Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дисциплина**

«Теоретическая механика и основы математического моделирования»

**Лабораторная работа №3**

Выполнил: студент группы М8О-206Б-20

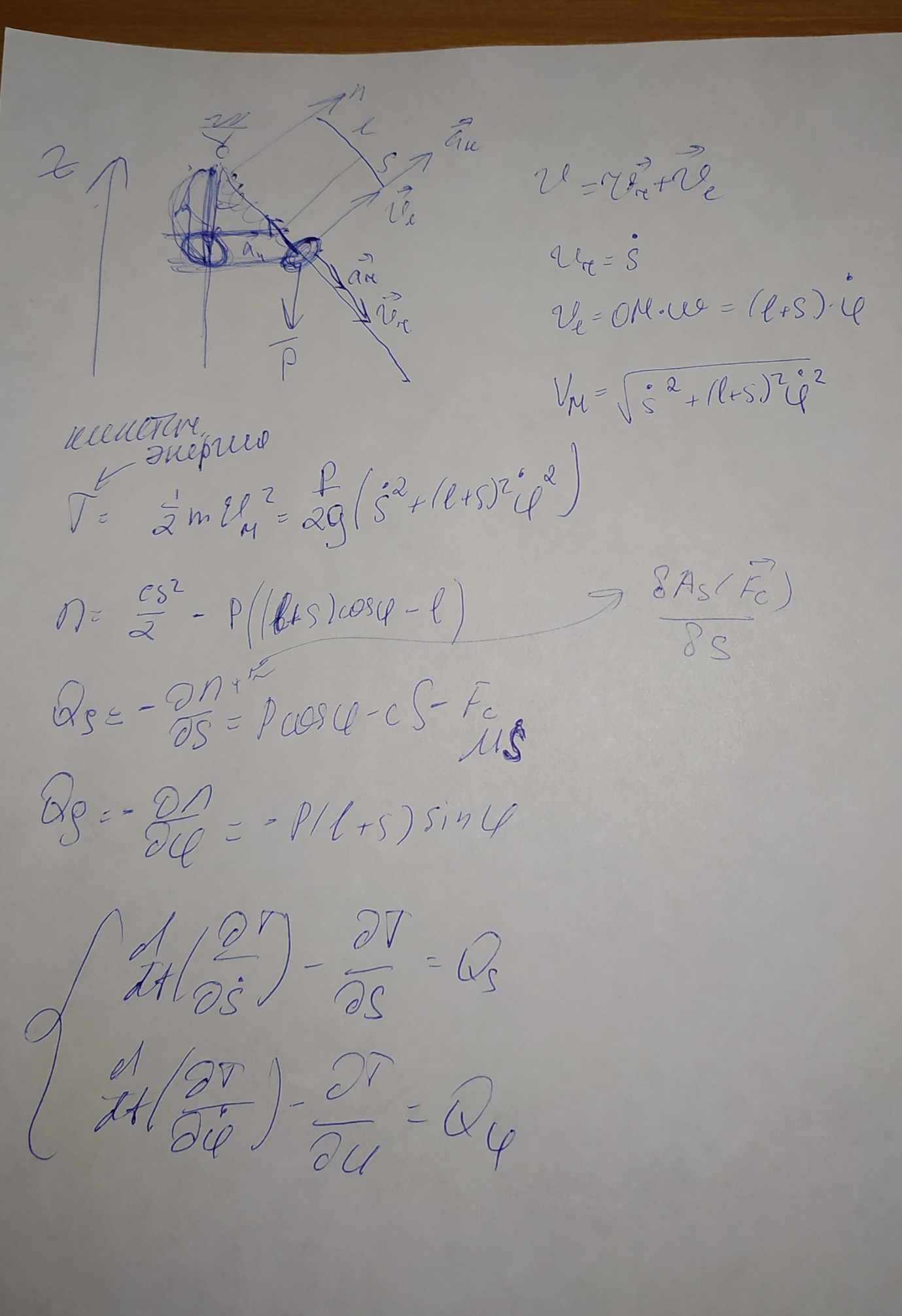
Почечура Артемий Андреевич

Вариант: 18

Москва

2021

**Вывод формул**



**Код программы**

import numpy as np

import sympy as sp

import math

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

from scipy.integrate import odeint

#y[2] = ds

#y[3] = dphi

#y[1] = phi

#y[0] = s

def odesys(y, t, m, g, P, c, lens):

