Федеральное государственное автономное образовательное учреждение высшего образования
«Российский университет дружбы народов имени Патриса Лумумбы»
Инженерная академия
Департамент механики и процессов управления

ОТЧЕТ

По **Курсовой работе по Механике Космического Полета**

Направление: **01.03.02 Прикладная математика и Информатика**
(код направления / название направления)

Профиль: **Математические методы механики полета ракет-носителей и космических аппаратов**
(название профиля)

Тема: **Локально-геостанционарные орбиты. Кеплеровы элементы орбиты.**
(название лабораторной / курсовой)

Выполнено студентом: **Критским Матвеем Димитриевичем**
(ФИО)

Группа: **ИПМбд-02-22**
№ студенческого: 1132226149

Москва, 2023

Оглавление.

1.Определения и обозначения.

1.1 Основные определения.

1.2 Кеплеровы элементы.

1.3 Виды орбит. Трассы орбит. Периоды обращения орбит.

2. Формулы и графики скоростей.

2.1 Зависимость скорости спутника на круговой орбите от расстояния спутника до Земли.

2.2 Зависимость скорости конца геоцентрического радиус-вектора от высоты спутника при постоянном наклонении.

2.3 Зависимость скорости спутника в апогее орбиты при зафиксированной высоте перигея от высоты апогея.

3. Вывод.

4. Приложение.

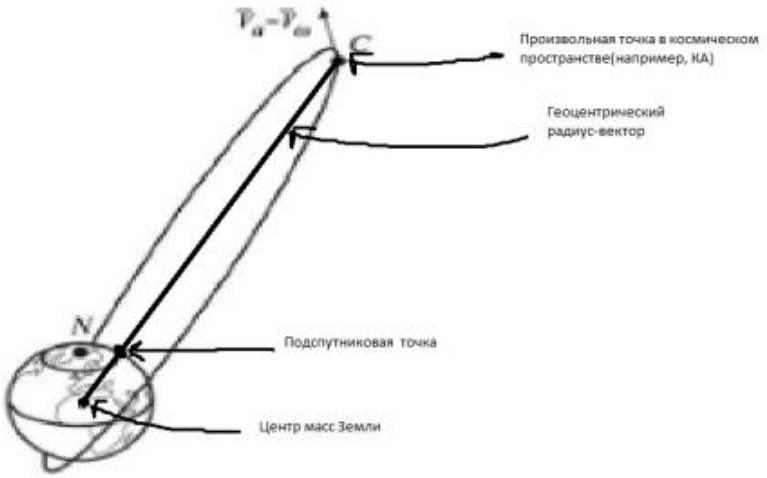
5. Используемая литература.

1. Определения и обозначения.

1.1 Основные определения.

Геоцентрический радиус-вектор(R) - вектор, начало которого совпадает с центром масс Земли, а конец - с определенной точкой в космическом пространстве(например, с положением космического аппарата(КА)).

Подспутниковая точка - точка пересечения геоцентрического радиус-вектора с поверхностью Земли. (см рис. 1)



(рис. 1)

Гравитационный параметр(□) - это произведение гравитационной постоянной G и массы тела M. □ = $G \cdot M$. □ меняется от планеты к планете из-за разной массы M. G - постоянна.

Восходящий узел - точка, в которой орбита проходит через базовую плоскость

H - расстояние между КА и подспутниковой точкой. (км)

V_c - скорость КА на круговой орбите. (км/с)

Vω - проекция скорости подспутниковой точки. (км/с)

V_i - скорость конца геоцентрического радиус-вектора, вращающегося вместе с Землей. (км/с)

V_A - скорость спутника в апогее. (км/с)

1.2 Кеплеровы элементы.

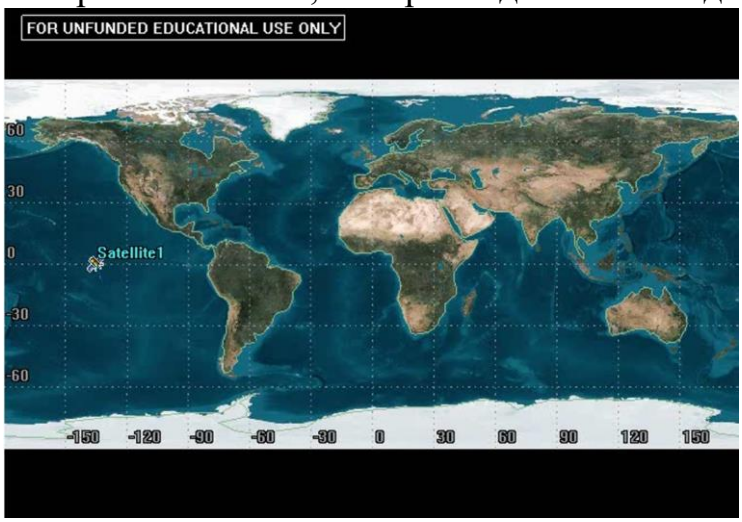
Кеплеровы элементы - это 6 элементов орбиты, определяющие положение КА в пространстве.

