МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)» (МАИ)

Кафедра 604 “Системный анализ и управление”

Специальность 24.03.03 “Баллистика и гидроаэродинамика”

Проектная работа

«Моделирование баллистического спуска КА на Венеру»

по дисциплине **«Моделирование АКС»**.

Выполнили студенты

группы М6О-401Б-19:

Азымова А. Р.
Белозерцева А. Н.
Бюн Ю.
Медведев С. Ю.
Скакун В. С.
Щербаносов А. Д.

Принял:
Шмигирилов

Сергей Юрьевич

Москва 2022 г.

Содержание

Введение

Спускаемые аппараты автоматических космических станций, предназначенных для исследования планеты Венера, отличаются конструктивно от спускаемых аппаратов космических кораблей. Планета Венера обладает достаточно мощной атмосферой: атмосферное давление на поверхности планеты более чем в 90 раз превышает земное. Температура на поверхности равна почти 700 °C. Всё это влияет на характер движения спускаемого аппарата на Венеру.

Цель работы: создание программного обеспечения для решений уравнений, описывающих движение спускаемого КА в атмосфере Венеры и исследование изменений параметров движения КА.

Для этого необходимо:

- Составить математическую модель движения КА.
- Разработать ПО для математической модели.
- Построить графики зависимостей параметров движения КА от времени.

Решение поставленной задачи реализовано в средах разработки ПО Matlab и Python.

Построение математической модели

Рассматривается атмосферное движение спускаемого космического аппарата до раскрытия парашюта увода.

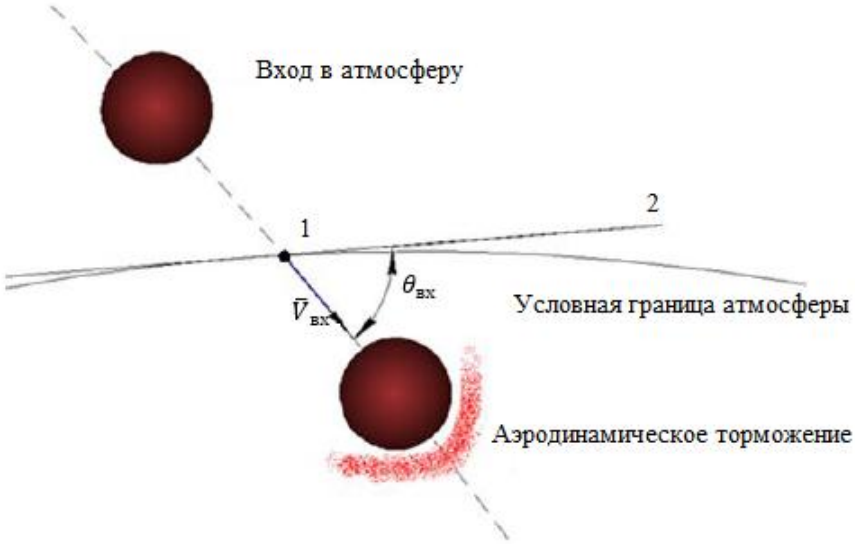

Рисунок 1. Движение спускаемого КА при входе в атмосферу Венеры. 1 - точка входа КА в атмосферу; 2 - местный горизонт; $\bar{V}_{вх}$ – скорость входа в атмосферу; $\theta_{вх}$ – угол входа.

В качестве дифференциальных уравнений движения центра масс СА будем рассматривать уравнения в проекциях на оси планетоцентрической прямоугольной экваториальной системы координат $OX_AY_AZ_A$ с началом в центре масс планеты, принимаемой за инерциальную систему, с направлениями осей OZ_A – из центра Венеры к северному полюсу Мира, OX_A – из центра Венеры в точку весеннего равноденствия, OY_A – из центра Венеры и дополняет систему до правой. А также в проекциях на оси скоростной барицентрической системы координат $Axyz$ с началом в центре масс КА.

Введем ряд допущений:

- все силы, действующие на КА, приложены к его центру масс;
- масса КА считается постоянной;
- спуск аппарата происходит под действием только силы веса G и аэродинамической силы R;
- Венера – шар с радиусом R_E с равномерно распределенной плотностью;

– движение КА вокруг центра масс не рассматривается;
– ускорение, обусловленное вращением Венеры не велико и поэтому центробежной силой можно пренебречь;
– суммарной силой притяжения Солнца и планет можно пренебречь;
– рассматривается плоское движение.

Для случая баллистического спуска можно использовать следующие положения:

– $C_{Y_A} = 0$;
– атмосфера планеты не вращается;
– атмосфера планеты изотермическая;
– высота спуска мала по сравнению с радиусом планеты;
– проекция ускорения свободного падения на касательную траектории мала по сравнению с ускорением от силы лобового сопротивления.

При приведенных допущениях и положениях уравнения движения КА:

$$\begin{aligned} \frac{dV}{dt} &= -\frac{C_x S}{m}\frac{\rho V^2}{2} - g\sin\theta \\ V\frac{d\theta}{dt} &= \frac{V^2}{r}\cos\theta - g\cos\theta \\ \frac{dh}{dt} &= V\sin\theta \\ \frac{dL}{dt} &= V\frac{R_E}{r}\cos\theta \end{aligned} \qquad (1)$$

Здесь V – скорость, θ – угол наклона траектории, h - высота, L – дальность по поверхности, m – масса КА, $g = \mu/r^2$ – гравитационное ускорение, $\mu = 324853.4$ км^3/с^2 – гравитационный параметр Венеры, $r = R_E + h$ - расстояние от центра Венеры до КА, $R_E = 6052$ км – радиус Венеры, C_x – коэффициент лобового сопротивления, S – площадь миделя, ρ – плотность атмосферы Венеры.

