**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Теоретическая механика»

**Лабораторная работа № 2**

**по курсу «Теоретическая механика»**

**Анимация системы.**

Студент: Погожев М.Ю.

Группа: М80-201Б-23

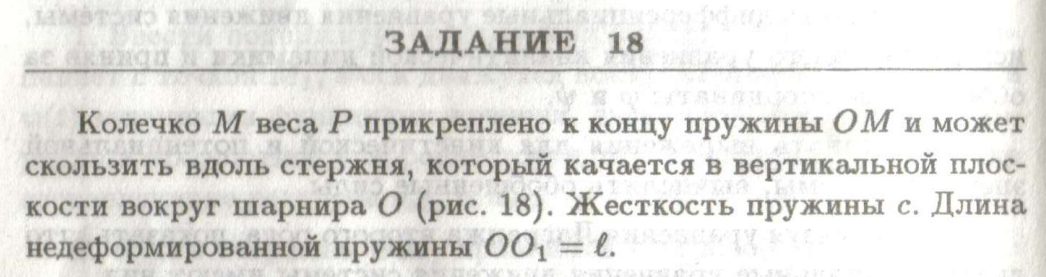
Преподаватель:

Дата: 12.11.2024

Оценка:

Москва, 2024

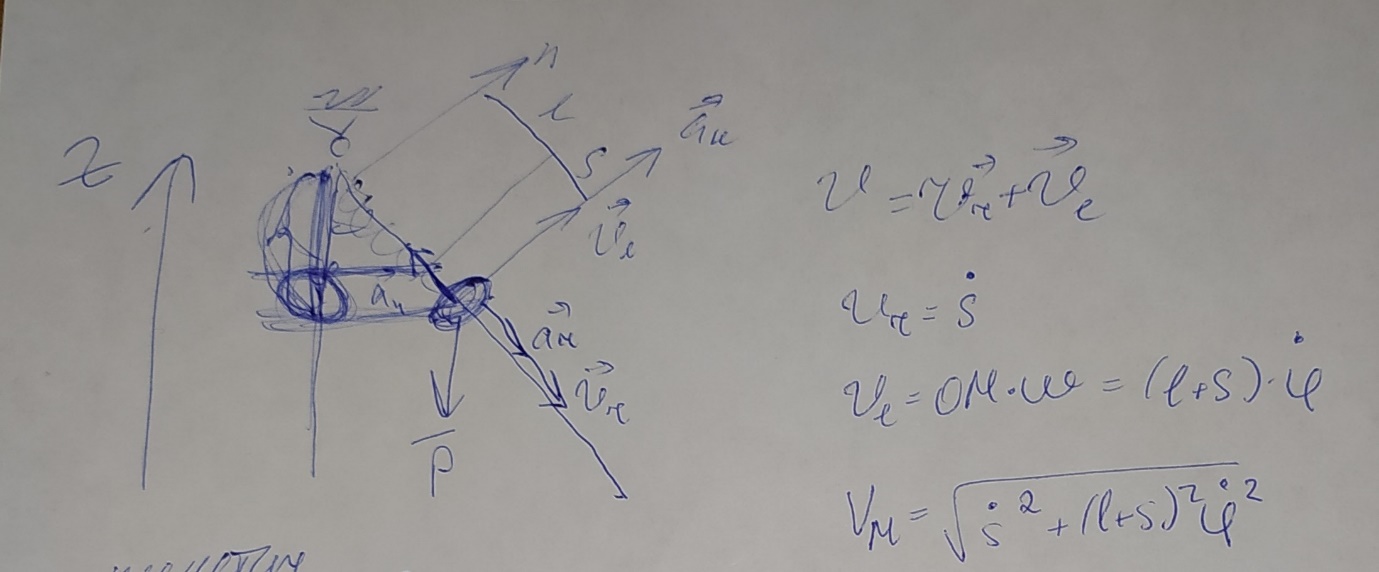
**Вариант 18**



**Задание**

Реализовать анимацию движения механической системы используя язык программирования Python.

