(t) > 0$: | Если $m_T(t) < 0$: |
|---|---|
| $m_{PH} = 438665\,кг - 348400\,кг + m_T(t)$ | $m_{PH} = 438665\,кг - 348400\,кг$ |
| $P(t) = P$ | $P(t) = 0$ |
| $P_{упр}(t) = 4H$ | $P_{упр}(t) = 0$ |

$M_{упр}(t) = 4H \cdot \frac{1}{2}$,

$I_И = \frac{2H}{\omega'}$,

$\omega(t_{i+1}) = \omega(t_i) + \omega' \cdot dt$,

$\varphi(t_{i+1}) = \varphi + \omega_i \cdot dt$, где $\omega_i = \{ \omega_y, \omega_{1,2}, \omega_3, \omega_4 \}$.

$$V_X(t_{i+1}) = V_X(t_i) + \frac{(P(t) - \frac{1}{n}) \cdot \cos\left(\frac{\pi}{2} - \varphi\right) - 2H \cdot \sin\left(\frac{\pi}{2} - \varphi\right)}{m_{PH}(t)} \cdot dt,$$

$$V_Y(t_{i+1}) = V_Y(t_i) + \frac{(P(t) - \frac{1}{n}) \cdot \cos\left(\frac{\pi}{2} - \varphi\right) + 2H \cdot \sin\left(\frac{\pi}{2} - \varphi\right)}{m_{PH}(t)} \cdot dt,$$

$x(t_{i+1}) = x(t_i) + V_X(t_{i+1}) \cdot dt$,

$y(t_{i+1}) = y(t_i) + V_Y(t_{i+1}) \cdot dt$,

С помощью этих формул и функций можно построить графики, описывающие движение ракеты-носителя. Подробнее про них написано в заключении.

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2. 1 Моделирование ракеты-носителя

Ракета-носитель марсохода Martia-1 состоит из 6 ступеней.

Шестая ступень — спусковой модуль. Для целей моделирования полёта он состоит из кабины с пилотом и посадочных опор, однако, ввиду ограничений программы, это стоит принимать за сам марсоход. Марсоход оснащён аккумуляторными батареями и солнечными панелями, что позволит ему накапливать и сохранять энергию для работы. Также имеются парашюты — тормозные и посадочные, что позволит марсоходу совершить мягкую посадку на Дюне.

Пятая ступень состоит из топливного бака и двигателя на жидком топливе. Предназначена для корректировки курса при приближении к Дюне.

Четвёртая ступень состоит из 5 топливных баков и ядерного ракетного двигателя. Предназначена для манёвров в открытом космосе, т. е. после покидания орбиты Кербина и до входа в орбиту Дюны.

Вторая и третья ступени состоят из топливных баков и двигателей на жидком топливе. Предназначены для ускорения при выходе из орбиты Кербина.

Первая ступень состоит из 6 твердотопливных ускорителей. Предназначена для выхода на орбиту Кербина.

У ракеты имеется защитный обтекатель, который защищает последние ступени от термального урона и придает аэродинамичности. Отбрасывается вместе с третьей ступенью, т. к. в этот момент ракета уже не находится в атмосфере и оба аспекта обтекателя больше не нужны.

## 2. 2 Написание автопилота

**Подробное описание автопилота выхода на орбиту (orbit.py).** На Рисунках 6-12 показаны части программного кода.

1. **Первый этап. Подготовка к взлету.**

```
conn = krpc.connect()  # подключаемся к серверу
vessel = conn.space_center.active_vessel  # активный корабль
control = vessel.control  # контролировать корабль
ap = vessel.auto_pilot  # работать с автопилотом

ut_start = conn.space_center.ut
ut_now = conn.space_center.ut
# готовимся к запуску
ap.target_pitch_and_heading(90, 90)
ap.engage()

# переменные потоки, при вызове которых мы получаем данные из KSP
ut = conn.add_stream(getattr, conn.space_center, 'ut')  # текущее время в KSP
stage_5_resources = vessel.resources_in_decouple_stage(stage=5, cumulative=False)  # пятая ступень (та, где отделяются ускорители)
srb_fuel = conn.add_stream(stage_5_resources.amount, 'SolidFuel')  # количество топлива во всех ускорителях в сумме
altitude = conn.add_stream(getattr, vessel.flight(), 'mean_altitude')  # высота над уровнем моря в метрах
apoapsis = conn.add_stream(getattr, vessel.orbit, 'apoapsis_altitude')  # высота апоцентра в метрах, если считать от уровня моря
periapsis = conn.add_stream(getattr, vessel.orbit, 'periapsis_altitude')  # высота перицентра в метрах, если счтиать от уровня моря
pitch = conn.add_stream(getattr, vessel.flight(), 'pitch')  # рысканье ракеты
```

*Рис. 6 — Часть кода*

**conn = krpc.connect()**

- Устанавливаем соединение с сервером **kRPC**, который является интерфейсом между Python и KSP.
- Это позволяет управлять кораблём и получать телеметрию из игры в реальном времени.

**vessel = conn.space_center.active_vessel**

- Получает ссылку на активный космический корабль.

**control = vessel.control**

- Ссылка на систему управления кораблём.
- Через control можно активировать ступени, регулировать тягу, изменять направление полёта и выполнять другие действия.