Название элемента	Описание элемента	Что меняет
Большая полуось(a)	Половина диаметра орбиты, проходящего через ее центр и 2 фокуса.	Размер орбиты.
Эксцентриситет(e)	Числовая характеристика конисеского сечения орбиты, показывающий степень ее отклонения от окружности	Форму орбиты.
Наклонение(i)	Угол между плоскостью орбиты и базовой плоскостью.	Положение орбиты в плоскости.
Долгота восходящего узла(Ω)	Угол между направлением весеннего равноденствия и направлением, где орбита пересекает плоскость эклиптики.	Положение орбиты в плоскости.
Аргумент перицентра(ω)	Угол между восходящим узлом орбиты и перицентром.	Положение орбиты в плоскости.
Средняя аномалия(M)	Угловое расстояние от перицентра тела, движущегося с постоянной угловой скоростью.	Положение орбиты в пространстве.

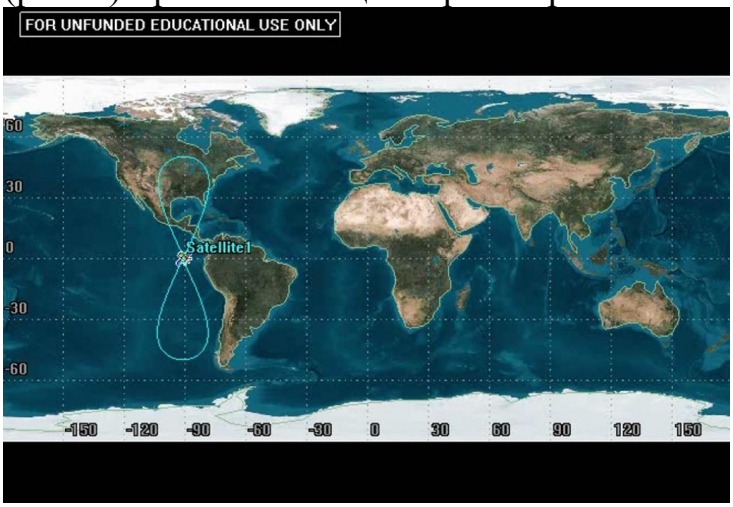
1.3 Виды орбит. Трассы орбит. Периоды обращения орбит.

Геосинхронная орбита(Geosynchronous Orbit) - орбита обращающегося вокруг Земли спутника, период которой равен периоду вращения Земли. Частным случаем геосинхронной орбиты является **геостационарная орбита(Geostationary orbit – GEO)** - круговая орбита, лежащая в плоскости земного экватора. Если наклонение геосинхронной орбиты отлично от 0 и эксцентриситет равен 0, то при наблюдении спутник описывает в небе “восьмерку”. Если же наклонение и эксцентриситет отличны от 0, то в зависимости от этих величин “восьмерка” выродится в эллипс или в отрезок прямой, лежащей в плоскости экватора(при нулевом наклонении и ненулевом эксцентриситете).

Трасса орбиты(Ground Track) - проекция орбиты спутника Земли на поверхность Земли, т.е. трасса движения подспутниковой точки.