dy = np.zeros(4)

dy[0] = y[2]

dy[1] = y[3]

a11 = 1

a12 = 0

a21 = 0

a22 = lens+y[0]

b1 = g\*np.cos(y[1])+(lens+y[0])\*y[3]\*\*2-c\*g/P\*y[0]-m\*g/P\*y[2]

b2 = -y[2]\*y[3]-g\*(lens+y[0])\*np.sin(y[1])

dy[2] = (b1\*a22 - b2\*a12)/(a11\*a22 - a12\*a21)

dy[3] = (b2\*a11 - b1\*a21)/(a11\*a22 - a12\*a21)

return dy

m = 1

g = 9.81

P = 20

c = 10

lens = 4

t\_fin = 20

t=np.linspace(0,t\_fin,1001)

s0 = 0

phi0 = 0.5

ds0 = 0

dphi0 = 0

y0 = [s0, phi0, ds0, dphi0]

Y = odeint(odesys, y0, t, (m, g, P, c, lens))

s = Y[:, 0]

phi = Y[:, 1]

ds = Y[:, 2]

dphi = Y[:, 3]

fig\_for\_graphs = plt.figure(figsize=[13,7])

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,2,1)

ax\_for\_graphs.plot(t,s,color='blue')

ax\_for\_graphs.set\_title("s(t)")

ax\_for\_graphs.set(xlim=[0,t\_fin])

ax\_for\_graphs.grid(True)

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,2,2)

ax\_for\_graphs.plot(t,phi,color='red')

ax\_for\_graphs.set\_title('phi(t)')

ax\_for\_graphs.set(xlim=[0,t\_fin])

ax\_for\_graphs.grid(True)

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,2,3)

ax\_for\_graphs.plot(t,ds,color='green')

ax\_for\_graphs.set\_title("s'(t)")

ax\_for\_graphs.set(xlim=[0,t\_fin])

ax\_for\_graphs.grid(True)

ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,2,4)

ax\_for\_graphs.plot(t,dphi,color='black')

ax\_for\_graphs.set\_title('phi\'(t)')

ax\_for\_graphs.set(xlim=[0,t\_fin])

ax\_for\_graphs.grid(True)

l = 9#lens+s

m = 0.2

a = phi

p1xn = l\*np.sin(a)-m/2\*np.cos(a)

p2xn = l\*np.sin(a)+m/2\*np.cos(a)

p3xn = m/2\*np.cos(a)

p4xn = -m/2\*np.cos(a)

p1yn = l-l\*np.cos(a)-m/2\*np.sin(a)

p2yn = l-l\*np.cos(a)+m/2\*np.sin(a)

p3yn = l+m/2\*np.sin(a)

p4yn = l-m/2\*np.sin(a)

h=0.2

d=1

xn=s\*np.sin(a)

yn=l-s\*np.cos(a)

vs = ds

vxsn=vs\*np.sin(a)

vysn=-vs\*np.cos(a)

w = dphi

vxn = w \* np.cos(a) \*s

vyn = w \* np.sin(a) \*s

vr = np.sqrt((vxsn+vxn)\*(vxsn+vxn)+(vysn+vyn)\*(vysn+vyn))

vs = np.sqrt(vxsn\*vxsn+vysn\*vysn)

v = np.sqrt(vxn\*vxn+vyn\*vyn)