**ap = vessel.auto_pilot**

- Ссылка на автопилот корабля.
- Позволяет задавать направление (угол наклона, курс) и стабилизировать полёт без ручного управления.

**ut_start = conn.space_center.ut и ut_now = conn.space_center.ut**

- Переменные для хранения текущего времени в игре (**Universal Time**).
- ut_start фиксирует начальный момент времени, когда начинается работа программы.
- ut_now используем для отслеживания времени в процессе выполнения полёта.

**ap.target_pitch_and_heading(90, 90)**

- Устанавливает начальный угол наклона и курс:
  - **Pitch (угол наклона):** 90° — вертикальный взлёт.
  - **Heading (курс):** 90° — направление на восток.

**ap.engage()**

- Активирует автопилот, который будет поддерживать заданный угол наклона и курс.

**Создание потоков данных**

***ut = conn.add_stream(getattr, conn.space_center, 'ut')***

- Поток, возвращающий текущее время в игре (**Universal Time**).
- Используется для расчёта продолжительности полёта или тайминга манёвров.

***stage_5_resources = vessel.resources_in_decouple_stage(stage=5, cumulative=False)***

- Получает доступ к ресурсам пятой ступени (ускорители) корабля.
- Аргументы:

- stage=5 — номер ступени.
- cumulative=False — ресурсы учитываются только для указанной ступени, без объединения с другими.

***srb_fuel = conn.add_stream(stage_5_resources.amount, 'SolidFuel')***

- Создаёт поток для отслеживания оставшегося твёрдотопливного топлива в пятой ступени (ускорители).

***altitude = conn.add_stream(getattr, vessel.flight(), 'mean_altitude')***

- Поток для отслеживания высоты над уровнем моря.

***apoapsis = conn.add_stream(getattr, vessel.orbit, 'apoapsis_altitude')***

- Поток для отслеживания высоты апоцентра орбиты.
- Апоцентр — самая высокая точка орбиты.

***periapsis = conn.add_stream(getattr, vessel.orbit, 'periapsis_altitude')***

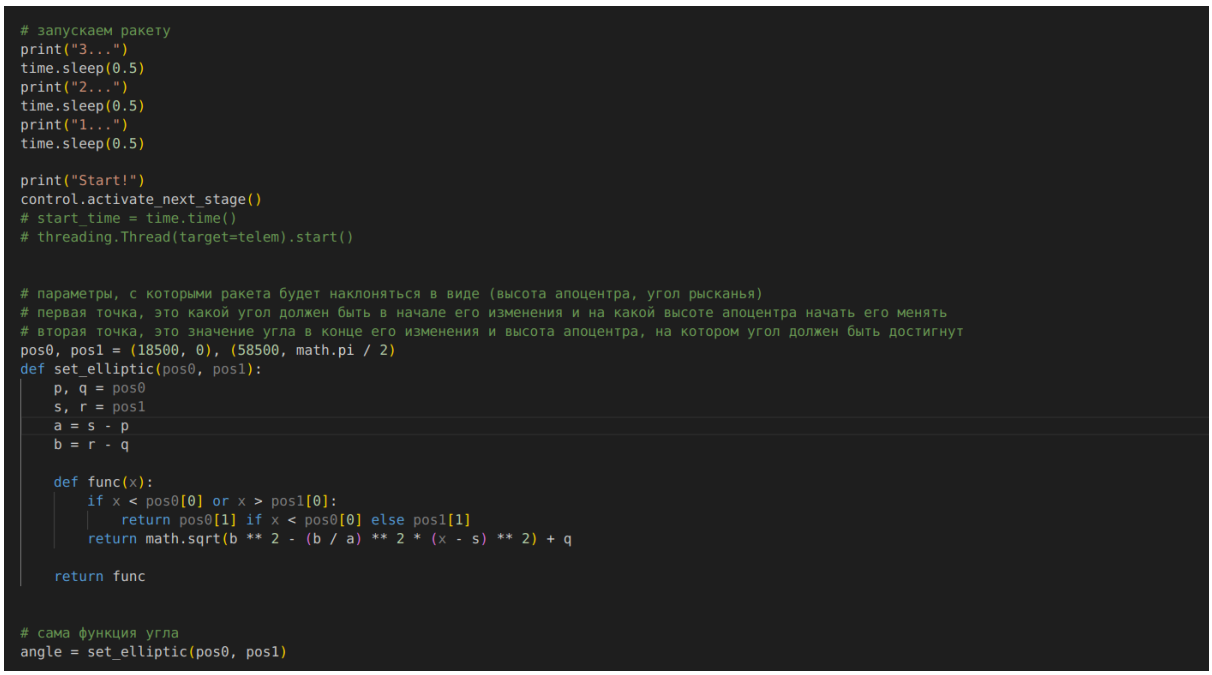
- Поток для отслеживания высоты перицентра орбиты.
- Перицентр — самая низкая точка орбиты.

***pitch = conn.add_stream(getattr, vessel.flight(), 'pitch')***

- Поток для получения текущего угла наклона корабля относительно горизонта.