Решение системы нелинейных дифференциальных уравнений (1) осуществлялось методом численного интегрирования – метод Рунге-Кутта четвертого порядка.

Произведение расчетов

Программное обеспечение для расчета движения спускаемого КА написано в средах разработки Matlab и Python.

Для расчетов принимались следующие начальные условия:

- масса КА 600 кг;
- диаметр КА 2,4 м;
- скорость входа в атмосферу 11 км/с;
- условная граница атмосферы 130 км;
- угол входа в атмосферу 30^{o};
- время срабатывания парашюта увода 45 с.

Результаты

Результаты расчетов представлены в качестве графиков зависимости скорости КА, угла траектории КА, высоты КА и дальности вдоль поверхности Венеры от времени.

Результаты полученные в Matlab

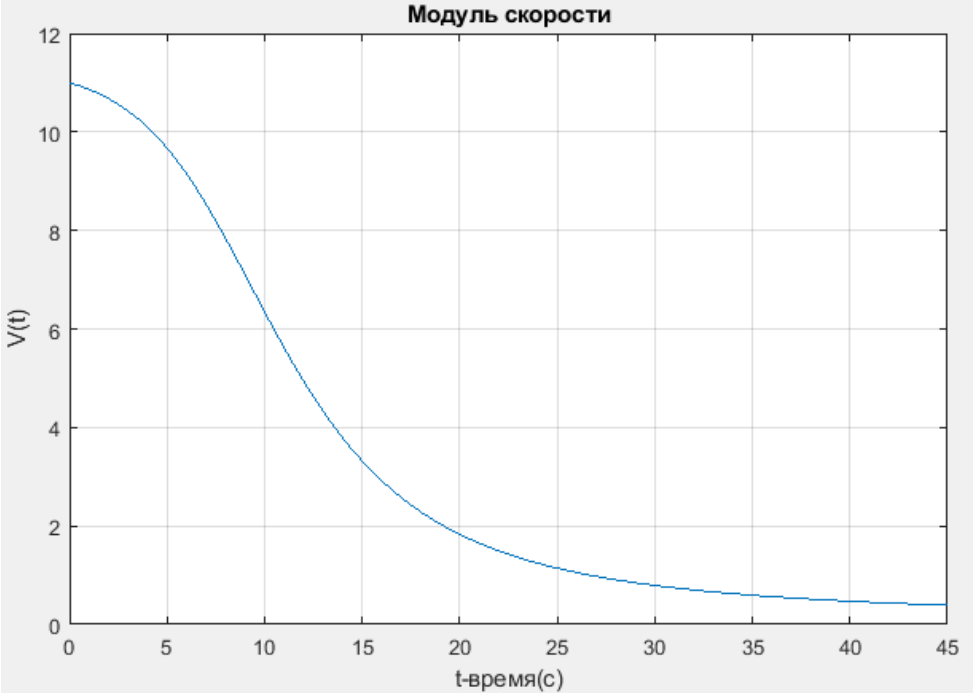

Рисунок 2. График изменения скорости в Matlab.

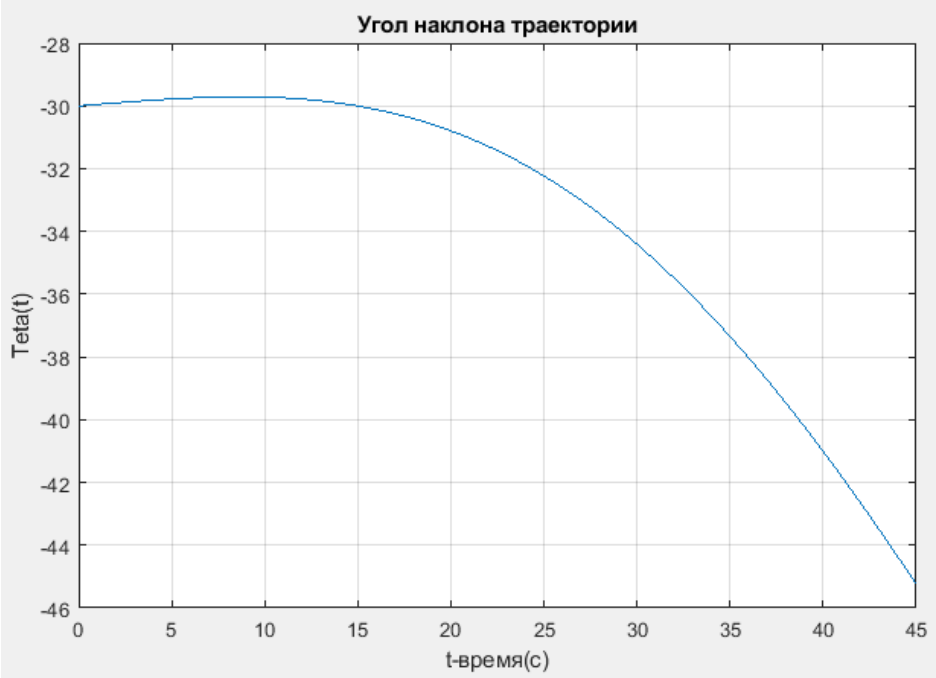

Рисунок 3. График изменения угла наклона траектории в Matlab.