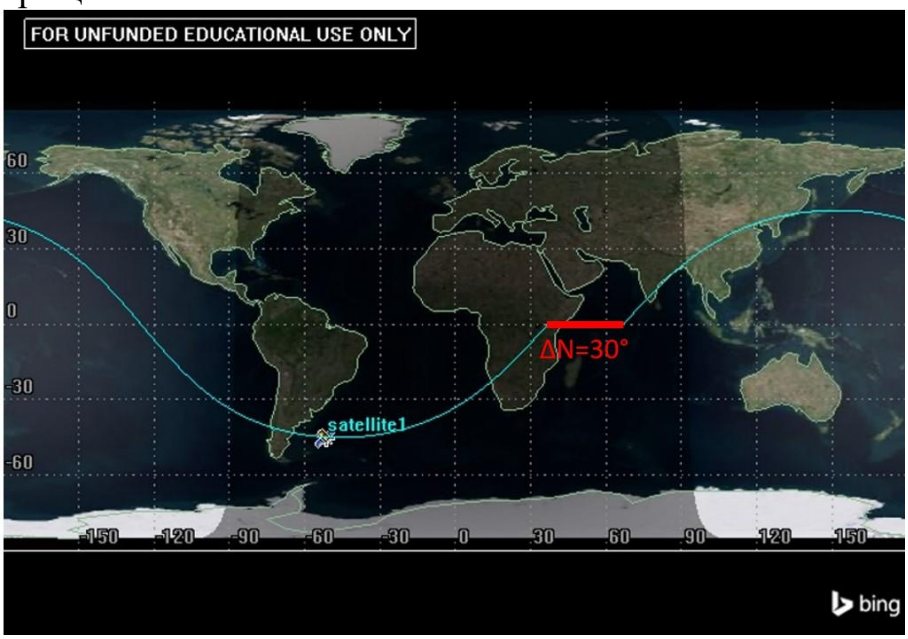


(рис. 2) Трасса геостационарной орбиты



(рис. 3) Трасса геосинхронной орбиты

Закрытая трасса орбиты(Closed Ground Track) - частный случай трассы, когда после полного оборота спутника следующая трасса накладывается на предыдущую. Например, геостационарная и геосинхронная орбиты имеют закрытую трассу (рис. 2 и рис. 3). Из-за поворота Земли трассы орбит “разрываются”(как видно на рис.4). Из-за этого “разрыва” после полного оборота спутника трасса следующего оборота не совпадает с предыдущей трассой. В случае рис. 2 и рис. 3 “разрыва” нет, т.к. период обращения орбит равен периоду вращения Земли.



(рис. 4) Пример открытой трассы.

Сидерический период обращения или **Период обращения по орбите(Sidereal Perion** or **Orbital Period T)** - время полета между двумя точками A_0 и A_1 на двух соседних витках. Сидерический период зависит от широты точки наблюдения A_0. Из 3-его закона Кеплера следует, что

$T = \sqrt{\frac{a^3 \cdot T_з^2}{a_з^3}}$, где a - большая полуось орбиты спутника при сидерическом периоде обращения; $a_з$ - большая полуось Земли, $T_з$ - период обращения Земли.

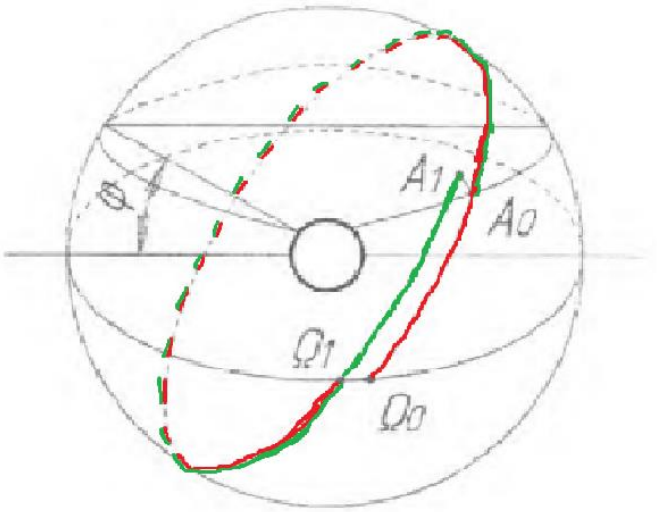
Драконический период обращения(Nodal Period T_{nod}) - время между двумя последовательными прохождениями спутника через плоскость экватора при движении с юга на север, т.е. время между двумя прохождениями спутника через два соседних восходящих узла орбиты Ω_1 и Ω_2. Драконический период имеет более практическое значение, чем сидерический, поскольку каждое прохождение спутником плоскости экватора может быть точно зафиксировано.

$T_{nod} = 2\pi \frac{a^{\frac{3}{2}}}{\sqrt{\mu}}\left(1 - \frac{\frac{3}{2}J_2 R_E^2}{a^2(1-e^2)^2}\left(\frac{10\cos^2(i)-2}{4} - \frac{3\cos^2(i)-1}{4}(1-e^2)^{\frac{1}{2}}\right)\right)$, где J_2 - полярная сплюснутость, $J_2 = 0.0010827$.