-

2. **Второй этап. Запуск ускорителей.**



```
# запускаем ракету
print("3...")
time.sleep(0.5)
print("2...")
time.sleep(0.5)
print("1...")
time.sleep(0.5)

print("Start!")
control.activate_next_stage()
# start_time = time.time()
# threading.Thread(target=telem).start()


# параметры, с которыми ракета будет наклоняться в виде (высота апоцентра, угол рысканья)
# первая точка, это какой угол должен быть в начале его изменения и на какой высоте апоцентра начать его менять
# вторая точка, это значение угла в конце его изменения и высота апоцентра, на котором угол должен быть достигнут
pos0, pos1 = (18500, 0), (58500, math.pi / 2)
def set_elliptic(pos0, pos1):
    p, q = pos0
    s, r = pos1
    a = s - p
    b = r - q

    def func(x):
        if x < pos0[0] or x > pos1[0]:
            return pos0[1] if x < pos0[0] else pos1[1]
        return math.sqrt(b ** 2 - (b / a) ** 2 * (x - s) ** 2) + q

    return func


# сама функция угла
angle = set_elliptic(pos0, pos1)
```

*Рис. 7 — Часть кода*

***Обратный отсчёт***

print("3...")

time.sleep(0.5)

print("2...")

time.sleep(0.5)

print("1...")

time.sleep(0.5)

- Выводит на экран последовательный отсчёт перед запуском ракеты.
- Использует time.sleep(0.5) для задержки выполнения программы на 0.5 секунды между сообщениями.

***Активация ускорителей***

print("Start!")

control.activate_next_stage()

start_time = time.time()

- **control.activate_next_stage()**: Запускает ускорители.

### Функция изменения угла наклона (сердце автопилота)

Ракета сначала летит строго вертикально (90°), но должна постепенно ложиться на горизонтальный полёт. Этот процесс управляется гравитационным поворотом.

***Параметры гравитационного поворота***

**pos0, pos1 = (18500, 0), (58500, math.pi / 2)**

- **pos0 = (18500, 0)**:
  - При апоцентре (высота самой дальней точки траектории) **18,500 м**, ракета начинает изменять угол наклона.
  - Угол в этот момент — **0°** (полет строго вертикально).
- **pos1 = (58500, math.pi / 2)**:
  - При апоцентре **58,500 м**, угол наклона ракеты должен быть **90°** (горизонтальный полёт).

***Определение функции изменения угла***

```
# параметры, с которыми ракета будет наклоняться в виде (высота апоцентра, угол рысканья)
# первая точка, это какой угол должен быть в начале его изменения и на какой высоте апоцентра начать его менять
# вторая точка, это значение угла в конце его изменения и высота апоцентра, на котором угол должен быть достигнут
pos0, pos1 = (18500, 0), (58500, math.pi / 2)
def set_elliptic(pos0, pos1):
    p, q = pos0
    s, r = pos1
    a = s - p
    b = r - q

    def func(x):
        if x < pos0[0] or x > pos1[0]:
            return pos0[1] if x < pos0[0] else pos1[1]
        return math.sqrt(b ** 2 - (b / a) ** 2 * (x - s) ** 2) + q

    return func
```

*Рис. 8 — Часть кода*

1. **p, q = pos0 и s, r = pos1:**
    - Переменные для высоты (p и s) и углов (q и r).
2. **a = s - p и b = r - q:**
    - Расстояния между начальной и конечной точками по осям высоты и угла.
3. **if x < pos0[0] or x > pos1[0]:**
    - Если текущая высота x находится вне заданного диапазона [p, s]:
        - Вернуть угол начальной точки q для высоты ниже pos0[0].
        - Вернуть угол конечной точки r для высоты выше pos1[0].
4. **return math.sqrt(...)**:
    - Если x находится в диапазоне [p, s], угол рассчитывается по эллиптической формуле:

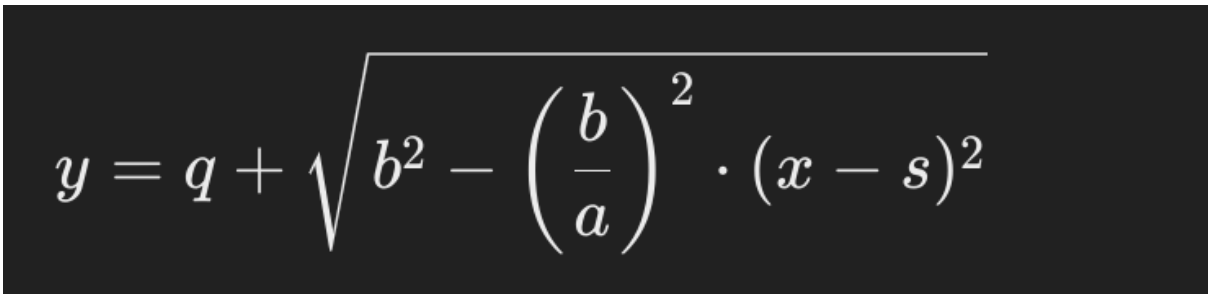



*Рис. 9 — Формула, использованная в части кода*

        - Гарантирует плавное изменение угла от начального значения до конечного.

***Инициализация функции***

angle = set_elliptic(pos0, pos1)

- angle — это теперь функция, которая принимает высоту апоцентра и возвращает угол наклона ракеты.