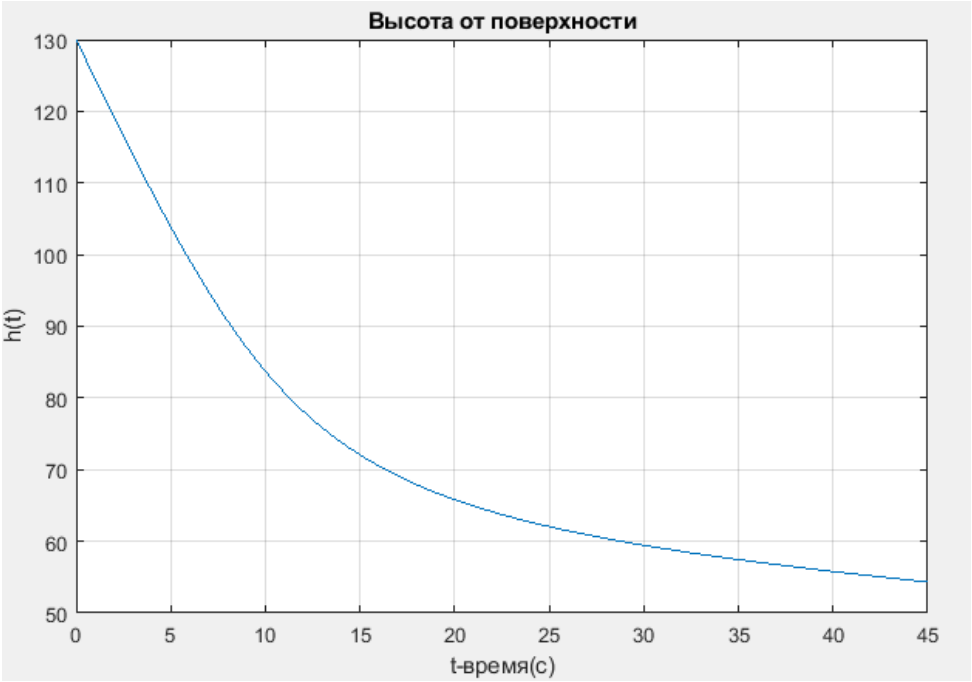

Рисунок 4. График изменения высоты в Matlab.

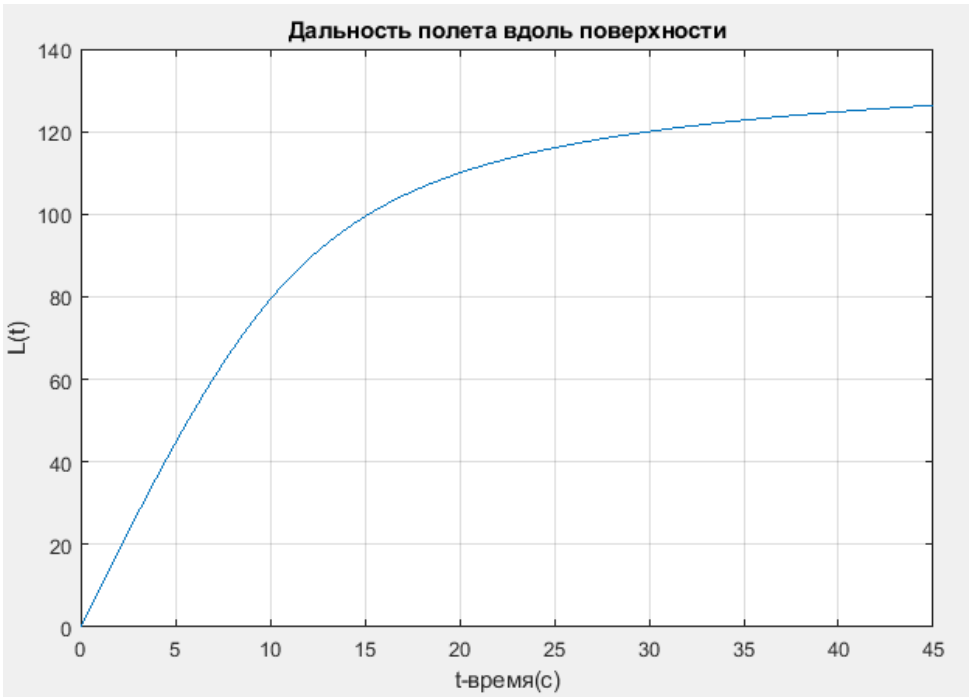

Рисунок 5. График дальности в Matlab.

