|  |  |  |
| --- | --- | --- |
| Министерство образования и науки РФ  Федеральное государственное бюджетное образовательное учреждение | | |
| Moscow-Aviation-Institute | высшего профессионального образования  "Московский авиационный институт"  (национальный исследовательский университет)  **факультет №8 Компьютерные науки и прикладная математика** |  |
| **Кафедра 806 «Вычислительная математика и программирование»** | | |
| **Специальность 01.03.02 «Прикладная математика и информатика»**  **Профиль «Информатика»** | | |

Курсовой проект

по курсу **«Введение в авиационную и ракетно-космическую технику»**

на тему «**Космический аппарат “Вега”**»

|  |  |  |
| --- | --- | --- |
|  | Работу выполнили:  Студенты группы М8О-103БВ-24  Пятницкий Артём Вячеславович  Демидов Георгий Константинович  Рубан Кирилл Александрович  Цицкиев Дени Русланович  Работу принял:  к.ф.-м.н., доцент,  Крылов Сергей Сергеевич | |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Крылов С.С. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Пятницкий А.В. |

Демидов Г.К.

Рубан К.А.

Цицкиев Д.Р.

Москва 2024

**Оглавление**

**Состав**

**КосМАИческие стрижи**

**М8О-103БВ-24**

|  |  |
| --- | --- |
| Участник команды | Роль |
| Пятницкий А.В. | Тимлид, физ. и мат. модель |
| Демидов Г.К. | Программист, KSP |
| Рубан К.А. | Программист, KSP |
| Цицкиев Д.Р. | Физ. и мат. модель |

# **Введение**

Мы вдохновились миссией “Вега” и хотели бы воссоздать часть миссии с изучением Венеры.

**Цель проекта:**

Изучить движение спускаемого аппарата в атмосфере планеты Венера.

**Задачи проекта:**

1. Найти материалы и данные по миссии “Вега”
2. Создать математическую и физическую модель полёта космического аппарата
3. Создать модель космического аппарата в рамках симулятора KSP
4. Собрать необходимые данные для анализа движения спускаемого аппарата в атмосфере Венеры
5. Проанализировать данные и сделать выводы
6. Оформить отчёт о проделанной работе
7. **Описание миссии**

Миссия «Вега» — советский проект, включавший автоматические межпланетные станции «Вега-1» и «Вега-2», предназначенные для исследования Венеры и кометы Галлея. Название «Вега» образовано от слов «Венера» и «Галлей».

## **1.1 Устройство аппарата**

Каждая станция состояла из двух основных частей:

1. **Пролётный аппарат (массой около 3170 кг):**

Научная аппаратура:

* Телевизионная система: для получения изображений ядра кометы Галлея.
* Спектрометры и анализаторы: для изучения состава и свойств кометного вещества.

1. **Спускаемый аппарат (массой около 1750 кг):**

Посадочный модуль (около 680 кг):

Научные приборы:

* Датчики температуры и давления для измерения параметров атмосферы Венеры.
* Спектрометры и хроматографы для анализа химического состава атмосферы и облаков.
* Грунтозаборное устройство с буровой установкой для анализа венерианского грунта

Аэростатный зонд (около 120 кг):

* Оболочка: фторопластовая, диаметром 3,4 метра, наполненная гелием.
* Гондола (6,9 кг): содержала датчики для измерения метеорологических параметров, радиосистему и блок питания.



Рисунок 1. Межпланетная станция «Вега»

## **1.2 План полёта:**

Миссия «Вега» включала сложный многоэтапный план полета, направленный на исследование Венеры и кометы Галлея. Аппараты «Вега-1» и «Вега-2» были запущены с космодрома Байконур на ракете-носителе «Протон-К» в декабре 1984 года, с интервалом в несколько дней.

После выхода на межпланетную траекторию аппараты начали шестимесячный перелет к Венере, используя свои двигательные установки для точной корректировки траектории. В июне 1985 года, спустя около шести месяцев после запуска, оба аппарата достигли окрестностей Венеры. На подлёте к планете, на расстоянии примерно 150 тысяч километров, каждый из них отделил спускаемый модуль, который начал самостоятельное движение по траектории входа в атмосферу.

Посадочные аппараты были оснащены системами защиты, чтобы выдержать экстремальные условия спуска через плотную и горячую атмосферу Венеры. Во время этого процесса они собирали данные о структуре атмосферы, измеряя её температуру, давление и химический состав на разных высотах. Также проводились анализы облаков, включая исследования их состава и структуры. После достижения поверхности Венеры аппараты продолжали работу в течение нескольких десятков минут, собирая информацию о составе грунта и физических условиях на планете. Это время было ограничено из-за экстремально высокой температуры (около 460 °C) и давления (более 90 атмосфер) на поверхности.

1. **Описание математической модели**

Графики ksp:

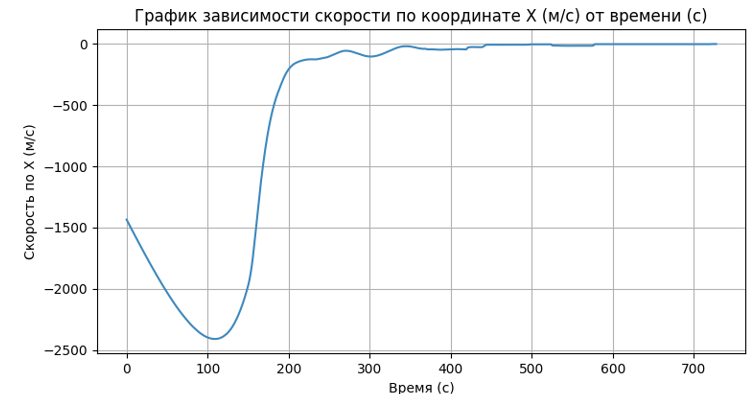


Рисунок 2. График зависимости скорости по X от времени



Рисунок 3. График зависимости скорости по координате Y от времени

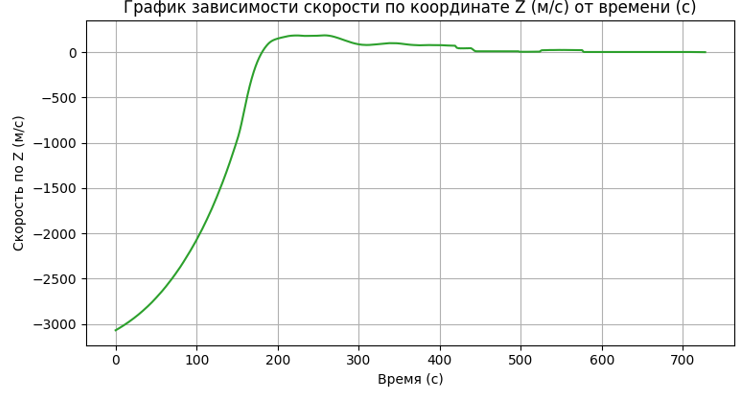


Рисунок 4. График зависимости скорости по Z от времени

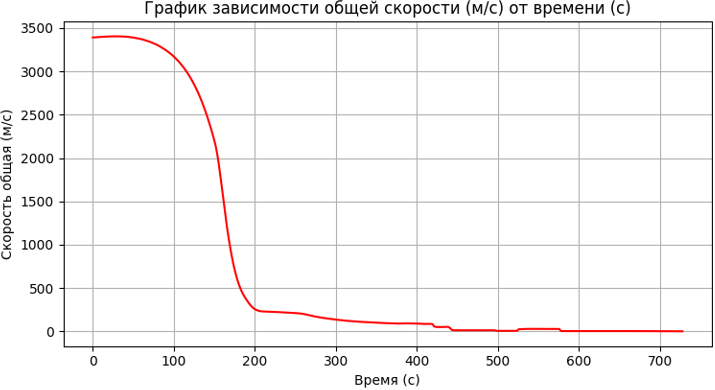


Рисунок 5. График зависимости общей скорости от времени