Если в формулу сидерического периода подставить формулу гравитационного параметра для эллиптических орбит, получим:

$\mu = 4\pi^2 \frac{a^3}{T^2}$. Для Земли: $\mu = 4\pi^2 \frac{a_з^3}{T_з^2} => \frac{T_з^2}{a_з^3} = \frac{4\pi^2}{\mu}$

$T = \sqrt{a^3 \frac{4\pi^2}{\mu}} = 2\pi \cdot \frac{a^{\frac{3}{2}}}{\sqrt{\mu}}$, т.е. T_{nod} - синодический период с учетом сплющенности Земли.



(рис. 5) Возмущенная орбита. Красным цветом показана траектория спутника для драконического периода, а зеленым - для сидерического.

Эффективный период обращения Земли(Efficient Period of Earth Rotation T_{eff}) - период невозмущенного вращения Земли, корректируемого с учетом прецессии орбиты спутника.

$T_{eff} = \frac{2\pi + \left(\frac{m}{n}\right)\cdot\delta\Omega}{\varpi_E}$, где m - кол-во оборотов, n - кол-во эффективных астрономических дней, $\delta\Omega$ - угловое смещение восходящего узла орбиты спутника за один оборот. Отсюда $\frac{m}{n}$ - коэф. повторяемости геосинхронной орбиты. Если сравнивать это со школьной формулой периода движения тела по окружности $T = \frac{2\pi}{\varpi}$, можно заметить, что в эффективном периоде обращения к 2π

прибавляется $\frac{m}{n}\delta\Omega$. Это необходимо из-за прецессии орбиты спутника, возникающей по причине несферичности Земли.

2. Формулы и графики скоростей.

2.1. Зависимость скорости спутника на круговой орбите от расстояния спутника до Земли.

Для подсчета скорости спутника на круговой орбите V_c воспользуемся следующей формулой:

$V_a = \sqrt{\frac{2\mu \cdot r_p}{r_A \cdot (r_A + r_p)}}$ (1), где μ - гравитационный параметр Земли, μ = 398600.4 км^3/с^2, r_A - радиус апогея, r_p - радиус перигея. Учитывая, что e = 0(круговая орбита) и что

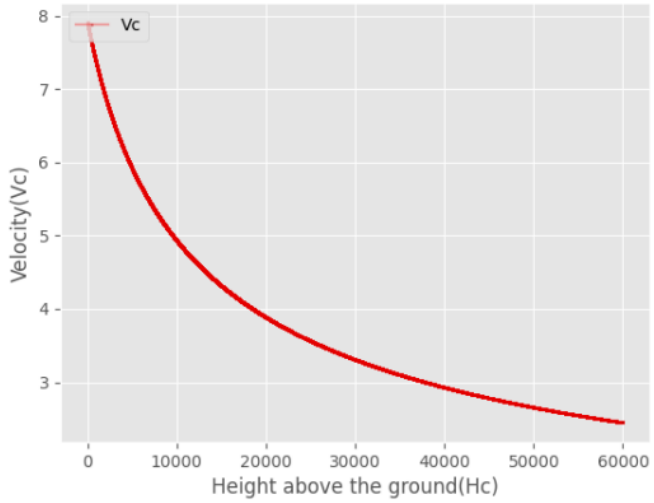
$r_A = \frac{p}{1 - e \cdot \cos(\theta)}$ = p (p - фокальное расстояние)

=> $r_A = r_p$

$r_p = \frac{p}{1 + e \cdot \cos(\theta)}$ = p

Подставив полученные радиусы в формулу (1), получим:

$V_a = \sqrt{\frac{2\mu \cdot r_p}{r_p \cdot (r_p + r_p)}} = \sqrt{\frac{\mu}{r_p}} = V_c$. Если представить, что $r_p = R_з + H$, где $R_з$ - радиус Земли, $R_з$ = 6378 км, H_c - расстояние от поверхности Земли до спутника, мы получим: $V_c = \sqrt{\frac{\mu}{R_з + H_c}}$. Если H_c будет принимать значения в отрезке [0; 60000] км с шагом 1, то мы получаем следующий график:



(рис. 6) график зависимости $V_c(H_c)$

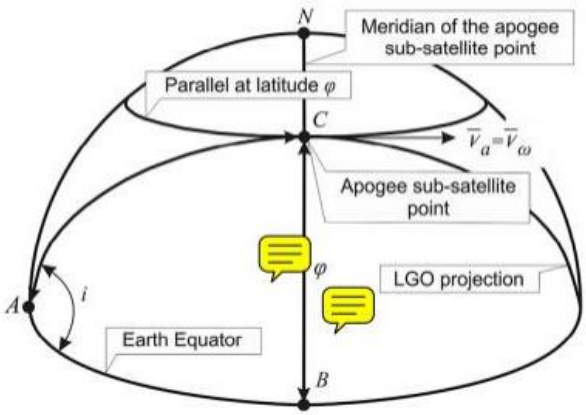
2.2 Зависимость скорости конца геоцентрического радиус-вектора от высоты спутника при постоянном наклонении.

