ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«ДИНАМИКА СИСТЕМЫ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ № 29

Выполнил(а) студент группы М8О-203Б-22

Полиха Александр Владимирович_____
подпись, дата

Проверил и принял

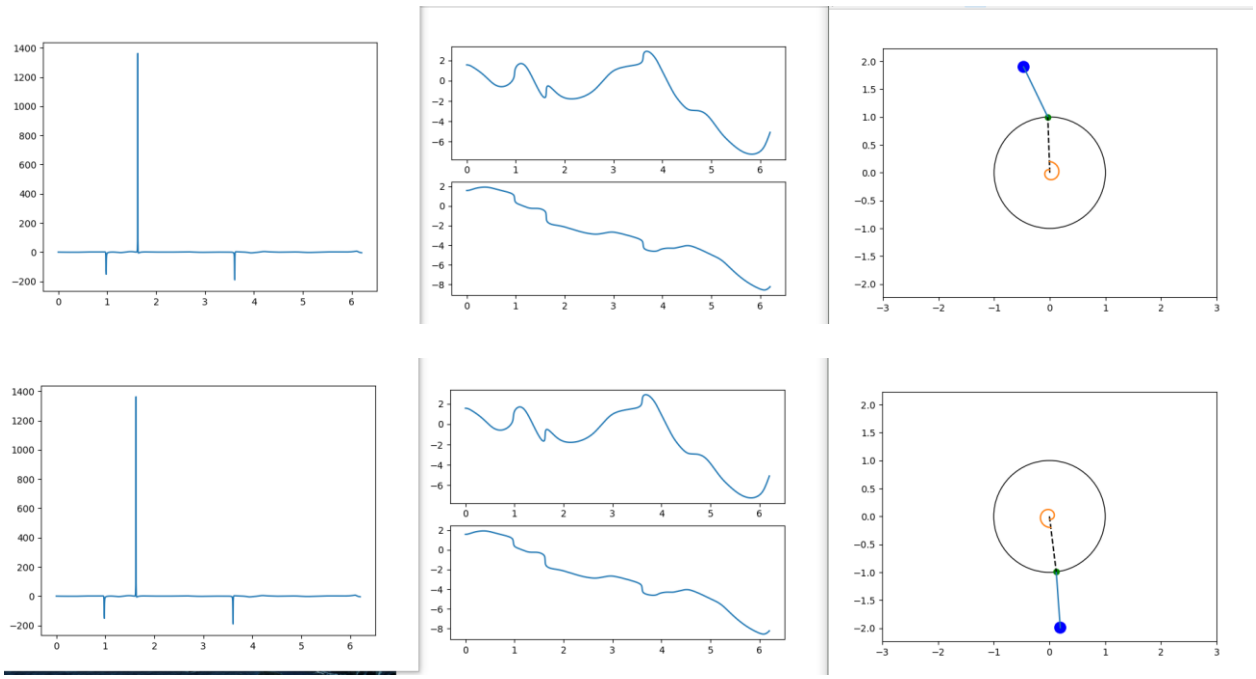
Авдюшкин А.Н._____
подпись, дата

Москва, 2023

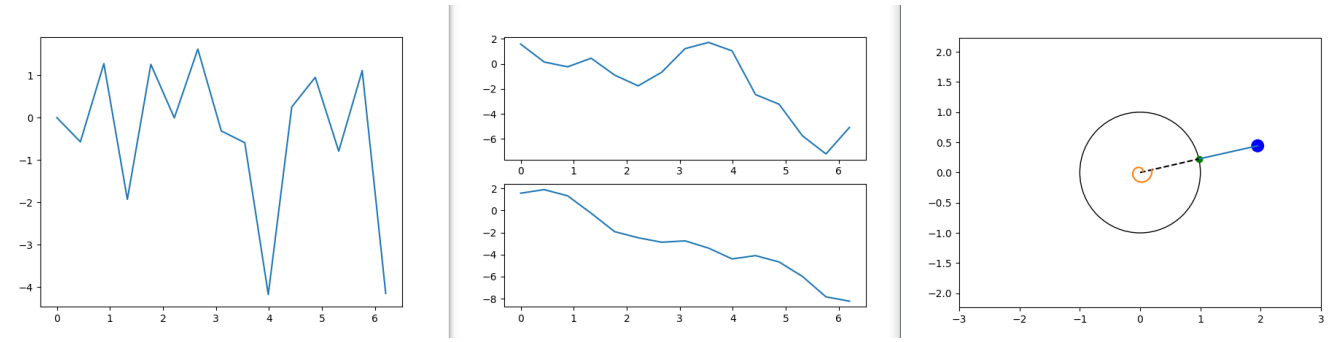
Задание

Проинтегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python. Построить анимацию движения системы, а также графики законов движения системы и указанных в задании реакций для разных случаев системы.