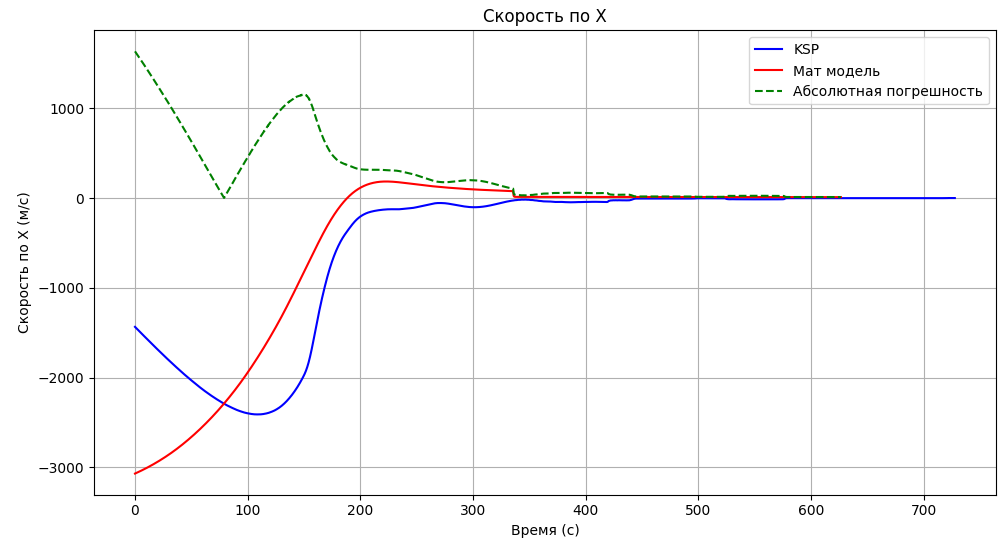


Рисунок 2. График зависимости скорости по X от времени

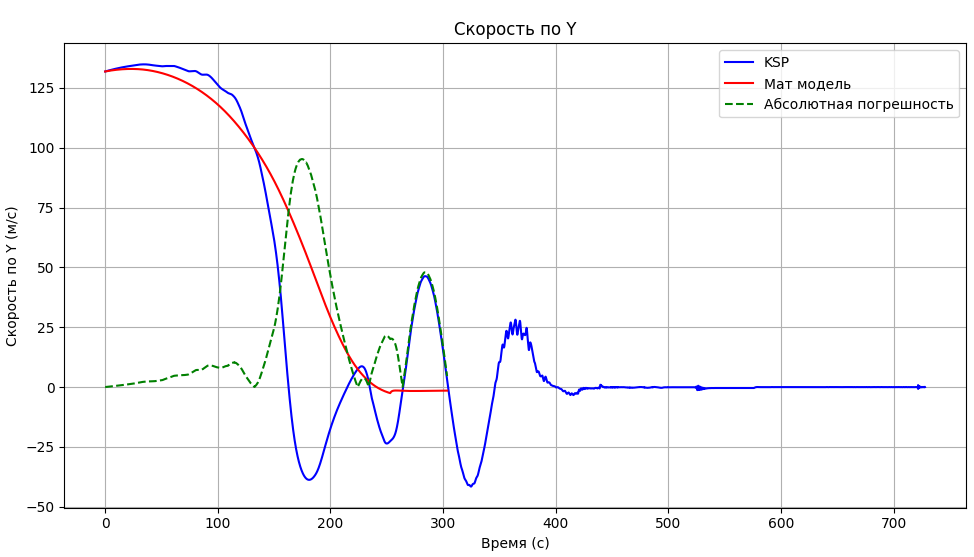


Рисунок 3. График зависимости скорости по координате Y от времени

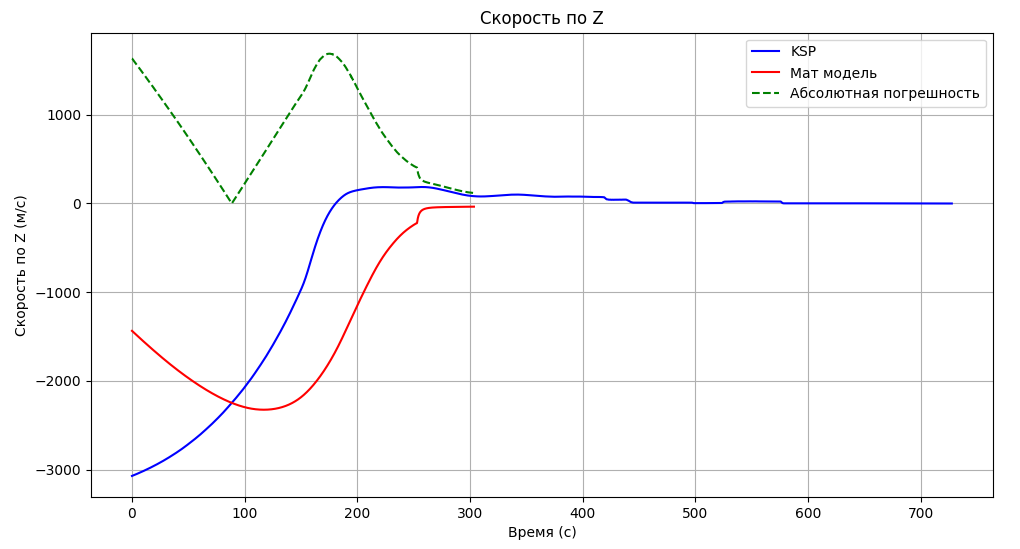


Рисунок 4. График зависимости скорости по Z от времени

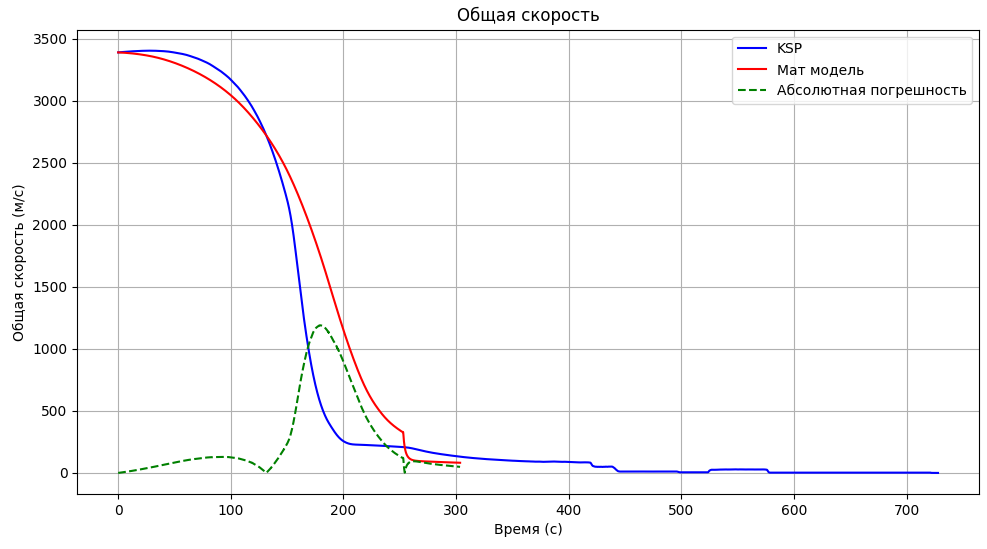