an=a

def cir(xl, yl, r):

angle = np.linspace( 0 , 2 \* np.pi , 150 )

px = xl + r \* np.cos( angle )

py = yl + r \* np.sin( angle )

return px, py

def pr(x1, y1, l, h, a):

angle = np.linspace( 0 , 10 \* np.pi , 150 )

x1p = np.linspace( 0 , x1-h/2\*np.sin(a) , 150 )

y1p = np.linspace( l , y1+h/2\*np.cos(a) , 150 )

x = x1p+np.sin(angle)\*0.5\*np.cos(a)

y = y1p+np.sin(angle)\*0.5\*np.sin(a)

return x, y

def ring(xl, yl, l, h, a):

Mx = xl + l/2\*np.cos(a)

My = yl + l/2\*np.sin(a)

Nx = xl - l/2\*np.cos(a)

Ny = yl - l/2\*np.sin(a)

PX=[Mx - h/2\*np.sin(a), Mx + h/2\*np.sin(a), Nx + h/2\*np.sin(a), Nx - h/2\*np.sin(a),Mx - h/2\*np.sin(a)]

PY=[My + h/2\*np.cos(a), My - h/2\*np.cos(a), Ny - h/2\*np.cos(a), Ny + h/2\*np.cos(a),My + h/2\*np.cos(a)]

return PX, PY

def rotate(x, y, a):

x\_rotated = x \* np.cos(a) - y \* np.sin(a)

y\_rotated = x \* np.sin(a) + y \* np.cos(a)

return x\_rotated, y\_rotated

fig = plt.figure(figsize=[13,7])

ax = fig.add\_subplot(1,1,1)

ax.axis('equal')

ax.set(xlim=[-l-5, l+5], ylim=[-3, m+l+2])

cx, cy = cir(0, l, m/2)

ax.plot(cx, cy, 'black')

p1s = ax.plot([p1xn[0],p2xn[0]],[p1yn[0],p2yn[0]],'black')[0]

p2s = ax.plot([p2xn[0],p3xn[0]],[p2yn[0],p3yn[0]],'black')[0]

p4s = ax.plot([p4xn[0],p1xn[0]],[p4yn[0],p1yn[0]],'black')[0]

RX, RY = ring(xn[0],yn[0],d,h,an[0])

rs = ax.plot(RX, RY, 'red')[0]

htr=0.5

tr = ax.plot([0,-htr,htr,0], [l,l+htr,l+htr,l], 'black')

cx,cy = pr(xn[0],yn[0],l,h,an[0])

cc, = ax.plot(cx,cy,'blue')

vvs=ax.plot([xn[0],xn[0]+vxsn[0]],[yn[0],yn[0]+vysn[0]],'green')[0]

vv=ax.plot([xn[0],xn[0]+vxn[0]],[yn[0],yn[0]+vyn[0]],'y')[0]

vres=ax.plot([xn[0],xn[0]+vxn[0]+vxsn[0]],[yn[0],yn[0]+vyn[0]+vysn[0]],'m')[0]

Phis = math.atan2(vysn[0], vxsn[0])

V\_arrow\_x = np.array([-vs[0]\*0.1, 0.0, -vs[0]\*0.1], dtype=float)

V\_arrow\_y = np.array([vs[0]\*0.05, 0.0, -vs[0]\*0.05], dtype=float)

V\_arrow\_rotx, V\_arrow\_roty = rotate(V\_arrow\_x, V\_arrow\_y, Phis)

V\_arrow, = ax.plot(xn[0] + vxsn[0] + V\_arrow\_rotx, yn[0] + vysn[0] + V\_arrow\_roty, color="green")

Phi = math.atan2(vyn[0], vxn[0])

Vk\_arrow\_x = np.array([-v[0]\*0.1, 0.0, -v[0]\*0.1], dtype=float)

Vk\_arrow\_y = np.array([v[0]\*0.05, 0.0, -v[0]\*0.05], dtype=float)

