МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
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,

Тимохин Максим Юрьевич

_____ Тимохин М.Ю.

_____ 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

1. 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.

2) **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.

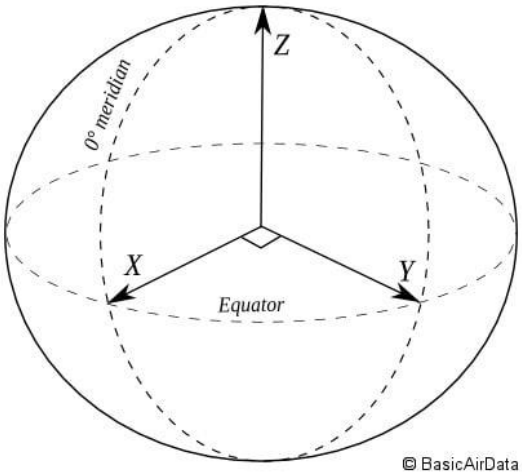
После выхода на межпланетную траекторию аппараты начали шестимесячный перелет к Венере, используя свои двигательные установки для точной корректировки траектории. В июне 1985 года, спустя около шести месяцев после запуска, оба аппарата достигли окрестностей Венеры. На подлёте к планете, на расстоянии 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.

2. Физическая модель

Рассматривается атмосферное движение спускаемого космического аппарата до приземления.

В качестве системы координат используется планетоцентрическая прямоугольная система координат $Oxyz$ с началом в центре планеты, принимаемая за инерциальную систему.



Система координат

Введем ряд допущений:

1) Все силы, действующие на КА, приложены к его центру масс;
2) Спуск аппарата происходит под действием только гравитационной силы F_{grav} и силы аэродинамического сопротивления F_c;
3) Ева– шар с радиусом R_{Eve} с равномерно распределенной плотностью;
4) Ускорение, обусловленное вращением Евы не велико и поэтому центробежной и Кориолисовой силами можно пренебречь;
5) Суммарной силой притяжения Солнца и планет можно пренебречь;
6) Атмосфера планеты не вращается;
7) Атмосфера планеты изотермическая;

Таким образом, физическая модель будет выглядеть так:

По второму закону Ньютона:

$$m\vec{a} = \sum_{i=1}^{n} F_i$$

Где:

m – масса спускаемого аппарата

a – ускорение спускаемого аппарата

F_i - силы, действующие на аппарат, а именно:

$\overrightarrow{F_{grav}} = m\overrightarrow{a_{grav}}$ - Гравитационная сила.

$\overrightarrow{a_{grav}} = \begin{cases} a_x = -\frac{\mu}{r^3}x \\ a_y = -\frac{\mu}{r^3}y \\ a_z = -\frac{\mu}{r^3}z \end{cases}$ - разложение гравитационного ускорения

μ = 8.1717302 * 10^{12} $м^3/с^2$ – гравитационный параметр планеты Ева

$r = R_{Eve} + h$ - расстояние от центра Венеры до КА

$$r = \sqrt{x^2 + y^2 + z^2}$$

$\vec{F_c} = \frac{C_d S \rho \vec{v}^2}{2}$ - Сила аэродинамического сопротивления

Где,

C_d - коэффициент аэродинамического сопротивления

S – площадь миделя (наибольшее по площади поперечное сечение аппарата)

ρ – плотность атмосферы, меняется по динамическому закону:

$$\rho = \rho_0 e^{-\frac{h}{H}}$$

P – плотность атмосферы, меняется по динамическому закону:

$$P = P_0 e^{-\frac{h}{H}}$$

$\vec{v}$ - скорость аппарата

$G \approx 6.67*10^{-11}$– гравитационная постоянная

$M = 1.224398 * 10^{23}$–масса планеты Ева

$R_{Eve} = 700\ 000$м - радиус планеты Ева

h - высота, на которой находится аппарат (от уровня моря)

$R = 287.052874$ Дж/(кг*м^3)– удельная газовая постоянная

$P_0 = 5$ атм – давление на уровне моря

$\rho_0 = 3.8352057$ кг/м^3

$H = 7921$м – Характеристическая высота

Разложим силы по каждой из осей поделим всё на массу аппарата:

$$\begin{cases} a_x = -\frac{\mu}{r^3}x - \frac{C_d S}{m}\frac{\rho v v_x}{2} \\ a_y = -\frac{\mu}{r^3}y - \frac{C_d S}{m}\frac{\rho v v_y}{2} \\ a_z = -\frac{\mu}{r^3}z - \frac{C_d S}{m}\frac{\rho v v_z}{2} \end{cases}$$

Таким образом, мы имеем формулу для расчёта ускорений по каждой из осей координат, с помощью которой мы можем перейти к моделированию математической модели.

3. Математическая модель

При приведённых допущениях уравнения движения спускаемого аппарата получим систему дифференциальных уравнений:

$$\begin{cases} \dot{\vec{r}} = \vec{v} \\ \dot{\vec{v}} = \vec{a} \end{cases} \quad (1)$$

$$\begin{cases} \dot{\vec{r}} = \vec{v} \\ \dot{\vec{v}} = -\frac{\mu}{r^3}\vec{r} + \frac{\overrightarrow{F_c}}{m} \end{cases} \quad (2)$$

$$\begin{cases} \dot{x} = v_x \\ \dot{y} = v_y \\ \dot{z} = v_z \\ \dot{v_x} = -\frac{\mu}{r^3}x - \frac{C_d S}{m}\frac{\rho v v_x}{2} \\ \dot{v_y} = -\frac{\mu}{r^3}y - \frac{C_d S}{m}\frac{\rho v v_y}{2} \\ \dot{v_z} = -\frac{\mu}{r^3}z - \frac{C_d S}{m}\frac{\rho v v_z}{2} \end{cases} \quad (3)$$

Для решения данной системы воспользуемся языком программирования Python и библиотеками

- SciPy
- NumPy

В качестве метода интегрирования будем использовать метод интегрироыания Рунге — Кутты 4 порядка.