Рисунок 5. График зависимости общей скорости от времени

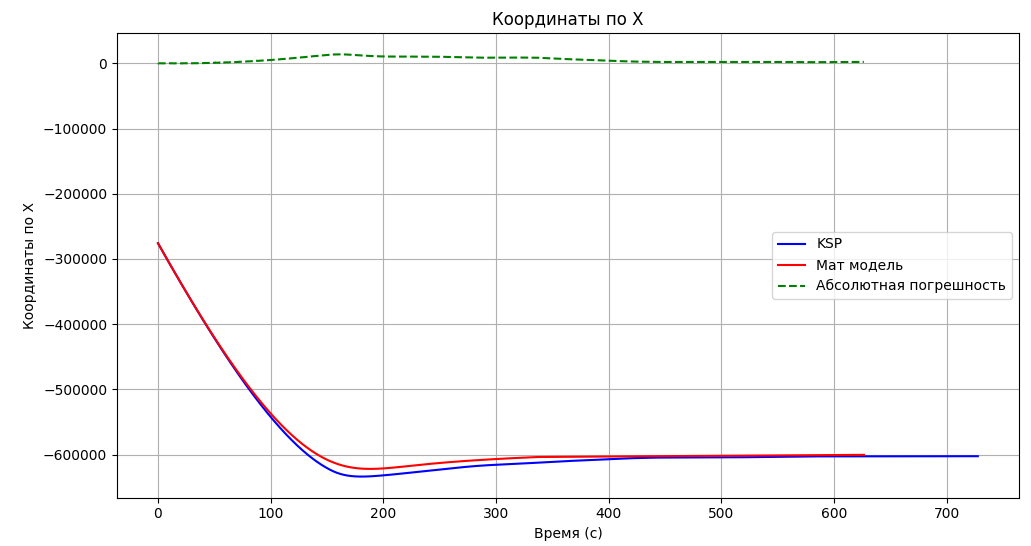


Рисунок 6. График зависимости координаты X от времени

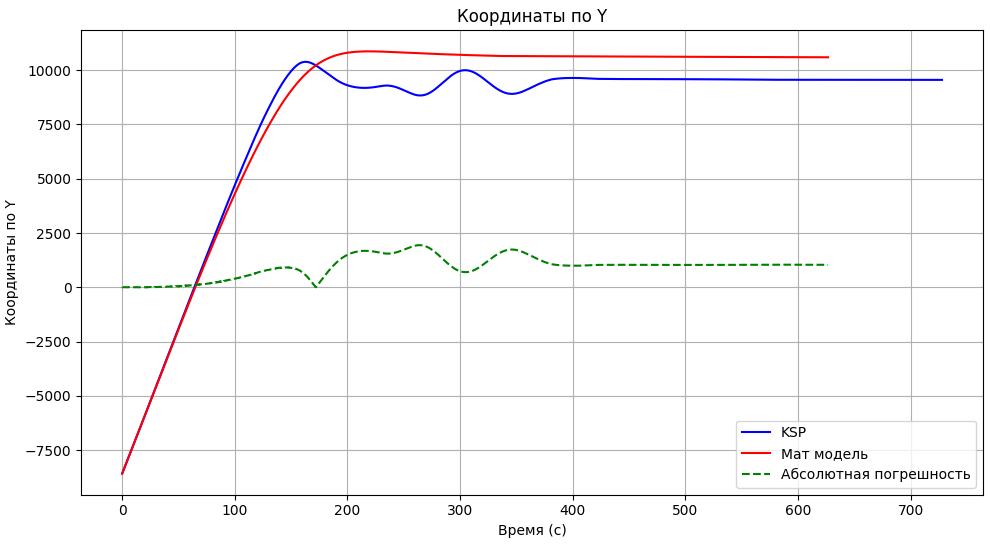


Рисунок 7. График зависимости координаты Y от времени

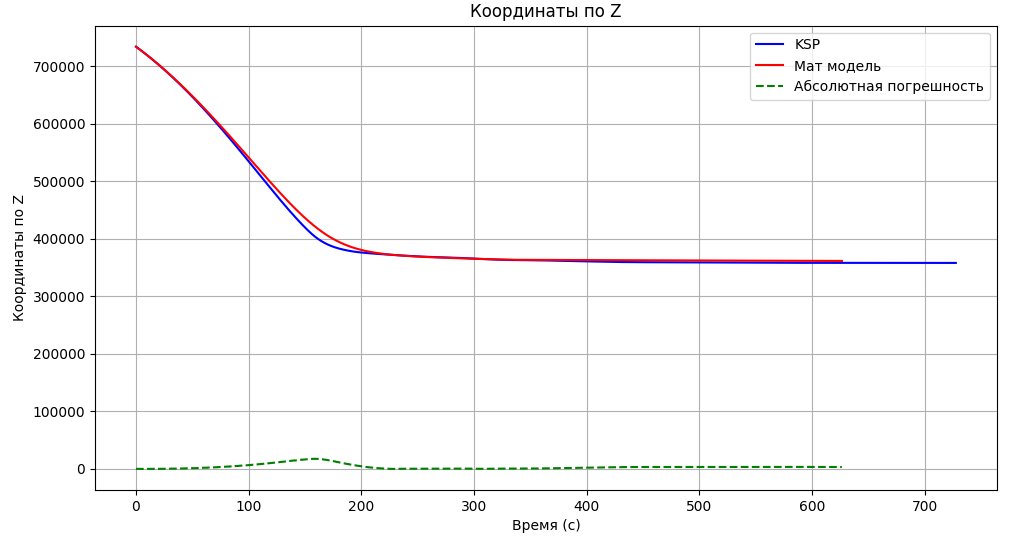


Рисунок 8. График зависимости координаты Z от времени