Vk\_arrow\_rotx, Vk\_arrow\_roty = rotate(Vk\_arrow\_x, Vk\_arrow\_y, Phi)

Vk\_arrow, = ax.plot(xn[0] + vxn[0] + Vk\_arrow\_rotx, yn[0] + vyn[0] + Vk\_arrow\_roty, color="y")

Phir = math.atan2(vysn[0]+vyn[0], vxsn[0]+vxn[0])

Vr\_arrow\_x = np.array([-vr[0]\*0.1, 0.0, -vr[0]\*0.1], dtype=float)

Vr\_arrow\_y = np.array([vr[0]\*0.05, 0.0, -vr[0]\*0.05], dtype=float)

Vr\_arrow\_rotx, Vr\_arrow\_roty = rotate(Vr\_arrow\_x, Vr\_arrow\_y, Phir)

Vr\_arrow, = ax.plot(xn[0] + vxsn[0] + vxn[0] + Vr\_arrow\_rotx, yn[0] + vysn[0]+ vyn[0] + Vr\_arrow\_roty, color="m")

def cha(i):

p1s.set\_data([p1xn[i],p2xn[i]],[p1yn[i],p2yn[i]])

p2s.set\_data([p2xn[i],p3xn[i]],[p2yn[i],p3yn[i]])

p4s.set\_data([p4xn[i],p1xn[i]],[p4yn[i],p1yn[i]])

RX, RY = ring(xn[i],yn[i],d,h,an[i])

rs.set\_data(RX, RY)

cx, cy = pr(xn[i],yn[i],l,h,an[i])

cc.set\_data(cx, cy)

vvs.set\_data([xn[i],xn[i]+vxsn[i]],[yn[i],yn[i]+vysn[i]])

vv.set\_data([xn[i],xn[i]+vxn[i]],[yn[i],yn[i]+vyn[i]])

vres.set\_data([xn[i],xn[i]+vxn[i]+vxsn[i]],[yn[i],yn[i]+vyn[i]+vysn[i]])

Phis = math.atan2(vysn[i], vxsn[i])

V\_arrow\_x = np.array([-vs[i]\*0.1, 0.0, -vs[i]\*0.1], dtype=float)

V\_arrow\_y = np.array([vs[i]\*0.05, 0.0, -vs[i]\*0.05], dtype=float)

V\_arrow\_rotx, V\_arrow\_roty = rotate(V\_arrow\_x, V\_arrow\_y, Phis)

V\_arrow.set\_data(xn[i] + vxsn[i] + V\_arrow\_rotx, yn[i] + vysn[i] + V\_arrow\_roty)

Phi = math.atan2(vyn[i], vxn[i])

Vk\_arrow\_x = np.array([-v[i]\*0.1, 0.0, -v[i]\*0.1], dtype=float)

Vk\_arrow\_y = np.array([v[i]\*0.05, 0.0, -v[i]\*0.05], dtype=float)

Vk\_arrow\_rotx, Vk\_arrow\_roty = rotate(Vk\_arrow\_x, Vk\_arrow\_y, Phi)

Vk\_arrow.set\_data(xn[i] + vxn[i] + Vk\_arrow\_rotx, yn[i] + vyn[i] + Vk\_arrow\_roty)

Phir = math.atan2(vysn[i]+vyn[i], vxsn[i]+vxn[i])

Vr\_arrow\_x = np.array([-vr[i]\*0.1, 0.0, -vr[i]\*0.1], dtype=float)

Vr\_arrow\_y = np.array([vr[i]\*0.05, 0.0, -vr[i]\*0.05], dtype=float)

Vr\_arrow\_rotx, Vr\_arrow\_roty = rotate(Vr\_arrow\_x, Vr\_arrow\_y, Phir)