Графики KSP

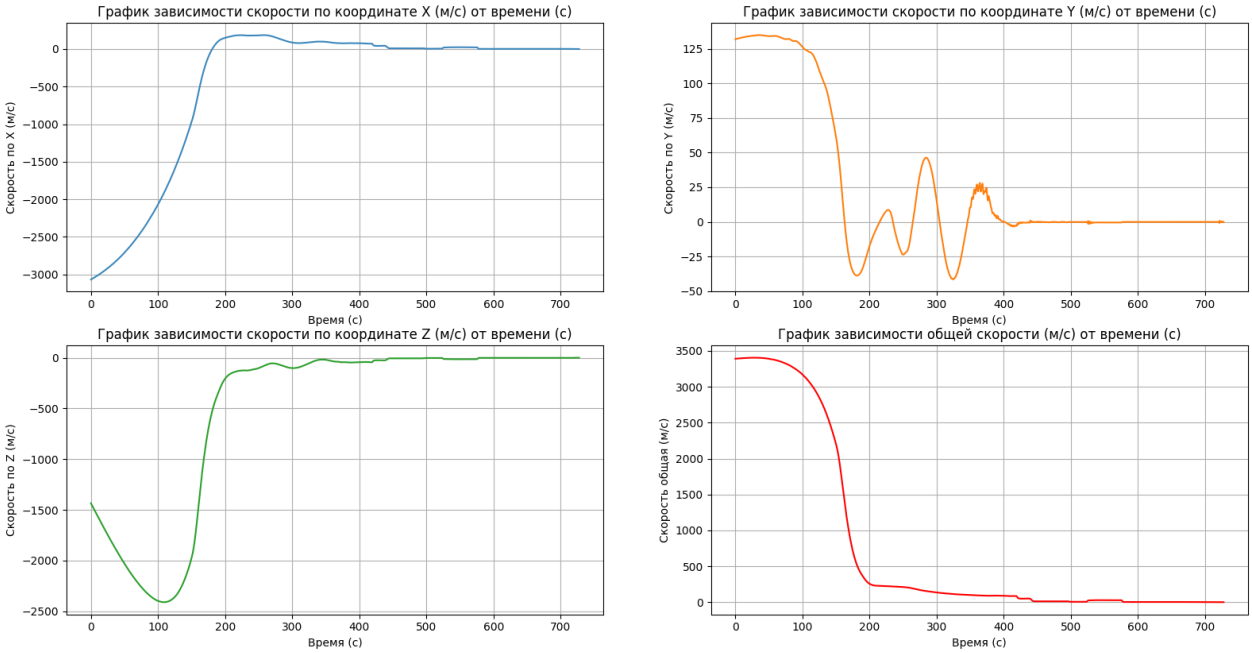


Рисунок 2. Графики зависимостей скорости координат от времени

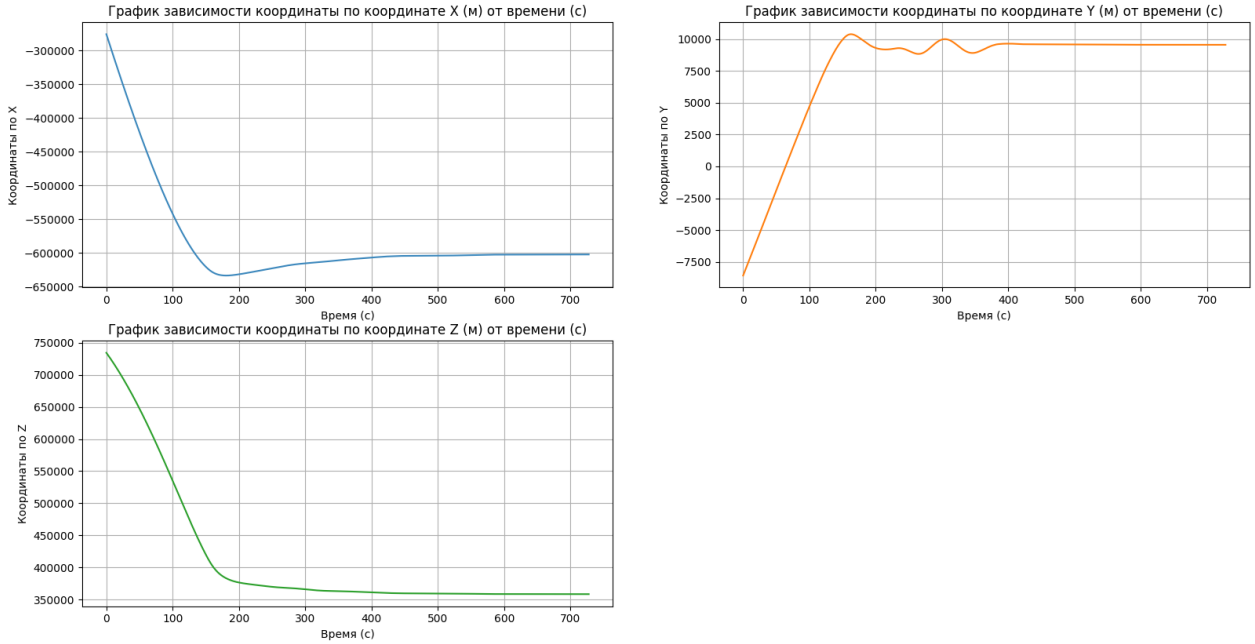


Рисунок 3. Графики зависимостей координат от времени

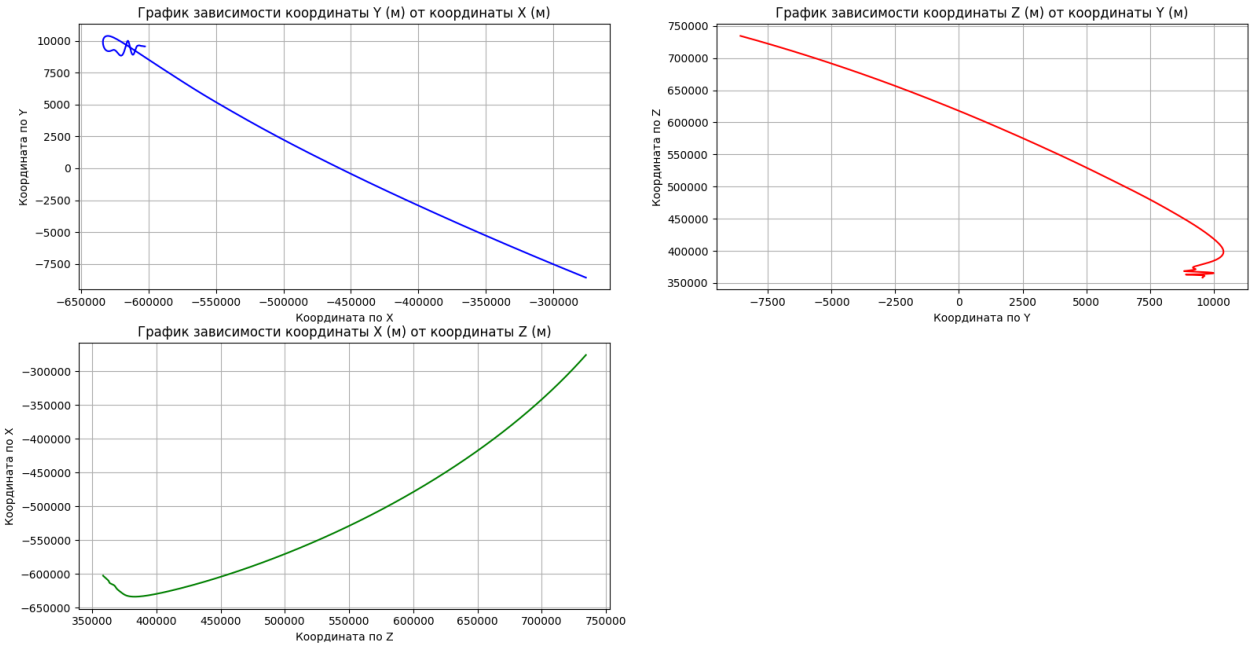


Рисунок 4. Графики траекторий в плоскостях

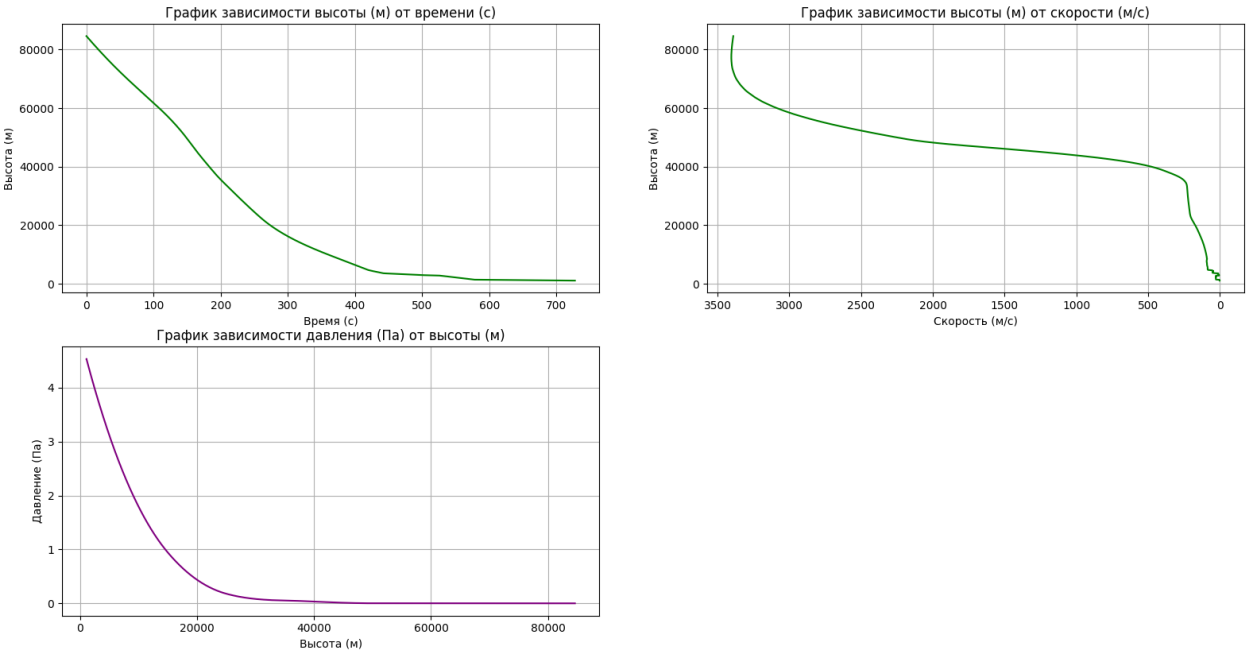


Рисунок 5. Графики зависимостей от высоты

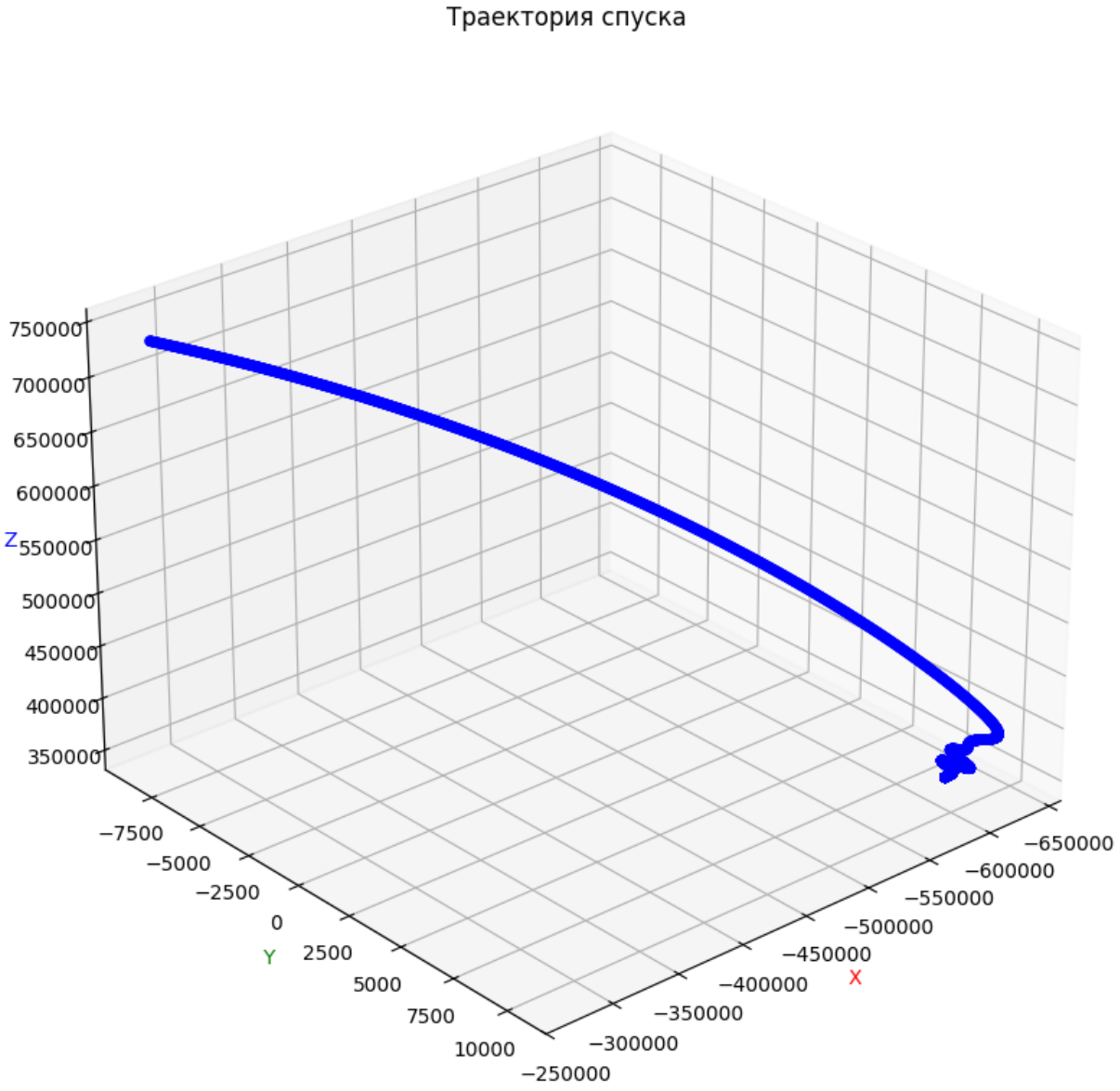


Рисунок 6. Траектории спуска

Графики математической модели

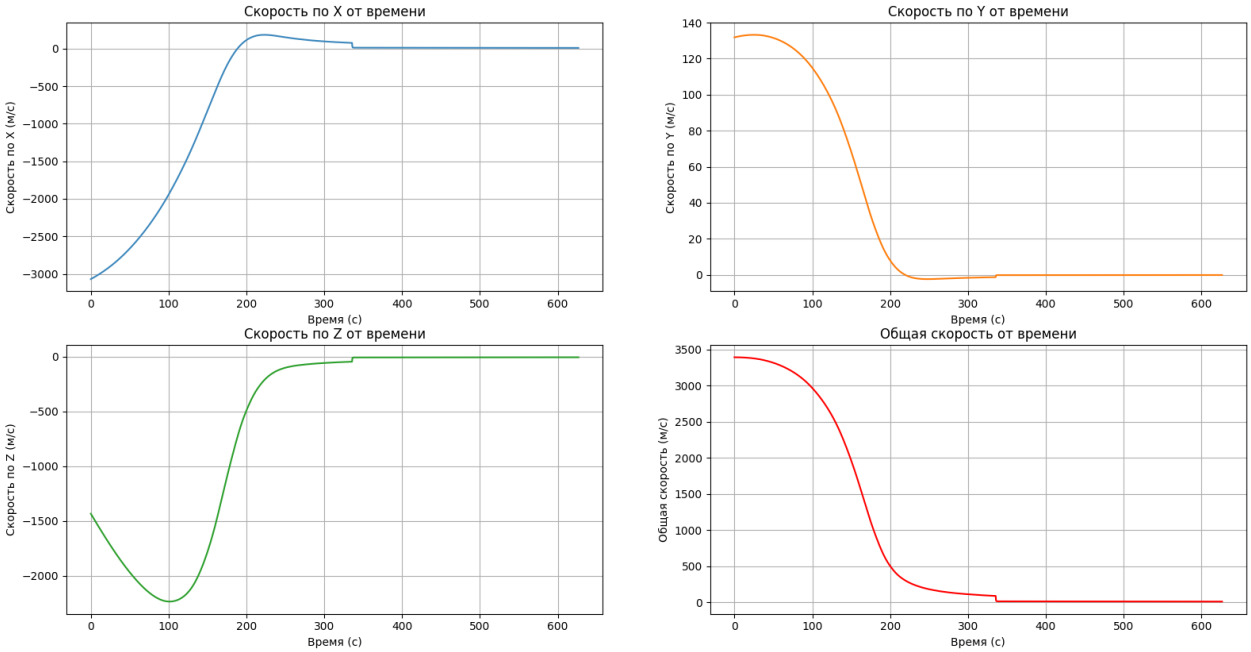


Рисунок 7. Графики зависимостей скорости координат от времени

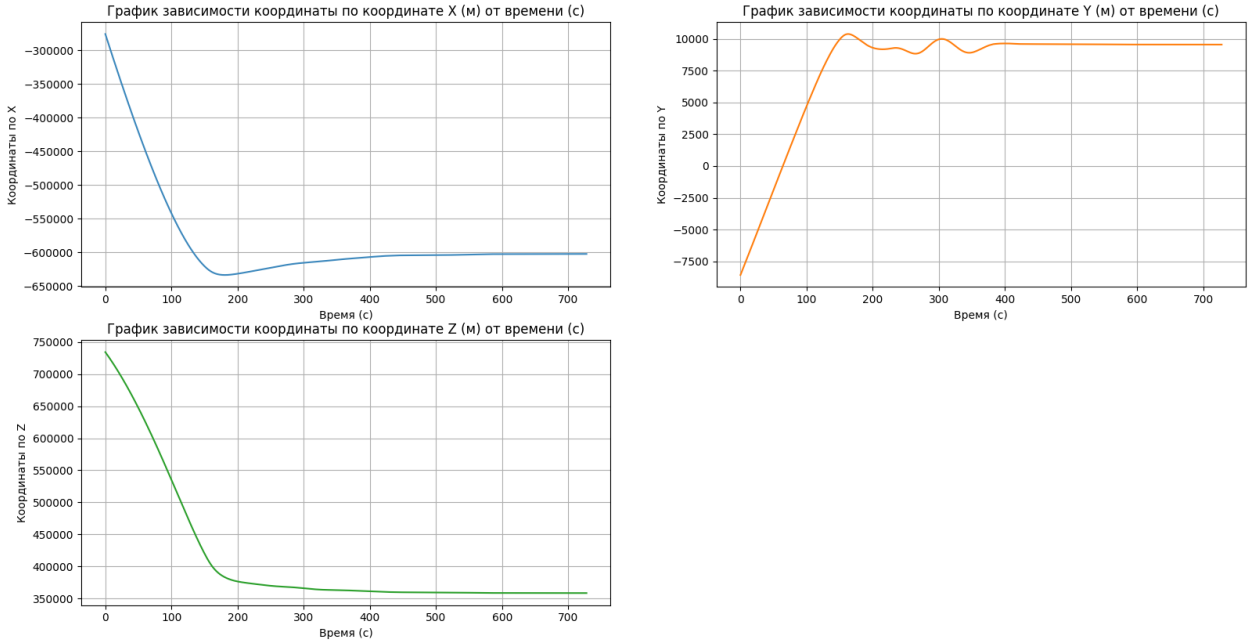


Рисунок 8. Графики зависимостей координат от времени

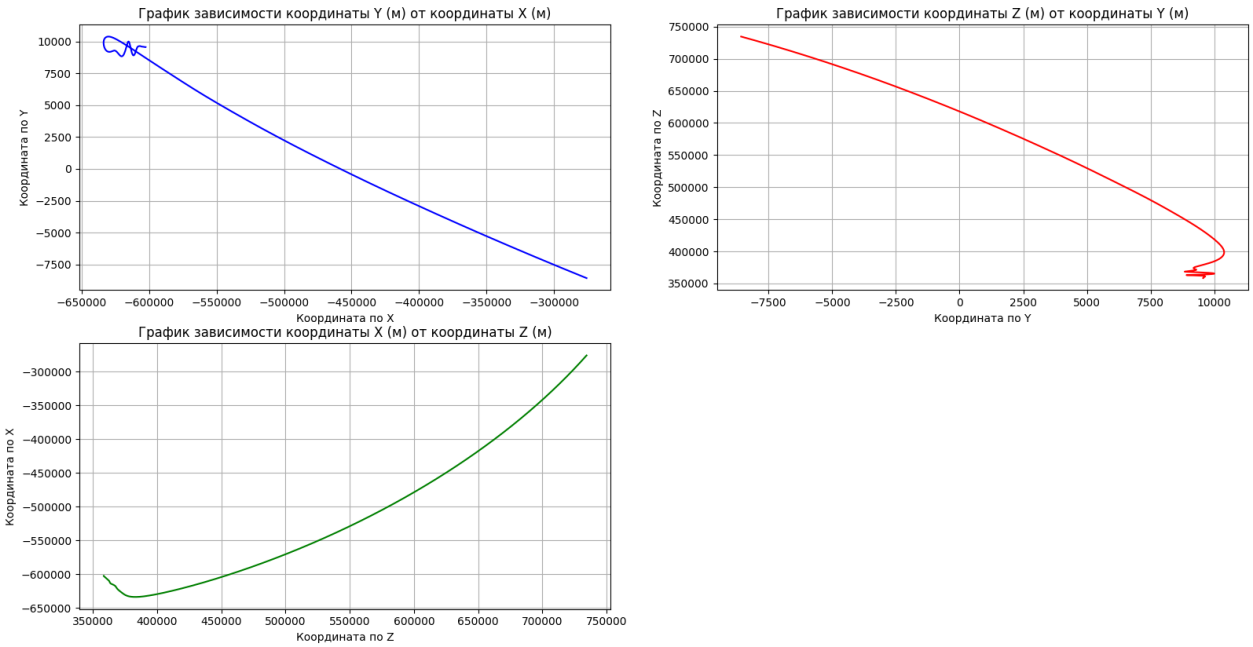


Рисунок 9. Графики траекторий в плоскостях

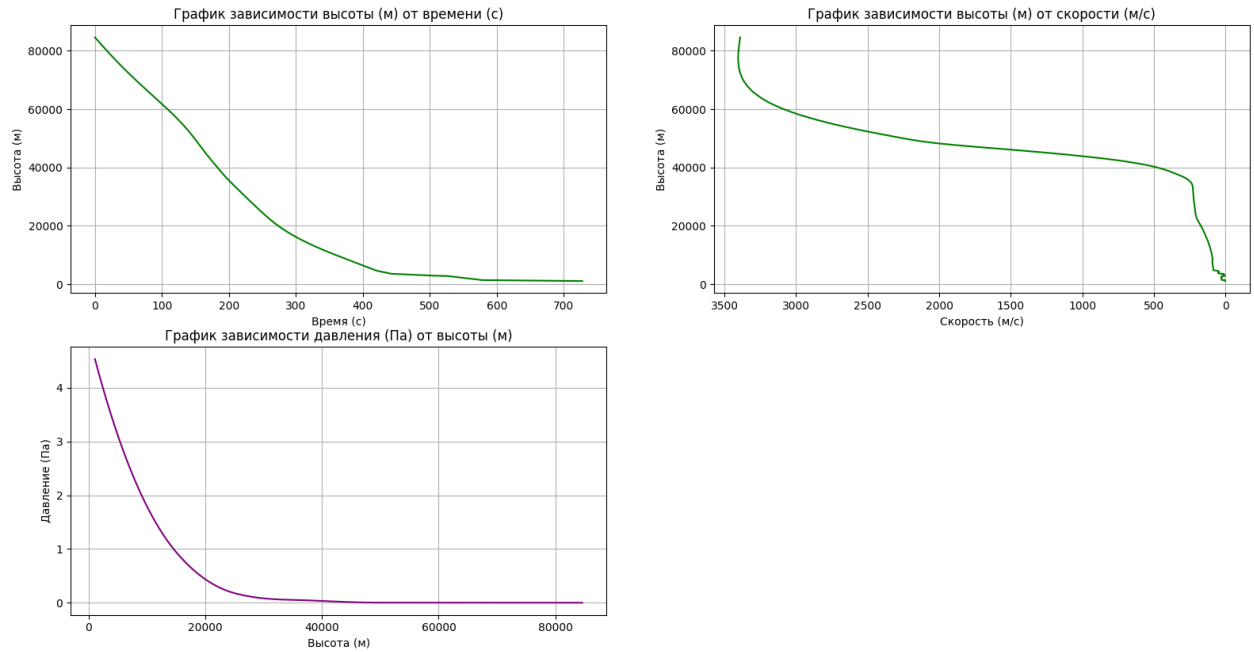