Результаты полученные в Python

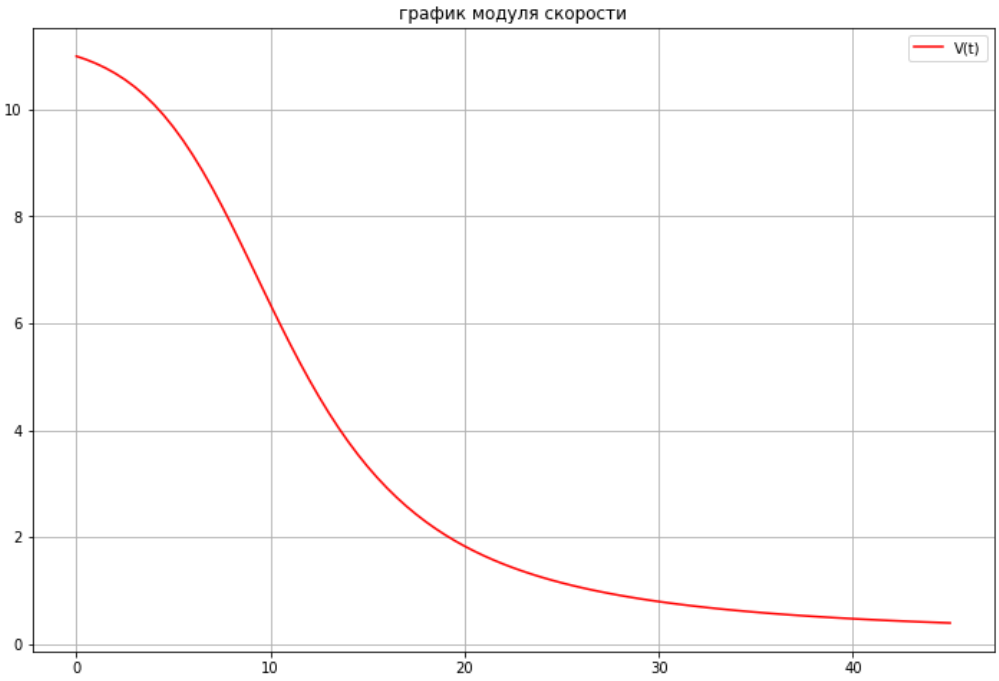

Рисунок 6. График изменения скорости в Python.

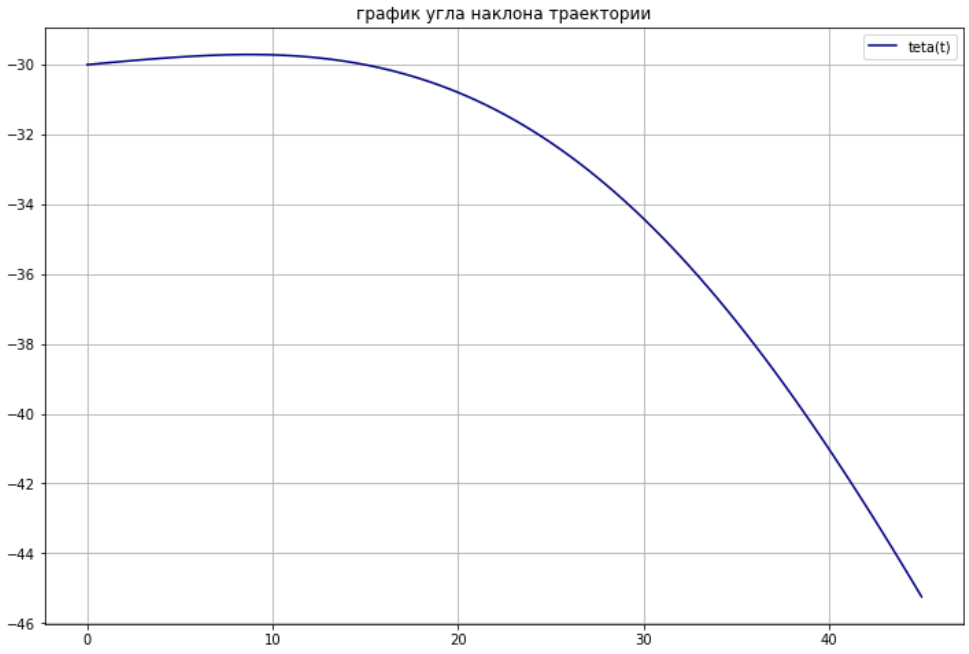

Рисунок 7. График изменения угла наклона траектории в Python.

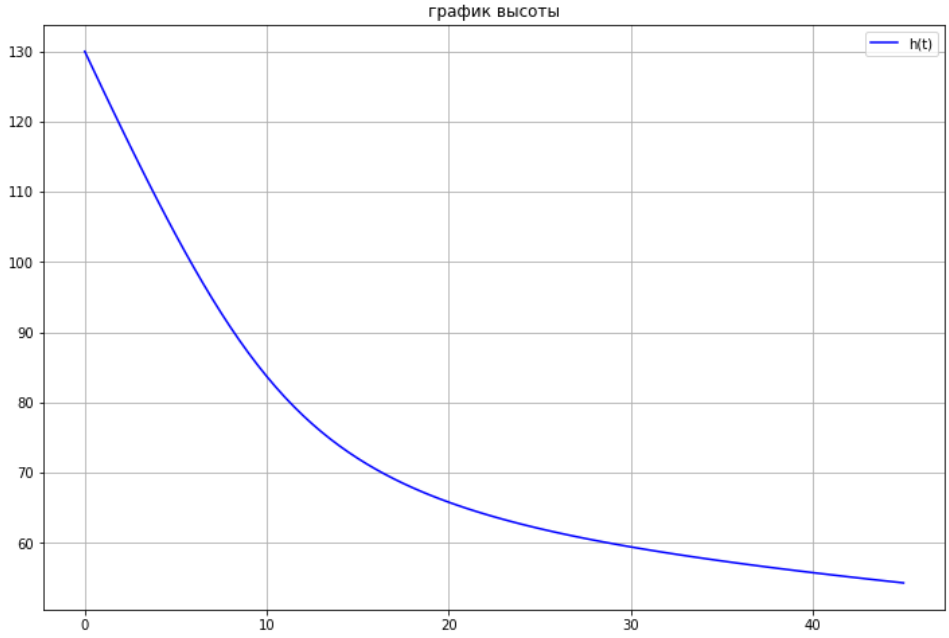

Рисунок 8. График изменения высоты в Python.