Пример использования:
print(angle(20000))  # Угол наклона при апоцентре 20,000 м

```
# ждём, пока апоцентр достигнет нужной высоты
while altitude() < pos0[0]:
    time.sleep(0.5)

# наклоняем ракету до тех пор, пока топливо в ускорителях не закончится
while srb_fuel() >= 0.1:
    # print(apoapsis(), angle(apoapsis()))
    ap.target_pitch = math.degrees(math.pi / 2 - angle(altitude()))
    # print(srb_fuel(), math.degrees(math.pi / 2 - angle(altitude())))
    time.sleep(0.2)
ap.target_pitch = 0

# отсоединяем их
control.activate_next_stage()
print("Ускорители отброшены")
```

*Рис. 10 — Часть кода*

**3 Этап. Доорбитальный полет.**

***Задачи:***

1. Плавно изменить угол наклона ракеты, переходя к горизонтальному полёту (гравитационный поворот).
2. Отбросить ускорители после их израсходования.

***Ждём достижения нужной высоты апоцентра***

while altitude() < pos0[0]:

time.sleep(0.5)

- Цикл проверяет текущую высоту ракеты с помощью altitude().
- **Цель:** дождаться, пока высота апоцентра достигнет **18500 м** (значение pos0[0]).
- Использует time.sleep(0.5) чтобы KSP не завис.

***Наклон ракеты***

while srb_fuel() >= 0.1:

ap.target_pitch = math.degrees(math.pi / 2 - angle(altitude()))

time.sleep(0.2)

- Пока у ускорителей остаётся хотя бы 10% топлива:

- **angle(altitude())**: Рассчитывает целевой угол наклона на основе текущей высоты ракеты.
- **math.pi / 2 - angle(...)**: Преобразует значение угла в наклон ракеты.
- **ap.target_pitch = ...**: Автопилот меняет угол ракеты на рассчитанный.

- Задержка time.sleep(0.2) снижает нагрузку на KSP и ПК.

***Устанавливаем горизонтальный полёт***

ap.target_pitch = 0

- После окончания топлива у ускорителей:
  - Устанавливает угол наклона на **0°**, переходя к горизонтальному полёту.

***Отделяем ускорители***

control.activate_next_stage()

print("Ускорители отброшены")

- **control.activate_next_stage()**: Выполняет отделение ускорителей.
- Сообщение **"Ускорители отброшены"** выводится в терминал для информирования.

```
# Функция, которая считает гомановский переход
def gam(mu, r1, r2):
    # принимает стандартный гравитационный параметр mu
    # радиус круговой орбиты r1, радиус круговой орбиты r2
    # r1 < r2
    a = (r1 + r2) / 2
    dv1 = math.sqrt(mu / r1) * (math.sqrt(r2 / a) - 1)
    dv2 = math.sqrt(mu / r2) * (1 - math.sqrt(r1 / a))
    # dv1 - На такое значение нужно увеличить скорость, будучи на круговой орбите r1
    # dv2 - на Такое значение нужно увеличить скорость, будучи на переходной траектории гомана (высота апоцентра это r2)
    return dv1, dv2

time.sleep(3)
mu = vessel.orbit.body.gravitational_parameter
delta_v = gam(mu, vessel.orbit.periapsis, vessel.orbit.apoapsis)[1]
node = control.add_node(ut() + vessel.orbit.time_to_apoapsis, prograde=delta_v)
time.sleep(1)
```

*Рис. 11 — Часть кода*

***Функция для расчёта манёвра (гомановский переход)***

**Описание параметров:**

1. **mu**: Стандартный гравитационный параметр тела.

    - Определяется как произведение гравитационной постоянной на массу тела.
2. **r1**: Радиус начальной орбиты (например, перицентр текущей орбиты).
3. **r2**: Радиус целевой орбиты (например, апоцентр переходной орбиты).

**Рассчитывает:**

**dv1 - На такое значение нужно увеличить скорость, будучи на круговой орбите r1**

**dv2 - на Такое значение нужно увеличить скорость, будучи на переходной траектории гомана (высота апоцентра это r2)**

***Применение функции и расчёт манёвра***

```
mu = vessel.orbit.body.gravitational_parameter


delta_v = gam(mu, vessel.orbit.periapsis, vessel.orbit.apoapsis)[1]
```

1. **mu**: Вычисляется из гравитационного параметра текущей планеты или тела.
    - Доступен через vessel.orbit.body.gravitational_parameter.
2. **delta_v**: Берётся второе значение из функции test3, которое соответствует скорость для выхода на целевую орбиту.

***Добавление манёвра***

```
node = control.add_node(ut() + vessel.orbit.time_to_apoapsis,
prograde=delta_v)
```

- **ut()**: Текущее универсальное время в игре.
- **vessel.orbit.time_to_apoapsis**: Время до апоцентра текущей орбиты.
- **control.add_node**: Добавляет узел манёвра с указанными параметрами:
    - Время манёвра: ut() + vessel.orbit.time_to_apoapsis
    - Прирост скорости (prograde): prograde=delta_v

***Добавление задержки***

```
time.sleep(1)
```

- Делает паузу в 1 секунду перед продолжением следующей команды.

```
F = vessel.available_thrust
Isp = vessel.specific_impulse * 9.82
m0 = vessel.mass
m1 = m0 / math.exp(delta_v / Isp)
flow_rate = F / Isp
burn_time = (m0 - m1) / flow_rate

print("Выключаем автопилот")
ap.disengage()

control.sas = True
time.sleep(1)
                                        (function) maneuver: object | Any
control.sas_mode = conn.space_center.SASMode.maneuver

print("Ждём момента до ускорения")
burn_ut = ut() + vessel.orbit.time_to_apoapsis - (burn_time/2)
lead_time = 5
time_to_apoapsis = conn.add_stream(getattr, vessel.orbit, "time_to_apoapsis")
while time_to_apoapsis() - (burn_time / 2) > 0:
    time.sleep(0.05)

print("Запускаем двигатели")
control.throttle = 1.0
time_when_end = ut() + vessel.orbit.time_to_apoapsis + burn_time - 16
while ut() < time_when_end:
    time.sleep(0.05)


print("Ракета успешно выведена на орбиту 250 км")
control.activate_next_stage()
control.throttle = 0

control.remove_nodes()
```

*Рис. 12 — Часть кода*

***Расчёт параметров для манёвра:***
**Определение доступной тяги и других характеристик ракеты:**

F = vessel.available_thrust

- **F**: Максимальная доступная тяга ракеты (в Ньютонах).