Рисунок 10. Графики зависимостей от высоты

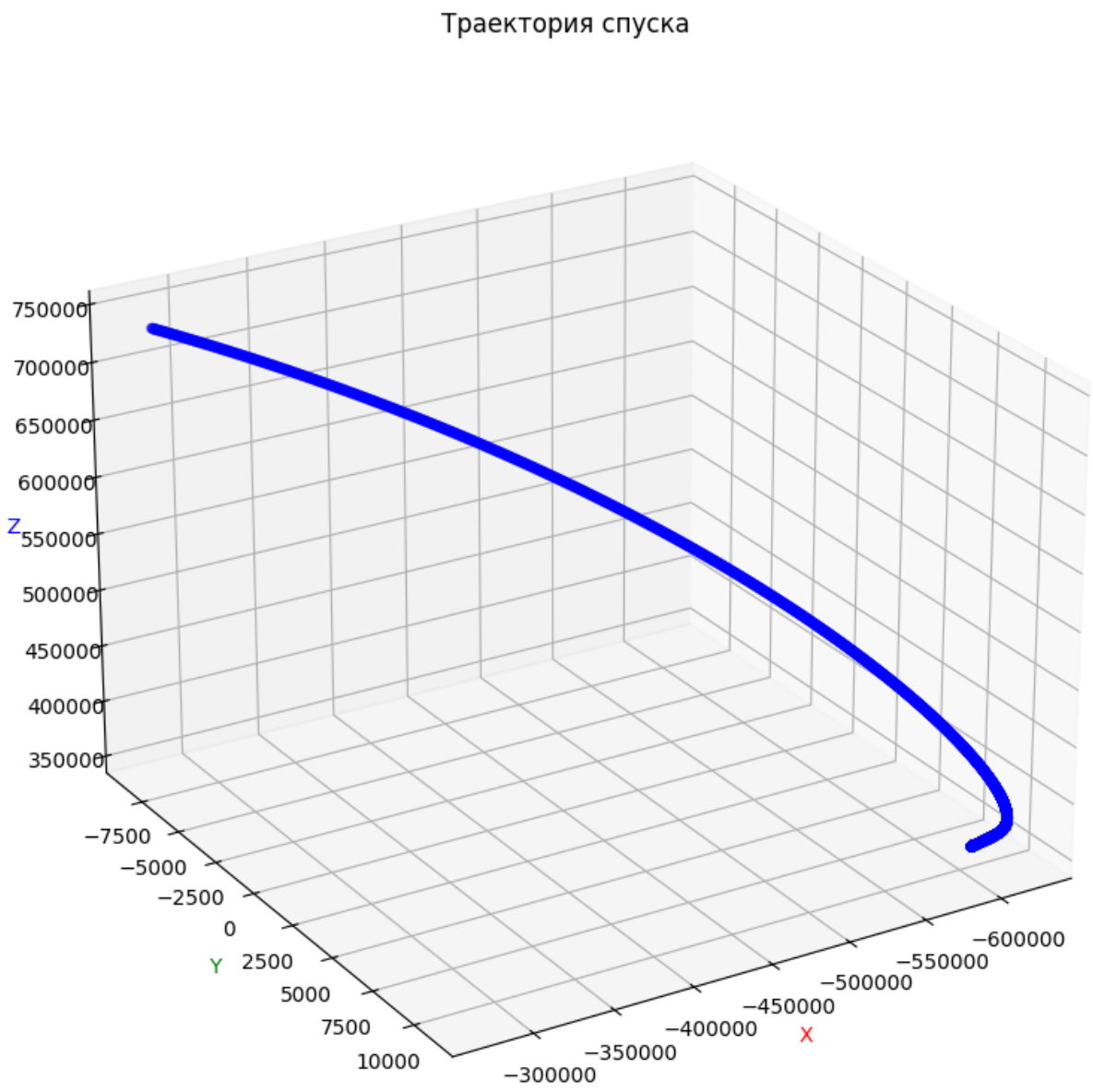


Рисунок 11. Траектория спуска

Графики с погрешностью

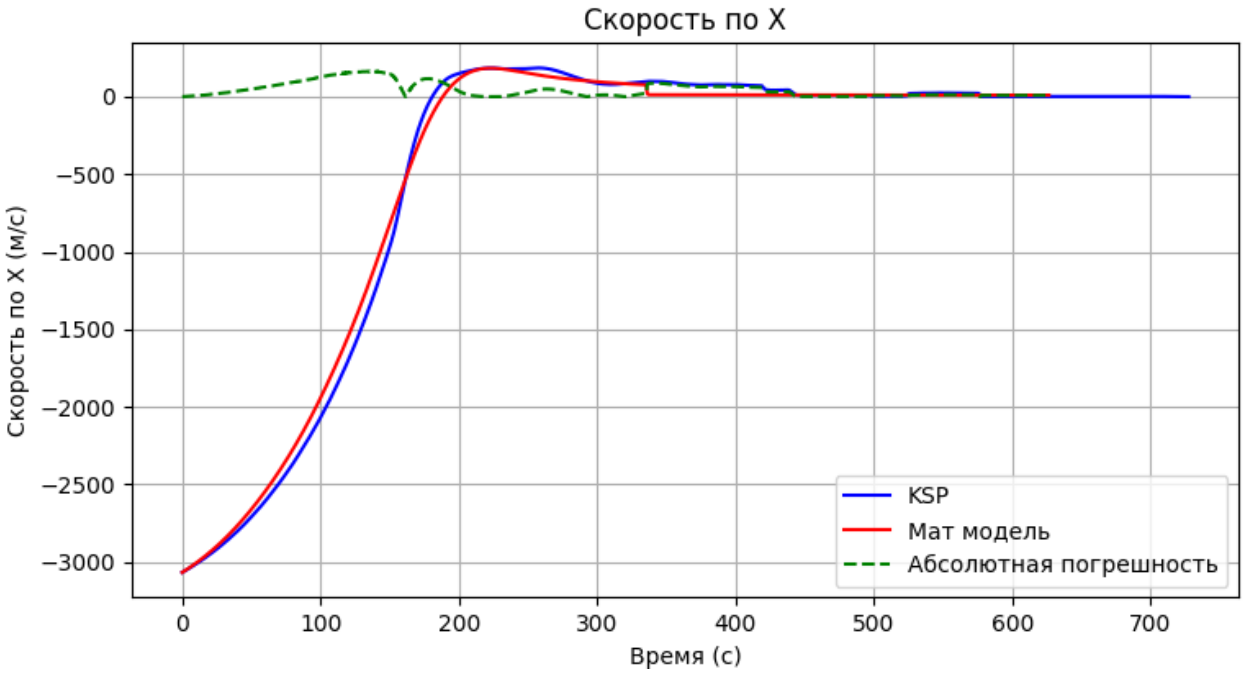


Рисунок 12. График зависимости скорости по координате X от времени

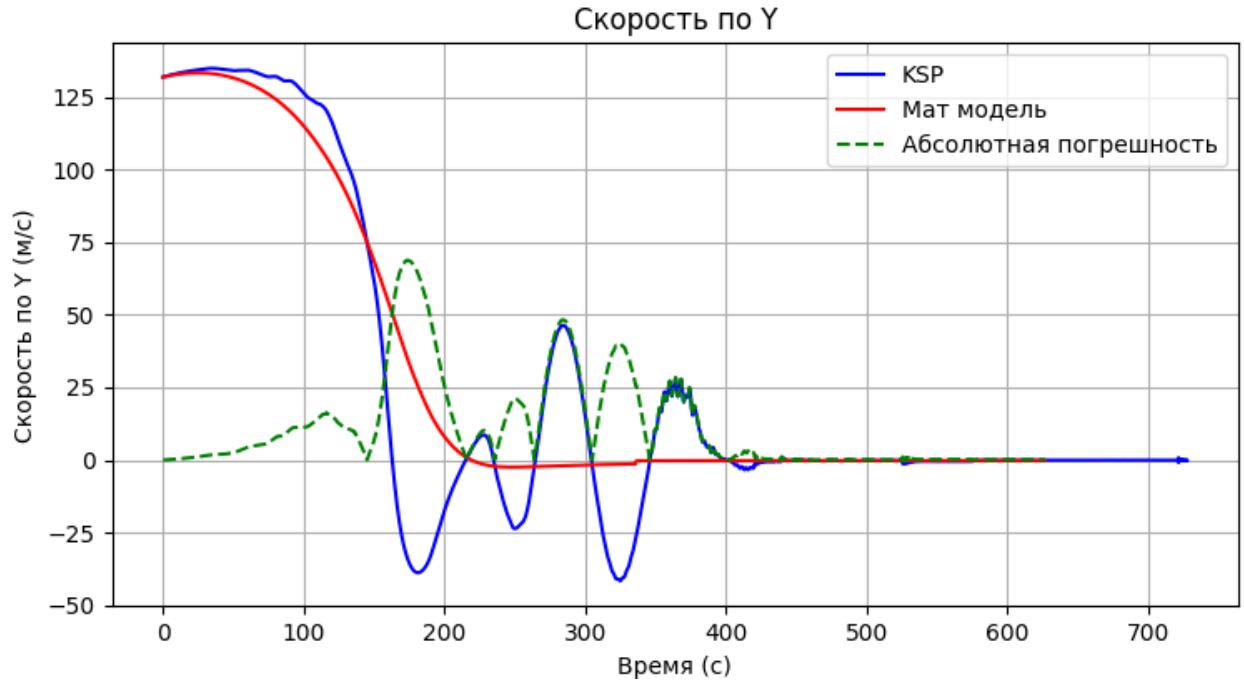


Рисунок 13. График зависимости скорости по координате Y от времени

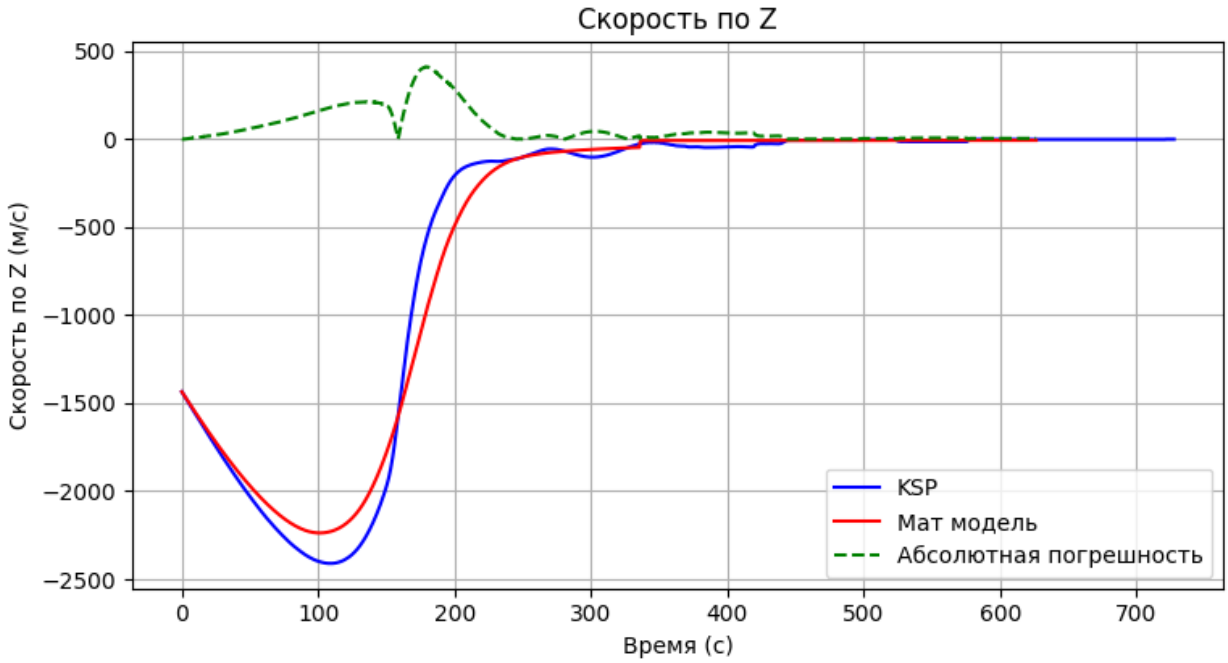


Рисунок 14. График зависимости скорости по Z от времени

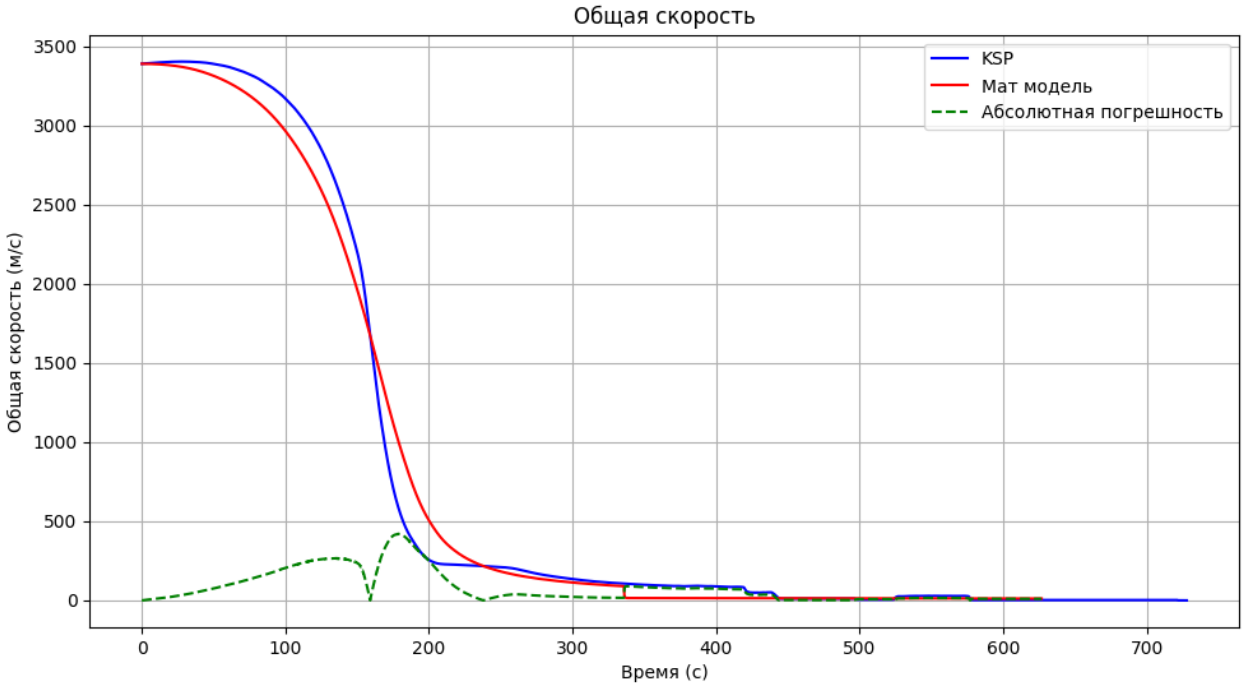


Рисунок 15. График зависимости общей скорости от времени

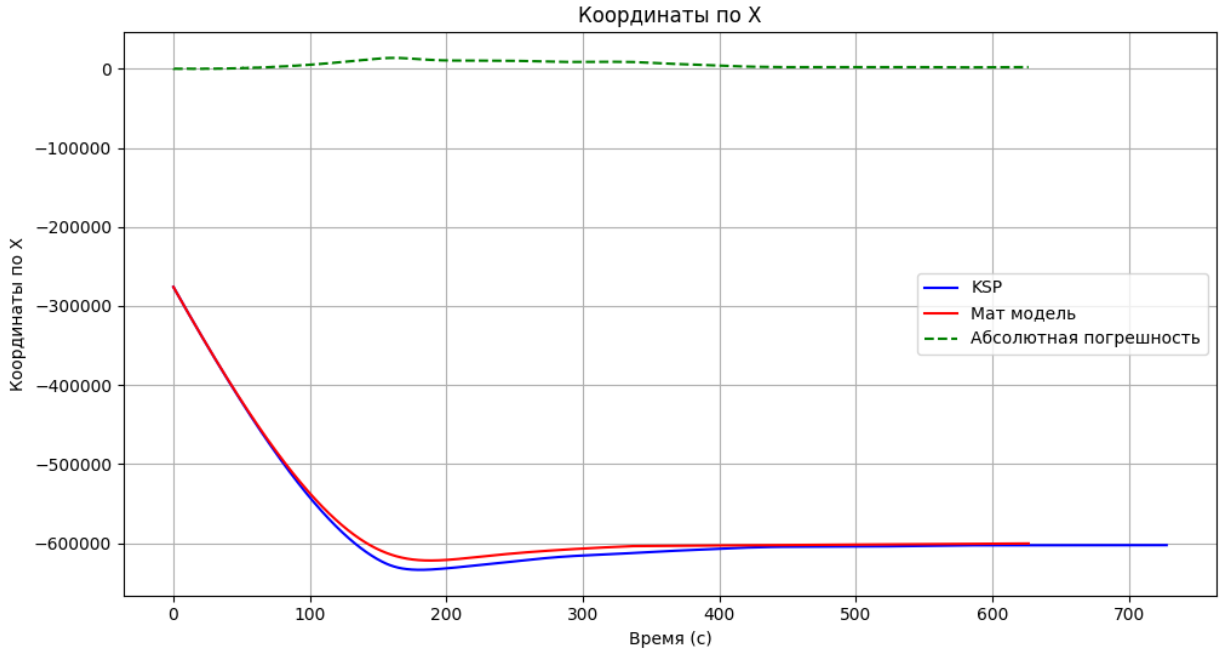


Рисунок 16. График зависимости координаты X от времени

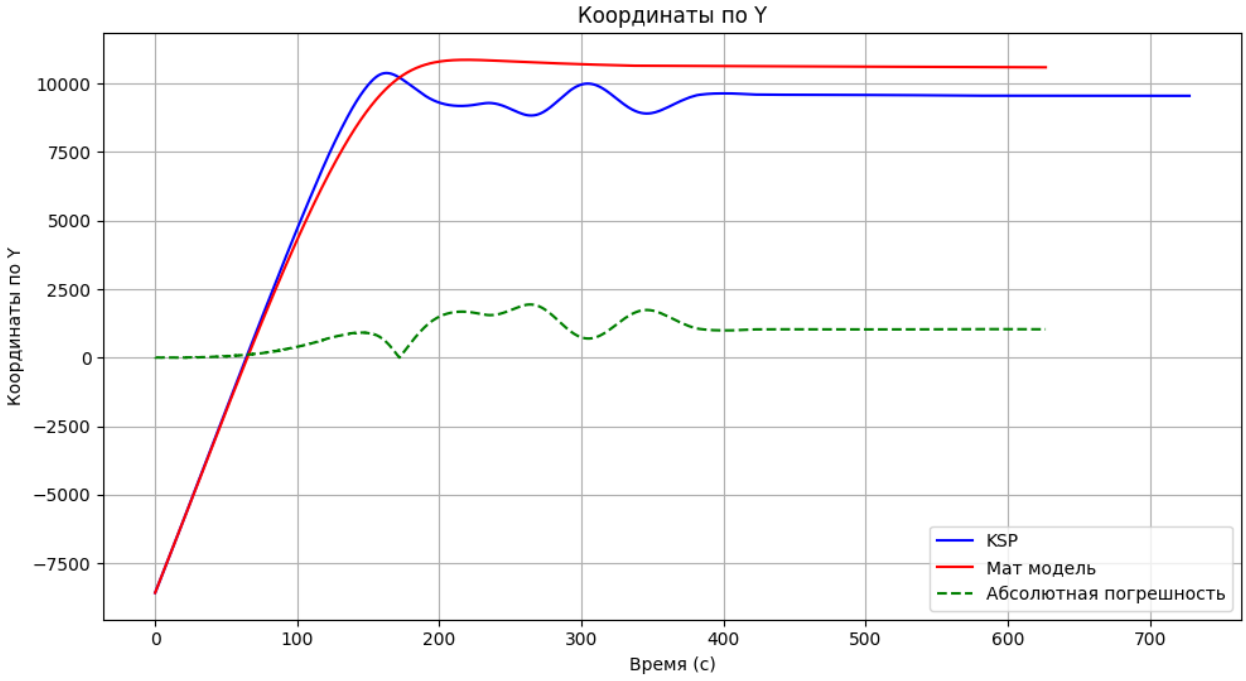


Рисунок 17. График зависимости координаты Y от времени

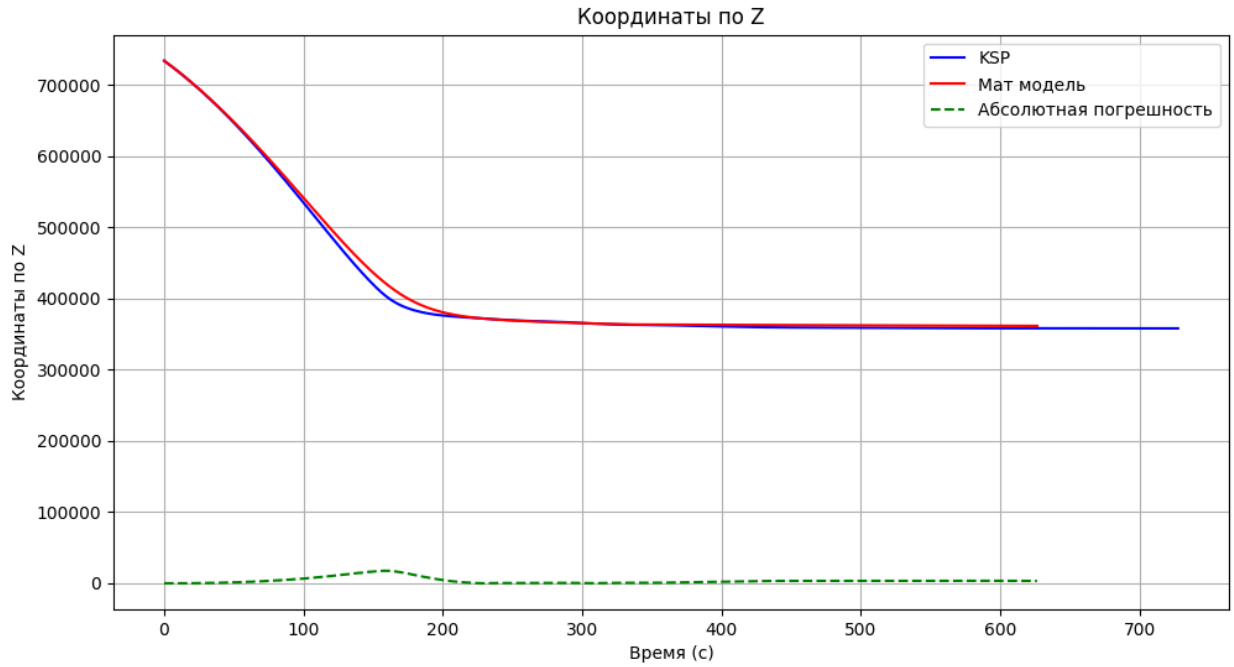


Рисунок 18. График зависимости координаты Z от времени