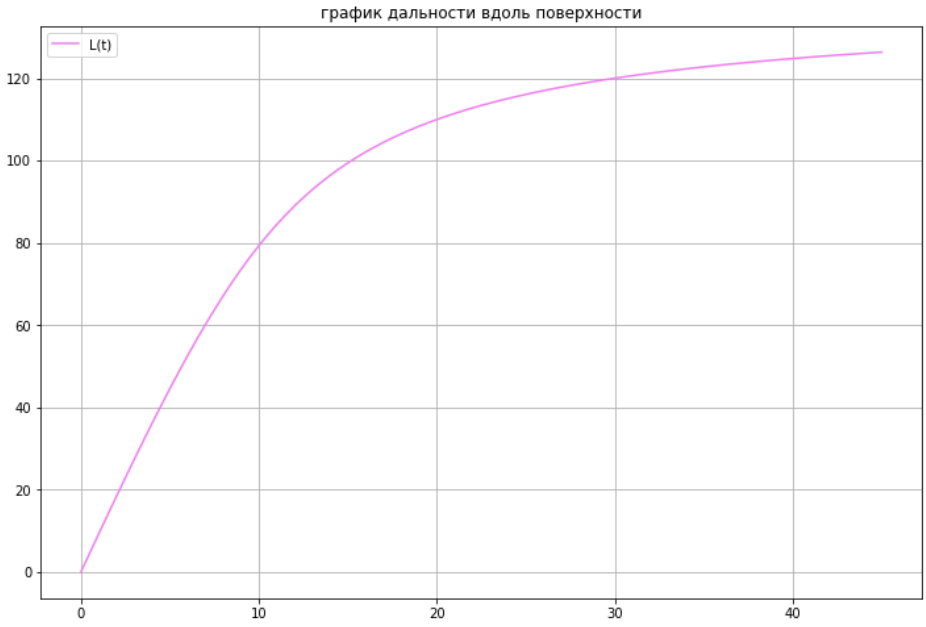

Рисунок 9. График дальности в Python.

Вывод.

В ходе баллистического спуска в процессе торможения в атмосфере скорость КА уменьшилась с 11 км/с до 392 м/с (0.392 км/с), высота со 130 км до 54.3 км, угол наклона траектории в конце баллистического спуска составляет −45.245°, пройденная дальность вдоль поверхности Венеры составила 126.36 км. Предполагается, что дальнейший спуск будет осуществлён с помощью парашютной системы. Достигнуты условия для начала участка парашютного спуска.