Результат



При Step = 15 получаем более ломанный график изменения



Код

```
import numpy as n
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from matplotlib.patches import Circle
import math
from scipy.integrate import odeint
```

```
def SystDiffEq(y, t, m1, m2, R, L, c, k, g,phist):
    # y = [phi, psi, phi', psi'] -> dy = [phi', psi', phi'', psi'']
    dy = n.zeros_like(y)
    dy[0] = y[2]
    dy[1] = y[3]

    phi =  y[0]
    psi= y[1]
    dphi = y[2]
    dpsi = y[3]
    # a11 * phi'' + a12 * psi'' = b1
    # a21 * phi'' + a22 * psi'' = b2
    a11 = ((m1/2) + m2)*R
    a12 = m2*L*n.cos(y[0]-y[1])
    b1 = -m2*g*n.sin(y[0])-(y[0]-phist)*c/R+k/R*(y[3]-y[2])-
((y[3])**2)*n.sin(y[0]-y[1])

    a21 = n.cos(y[0]-y[1])
    a22 = L/R
    b2 = ((y[3])**2)*n.sin(y[0]-y[1]) - (g/R)*n.sin(y[1]) - k*(y[3]-
y[2])/(m2*R*L)



    detA = a11 * a22 - a12 * a21
    detA1 = b1 * a22 - a12 * b2
    detA2 = a11 * b2 - a21 * b1


    dy[2] =  detA1  / detA
    dy[3] =  detA2  / detA

    return dy

R = 1
RB = 0.1
L = 1
m1 = 0.1
m2 = 0.05
c = 1
k = 0
g = 9.81
phist  = 0

step = 15
t = n.linspace(0,6.2,step)
#x = n.sin(t)
#phi = n.sin(2*t)

y0 = [n.pi/2, n.pi/2, 0, 0]
```

```
Y,info = odeint(SystDiffEq, y0, t, (m1, m2, R, L, c, k, g, phist), full_output=1)
print(info)
phi = Y[:,0]
psi = Y[:,1]
dphi = Y[:,2]
dpsi = Y[:,3]


ddphi = n.zeros_like(t)
ddpsi = n.zeros_like(t)
ROX = n.zeros_like(t)
for i in range(len(t)):
    ddphi[i] =  SystDiffEq(Y[i], t[i], m1, m2, R, L, c, k, g, phist)[2]
    ddpsi[i] =  SystDiffEq(Y[i], t[i], m1, m2, R, L, c, k, g, phist)[3]
    ROX[i] = m2 * (R*(ddphi[i]*n.cos(phi[i]) - (dphi[i]**2)*n.sin(phi[i])) +
L*(ddpsi[i]*n.cos(phi[i])-(dpsi[i]**2)*n.sin(psi[i])))
fgrt  =  plt.figure()
phiplt = fgrt.add_subplot(2,1,1)
phiplt.plot(t,phi)
psiplt = fgrt.add_subplot(2,1,2)
psiplt.plot(t,psi)

fgr_ROX = plt.figure()
gr_ROX = fgr_ROX.add_subplot(1, 1, 1)
gr_ROX.plot(t, ROX)

fgrt.show()


fgr = plt.figure()
gr = fgr.add_subplot(1,1,1)
gr.axis('equal')
gr.set_xlim((-3, 3))
gr.set_ylim((-3, 3))



circle = Circle((0, 0), R, fill=False)
circle_patch = gr.add_patch(circle)

Xa = n.cos(phi)
Ya = n.sin(phi)
Xb = Xa + n.cos(phi)
Yb = Ya + n.sin(phi)

pA = gr.plot(Xa[0],Ya[0], marker='o',color='g')[0]
pB = gr.plot(Xb[0],Yb[0], marker='o',color='b')[0]
AB = gr.plot([Xa[0], Xb[0]],[Ya[0], Yb[0]])[0]
B_circle = Circle((Xb[0], Yb[0]), RB, fill=True, color='b')
```

```
gr.add_patch(B_circle)

OA = gr.plot([0, Xa[0]], [0, Ya[0]], 'k--')[0]

# Spiral
Ns = 1
r1 = 0.05
r2 = 0.2
numpts = n.linspace(0, 1, 50*Ns+1)
Betas = numpts * (Ns * 2*n.pi -  phi[0])
Xs = (r1 + (r2-r1) * numpts)  * n.cos(Betas + n.pi/2)
Ys = (r1 + (r2-r1) * numpts) *  n.sin(Betas + n.pi/2)

Spiral = gr.plot(Xs, Ys)[0]

def update(i):
    global Xa, Ya, Xb, Yb, B_circle, OA
    psi = n.cos(t[i] * 3)
    theta  = n.radians(i)
    Xa = n.cos(theta)
    Ya = n.sin(theta)
    Xb = Xa + n.cos(theta + psi) * L
    Yb = Ya + n.sin(theta + psi) * L
    pA.set_data(Xa, Ya)
    pB.set_data(Xb, Yb)
    AB.set_data([Xa, Xb], [Ya, Yb])
    B_circle.set_center((Xb, Yb))
    OA.set_data([0, Xa], [0, Ya])


    Betas = numpts * (Ns * 2*n.pi)
    Xs = (r1 + (r2-r1) * numpts)  * n.cos(Betas+theta)
    Ys = (r1 + (r2-r1) * numpts) *  n.sin(Betas+theta)

    Spiral.set_data(Xs, Ys)


    return circle_patch, pA, pB, AB, B_circle, OA,  Spiral

anim = FuncAnimation(fgr, update, frames=step, interval=1, blit=True)

plt.show()
```

Вывод

Научился интегрировать систему дифференциальных уравнений движения системы с двумя степенями свободы с помощью средств Python.

Также научился строить анимации таких систем.