Для подсчета скорости конца геоцентрического радиус-вектора, воспользуемся следующей формулой:

$V_\omega = \omega_E \cdot r_A \cdot \cos(\varphi)$ (2)

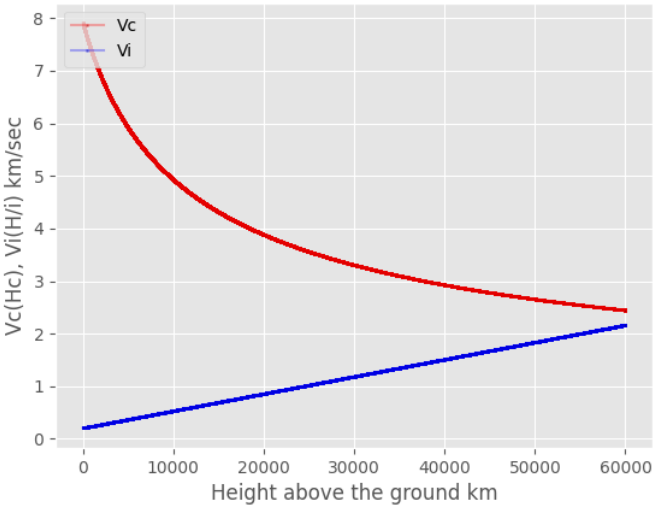
Если учесть, что $\cos(i) = \cos(\varphi) \cdot \sin(\alpha)$ и что в нашем случае $\alpha = 90°$(рис. 7)

получим: $V_\omega = \varpi_E \cdot r_A \cdot \cos(i)$, где ϖ_E - угловая скорость вращения Земли, $\varpi_E = 7.2921159 \cdot 10^{-5}$ c^{-1} , i - наклонение орбиты, в нашем случае является постоянной величиной i, которая принимает значения [0; 63.4]°. В моем случае i = 63.4°(орбита Молния), $r_A = (R_з + H)$. H аналогично H_c - расстояние от поверхности Земли до спутника.



(рис. 7) Проекция локально-геостационарой орбиты с апогеем, соответствующим максимальной(минимальной) широте.

Если H будет принимать значения [0; 60000] км с шагом 1, мы получим следующий график:



(рис. 8) График зависимостей Vc(Hc) и Vi(H/i)

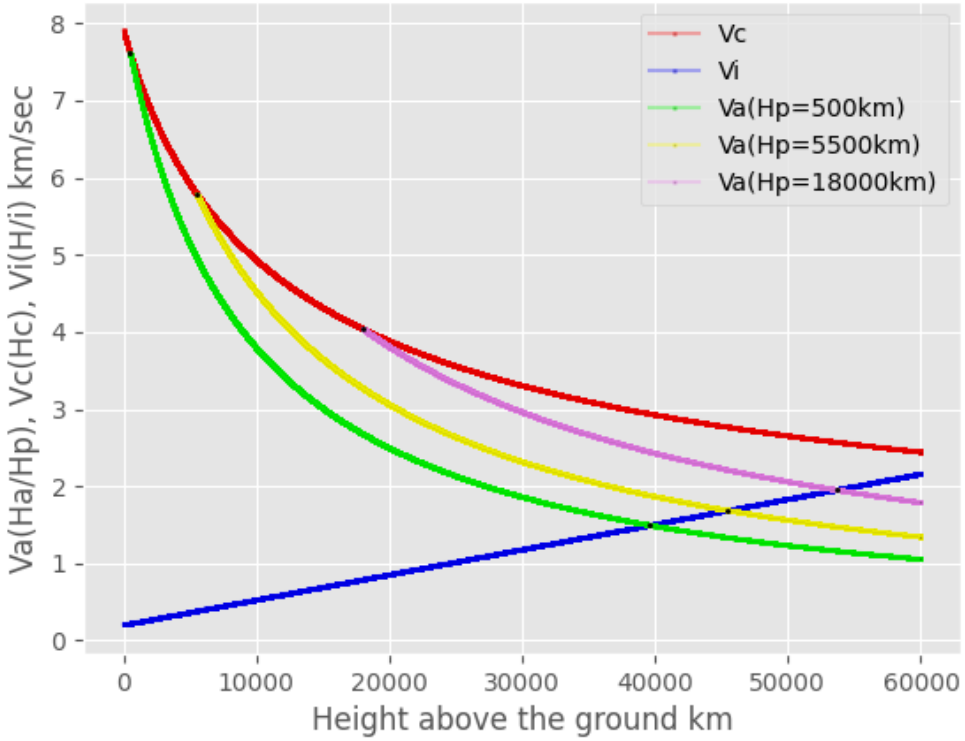
2.3 Зависимость скорости спутника в апогее орбиты при зафиксированной высоте перигея от высоты апогея.