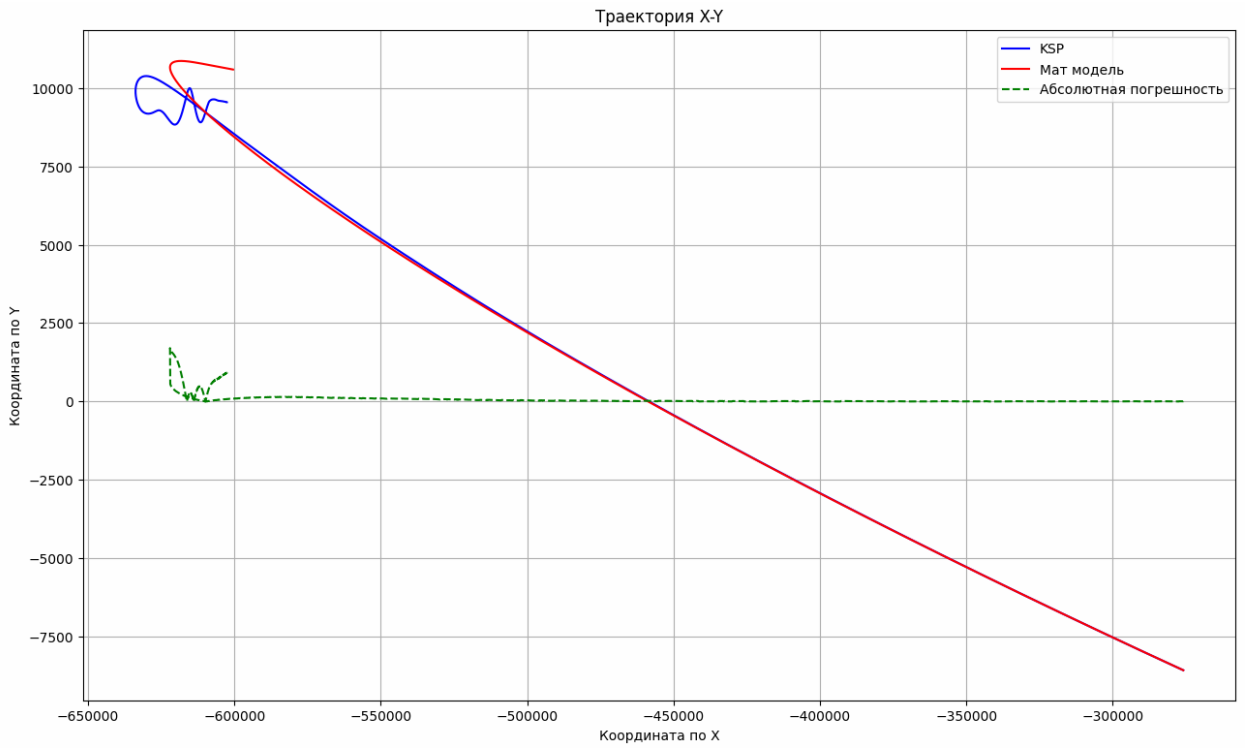


Рисунок 19. График зависимости координаты Y от X

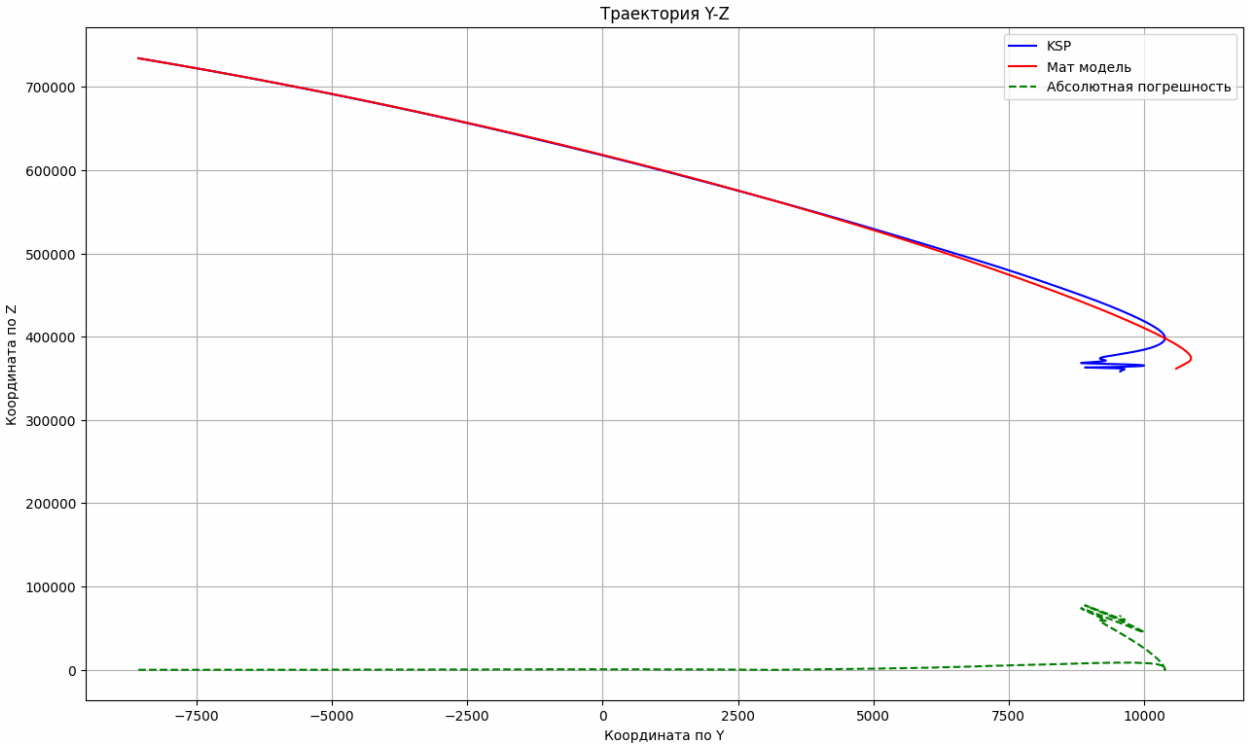


Рисунок 20. График зависимости координаты Z от Y

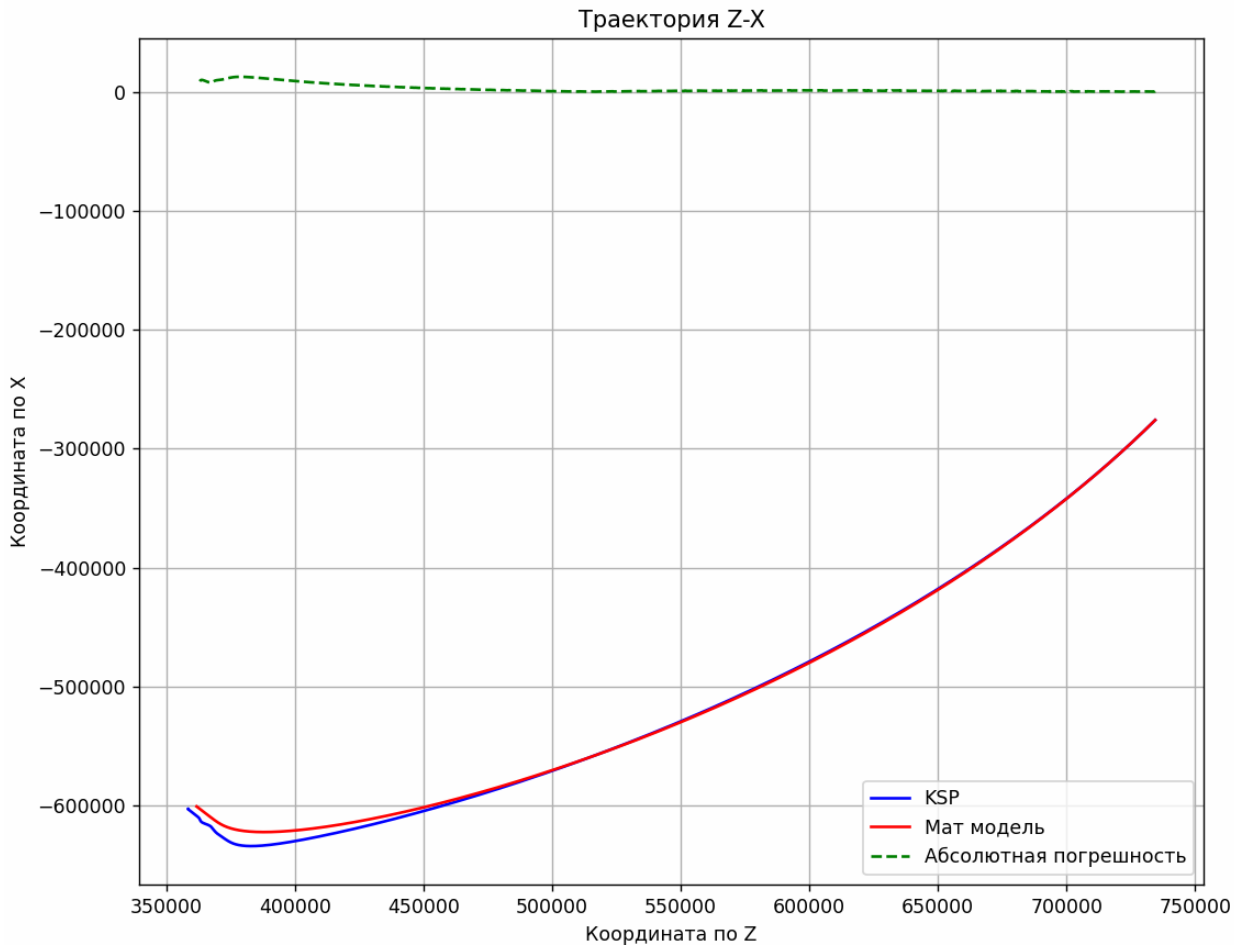


Рисунок 21. График зависимости координаты X от Z

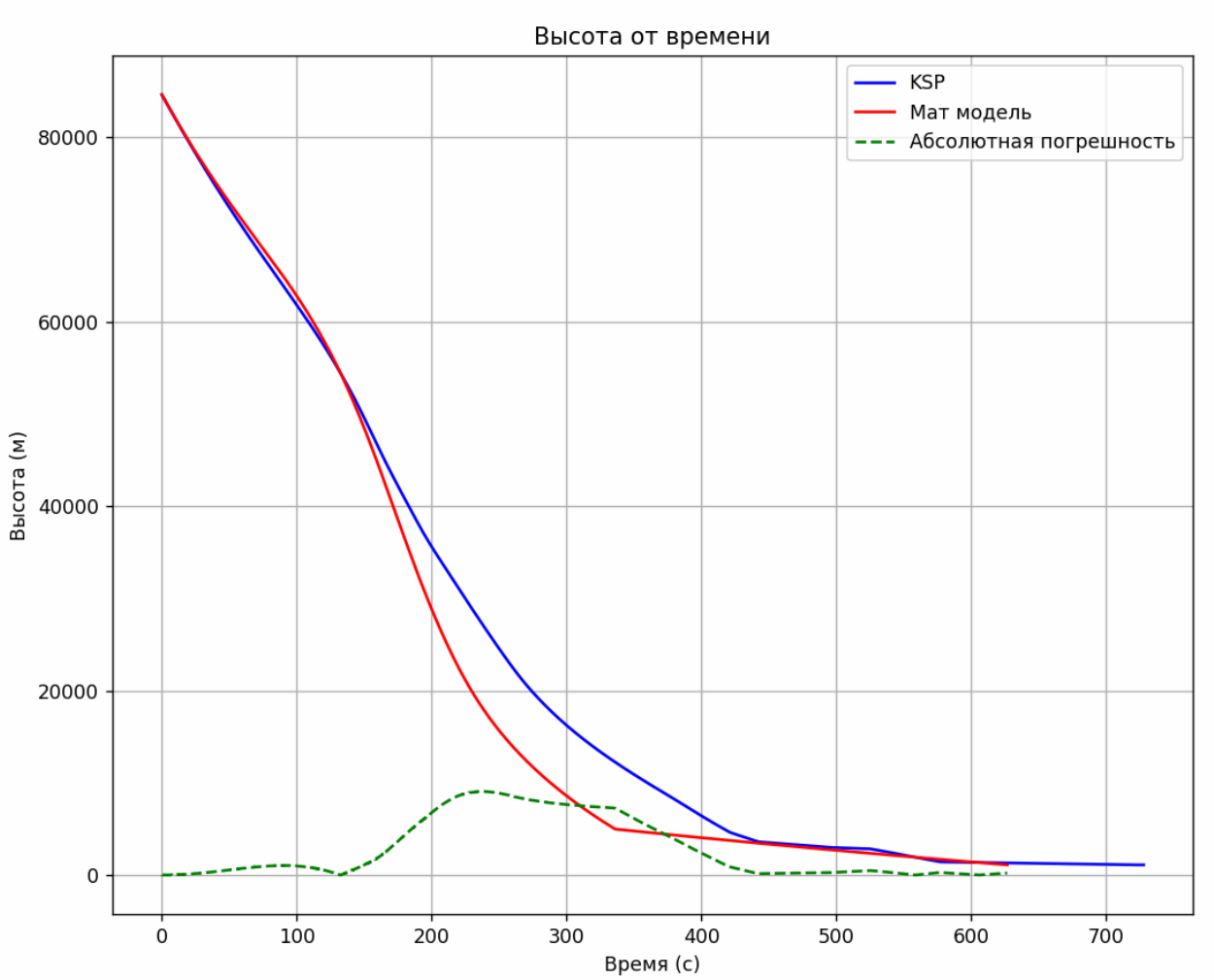


Рисунок 22. График зависимости высоты от времени

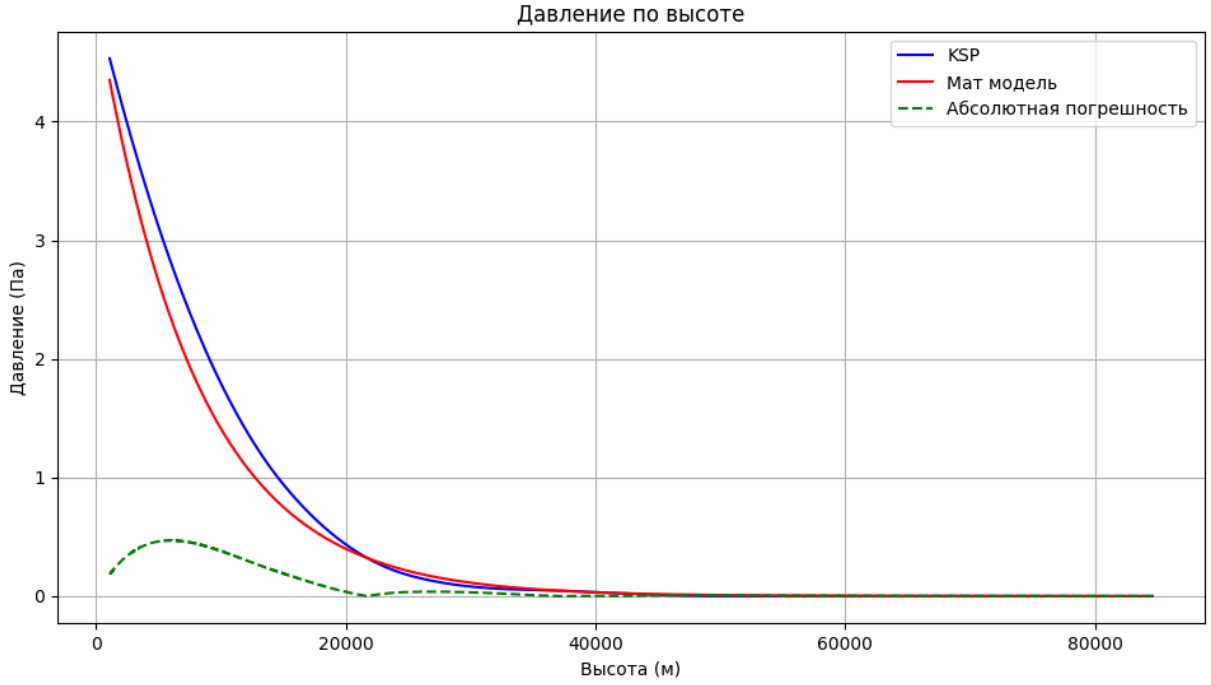


Рисунок 23. График зависимости высоты от времени

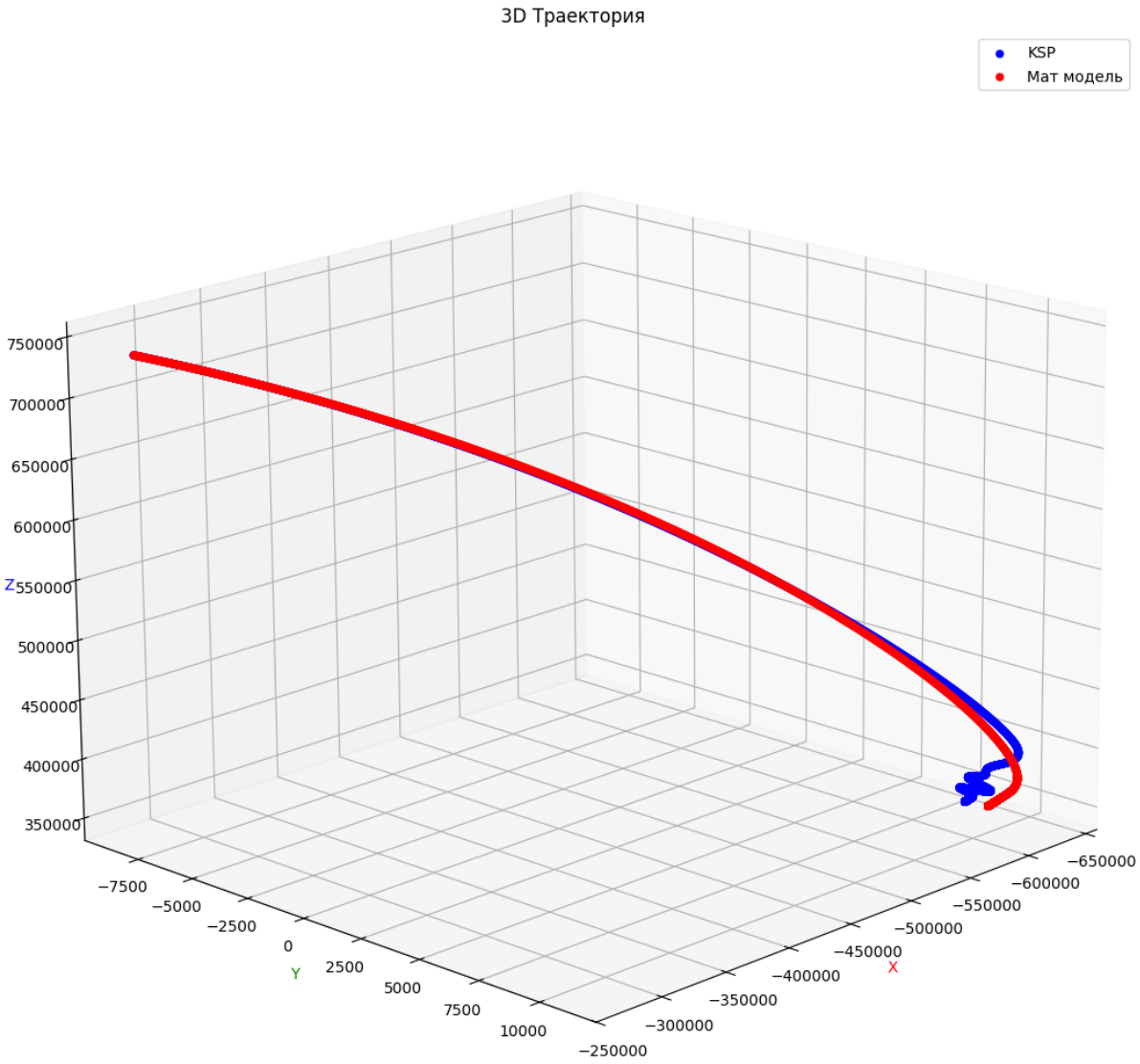


Рисунок 24. Траектория спуска

4. 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)
- time
- SciPy

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/с)",
'#3c88bd', "График зависимости скорости по координате X (м/с) от времени (с)")
    F(2, 2, 2, times, velocity_y, "Время (с)", "Скорость по Y (м/с)",