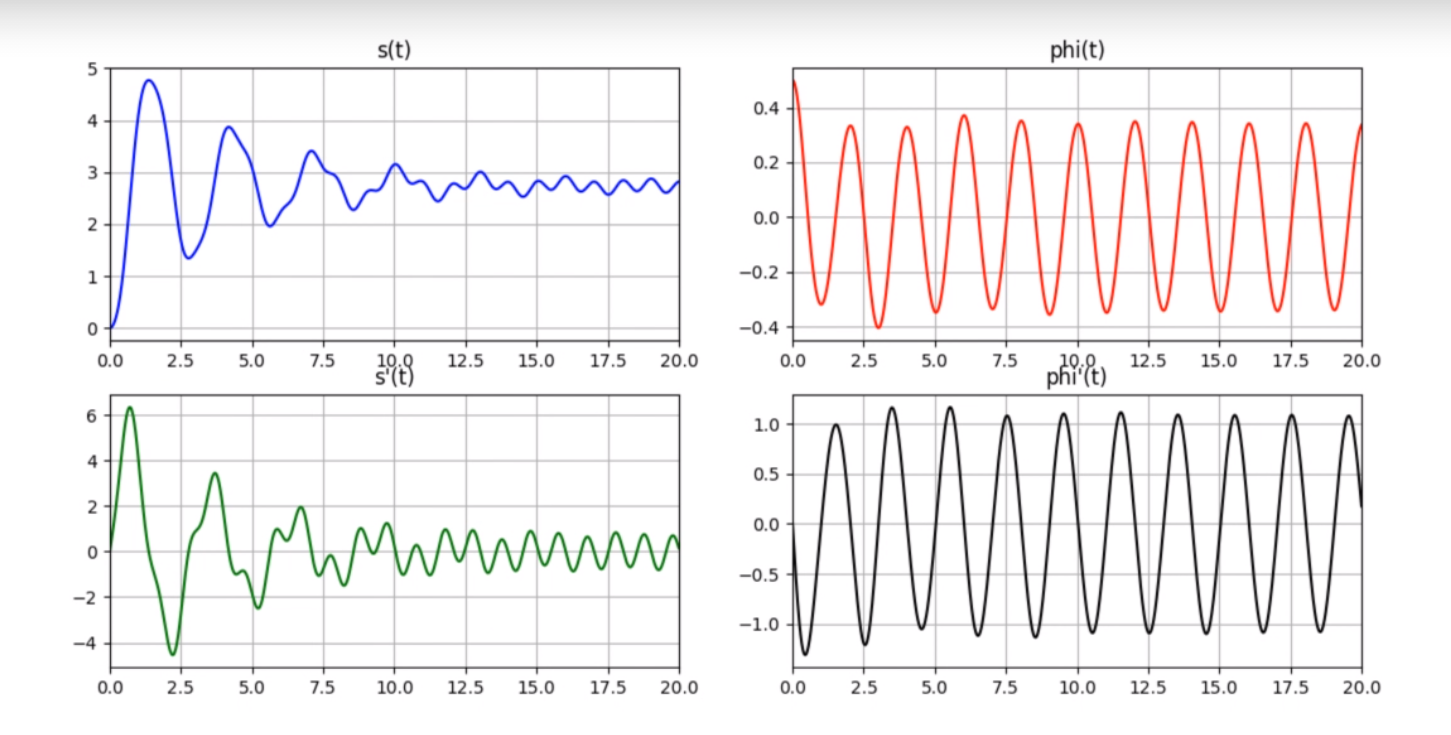
Vr\_arrow.set\_data(xn[i] + vxsn[i] + vxn[i] + Vr\_arrow\_rotx, yn[i] + vysn[i]+ vyn[i] + Vr\_arrow\_roty)

return [p1s], [p2s], [p4s], [rs], [cc], [vvs]

anima = FuncAnimation(fig,cha,frames=len(t),interval=10)

plt.show()

**Графики функций**

****