Для подсчета скорости спутника в апогее орбиты при зафиксированной высоте перигея воспользуемся уже упомянутой в (2.1) формуле (1):

$$V_a = \sqrt{\frac{2\mu \cdot r_p}{r_A \cdot (r_A + r_p)}}.$$

Если $r_p = (R + H_p)$, H_p – высота перигея, H_p = 500, 5500, 18000 км,

$r_A =$ $(R + H_A)$, H_A - высота апогея, то подставив соответствующие значения H_p и значения H_A в пределе [0; 60000] км, получим следующий график:



(рис. 9) График зависимостей $V_c(H_c)$, $V_i(H/i)$, $V_A(H_A,H_p)$.

Лаймовый график - V_{a1}, Желтый - V_{a2}, Фиолетовый - V_{a3}.

На данном графике также отображены точки пересечения V_A с V_c и V_i с точностью до 10^{-6} черными точками.

3. Вывод.

Из (рис. 6), (рис. 8), (рис. 9) можно заметить, что с отдалением спутника от поверхности Земли скорость в апогее уменьшается, а скорость точки конца геоцентрического радиус-вектора наоборот увеличивается.

Скорость спутника в апогее круговой орбиты V_c принимает значения [2.450512; 7.904830]. При чем при H = 1 V_c - максимален, а при H = 60000 - минимален. Аналогично и с V_{a1-3}:

V_{a1} принимает значения [1.061894; 8.052227]

V_{a2} принимает значения [1.350158; 9.017051]

V_{a3} принимает значения [1.796111; 9.952556]

Стоит заметить, что максимумы и минимумы V_{a1-3} возрастают по мере увелечения индекса (т.е. $\max(V_{a1}) < \max(V_{a2})$ и т.д.).

Графики V_c и V_{a1-3} - это графики квадратного корня. Графики нисходящие.

Скорость конца геоцентрического радиус-вектора V_i принимает значения [0.208281; 2.167315]. Графики V_i - прямые и восходящие.

4. Приложения.