**Вывод формул**

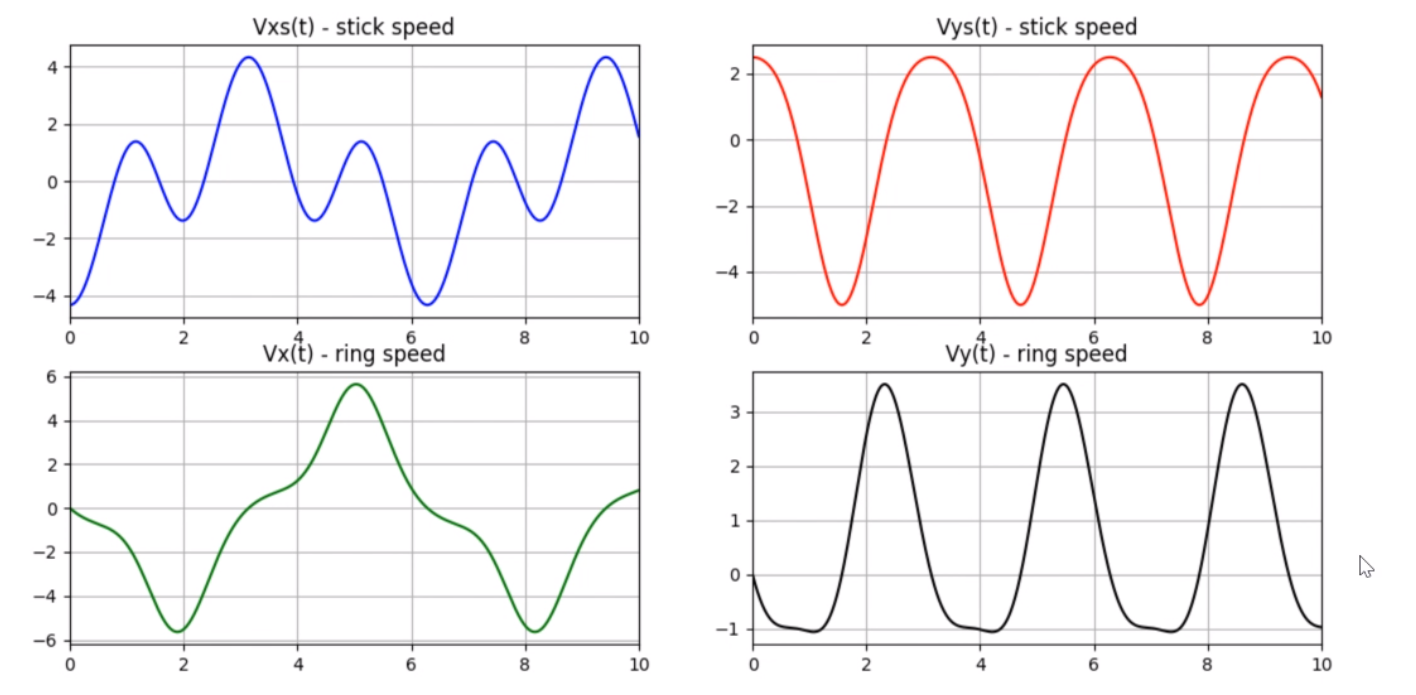


В программе считается и выводятся только скорости. Графики выводятся для переносной и абсолютной скоростей, на анимации показано направление относительной и переносной скоростей в каждый момент времени.

**Код программы**

import numpy as np  
import sympy as sp  
import math  
import matplotlib.pyplot as plt  
from matplotlib.animation import FuncAnimation  
  
l = 9  
m = 0.2  
t = sp.Symbol('t')  
a = sp.cos(t) \* math.pi / 3  
p1x = l \* sp.sin(a) - m / 2 \* sp.cos(a)  
p2x = l \* sp.sin(a) + m / 2 \* sp.cos(a)  
p3x = m / 2 \* sp.cos(a)  
p4x = -m / 2 \* sp.cos(a)  
p1y = l - l \* sp.cos(a) - m / 2 \* sp.sin(a)  
p2y = l - l \* sp.cos(a) + m / 2 \* sp.sin(a)  
p3y = l + m / 2 \* sp.sin(a)  
p4y = l - m / 2 \* sp.sin(a)  
  
h = 0.2  
d = 1  
s = l / 2 + sp.cos(2 \* t + math.pi / 2) \* (l / 2 - 2)  
x = s \* sp.sin(a)  
y = l - s \* sp.cos(a)  
vs = sp.diff(s, t)  
vxs = vs \* sp.sin(a)  
vys = -vs \* sp.cos(a)  
w = sp.diff(a, t)  
vx = w \* sp.cos(a) \* s  
vy = w \* sp.sin(a) \* s  
vpr = sp.sqrt((vxs + vx) \* (vxs + vx) + (vys + vy) \* (vys + vy))  
vps = sp.sqrt(vxs \* vxs + vys \* vys)  
vp = sp.sqrt(vx \* vx + vy \* vy)  
  