'#ff7f0e', "График зависимости скорости по координате Y (м/с) от времени (с)")
    F(2, 2, 3, times, velocity_z, "Время (с)", "Скорость по Z (м/с)",
'#2ca02c', "График зависимости скорости по координате Z (м/с) от времени (с)")
    F(2, 2, 4, times, speed, "Время (с)", "Скорость общая (м/с)", 'red',
"График зависимости общей скорости (м/с) 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/с)", "Высота (м)", 'green', "График
зависимости высоты (м) от скорости (м/с)")
    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.txt (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*К)
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
```

```python
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.

```python
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
```

```
# 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]
```

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.

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

5. Симуляция

1. Вход в атмосферу Евы

Рисунок 35

2. Нагревание аппарата, быстрое торможение

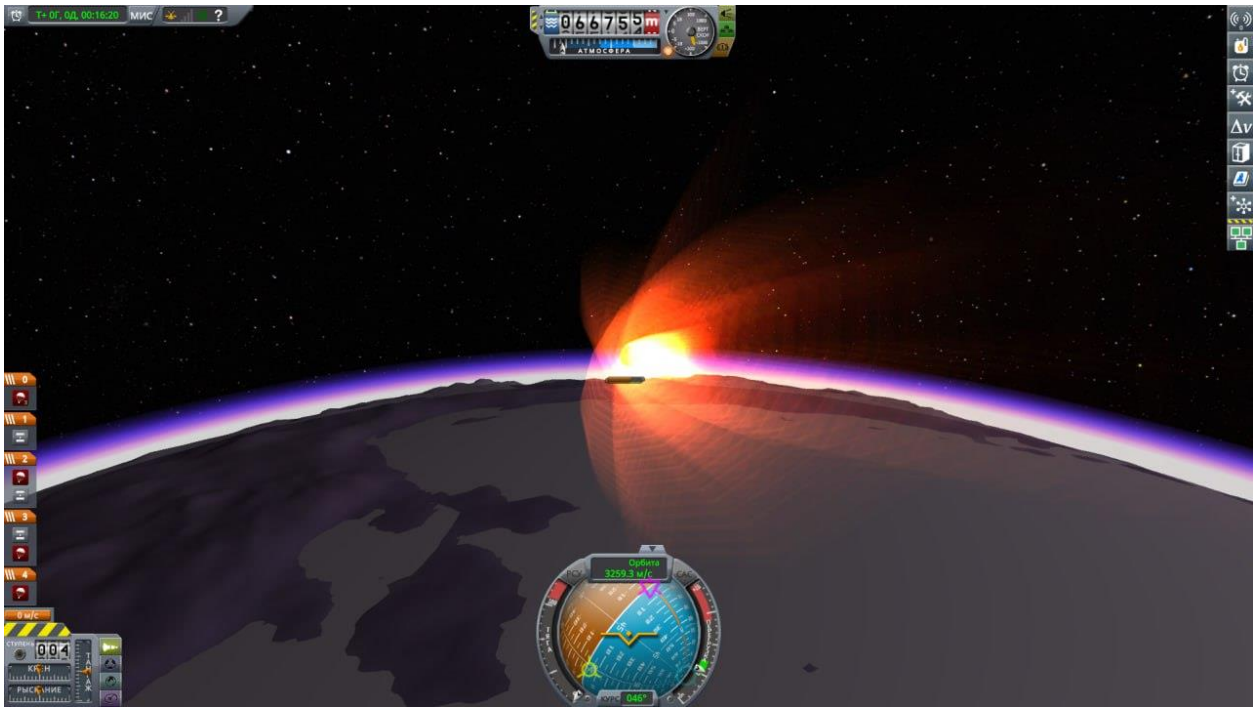


Рисунок 36

3. Аппарат перестаёт нагреваться, свободное падение

Рисунок 37

4. Открытие первого парашюта

Рисунок 38

5. Открытие второго парашюта

Рисунок 39

6. Открытие третьего парашюта

Рисунок 40

7. Отцепление второго парашюта

Рисунок 41

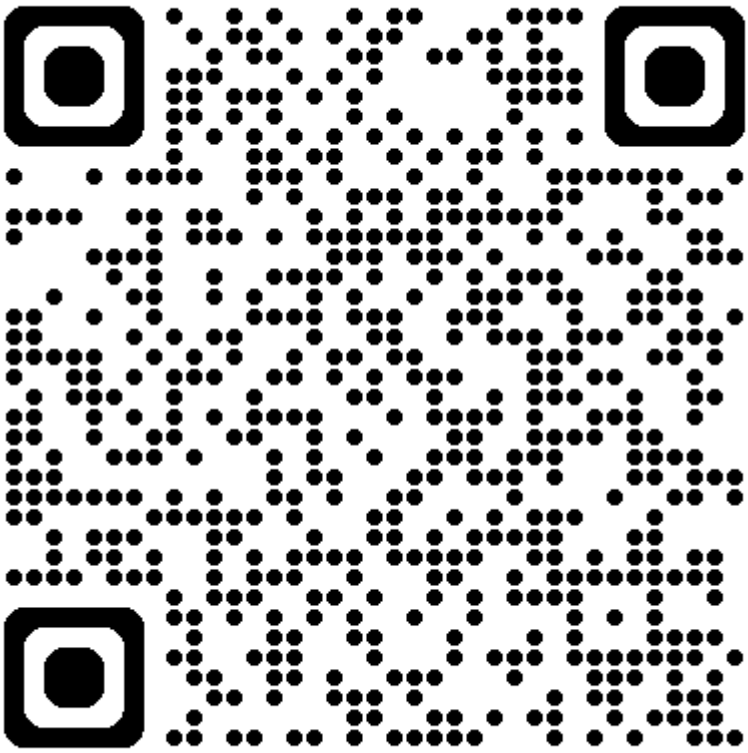
8. Открытие четвёртого парашюта (3 штуки)

Рисунок 42

9. Приземление

Рисунок 43

6. Медиа



Ссылка на GitHub репозиторий

7. Деятельность участников команды

Пятницкий Артём:

Создание математической и физической моделей и координация проекта в целом.

Демидов Георгий:

Помощь в создании алгоритма управления, сбор данных из KSP, написание отчёта, поиск информации, связанной с миссией.

Рубан Кирилл:

Написание алгоритма построения графиков, работа с данными, полученными из KSP и математической модели.

Цицкиев Дени:

Помощь в создании математической модели, сбор данных из KSP, написание алгоритма управления. Написание отчётов, монтаж видеоотчётов.

Заключение

В рамках данного проекта мы изучили движение спускаемого аппарата в атмосфере Венеры (Евы). Мы достигли поставленных задач.

Анализируя полученные из KSP и математической модели данные, можем сказать, что графики модели являются качественной версией графиков из симулятора, но из-за некоторых неучтённых факторов (подъёмная и боковая силы, изменение температуры от высоты и т.д.) и особенностей вычисления погрешностей, данные имеют значительные расхождения под конец спуска.

Список источников

1. Wikipedia contributors. Vega program. Wikipedia, The Free Encyclopedia. URL: https://en.wikipedia.org/wiki/Vega_program
2. Ла-Спейс. Проект «Вега-1, 2». URL: https://www.laspace.ru/ru/activities/projects/vega_1_2/
3. Эпизоды Космоса. Библиотека. Вега. URL: https://epizodyspace.ru/bibl/vega/01.html
4. Wiki. Kerbal Space Program. Atmosphere. URL: https://wiki.kerbalspaceprogram.com/wiki/Atmosphere
5. Wiki. Kerbal Space Program. Parachute. URL: https://wiki.kerbalspaceprogram.com/wiki/Parachute/ru?__cf_chl_rt_tk=Ef2gciL4b4G.6rLv42QychXNMGpdjZSi0OE.l2Ew8_k-1734904707-1.0.1.1-I_0qNtfi9mi_dKJwpOkUjWhcJ0VDL4dGqnQ_GJr44Ho
6. Мякишев Г. Я., Буховцев Б. Б., Чаругин В. М. Классический курс физики (базовый/углубленный) / Под ред. Н. А. Парфентьевой. Москва: Издательство «Просвещение», 2017.