Приложение А. ПО на Matlab

```
%Этап баллистического спуска
pi=3.14159;
Cx=0.015;%аэродинамический коэффициент
S=pi*(2.4^2)/4;%площадь миделевого сечения
V=11000;%модуль скорости входа в атмосферу
mu=324853.4 *10^9;%гравитационный параметр Венеры
Re=6052000;%радиус Венеры
h=130000;%высота входа в атмосферу
m=600;%масса аппарата
r=Re+h;%расстояние от апарата до начала координат
g=mu/(r^2)%ускорение свободного падения Венеры на высоте h
teta=-30*(pi/180);%угол входа в атмосферу
rro0=67;%плотность атмосферы Венеры у поверхности(кг/м^3)
Hatm=15900;%"высота однородной атмосферы" Венеры
L=0%дальность по поверхности

t0=0;
t=t0;%время
dt=0.5;% шаг интегрирования

%интегрирование системы ДУ методом Рунге-Кутты 4-го порядка
K11=0;K21=0;K31=0;K41=0;
K1=[K11;K21;K31;K41];

K12=0;K22=0;K32=0;K42=0;
K2=[K12;K22;K32;K42];

K13=0;K23=0;K33=0;K43=0;
K3=[K13;K23;K33;K43];

K14=0;K24=0;K34=0;K44=0;
K4=[K14;K24;K34;K44];
i=1;
for t=0:dt:45
   rro=rro0*exp(-h/Hatm);%изменение плотности атмосферы с высотой
   r=Re+h; %расстояние от центра планеты до спускаемого аппарата
   g=mu/(r^2);%изменение ускорения свободного падения с высотой g(r)=g(h)

   nx=Cx*S*rro*V^2/(2*m*g);%перегрузка

   K1(1)=-((Cx*S*rro*V^2)/(2*m))-g*sin(teta);%для V
   K1(2)=(cos(teta)/V)*((V^2/r)-g);%для teta
   K1(3)=V*sin(teta);%для h
   K1(4)=V*(Re/r)*cos(teta);% для L

   K2(1)=-((Cx*S*rro*(V+0.5*K1(1)*dt)^2)/(2*m))-g*sin(teta+0.5*K1(2)*dt);
   K2(2)=(cos(teta+0.5*K1(2)*dt)/(V+0.5*K1(1)*dt))*(((V+0.5*K1(1)*dt)^2/r) -
g);
   K2(3)=(V+0.5*K1(1)*dt)*sin(teta+0.5*K1(2)*dt);
   K2(4)=(V+0.5*K1(1)*dt)*(Re/r)*cos(teta+0.5*K1(2)*dt);



   K3(1)=-((Cx*S*rro*(V+0.5*K2(1)*dt)^2)/(2*m))-g*sin(teta+0.5*K2(2)*dt);
   K3(2)=(cos(teta+0.5*K2(2)*dt)/(V+0.5*K2(1)*dt))*(((V+0.5*K2(1)*dt)^2/r) -
g);
   K3(3)=(V+0.5*K2(1)*dt)*sin(teta+0.5*K2(2)*dt);
   K3(4)=(V+0.5*K2(1)*dt)*(Re/r)*cos(teta+0.5*K2(2)*dt);

   K4(1)=-((Cx*S*rro*(V+K3(1)*dt)^2)/(2*m))-g*sin(teta+K3(2)*dt);
```

```
    K4(2)=(cos(teta+K3(2)*dt)/(V+K3(1)*dt))*(((V+K3(1)*dt)^2/r) -g);
    K4(3)=(V+K3(1)*dt)*sin(teta+K3(2)*dt);
    K4(4)=(V+K3(1)*dt)*(Re/r)*cos(teta+K3(2)*dt);

    t;
    Vkilm=V/1000
    hkilm=h/1000;
    tetagr=teta*180/pi;
    Lkilm=L/1000;
    tstr(i)=t
    vstr(i)=Vkilm
    hstr(i)=hkilm
    tetastr(i)=tetagr
    lstr(i)=Lkilm
    i=i+1
    V=V+dt*(K1(1)+2*K2(1)+2*K3(1)+K4(1))/6;
    teta=teta+dt*(K1(2)+2*K2(2)+2*K3(2)+K4(2))/6;
    h=h+dt*(K1(3)+2*K2(3)+2*K3(3)+K4(3))/6;
    L=L+dt*(K1(4)+2*K2(4)+2*K3(4)+K4(4))/6;
end

figure(1)
plot(tstr,vstr),grid %Построение графика модуля скороости
xlabel('t-время(c)')
ylabel('V(t)')
title ('Модуль скорости')

figure(2)
plot(tstr,hstr),grid %Построение графика высоты от поверхности
xlabel('t-время(c)')
ylabel('h(t)')
title ('Высота от поверхности')

figure(3)
plot(tstr,tetastr),grid %Построение графика угла наклона траектории
xlabel('t-время(c)')
ylabel('Teta(t)')
title ('Угол наклона траектории')

figure(4)