Ссылка на GitHub: https://github.com/rudnmk/Coursework

Код программы для подсчета и сохранения информации (находится по пути Coursework/code/code.c):

```c
#include <stdio.h>
#include<math.h>
#define M_PI 3.14159265358979323846
#define Nu 398600.4
#define i 63.4
#define R 6378.0

int main() {
    float Vc;
    float Vi;
    float Va1;
    float Va2;
    float Va3;
    float We = 7.2921159;
    const float Hp1 = 500.0;
    const float Hp2 = 5500.0;
    const float Hp3 = 18000.0;
    int flag1 = 0;
    int flag2 = 0;
    int flag3 = 0;
    FILE *Vc_file = fopen("Vc_DATA.txt", "w");
    FILE *Vi_file = fopen("Vi_DATA.txt", "w");
    FILE *Va1_file = fopen("Va1_DATA.txt", "w");
    FILE *Va2_file = fopen("Va2_DATA.txt", "w");
    FILE *Va3_file = fopen("Va3_DATA.txt", "w");
    FILE *intersec = fopen("IntersectionPoints.txt", "w");

    for (int H = 1; H < 60001; H++) {
```

```
            Vc = sqrt(Nu / (R + H));
            Vi = We * pow(10, -5) * (R + H) * cos(i / 180.0 * M_PI);
            Va1 = sqrt((2 * Nu * (R + Hp1)) / ((R + H) * (R + H + R + Hp1)));
            Va2 = sqrt((2 * Nu * (R + Hp2)) / ((R + H) * (R + H + R + Hp2)));
            Va3 = sqrt((2 * Nu * (R + Hp3)) / ((R + H) * (R + H + R + Hp3)));

            if (Va1 < Vc && flag1 == 0) {
                for (float tmp = H - 2; tmp < H; tmp = tmp + 0.001) {
                    Va1 = sqrt((2 * Nu * (R + Hp1)) / ((R + tmp) * (R + tmp + R +
Hp1)));
                    Vc = sqrt(Nu / (R + tmp));
                    if (Va1 <= Vc) {
                        fprintf(intersec, "%f   %f\n", tmp, Vc);
                        break;
                    }
                }
                flag1 = 1;
            }
            if (Va2 < Vc && flag2 == 0) {
                for (float tmp = H - 2; tmp < H; tmp = tmp + 0.001) {
                    Va2 = sqrt((2 * Nu * (R + Hp2)) / ((R + tmp) * (R + tmp + R +
Hp2)));
                    Vc = sqrt(Nu / (R + tmp));
                    if (Va2 <= Vc) {
                        fprintf(intersec, "%f   %f\n", tmp, Vc);
                        break;
                    }
                }
                flag2 = 1;
            }
```

```
            if (Va3 < Vc && flag3 == 0) {
                for (float tmp = H - 2; tmp < H; tmp = tmp + 0.001) {
                    Va3 = sqrt((2 * Nu * (R + Hp3)) / ((R + tmp) * (R + tmp + R +
Hp3)));
                    Vc = sqrt(Nu / (R + tmp));
                    if (Va3 <= Vc) {
                        fprintf(intersec, "%f   %f\n", tmp, Vc);
                        break;
                    }
                }
                flag3 = 1;
            }

            if (Va1 < Vi && flag1 == 1) {
                for (float tmp = H - 2; tmp < H; tmp = tmp + 0.01) {
                    Va1 = sqrt((2 * Nu * (R + Hp1)) / ((R + tmp) * (R + tmp + R +
Hp1)));
                    Vi = We * pow(10, -5) * (R + tmp) * cos(i / 180.0 * M_PI);
                    if (Va1 <= Vi) {
                        fprintf(intersec, "%f   %f\n", tmp, Vi);
                        break;
                    }
                }
                flag1 = 2;
            }
            if (Va2 < Vi && flag2 == 1) {
                for (float tmp = H - 2; tmp < H; tmp = tmp + 0.01) {
                    Va2 = sqrt((2 * Nu * (R + Hp2)) / ((R + tmp) * (R + tmp + R +
Hp2)));
                    Vi = We * pow(10, -5) * (R + tmp) * cos(i / 180.0 * M_PI);
```

```
                if (Va2 <= Vi) {
                    fprintf(intersec, "%f   %f\n", tmp, Vi);
                    break;
                }
            }
            flag2 = 2;
        }
        if (Va3 < Vi && flag3 == 1) {
            for (float tmp = H - 2; tmp < H; tmp = tmp + 0.01) {
                Va3 = sqrt((2 * Nu * (R + Hp3)) / ((R + tmp) * (R + tmp + R +
Hp3)));
                Vi = We * pow(10, -5) * (R + tmp) * cos(i / 180.0 * M_PI);
                if (Va3 <= Vi) {
                    fprintf(intersec, "%f   %f\n", tmp, Vi);
                    break;
                }
            }
            flag3 = 2;
        }

        fprintf(Vc_file, "%i   %f \n", H, Vc);
        fprintf(Vi_file, "%i   %f \n", H, Vi);
        if (Va1 < Vc) {
            fprintf(Va1_file, "%i   %f\n", H, Va1);
        }

        if (Va2 < Vc) {
            fprintf(Va2_file, "%i   %f\n", H, Va2);
        }
```

```
            if (Va3 < Vc) {
                fprintf(Va3_file, "%i   %f\n", H, Va3);
            }
        }

        fclose(Vc_file);
        fclose(Vi_file);
        fclose(Va1_file);
        fclose(Va2_file);
        fclose(Va3_file);
        fclose(intersec);
        return 0;
    }
```