  
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  
  
  
tn = np.linspace(0, 10, 1001)  
an = np.zeros\_like(tn)  
  
p2xn = np.zeros\_like(tn)  
p2yn = np.zeros\_like(tn)  
p4xn = np.zeros\_like(tn)  
p4yn = np.zeros\_like(tn)  
p1xn = np.zeros\_like(tn)  
p1yn = np.zeros\_like(tn)  
p3xn = np.zeros\_like(tn)  
p3yn = np.zeros\_like(tn)  
xn = np.zeros\_like(tn)  
yn = np.zeros\_like(tn)  
vxsn = np.zeros\_like(tn)  
vysn = np.zeros\_like(tn)  
vxn = np.zeros\_like(tn)  
vyn = np.zeros\_like(tn)  
vs = np.zeros\_like(tn)  
v = np.zeros\_like(tn)  
vr = np.zeros\_like(tn)  
  
for i in range(len(tn)):  
 an[i] = sp.Subs(a, t, tn[i])  
  
 p2xn[i] = sp.Subs(p2x, t, tn[i])  
 p2yn[i] = sp.Subs(p2y, t, tn[i])  
 p4xn[i] = sp.Subs(p4x, t, tn[i])  
 p4yn[i] = sp.Subs(p4y, t, tn[i])  
 p1xn[i] = sp.Subs(p1x, t, tn[i])  
 p1yn[i] = sp.Subs(p1y, t, tn[i])  
 p3xn[i] = sp.Subs(p3x, t, tn[i])  
 p3yn[i] = sp.Subs(p3y, t, tn[i])  
 xn[i] = sp.Subs(x, t, tn[i])  
 yn[i] = sp.Subs(y, t, tn[i])  
 vxsn[i] = sp.Subs(vxs, t, tn[i])  
 vysn[i] = sp.Subs(vys, t, tn[i])  
 vxn[i] = sp.Subs(vx, t, tn[i])  
 vyn[i] = sp.Subs(vy, t, tn[i])  
 v[i] = sp.Subs(vp, t, tn[i])  
 vs[i] = sp.Subs(vps, t, tn[i])  
 vr[i] = sp.Subs(vpr, t, tn[i])  
  
fig\_for\_graphs = plt.figure(figsize=[13, 7])  
ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 1)  
ax\_for\_graphs.plot(tn, vxsn, color='blue')  
ax\_for\_graphs.set\_title("Vxs(t) - stick speed")  
ax\_for\_graphs.set(xlim=[0, 10])  
ax\_for\_graphs.grid(True)  
  
ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 2)  
ax\_for\_graphs.plot(tn, vysn, color='red')  
ax\_for\_graphs.set\_title('Vys(t) - stick speed')  
ax\_for\_graphs.set(xlim=[0, 10])  
ax\_for\_graphs.grid(True)  
  
ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 3)  
ax\_for\_graphs.plot(tn, vxn, color='green')  
ax\_for\_graphs.set\_title("Vx(t) - ring speed")  
ax\_for\_graphs.set(xlim=[0, 10])  
ax\_for\_graphs.grid(True)  
  
ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 4)  
ax\_for\_graphs.plot(tn, vyn, color='black')  
ax\_for\_graphs.set\_title('Vy(t) - ring speed')  
ax\_for\_graphs.set(xlim=[0, 10])  
ax\_for\_graphs.grid(True)  
  
fig = plt.figure(figsize=[13, 7])  
ax = fig.add\_subplot(1, 1, 1)  
ax.axis('equal')  
ax.set\_xlim([-l - 5, l + 5])  
ax.set(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]  
  
  
a = FuncAnimation(fig, cha, frames=len(tn), interval=10)  
plt.show()

**Примеры вывода программы**



Графики функций.