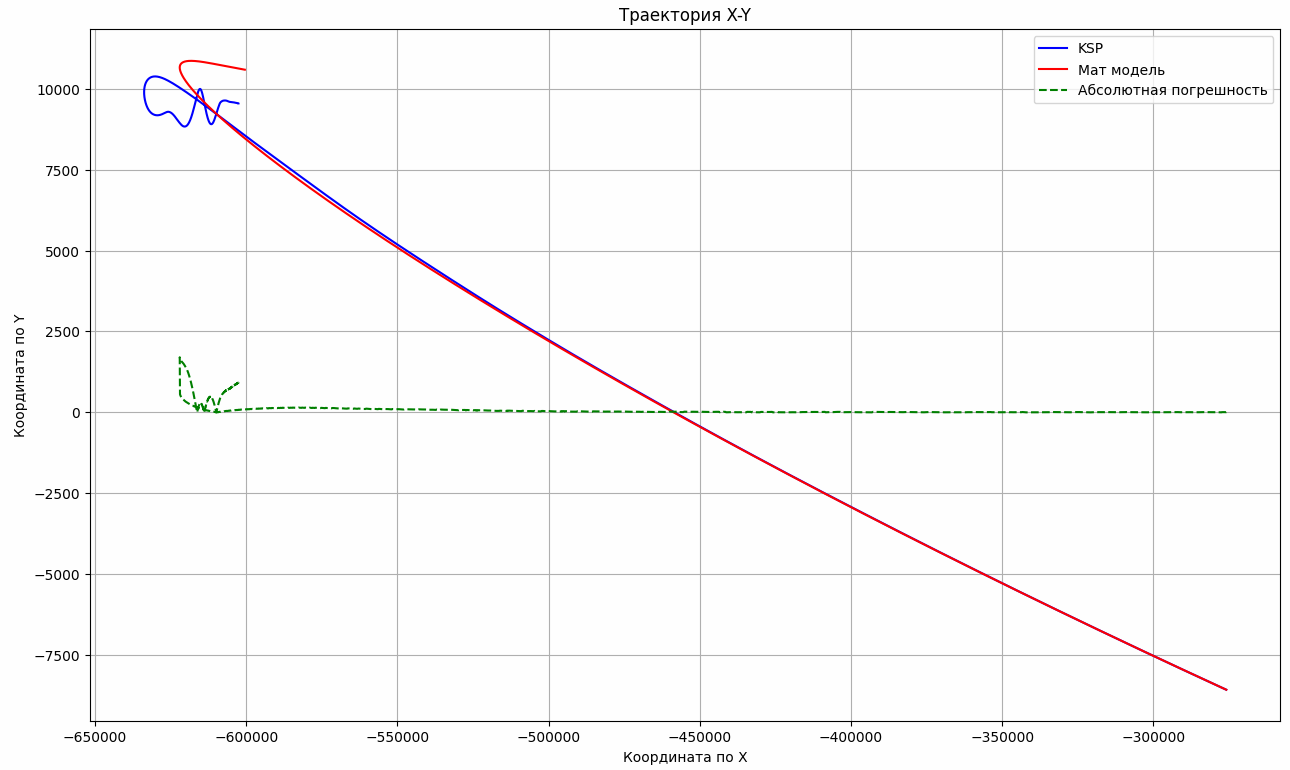


Рисунок 9. График зависимости координаты Y от X

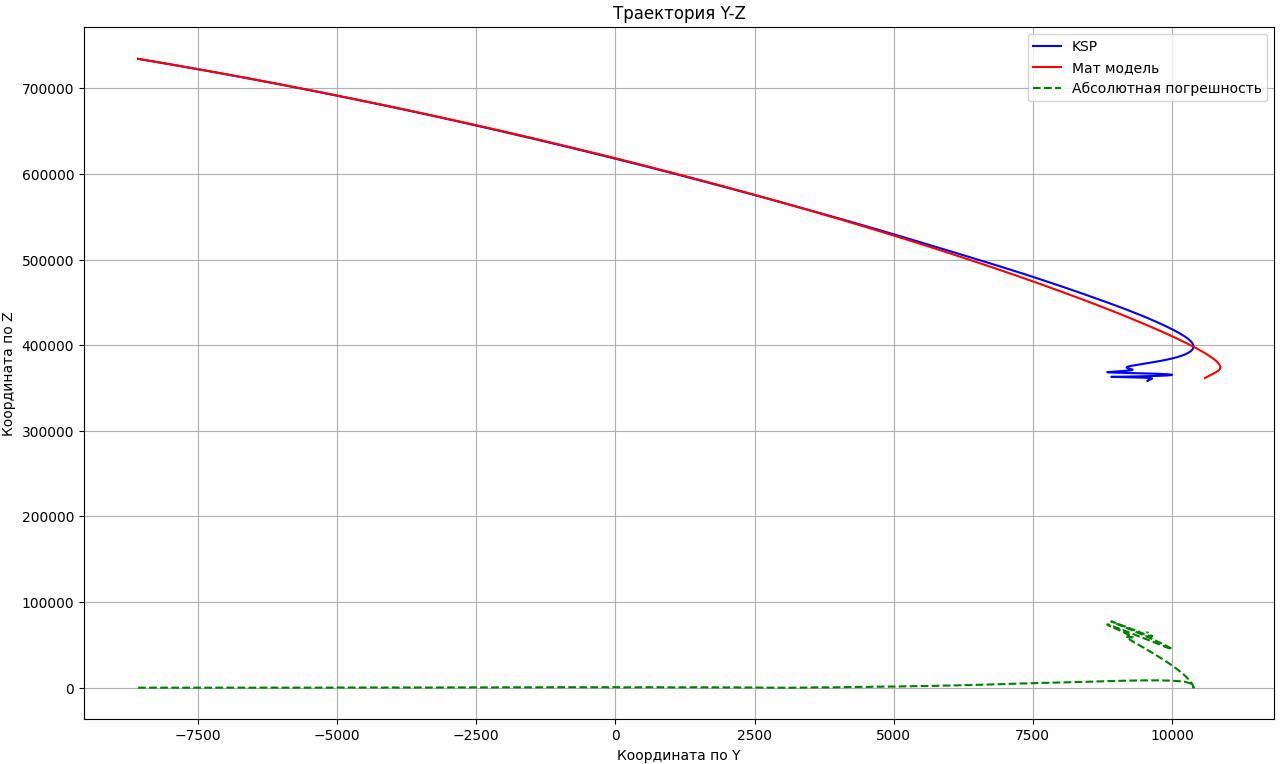


Рисунок 10. График зависимости координаты Z от Y

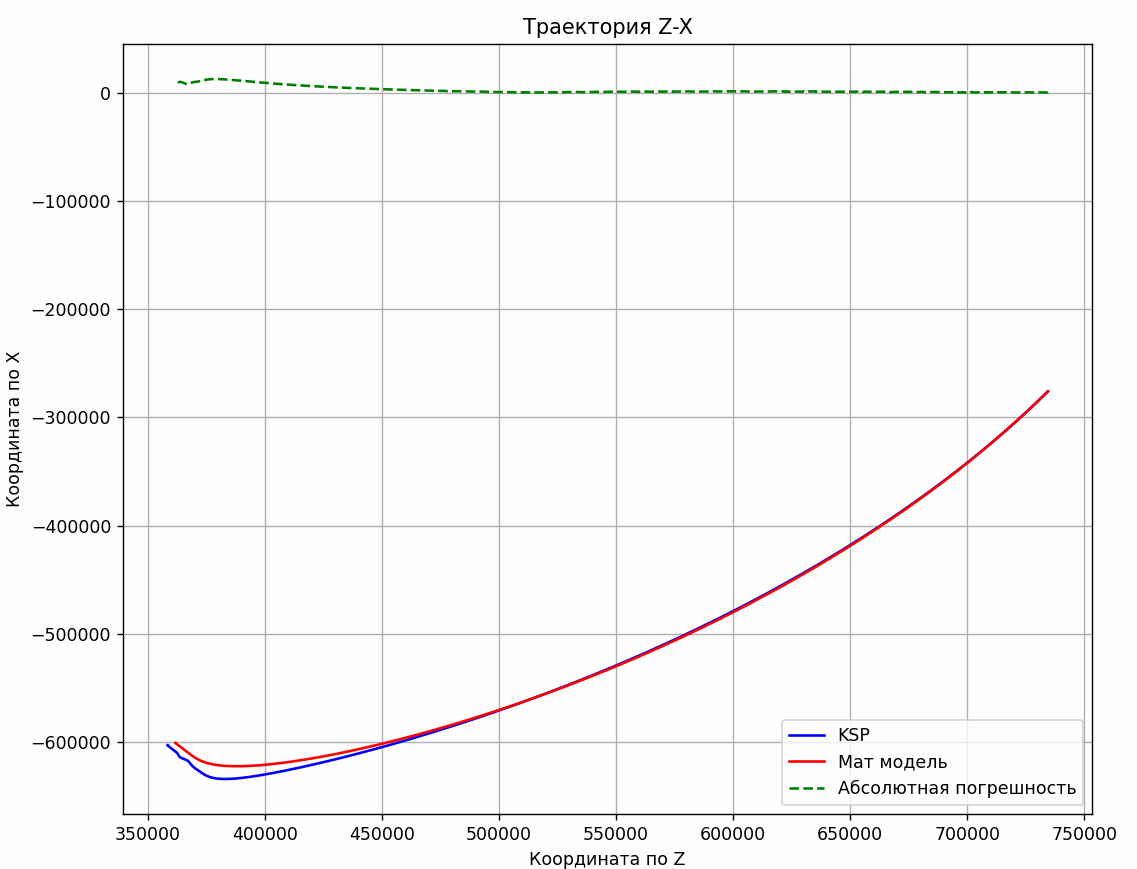


Рисунок 11. График зависимости координаты X от Z

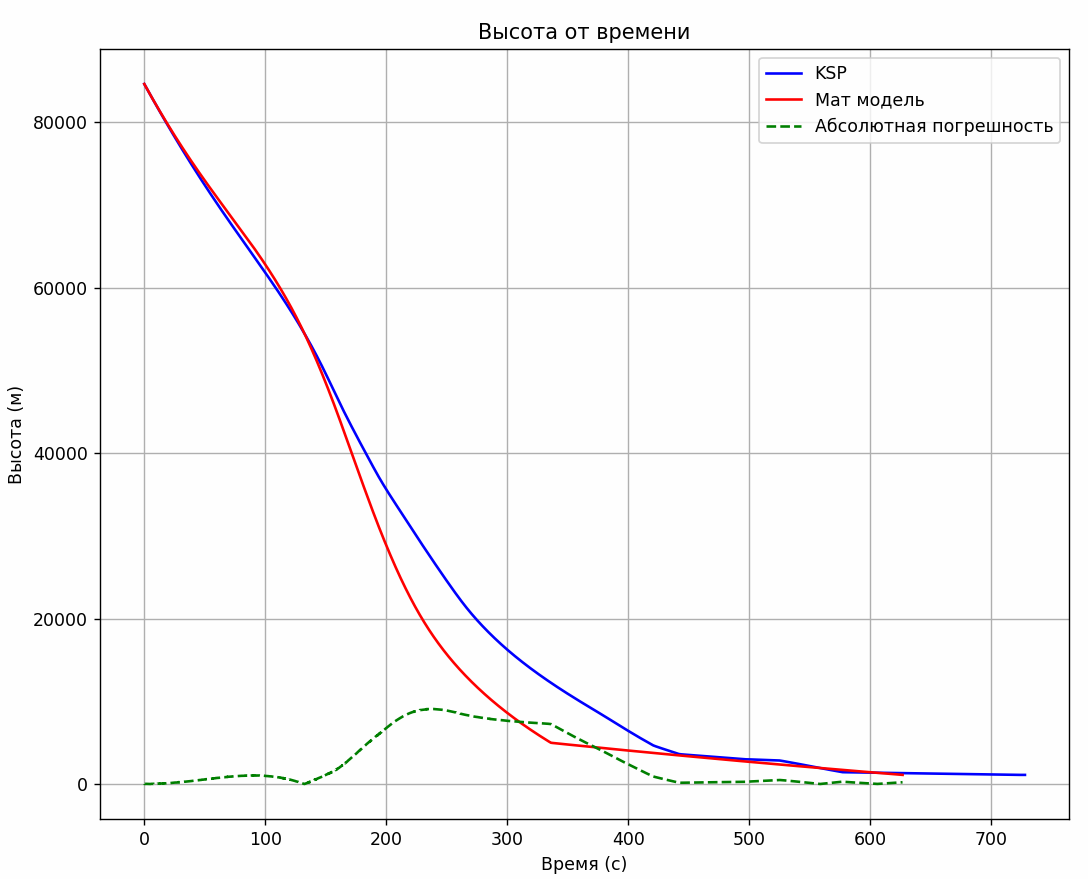


Рисунок 12. График зависимости высоты от времени