plot(tstr,lstr),grid %Построение графика дальности полета вдоль поверхности
xlabel('t-время(c)')
ylabel('L(t)')
title ('Дальность полета вдоль поверхности')
```

Приложение Б. ПО на Python

```
#Этап баллистического спуска
import math
import matplotlib.pyplot as plt
pi=3.14159
Cx=0.015#?
S=pi*(2.4**2)/4#площадь миделевого сечения
V=11000#модуль скорости входа в атмосферу
mu=324853.4 *10**9#гравитационный параметр Венеры
Re=6052000#радиус Венеры
h=130000#высота входа в атмосферу
m=600#масса аппарата
r=Re+h#расстояние от аппарата до начала координат
g=mu/(r**2)#ускорение свободного падения Венеры на высоте h
teta=-30*(pi/180)#угол входа в атмосферу
rro0=67#плотность атмосферы Венеры у поверхности(кг/м^3)
Hatm=15900#высота однородной атмосферы Венеры
L=0#дальность по поверхности
t0=0
t=t0
dt=0.5
K1=[0,0,0,0]
K2=[0,0,0,0]
K3=[0,0,0,0]
K4=[0,0,0,0]
Lm=[]
tetam=[]
Vm=[]
hm=[]
tm=[]
while t<=45:

    rro=rro0*math.exp(-h/Hatm)
    r=Re+h
    g=mu/(r**2)
    K1[0]=-((Cx*S*rro*V**2)/(2*m))-g*math.sin(teta) #для V
    K1[1]=(math.cos(teta)/V)*((V**2/r)-g) #для teta
    K1[2]=V*math.sin(teta) #для h
    K1[3]=V*(Re/r)*math.cos(teta) # для L

    K2[0]=-((Cx*S*rro*(V+0.5*K1[0]*dt)**2)/(2*m))-
g*math.sin(teta+0.5*K1[1]*dt);

K2[1]=(math.cos(teta+0.5*K1[1]*dt)/(V+0.5*K1[0]*dt))*(((V+0.5*K1[0]*dt)**2/r)
-g);
    K2[2]=(V+0.5*K1[0]*dt)*math.sin(teta+0.5*K1[1]*dt);
    K2[3]=(V+0.5*K1[0]*dt)*(Re/r)*math.cos(teta+0.5*K1[1]*dt);


    K3[0]=-((Cx*S*rro*(V+0.5*K2[0]*dt)**2)/(2*m))-
g*math.sin(teta+0.5*K2[1]*dt);

K3[1]=(math.cos(teta+0.5*K2[1]*dt)/(V+0.5*K2[0]*dt))*(((V+0.5*K2[0]*dt)**2/r)
-g);
    K3[2]=(V+0.5*K2[0]*dt)*math.sin(teta+0.5*K2[1]*dt);
    K3[3]=(V+0.5*K2[0]*dt)*(Re/r)*math.cos(teta+0.5*K2[1]*dt);

    K4[0]=-((Cx*S*rro*(V+K3[0]*dt)**2)/(2*m))-g*math.sin(teta+K3[1]*dt);
    K4[1]=(math.cos(teta+K3[1]*dt)/(V+K3[0]*dt))*(((V+K3[0]*dt)**2/r) -g);
    K4[2]=(V+K3[0]*dt)*math.sin(teta+K3[1]*dt);
    K4[3]=(V+K3[0]*dt)*(Re/r)*math.cos(teta+K3[1]*dt);

    Lkilm=L/1000
```

```
    Vkilm=V/1000;
    hkilm=h/1000;
    tetagr=teta*180/pi;

    tm.append(t)
    Lm.append(Lkilm)
    Vm.append(Vkilm)
    hm.append(hkilm)
    tetam.append(tetagr)

    V=V+dt*(K1[0]+2*K2[0]+2*K3[0]+K4[0])/6
    teta=teta+dt*(K1[1]+2*K2[1]+2*K3[1]+K4[1])/6
    h=h+dt*(K1[2]+2*K2[2]+2*K3[2]+K4[2])/6
    L=L+dt*(K1[3]+2*K2[3]+2*K3[3]+K4[3])/6
    print('t= ',t, '   Vkm= ', Vkilm, '   hkm= ',hkilm,'   tetagr= ',tetagr,'
Lkm= ',Lkilm)
    t+=dt

plt.figure(figsize=(12,8))
plt.plot(tm, Vm,"red",label="V(t)")
plt.grid(True);
plt.legend()
plt.title("график модуля скорости")

plt.figure(figsize=(12,8))
plt.plot(tm, hm,"blue",label="h(t)")
plt.grid(True);
plt.legend()
plt.title("график высоты")

plt.figure(figsize=(12,8))
plt.plot(tm, tetam,"navy",label="teta(t)")
plt.grid(True);
plt.legend()
plt.title("график угла наклона траектории")


plt.figure(figsize=(12,8))
plt.plot(tm, Lm,"violet",label="L(t)")
plt.grid(True);
plt.legend()
plt.title("график дальности вдоль поверхности")
```