Код программы для построения и вывода графиков (находится по пути Coursework/code/graph.py):

```
import pandas as ps
import matplotlib.pyplot as plt
import numpy as np



with open("Vc_DATA.txt", "r") as f:
    vc = f.readlines()
    f.close()

with open("Vi_DATA.txt", "r") as f:
```

```
    vi = f.readlines()
    f.close()

with open("Va1_DATA.txt", "r") as f:
    va1 = f.readlines()
    f.close()

with open("Va2_DATA.txt", "r") as f:
    va2 = f.readlines()
    f.close()

with open("Va3_DATA.txt", "r") as f:
    va3 = f.readlines()
    f.close()

with open("IntersectionPoints.txt", "r") as f:
    points = f.readlines()
    x1, y1 = (points[0].rstrip('\n')).split('   ')
    x2, y2 = (points[1].rstrip('\n')).split('   ')  #Точки пересечений Va с Vc
    x3, y3 = (points[2].rstrip('\n')).split('   ')
```

```
    x4, y4 = (points[3].rstrip('\n')).split('   ')
    x5, y5 = (points[4].rstrip('\n')).split('   ')  #Точки пересечений Va с Vi
    x6, y6 = (points[5].rstrip('\n')).split('   ')
    f.close()



vc_x = []
vi_x = []
va1_x = []
va2_x = []
va3_x = []
data_list_vc = []
data_list_vi = []
data_list_va1 = []
data_list_va2 = []
data_list_va3 = []
for i in vc:
    tmp = (i.rstrip('\n')).split('   ')
    vc_x.append(int(tmp[0]))
```

```
    data_list_vc.append(float(tmp[1]))


for i in vi:
    tmp = (i.rstrip('\n')).split('    ')
    vi_x.append(int(tmp[0]))
    data_list_vi.append(float(tmp[1]))


for i in va1:
    tmp = (i.rstrip('\n')).split('    ')
    va1_x.append(int(tmp[0]))
    data_list_va1.append(float(tmp[1]))


for i in va2:
    tmp = (i.rstrip('\n')).split('    ')
    va2_x.append(int(tmp[0]))
    data_list_va2.append(float(tmp[1]))


for i in va3:
    tmp = (i.rstrip('\n')).split('    ')
    va3_x.append(int(tmp[0]))
```

```
    data_list_va3.append(float(tmp[1]))



plt.style.use("ggplot")
figure, ax1 = plt.subplots()


result_array_vc = np.asarray(data_list_vc).T
result_array_vi = np.asarray(data_list_vi).T
result_array_va1 = np.asarray(data_list_va1).T
result_array_va2 = np.asarray(data_list_va2).T
result_array_va3 = np.asarray(data_list_va3).T
ax1.plot(vc_x, result_array_vc, color="red", alpha=0.3, label="Vc", marker="s", markerfacecolor="black", markersize=1)
ax1.plot(vi_x, result_array_vi, color="blue", alpha=0.3, label="Vi", marker="s", markerfacecolor="black", markersize=1)
ax1.plot(va1_x, result_array_va1, color="lime", alpha=0.3, label="Va(Hp=500km)", marker="s", markerfacecolor="black", markersize=1)
ax1.plot(va2_x, result_array_va2, color="yellow", alpha=0.3, label="Va(Hp=5500km)", marker="s", markerfacecolor="black", markersize=1)
ax1.plot(va3_x, result_array_va3, color="violet", alpha=0.3, label="Va(Hp=18000km)", marker="s", markerfacecolor="black", markersize=1)
```

```
ax1.plot(float(x1), float(y1), color="black", alpha=1, marker="s", markerfacecolor="black", markersize=1)

ax1.plot(float(x2), float(y2), color="black", alpha=1, marker="s", markerfacecolor="black", markersize=1)

ax1.plot(float(x3), float(y3), color="black", alpha=1, marker="s", markerfacecolor="black", markersize=1)


ax1.plot(float(x4), float(y4), color="black", alpha=1, marker="s", markerfacecolor="black", markersize=1)

ax1.plot(float(x5), float(y5), color="black", alpha=1, marker="s", markerfacecolor="black", markersize=1)

ax1.plot(float(x6), float(y6), color="black", alpha=1, marker="s", markerfacecolor="black", markersize=1)


ax1.legend(loc="upper right")


ax1.set_xlabel("Height above the ground km")

ax1.set_ylabel("Va(Ha/Hp), Vc(Hc), Vi(H/i) km/sec")

plt.show()
```

5. Используемая литература.

1. Locally Geostationary Orbits: Optimal Geometry of Elliptic Orbit for Earth Coverage by Yury N. Razoumny.

https://www.researchgate.net/publication/329948694_Locally_Geostationary_Orbits_Optimal_Geometry_of_Elliptic_Orbit_for_Earth_Coverage

2. Основы теории полета Белоконов В.М.

http://repo.ssau.ru/bitstream/Uchebnye-izdaniya/Osnovy-teorii-poleta-kosmicheskih-apparatov-konspekt-lekcii-Tekst-elektronnyi-83923/1/Белоконов%20В.М.%20Основы%20теории%20полета%202006.pdf