Isp = vessel.specific_impulse * 9.82

- **Isp**: Удельный импульс двигателя в м/с. Переводится из секунд (умножая на ускорение свободного падения, 9.82).

m0 = vessel.mass

- **m0**: Начальная масса ракеты (в кг).

m1 = m0 / math.exp(delta_v / Isp)

- **m1**: Окончательная масса ракеты после сжигания топлива для выполнения манёвра, рассчитывается по уравнению Циолковского:

flow_rate = F / Isp

- **flow_rate**: Расход топлива (кг/с)

burn_time = (m0 - m1) / flow_rate

- **burn_time**: Время сжигания топлива (в секундах).

***Выключение автопилота:***

```
print("Выключаем автопилот")
ap.disengage()
```

- Отключает текущий автопилот ракеты.

***Переход к ручному управлению с использованием SAS:***

```
control.sas = True
time.sleep(1)
control.sas_mode = conn.space_center.SASMode.maneuver
```

- **control.sas = True**: Включает SAS (система автоматической стабилизации).
- **control.sas_mode**: Устанавливает режим SAS в **maneuver** (наведение на точку манёвра).

***Ожидание момента для начала манёвра:***

```
burn_ut = ut() + vessel.orbit.time_to_apoapsis - (burn_time / 2)
lead_time = 5
time_to_apoapsis=conn.add_stream(getattr,vessel.orbit,"time_to_apoas”)
```

- **burn_ut**: Рассчитывает точное время старта манёвра:
  - Манёвр должен начаться за половину времени сжигания топлива до апоцентра.
- **time_to_apoapsis**: Создаёт поток, который отслеживает время до апоцентра.

```
while time_to_apoapsis() - (burn_time / 2) > 0:
   time.sleep(0.05)
```

- Ожидает, пока до апоцентра не останется меньше половины времени сжигания топлива.

***Запуск двигателей:***

```
print("Запускаем двигатели")
control.throttle = 1.0
```

- Устанавливает полный газ (throttle=1.0)

***Выполнение манёвра:***

```
time_when_end = ut() + vessel.orbit.time_to_apoapsis + burn_time - 16
while ut() < time_when_end:
    time.sleep(0.05)
```

- **time_when_end**: Рассчитывает момент окончания манёвра.
  - Манёвр продолжается до конца рассчитанного времени.
- **while ut() < time_when_end**: Следит за текущим временем и завершает цикл, когда манёвр выполнен.

***Завершение манёвра:***

```
print("Ракета успешно выведена на орбиту 250 км")
control.activate_next_stage()
control.throttle = 0
```

- Оповещает о завершении манёвра.
- Активирует следующую ступень ракеты.
- Останавливает двигатели (throttle=0\text{throttle} = 0).

***Удаление узлов манёвра:***

```
control.remove_nodes()
```

- Удаляет все узлы манёвра, чтобы не мешать дальнейшему управлению.

# ЗАКЛЮЧЕНИЕ

После выполнения работы были составлены графики для сравнения теоретических вычислений и практической симуляции.