Приложение С. Метод Рунге-Кутта с автовыбором шага.

```
import numpy as np
import matplotlib.pyplot as plt
pi=3.14159
Cx=0.015#?
S=pi*(2.4**2)/4#площадь миделевого сечения
V=11000#модуль скорости входа в атмосферу
mu=324853.4 *10**9#гравитационный параметр Венеры
Re=6052000#радиус Венеры
h=130000#высота входа в атмосферу
m=600#масса аппарата
r=Re+h#расстояние от аппарата до начала координат
g=mu/(r**2)#ускорение свободного падения Венеры на высоте h
teta=-30*(pi/180)#угол входа в атмосферу
rro0=67#плотность атмосферы Венеры у поверхности(кг/м^3)
Hatm=15900#высота однородной атмосферы Венеры
L=0#дальность по поверхности
t0=0
```

```
t=t0
dt=0.5
K1=[0,0,0,0]
K2=[0,0,0,0]
K3=[0,0,0,0]
K4=[0,0,0,0]
Lm=[]
tetam=[]
Vm=[]
hm=[]
tm=[]
def partstep(V,teta,h,L,dt):
    rro=rro0*np.exp(-h/Hatm)
    r=Re+h
    g=mu/(r**2)
    K1[0]=-Cx*S*(rro*V**2)/(2*m)-g*np.sin(teta);
    K1[1]=(1.0/V)*(((V**2)/r)*np.cos(teta)-g*np.cos(teta))
    K1[2]=V*np.sin(teta)
    K1[3]=V*(Re/r)*np.cos(teta)

    K2[0]=-Cx*S*(rro*(V+0.5*dt*K1[0])**2)/(2*m)-g*np.sin(teta+0.5*dt*K1[1])

K2[1]=(1.0/(V+0.5*dt*K1[0]))*((((V+0.5*dt*K1[0])**2)/r)*np.cos(teta+0.5*dt*K1
[1])-g*np.cos(teta+0.5*dt*K1[1]))
    K2[2]=(V+0.5*dt*K1[0])*np.sin(teta+0.5*dt*K1[1]);
    K2[3]=(V+0.5*dt*K1[0])*(Re/r)*np.cos(teta+0.5*dt*K1[1]);

    K3[0]=-Cx*S*(rro*(V+0.5*dt*K2[0])**2)/(2*m)-g*np.sin(teta+0.5*dt*K2[1])

K3[1]=(1.0/(V+0.5*dt*K2[0]))*((((V+0.5*dt*K2[0])**2)/r)*np.cos(teta+0.5*dt*K2
[1])-g*np.cos(teta+0.5*dt*K2[1]))
    K3[2]=(V+0.5*dt*K2[0])*np.sin(teta+0.5*dt*K2[1])
    K3[3]=(V+0.5*dt*K2[0])*(Re/r)*np.cos(teta+0.5*dt*K2[1]);

    K4[0]=-Cx*S*(rro*(V+dt*K3[0])**2)/(2*m)-g*np.sin(teta+dt*K3[1])
    K4[1]=(1.0/(V+dt*K3[0]))*((((V+dt*K3[0])**2)/r)*np.cos(teta+dt*K3[1])-
g*np.cos(teta+dt*K3[1]))
    K4[2]=(V+dt*K3[0])*np.sin(teta+dt*K3[1])
    K4[3]=(V+dt*K3[0])*(Re/r)*np.cos(teta+dt*K3[1])

    V=V+dt*(K1[0]+2*K2[0]+2*K3[0]+K4[0])/6
    teta=teta+dt*(K1[1]+2*K2[1]+2*K3[1]+K4[1])/6
    h=h+dt*(K1[2]+2*K2[2]+2*K3[2]+K4[2])/6
    L=L+dt*(K1[3]+2*K2[3]+2*K3[3]+K4[3])/6
    return(V,teta,h,L)

Vect1=[]
Vect2=[]
eps=[];
eps.append(1.0)#требования точности по величине скорости
eps.append(0.0001)#требования точности по углу наклона раектории
eps.append(2.0)#требования точности по высоте
eps.append(5.0)#требования точности по дальности
step=dt;
bol=False;
bolbuf=True;
itercountstep=0;
file=open('results.txt','w')
file.write("t\tV\tteta\th\tL\n")
```

```
while t<=45.0:
    step=dt;
    print("stepred=",step)
    #данные полученные на предыдущем шаге (в первый раз- начальные данные)
    h0=h
    L0=L
    teta0=teta
    V0=V
    V1,teta1,h1,L1=partstep(V0, teta0, h0, L0, step)#вычисляем с начальным
шагом
    V2,teta2,h2,L2=partstep(V0, teta0, h0, L0, step/2.0)# вычисляем с вдвое
меньшим шагом
    #заполняем массивы результатов
    Vect1.append(V1);Vect1.append(teta1);Vect1.append(h1);Vect1.append(L1);
    Vect2.append(V2);Vect2.append(teta2);Vect2.append(h2);Vect2.append(L2);

    #сравнение разности массивов результатов с шагами dt и dt/2.0 в с
массивом погрешностей
    for i in range(len(Vect1)):
        if(abs(Vect1[i]-Vect2[i])>=eps[i]):
            bol=True;#если bol=True, значит надо уменьшать шаг, насколько
пока не ясно
    Vect1.clear();#очищаем массивы результатов (потом снова будем их
заполнять)
    Vect2.clear();
    if(bol==True):#если надо уменьшать шаг...
        itercountstep=0;#счётчик итераций уменьшения шага
        #цикл уменьшания шага
        while(bol==True and itercountstep<=1000):#пока шаг требуется
уменьшать и пока максимальное число итераций уменьшения шага не превышено
            bolbuf=True;#другая логическая переменная, показывающая когда
надо выйти из цикла с уменьшением шага
            V1,teta1,h1,L1=partstep(V0, teta0, h0, L0, step)#первый раз dt
            V2,teta2,h2,L2=partstep(V0, teta0, h0, L0, step/2.0)#первый раз
dt/2.0

Vect1.append(V1);Vect1.append(teta1);Vect1.append(h1);Vect1.append(L1);#первы
й раз результаты с шагом dt (вообще с шагом step)

Vect2.append(V2);Vect2.append(teta2);Vect2.append(h2);Vect2.append(L2);#первы
й раз результаты с шагом dt/2.0 (вообще с шагом step/2.0)
            step=step/2.0;#при первом проходе dt/2.0
            #сравнение разности массивов результато с шагами step и step/2.0
в с массивом погрешностей
            for i in range(0,len(Vect1)):
                if(abs(Vect1[i]-Vect2[i])>=eps[i]):
                    bol=True;
                    bolbuf=False;#эта переменная false если хотя бы 1 элемент
сравнения больше eps[i], т.е. разность больше допустимой
            if(bolbuf==True):#если ранее bolbuf не стал false, значит пора
выходить из цикла уменьшения шага
                    bol=False;#для выхода из цикла уменьшения шага
                    #запоминаем актуальные значения
                    V=V1
                    teta=teta1
                    h=h1
                    L=L1
                    step=2*step;#первый раз dt (вообще step, используемый для
расчёта V1,teta1,h1,L1)
```

```
            itercountstep+=1;
            print(itercountstep)
            Vect1.clear();
            Vect2.clear();

    elif(bol==False):
            V=V1
            teta=teta1
            h=h1
            L=L1
            Vect1.clear();
            Vect2.clear();
    Lkilm=L/1000.0
    Vkilm=V/1000.0;
    hkilm=h/1000.0;
    tetagr=teta*180/pi;

    tm.append(t)
    Lm.append(Lkilm)
    Vm.append(Vkilm)
    hm.append(hkilm)
    tetam.append(tetagr)

file.write(str(t)+"\t"+f'{V:.5f}'+"\t"+f'{tetagr:.5f}'+"\t"+f'{h:.5f}'+"\t"+f
'{L:.5f}'+"\n");
    print("steppost=",step)
    t+=step;

file.close()

plt.figure(figsize=(12,8))
plt.plot(tm, Vm,"red",label="V(t)")
plt.grid(True);
plt.legend()
plt.title("график модуля скорости")

plt.figure(figsize=(12,8))
plt.plot(tm, hm,"blue",label="h(t)")
plt.grid(True);
plt.legend()
plt.title("график высоты")

plt.figure(figsize=(12,8))
plt.plot(tm, tetam,"navy",label="teta(t)")
plt.grid(True);
plt.legend()
plt.title("график угла наклона траектории")


plt.figure(figsize=(12,8))
plt.plot(tm, Lm,"violet",label="L(t)")
plt.grid(True);
plt.legend()
plt.title("график дальности вдоль поверхности")
```