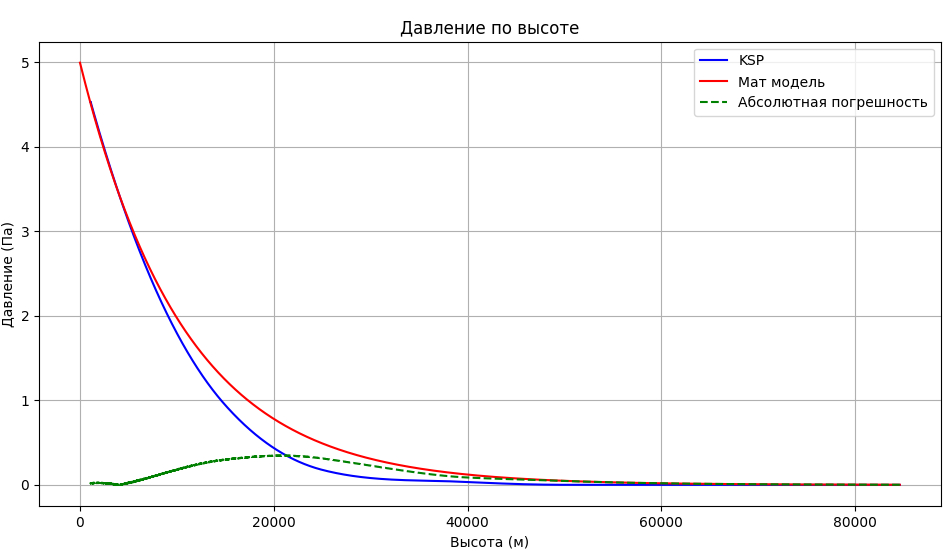


Рисунок 13. График зависимости высоты от времени

1. **Программная реализация**

Для своих программ мы использовали такие библиотеки как:

* krpc (для взаимодействия с ksp с помощью кода)
* numpy (для выполнения математических операций и работы с массивами)
* matplotlib (для построения графиков)