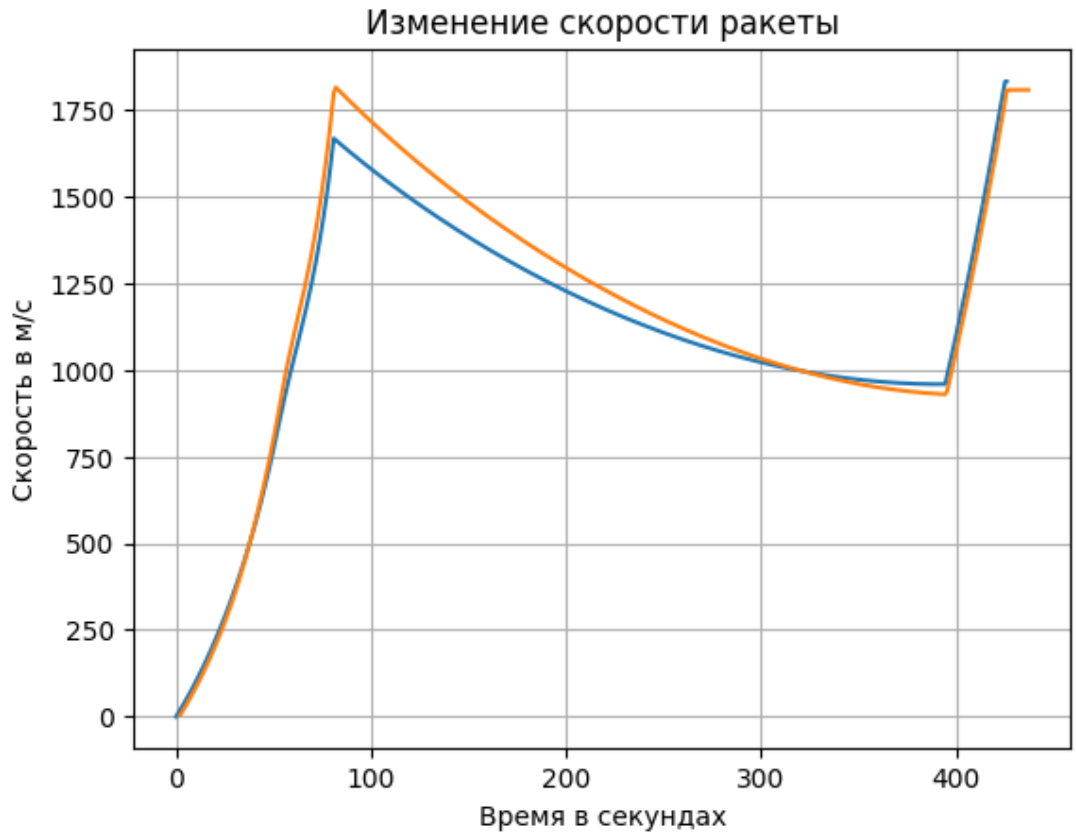



*Рис. 13 — График зависимости скорости ракеты от времени*

Скорость на графике (Рисунок 13) достигается меньшая потому, что мы не учитываем изменение давления атмосферы, которая падает на определенной высоте.

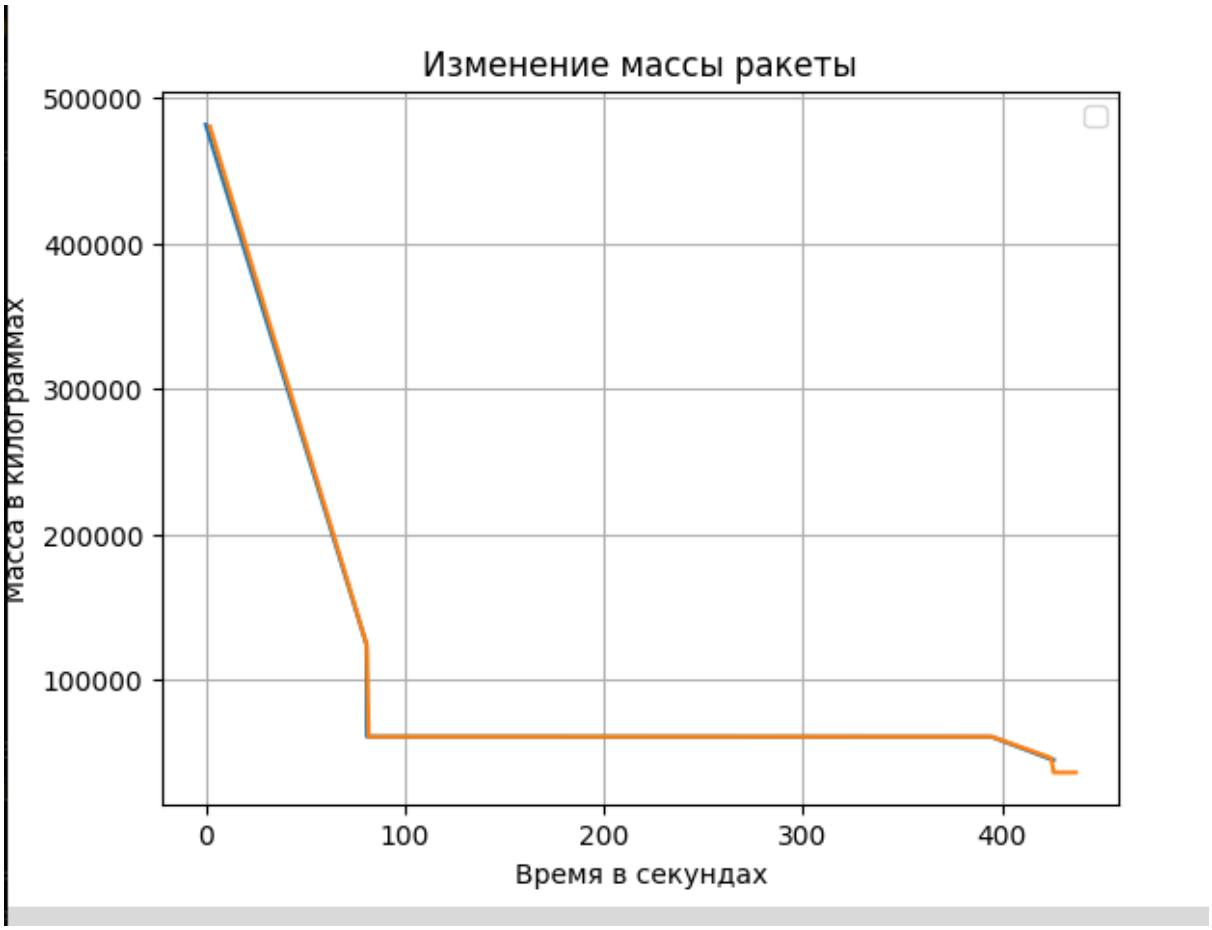



*Рис. 14 — График зависимости массы ракеты от времени*

График изменения масс — Рисунок 14. Тут погрешности почти нет.