**ksp.py (запись данных в файл)**

Подключение к ksp с помощью krpc:

conn = krpc.connect(name='Тест Ева орбита 1 14\_12 (SANDBOX)')

vessel = conn.space\_center.active\_vessel

Открываем файл для записи, в бесконечном цикле while собираем данные о состоянии корабля:

pressure = vessel.flight().static\_pressure # Получает текущее статическое атмосферное давление на корабль.

altitude = vessel.flight().mean\_altitude # Определяет текущую среднюю высоту корабля над уровнем моря планеты

eva = conn.space\_center.bodies['Eve'] # Получает ссылку на небесное тело

eva\_reference\_frame = eva.reference\_frame # Получает систему координат, привязанную к планете Eve

velocity = vessel.velocity(eva\_reference\_frame) # Получает текущую скорость корабля в системе отсчета планеты Eve

position = vessel.position(eva\_reference\_frame) # Получает текущие координаты корабля относительно планеты Eve в системе отсчета

speed = (velocity[0]\*\*2 + velocity[1]\*\*2 + velocity[2]\*\*2)\*\*0.5 # Вычисляет модуль полной скорости корабля

Записываем данные в файл:

file.write(f"{current\_time - start\_time} {altitude} {velocity[0]} {velocity[1]} {velocity[2]} {speed} {position[0]} {position[1]} {position[2]} {pressure}\n")

* Время (current\_time - start\_time);
* Высота (altitude);
* Компоненты скорости (velocity[0], velocity[1], velocity[2]);
* Модуль скорости (speed);
* Координаты положения корабля (position[0], position[1], position[2]);
* Атмосферное давление (pressure).

Далее выпускаем парашюты на определённых высотах.

**Graphics.py (построение графиков по данным из ksp)**

Открываем файл и извлекаем из него данные:

* t — время, прошедшее с начала записи данных (в секундах);
* h — высота над поверхностью планеты (в метрах);
* vx, vy, vz — компоненты скорости корабля по осям X, Y, Z (в м/с);
* sp — модуль полной скорости корабля (в м/с);
* px, py, pz — положение корабля относительно центра планеты по осям X, Y, Z (в метрах);
* pressure — атмосферное давление в точке корабля (в Па).

Добавляем данные в соответствующие списки.

Функция **F** нужна для построения графиков.

def F(n1, n2, n3, lst\_x, lst\_y, t\_x, t\_y, color, name\_graf):

    plt.subplot(n1, n2, n3)

    plt.plot(lst\_x, lst\_y, color=color)

    plt.xlabel(t\_x)

    plt.ylabel(t\_y)

    plt.grid()

    plt.title(name\_graf)

Аргументы функции:

* n1, n2, n3 — параметры сетки графиков;
* lst\_x, lst\_y — данные для осей X и Y;
* t\_x, t\_y — подписи осей;
* color — цвет линии графика;
* name\_graf — заголовок графика.

Функция **speed\_graf** строит 4 графика: зависимость скорости по осям от времени, полной скорости от времени.

def speed\_graf():

    F(2, 2, 1, times, velocity\_x, "Время (с)", "Скорость по X (м/c)", '#3c88bd', "График зависимости скорости по координате X (м/c) от времени (с)")

    F(2, 2, 2, times, velocity\_y, "Время (с)", "Скорость по Y (м/c)", '#ff7f0e', "График зависимости скорости по координате Y (м/c) от времени (с)")

    F(2, 2, 3, times, velocity\_z, "Время (с)", "Скорость по Z (м/c)", '#2ca02c', "График зависимости скорости по координате Z (м/c) от времени (с)")

    F(2, 2, 4, times, speed, "Время (с)", "Скорость общая (м/c)", 'red', "График зависимости общей скорости (м/c) от времени (с)")

Функция **coords\_graf** строит графики изменения координат корабля относительно времени.

def coords\_graf():

    F(2, 2, 1, times, position\_x, "Время (с)", "Координаты по X", '#3c88bd', "График зависимости координаты по координате X (м) от времени (с)")

    F(2, 2, 2, times, position\_y, "Время (с)", "Координаты по Y", '#ff7f0e', "График зависимости координаты по координате Y (м) от времени (с)")

    F(2, 2, 3, times, position\_z, "Время (с)", "Координаты по Z", '#2ca02c', "График зависимости координаты по координате Z (м) от времени (с)")

Функция **traectory\_graf** строит зависимости Y-X, Z-Y, X-Z.

def traectory\_graf():

    F(2, 2, 1, position\_x, position\_y, "Координата по X", "Координата по Y", 'blue', "График зависимости координаты Y (м) от координаты X (м)")

    F(2, 2, 2, position\_y, position\_z, "Координата по Y", "Координата по Z", 'red', "График зависимости координаты Z (м) от координаты Y (м)")

    F(2, 2, 3, position\_z, position\_x, "Координата по Z", "Координата по X", 'green', "График зависимости координаты X (м) от координаты Z (м)")

Функция **height\_graf** строит 2 графика: высота корабля относительно времени и полной скорости (инвертируем ось X, чтобы высота шла справа налево).

def height\_graf():

    F(2, 2, 1, times, height, "Время (с)", "Высота (м)", 'green', "График зависимости высоты (м) от времени (с)")

    F(2, 2, 2, speed, height, "Скорость (м/c)", "Высота (м)", 'green', "График зависимости высоты (м) от скорости (м/c)")

    plt.gca().invert\_xaxis()

Функция **pressure\_height\_graf** строит график зависимости давления от высоты.

def pressure\_height\_graf():

    F(2, 2, 3, height, Pressure, "Высота (м)", "Давление (Па)", 'purple', "График зависимости давления (Па) от высоты (м)")

Функция **graf\_3D** создает трехмерный график, отображающий траекторию корабля в пространстве.

def graf\_3D(position\_x, position\_y, position\_z):

    fig = plt.figure(figsize=(7, 4))

    ax\_3d = fig.add\_subplot(111, projection='3d')

    # Построение точек

    ax\_3d.scatter(position\_x, position\_y, position\_z, color='blue')

    # Добавляем подписи к осям

    ax\_3d.set\_xlabel('X', color='red')

    ax\_3d.set\_ylabel('Y', color='green')

    ax\_3d.set\_zlabel('Z', color='blue')

    # Заголовок графика

    ax\_3d.set\_title('Траектория спуска')

    # Убираем вывод координат точек в окошке

    ax\_3d.grid(True)

    ax\_3d.view\_init(elev=30, azim=30)

Далее выводим наборы графиков в отдельных окнах.

**Modeling of a mathematical model to file mathdata.py (запись данных математической модели и построение по ним графиков)**

Задаем константные значения и высчитываем некоторые значения.

# Константы планеты

M = 1.224398e23  # Масса планеты (Ева), кг

G = 6.672e-11  # Гравитационная постоянная, м^3/(кг·с^2)

R = 700000  # Радиус планеты, м

Mu = 8.1717302 \* 10 \*\* 12 # гравитационный параметр планеты ('м3/с2')

# Атмосферные параметры

P0 = 506625  # Давление у поверхности, Па

H = 7921  # Высота масштабирования атмосферы, м

# H = 10779.053

T = 401  # Температура атмосферы, К

R\_specific=8.314462618153 #Дж/(моль\*K)

g0=Mu/R/R#ускорение свободного падения у поверхности модуль

print("g0 = ",g0)

mmol=R\_specific\*T/(g0\*H)#молярная масса атмосферы у поверхности

print("mmol = ",mmol)

rro0=P0\*mmol/R\_specific/T

print("rro0 = ",rro0);#плотсноть атмосферы у поверхности kg/m^3

# Параметры аппарата

m = 6950  # Масса аппарата, кг

A = 4.8  # Площадь поперечного сечения, м^2

Cd = 1.2  # Коэффициент аэродинамического сопротивления

Далее считаем силы.

def mass(h):

    if h > 5000:

        return 6950

    elif h <= 5000:

        return 5300

    elif h <= 3000:

        return 2261

    else:

        return 2261

def pressure(h):

    """Давление в зависимости от высоты."""

    return P0 \* np.exp(-h / H)

def rho(h):

    """Плотность атмосферы в зависимости от высоты."""

    return rro0\*np.exp(-h / H)

def drag\_force(V, h):

    Cd = 3.2

    A = 4.5

    if h <= 5000:

        Cd = 3  #Mk25

        A = 125

    elif h <= 4000:

        Cd = 3  #Mk16

        A = 50

    elif h <= 3000:

        Cd = 3  #Mk12-R + Mk16

        A = 33 + 50

    elif h <= 2500:

        Cd = 3  #Mk12-R

        A = 33

    elif h <= 1500:

        Cd = 3  #Mk12-R + три Mk2-R

        A = 32 \* 3 + 700

    """Сила аэродинамического сопротивления."""

    return 0.5 \* Cd \* A \* rho(h) \* V

Функция **equations** описывает систему дифференциальных уравнений аппарата.

def equations(t, state):

    """Уравнения движения аппарата."""

    x, y, z, Vx, Vy, Vz = state

    r = np.sqrt(x \*\* 2 + y \*\* 2 + z \*\* 2)

    # Текущая высота и скорость

    h = r - R

    m = mass(h)

    if(h < 1101):

        return

    V = np.sqrt(Vx \*\* 2 + Vy \*\* 2 + Vz \*\* 2)  # модуль скорости

    # Силы

    #Fg = g(h) \* m  # Сила тяжести

    Fd = drag\_force(V, h)  # Сила сопротивления

    ax = (-Mu / (r \*\* 3)) \* x - (Fd \* Vx) / m

    ay = (-Mu / (r \*\* 3)) \* y - (Fd \* Vy) / m

    az = (-Mu / (r \*\* 3)) \* z - (Fd \* Vz) / m

# поДЪЁМНАЯ И БОКОВАЯ СИЛЫ

    # Дифференциальные уравнения

    dx\_dt = Vx

    dy\_dt = Vy

    dz\_dt = Vz

    dVx\_dt = ax

    dVy\_dt = ay

    dVz\_dt = az

    return [dx\_dt, dy\_dt, dz\_dt, dVx\_dt, dVy\_dt, dVz\_dt]

Дальше задаем начальные условия, решаем систему дифференциальных уравнений, сохраняем данные и записываем в файл. Потом строим графики с помощью функции **F**, которая была описана раннее.

# Силы

    #Fg = g(h) \* m  # Сила тяжести

    Fd = drag\_force(V, h)  # Сила сопротивления

    #Skalr\_r\_V = x \* Vx + y \* Vy + z \* Vz

    # Компоненты ускорений

    ax = (-Mu / (r \*\* 3)) \* x - (Fd \* Vx) / m

    ay = (-Mu / (r \*\* 3)) \* y - (Fd \* Vy) / m

    az = (-Mu / (r \*\* 3)) \* z - (Fd \* Vz) / m

# поДЪЁМНАЯ И БОКОВАЯ СИЛЫ

    # Дифференциальные уравнения

    dx\_dt = Vx

    dy\_dt = Vy

    dz\_dt = Vz

    dVx\_dt = ax

    dVy\_dt = ay

    dVz\_dt = az

    return [dx\_dt, dy\_dt, dz\_dt, dVx\_dt, dVy\_dt, dVz\_dt]

**grafiki.py (построение графика погрешности)**

Открываем файл и записываем в списки данные.

Функция **return\_pogr** вычисялет погрешность между двумя наборами данных.

Аргументы функций:

* x - ключ, указывающий, по каким значениям (обычно ось времени или высоты) нужно сравнивать два набора данных.
* y - ключ, указывающий, какие данные нужно сравнивать (например, давление или скорость)

Все функции идентичны, но они строят графики по данным, взятым из ksp, полученными с помощью математической модели и график погрешности.

1. **Симуляция**
2. **Медиа**
3. **Деятельность участников команды**

**Пятницкий Артём**

**Демидов Георгий**

**Рубан Кирилл**

**Цицкиев Дени**

**Заключение**

**Список источников**

1. <https://en.wikipedia.org/wiki/Vega_program>
2. <https://www.laspace.ru/ru/activities/projects/vega_1_2/>
3. https://epizodyspace.ru/bibl/vega/01.html