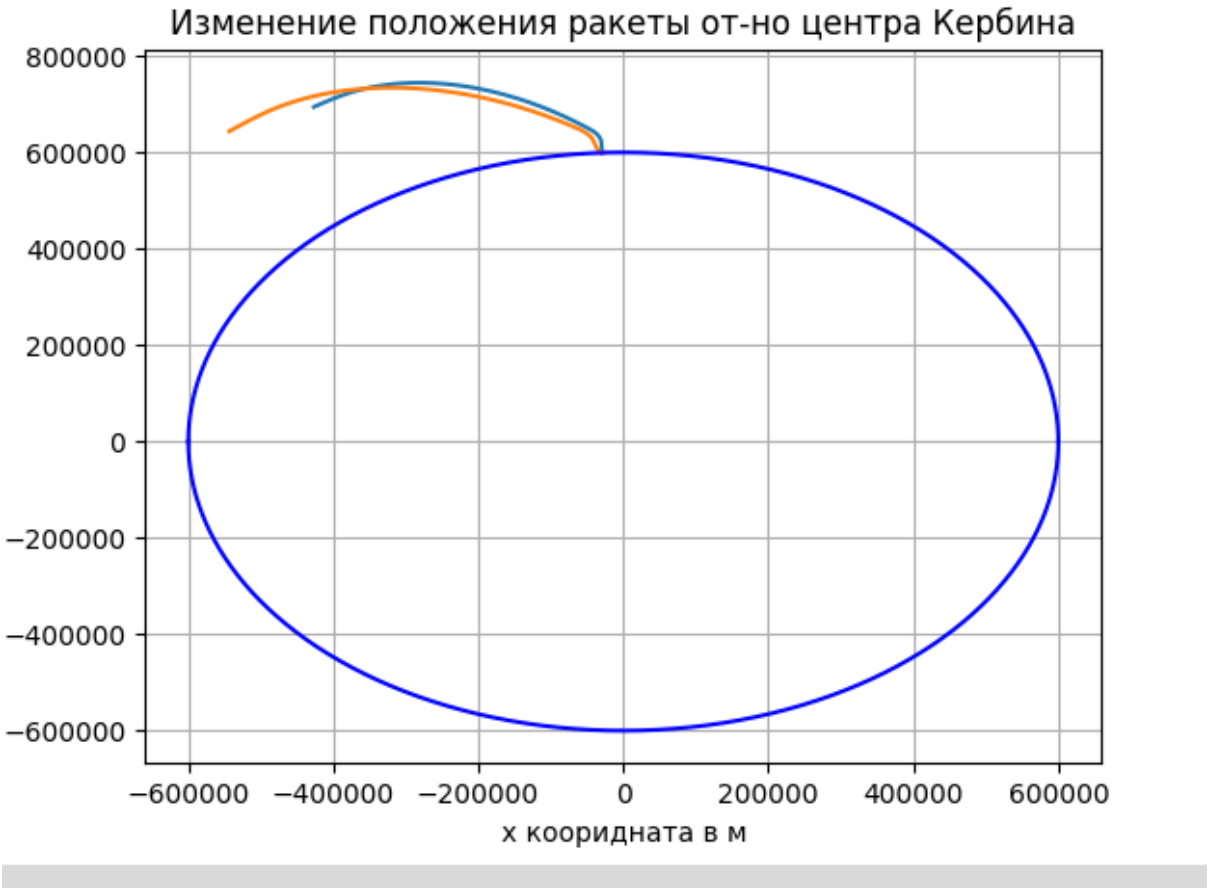



*Рис. 15 — График координат ракеты*

Расчетный график на Рисунке 15 медленнее заворачивает потому, что угол наклона зависит от высоты, а на расчетах мы развиваем меньшую скорость, следовательно медленнее набираем высоты, значит медленнее вращаемся.

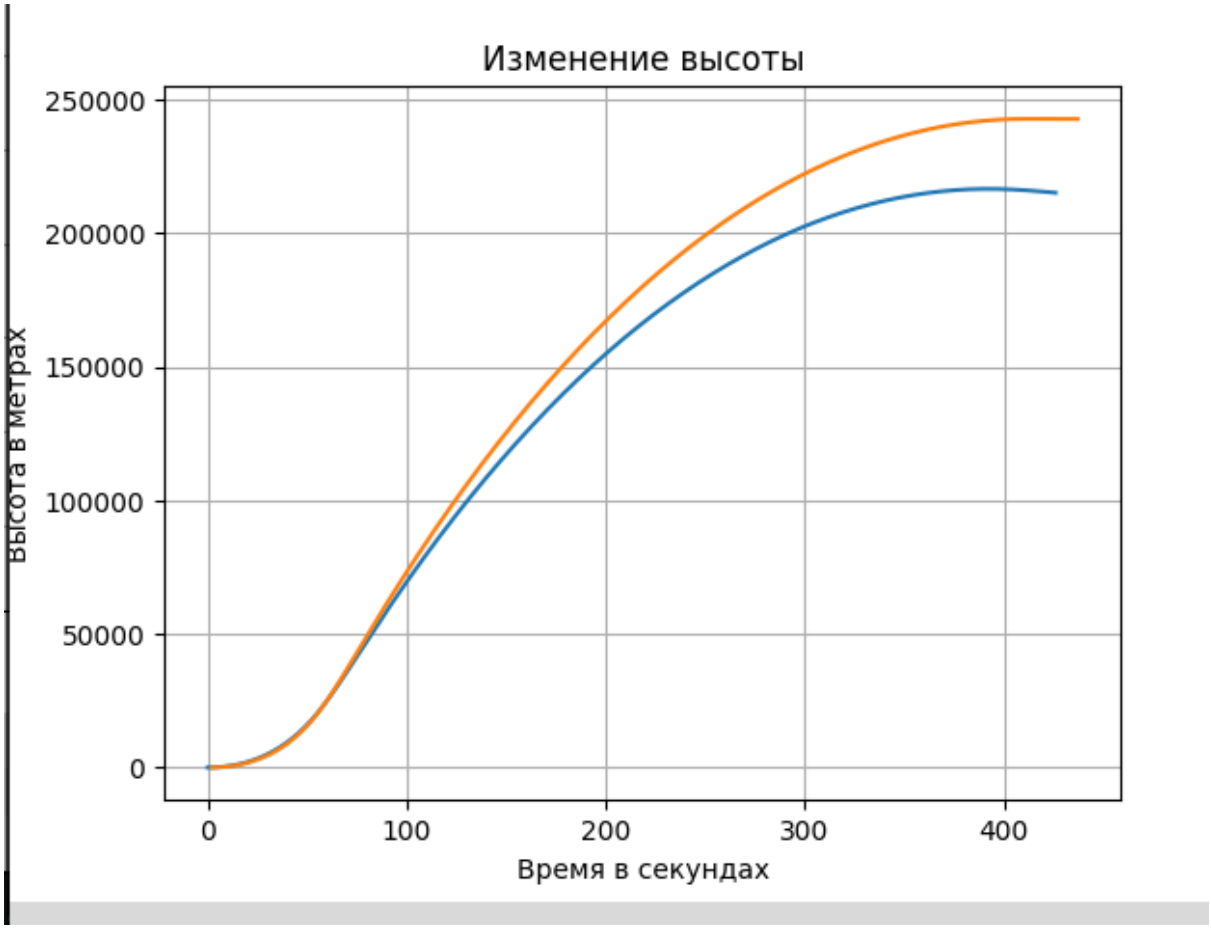



*Рис. 16 — График зависимости высоты ракеты от времени*

Ввиду того что скорость расчетная чуть меньше, а итоговая значительно меньше, то и высота набирается меньшая (Рисунок 16).

В ходе выполнения данной курсовой работы была проведена комплексная оценка процесса доставки марсохода на Марс, начиная с теоретических основ и заканчивая практическим моделированием в симуляторе Kerbal Space Program. В теоретической части были разработаны физическая и математическая модели, которые позволили проанализировать основные силы, действующие на

космический аппарат во время полета. Однако, для упрощения вычислений, некоторые силы, такие как влияние солнечного ветра, были исключены из рассмотрения. Это привело к расхождению графиков, полученных в теоретической и практической частях работы.

Практическая часть, посвященная моделированию полета и созданию автопилота, продемонстрировала, что несмотря на упрощения в физической модели, можно достичь результатов, близких к симуляции полета. Данные, собранные в процессе экспериментов, позволили визуализировать траектории и временные параметры, что подтвердило основные выводы теоретической части. Тем не менее, расхождения между теоретическими расчетами и практическими результатами подчеркивают важность учета всех действующих сил для более точного моделирования.

В приложении 1 можно найти ссылку и qr-код, ведущие ко всем файлам работы.

Работа показала, что теоретические модели являются важным инструментом для понимания процессов, связанных с космическими полетами, однако для достижения высокой точности необходимо учитывать все факторы, влияющие на движение космического аппарата. Полученные результаты открывают новые горизонты для дальнейших исследований и могут служить основой для более сложных моделей, учитывающих все аспекты полета.

# СПИСОК ЛИТЕРАТУРЫ

Mars Pathfinder Интернет-ресурс: https://science.nasa.gov/mission/mars-pathfinder/

Mars Exploration Rovers: Spirit and Opportunity Интернет-ресурс: https://science.nasa.gov/mission/mars-exploration-rovers-spirit-and-opportunity/

Mars Science Laboratory: Curiosity Rover Интернет-ресурс: https://science.nasa.gov/mission/msl-curiosity/

Mars 2020: Perseverance Rover Интернет-ресурс: https://science.nasa.gov/mission/mars-2020-perseverance/

Tianwen-1 Интернет-ресурс: https://habr.com/ru/articles/557564/

Emirates Mars Mission (Hope) Интернет-ресурс: https://space.gov.ae/en/initiatives-and-projects/emirates-mars-mission

The Kerbal Math & Physics Lab Интернет-ресурс: https://sites.google.com/view/kspmath

Википедия по Kerbal Space Program Интернет-ресурс: https://wiki.kerbalspaceprogram.com/wiki/Main_Page

Волоцуев, Владимир Валериевич - Введение в проектирование, конструирование и производство ракет: учеб. пособие / В. В. Волоцуев, И.С. Ткаченко. – Самара: Изд-во Самарского ун-та, 2017. – 88 с.

# ПРИЛОЖЕНИЕ 1 — Github

Ссылка: https://github.com/Aydar8888/Spatium

Qr-код: