ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«АНИМАЦИЯ ТОЧКИ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ № 23

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

Шипилов Кирилл Юрьевич_____
подпись, дата

Проверил и принял

Авдюшкин А.Н._____
подпись, дата

с оценкой _____

Москва, 2023

Задание: построить заданную траекторию, запустить анимацию движения точки, построить стрелки радиус-вектора, вектора скорости, вектора ускорения и радиуса кривизны.

Закон движения точки:

$r(t) = 1 - \sin(t)$

$\varphi(t) = 2t$

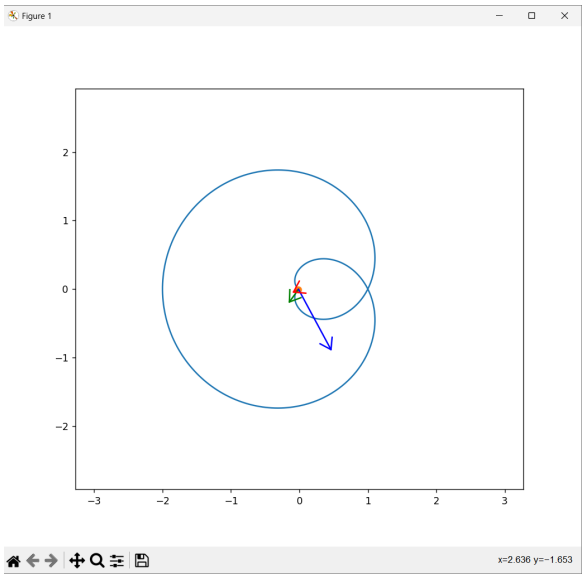
Текст программы:

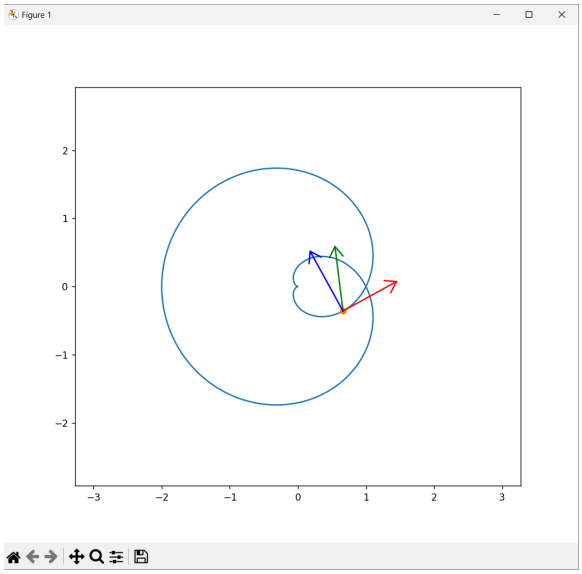
```
import math
import sympy as s
import matplotlib.pyplot as plot
import numpy as np
from matplotlib.animation import FuncAnimation

t = s.Symbol('t')

x = (1 - s.sin(t)) * s.cos(2 * t)
y = (1 - s.sin(t)) * s.sin(2 * t)

Vx = s.diff(x, t)
Vy = s.diff(y, t)

ax = s.diff(Vx, t)
ay = s.diff(Vy, t)

rx = -Vy / s.sqrt(Vx * Vx + Vy * Vy)
ry = Vx / s.sqrt(Vx * Vx + Vy * Vy)

step = 1000
T = np.linspace(0, 2 * math.pi, step)
X = np.zeros_like(T)
Y = np.zeros_like(T)
VX = np.zeros_like(T)
VY = np.zeros_like(T)
AX = np.zeros_like(T)
AY = np.zeros_like(T)
RX = np.zeros_like(T)
RY = np.zeros_like(T)


for i in range(len(T)):
    X[i] = x.subs(t, T[i])
    Y[i] = y.subs(t, T[i])
    VX[i] = 0.5 * Vx.subs(t, T[i])
    VY[i] = 0.5 * Vy.subs(t, T[i])
    AX[i] = 0.2 * ax.subs(t, T[i])
```

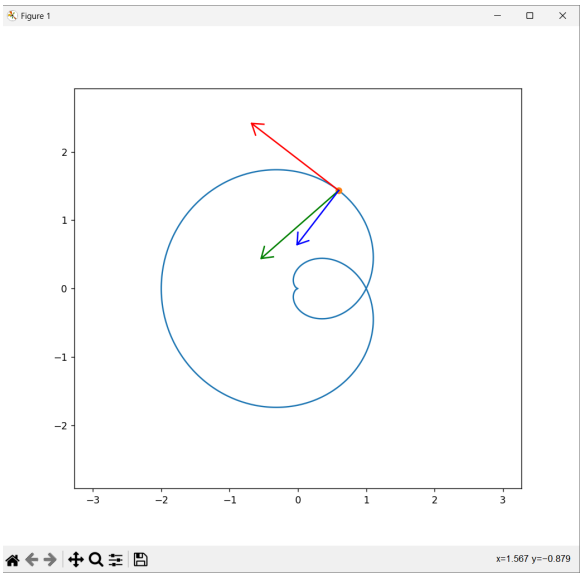
```
    AY[i] = 0.2 * ay.subs(t, T[i])
    RX[i] = rx.subs(t, T[i])
    RY[i] = ry.subs(t, T[i])

fgr = plot.figure()
grf = fgr.add_subplot(1, 1, 1)
grf.axis('equal')
grf.set(xlim=[-6, 6], ylim=[-6, 6])
grf.plot(X, Y)

Pnt = grf.plot(X[0], Y[0], marker='o')[0]
Vpl = grf.plot([X[0], X[0] + VX[0]], [Y[0], Y[0] + VY[0]], 'r')[0]
Apl = grf.plot([X[0], X[0] + AX[0]], [Y[0], Y[0] + AY[0]], 'g')[0]
Rpl = grf.plot([X[0], X[0] + RX[0]], [Y[0], Y[0] + RY[0]], 'b')[0]


def vect_arrow(vec_x, vec_y, _x, _y):
    a = 0.15
    b = 0.1
    arr_x = np.array([-a, 0, -a])
    arr_y = np.array([b, 0, -b])

    phi = math.atan2(vec_y, vec_x)

    rot_x = arr_x * np.cos(phi) - arr_y * np.sin(phi)
    rot_y = arr_x * np.sin(phi) + arr_y * np.cos(phi)

    arr_x = rot_x + _x + vec_x
    arr_y = rot_y + _y + vec_y

    return arr_x, arr_y


ArVX, ArVY = vect_arrow(VX[0], VY[0], X[0], Y[0])
V_arr = grf.plot(ArVX, ArVY, 'r')[0]

ArAX, ArAY = vect_arrow(AX[0], AY[0], X[0], Y[0])
A_arr = grf.plot(ArAX, ArAY, 'g')[0]

ArRX, ArRY = vect_arrow(RX[0], RY[0], X[0], Y[0])
R_arr = grf.plot(ArRX, ArRY, 'b')[0]


def anim(j):
    global ArVX, ArVY, ArAX, ArAY, ArRX, ArRY
    Pnt.set_data([X[j]], [Y[j]])

    Vpl.set_data([X[j], X[j] + VX[j]], [Y[j], Y[j] + VY[j]])
    ArVX, ArVY = vect_arrow(VX[j], VY[j], X[j], Y[j])
    V_arr.set_data(ArVX, ArVY)
```

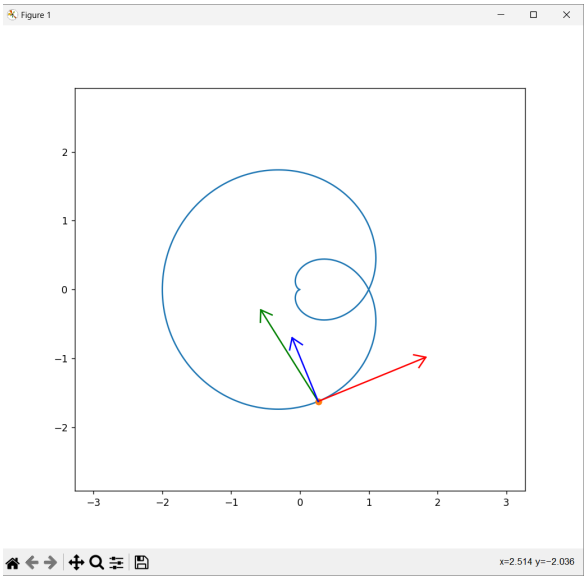
```
    Apl.set_data([X[j], X[j] + AX[j]], [Y[j], Y[j] + AY[j]])
    ArAX, ArAY = vect_arrow(AX[j], AY[j], X[j], Y[j])
    A_arr.set_data(ArAX, ArAY)

    Rpl.set_data([X[j], X[j] + RX[j]], [Y[j], Y[j] + RY[j]])
    ArRX, ArRY = vect_arrow(RX[j], RY[j], X[j], Y[j])
    R_arr.set_data(ArRX, ArRY)

    return [Pnt, Vpl, V_arr, Apl, A_arr, Rpl, R_arr]


an = FuncAnimation(fgr, anim, frames=step, interval=1)

plot.show()
```

Результат работы программы:









ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ОТЧЕТ

О ВЫПЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

«АНИМАЦИЯ СИСТЕМЫ»

ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»

ВАРИАНТ ЗАДАНИЯ № 8

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

Шипилов Кирилл Юрьевич_____
подпись, дата

Проверил и принял

Авдюшкин А.Н._____
подпись, дата

Москва, 2023

Задание: построить анимацию движения системы с помощью Python.

Система:

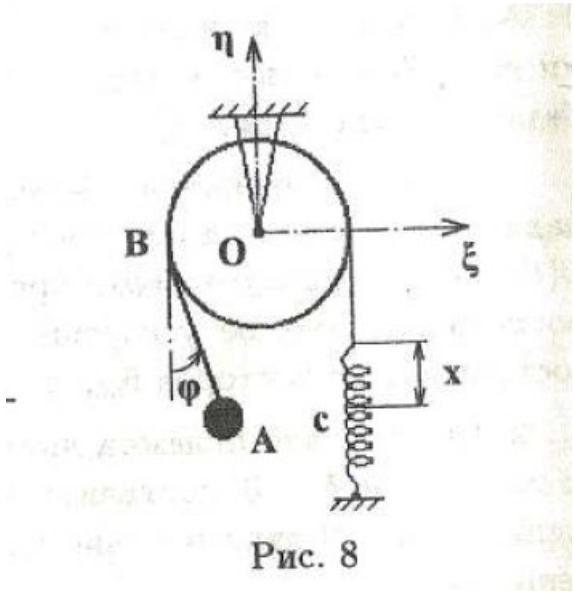


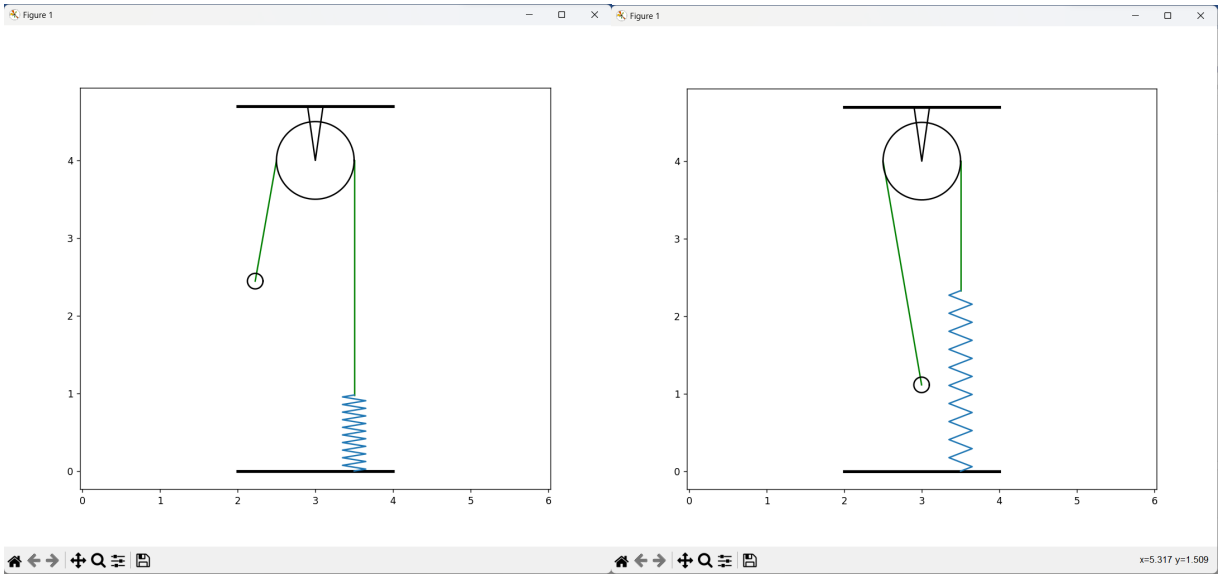
Рис. 8

Текст программы:

```
import math

import numpy as n
import matplotlib.pyplot as plot
from matplotlib.animation import FuncAnimation

fgr = plot.figure()
gr = fgr.add_subplot(1, 1, 1)
gr.axis('equal')

step = 500
t = n.linspace(0, 2*math.pi, step)

Xo = 3
Yo = 4
R = 0.5
y0 = 2.3
r = 0.1
y = n.sin(t)

gr.plot([2, 4], [0, 0], 'black', linewidth=3)
gr.plot([2, 4], [Yo+0.7, Yo+0.7], 'black', linewidth=3)
gr.plot([Xo-0.1, Xo, Xo+0.1], [Yo+0.7, Yo, Yo+0.7], 'black')

y_l = y + y0
y_r = y - y0
phi = n.pi/18 * n.sin(2*t)

Xb = Xo - R
```

```
Yb = Yo

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

AB = gr.plot([Xa[0], Xb], [Ya[0], Yb], 'green')[0]
L = gr.plot([Xo + R, Xo + R], [Yo, Yo + y_r[0]], 'green')[0]

Alp = n.linspace(0, 2*n.pi, 100)
Xc = n.cos(Alp)
Yc = n.sin(Alp)

Block = gr.plot(Xo + R * Xc, Yo + R * Yc, 'black')[0]

m = gr.plot(Xa[0] + r * Xc, Ya[0] + r * Yc, 'black')[0]

Np = 20
Yp = n.linspace(0, 1, 2*Np+1)
Xp = 0.15 * n.sin(n.pi/2*n.arange(2*Np+1))
Pruzh = gr.plot(Xo + R + Xp, (Yo + y_r[0]) * Yp)[0]


def run(i):
    m.set_data([Xa[i] + r * Xc], [Ya[i] + r * Yc])
    AB.set_data([Xa[i], Xb], [Ya[i], Yb])
    L.set_data([Xo + R, Xo + R], [Yo, Yo + y_r[i]])
    Pruzh.set_data(Xo + R + Xp, (Yo + y_r[i]) * Yp)
    return [m, AB, Block, Pruzh]


anim = FuncAnimation(fgr, run, frames=step, interval=1)

plot.show()
```

Результат работы:



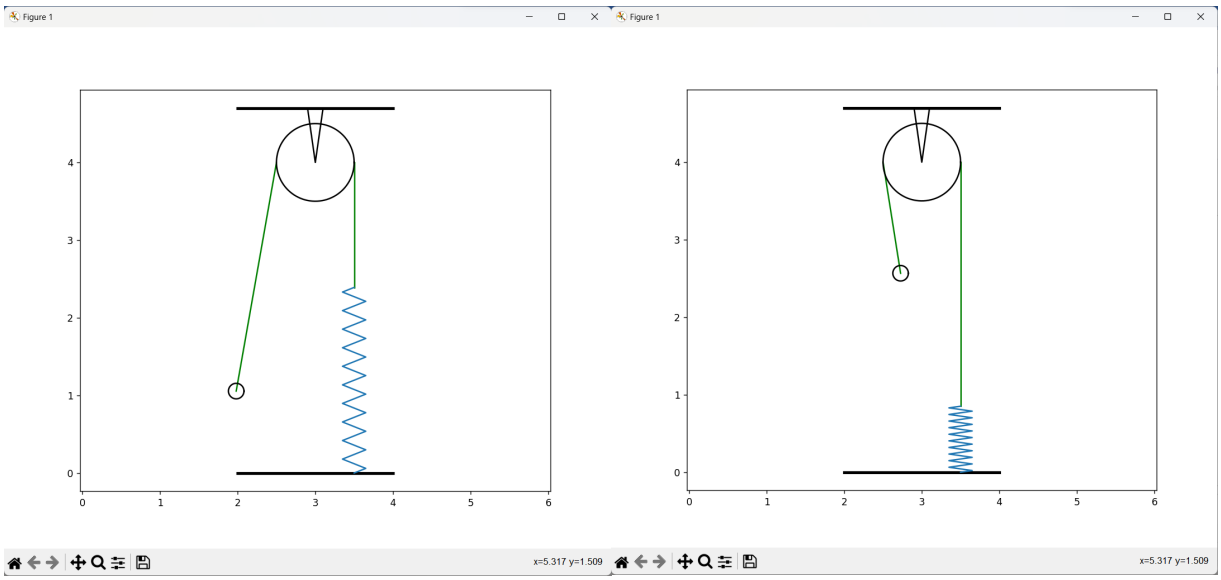


Figure 1
Figure 1
x=5.317 y=1.509
x=5.317 y=1.509

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 № 8

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

Шипилов Кирилл Юрьевич_____
подпись, дата

Проверил и принял

Авдюшкин А.Н._____
подпись, дата

с оценкой _____

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.

Система:

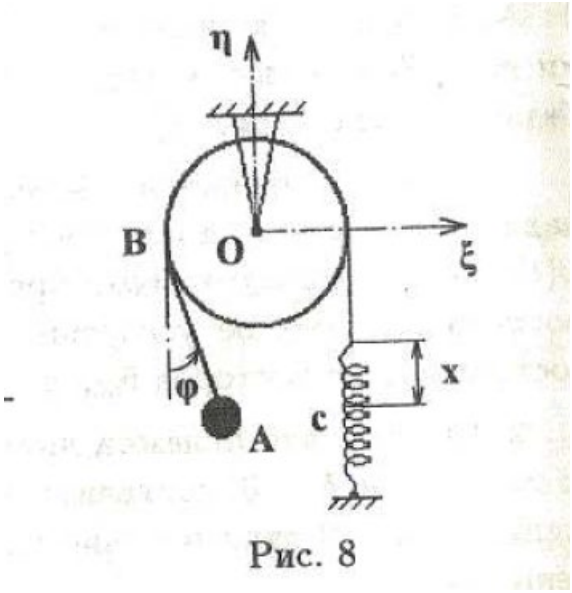


Рис. 8

Законы движения системы:

$$(\ell \geq 0,\ \varphi > -\pi)$$

$$[(M/2)+m]\,\ddot{x} - m\ell\dot{\varphi}^2 = mg\cos\varphi - c(x+\delta), \qquad \delta = mg/c,$$

$$\ell\ddot{\varphi} + \dot{\varphi}\,(2\dot{x} - r\dot{\varphi}) = -g\sin\varphi.$$

Проекции реакции оси блока:

$$N_\xi = -m\left(\ell\ddot{\varphi} + r\dot{\varphi}^2 + 2\dot{\ell}\dot{\varphi}\right)\cos\varphi - m\left(\ddot{x} - \ell\dot{\varphi}^2\right)\sin\varphi, \quad \ell = \ell_0 + x - r\varphi,$$

$$N_\eta = -m\left(\ell\ddot{\varphi} + r\dot{\varphi}^2 + 2\dot{\ell}\dot{\varphi}\right)\sin\varphi + m\left(\ddot{x} - \ell\dot{\varphi}^2\right)\cos\varphi - cx - (M+m)g.$$

Текст программы:

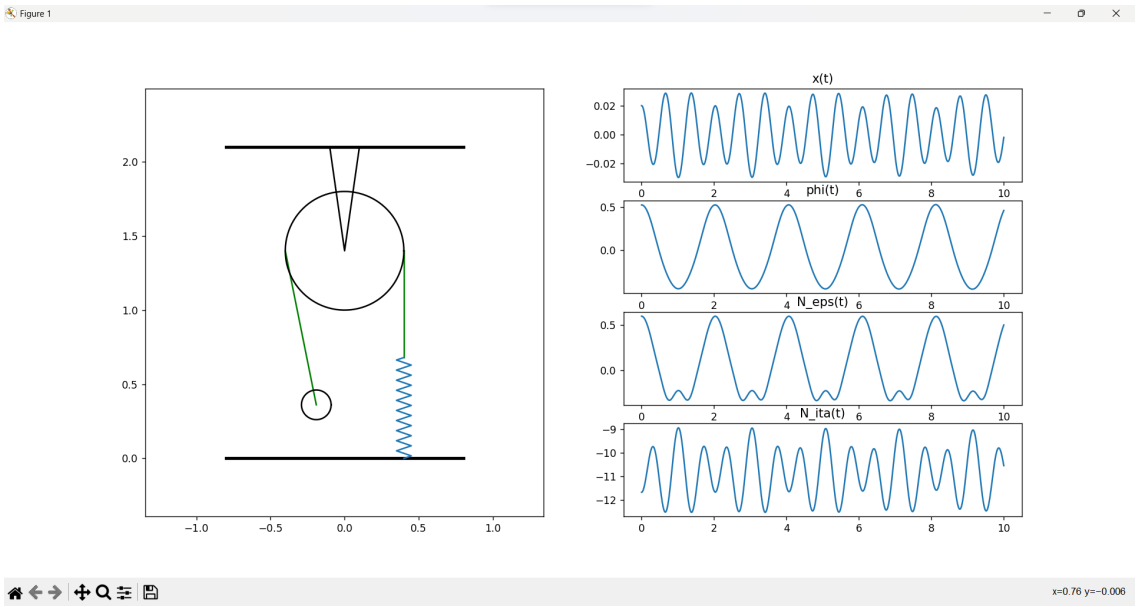
```
import numpy as n
import matplotlib.pyplot as plot
from matplotlib.animation import FuncAnimation
from scipy.integrate import odeint
```

```
def sys_diff_eq(y, t, m, M, l0, c, g, R):
    # y = [x, phi, x', phi'] -> dy = [x', phi', x'', phi'']
    dy = n.zeros_like(y)
    dy[0] = y[2]
    dy[1] = y[3]

    L = l0 + y[0] - R * y[1]
    # a11 * x'' + a12 * phi'' = b1
    # a21 * x'' + a22 * phi'' = b2

    a11 = M/2 + m
    a12 = 0
    b1 = m * g * n.cos(y[1]) - c * (y[0] + (m * g) / c) + m * L * y[3] * y[3]

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

    det_a = a11 * a22 - a12 * a21
    det_a1 = b1 * a22 - a12 * b2
    det_a2 = a11 * b2 - a21 * b1

    dy[2] = det_a1 / det_a
    dy[3] = det_a2 / det_a

    return dy


step = 1000
t = n.linspace(0, 10, step)

y0 = [0.02, n.pi/6, 0, 0]

M = 1
m = 0.1
R = 0.4
l0 = 1
c = 50
g = 9.81

Y = odeint(sys_diff_eq, y0, t, (m, M, l0, c, g, R))

x = Y[:, 0]
phi = Y[:, 1]
x_t = Y[:, 2]
phi_t = Y[:, 3]

L = n.zeros_like(t)
x_tt = n.zeros_like(t)
phi_tt = n.zeros_like(t)
N_eps = n.zeros_like(t)
```

```
N_ita = n.zeros_like(t)


for i in range(len(t)):
    L[i] = l0 + x[i] + R * phi[i]
    x_tt[i] = sys_diff_eq(Y[i], t[i], m, M, L[i], c, g, R)[2]
    phi_tt[i] = sys_diff_eq(Y[i], t[i], m, M, L[i], c, g, R)[3]
    N_eps[i] = -m*(L[i]*phi_tt[i] + R*phi_t[i]**2 + 2*(x_t[i] -
R*phi_t[i])*phi_t[i])*n.cos(phi[i]) - m*(x_tt[i] -
L[i]*phi_t[i]**2)*n.sin(phi[i])
    N_ita[i] = -m*(L[i]*phi_tt[i] + R*phi_t[i]**2 + 2*(x_t[i] -
R*phi_t[i])*phi_t[i])*n.sin(phi[i]) + m*(x_tt[i] -
L[i]*phi_t[i]**2)*n.cos(phi[i]) - c*x[i] - (M + m) * g


fgr = plot.figure()
gr = fgr.add_subplot(4, 2, (1, 7))
gr.axis('equal')

x_plt = fgr.add_subplot(4, 2, 2)
x_plt.plot(t, x)
x_plt.set_title('x(t)')

phi_plt = fgr.add_subplot(4, 2, 4)
phi_plt.plot(t, phi)
phi_plt.set_title('phi(t)')

n_eps_plt = fgr.add_subplot(4, 2, 6)
n_eps_plt.plot(t, N_eps)
n_eps_plt.set_title('N_eps(t)')

n_ita_plt = fgr.add_subplot(4, 2, 8)
n_ita_plt.plot(t, N_ita)
n_ita_plt.set_title('N_ita(t)')

Yo = 1.4
r = 0.1
h_p0 = 0.7

Xb = -R
Yb = Yo

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

gr.plot([-2 * R, 2 * R], [0, 0], 'black', linewidth=3)
gr.plot([-2 * R, 2 * R], [Yo+0.7, Yo+0.7], 'black', linewidth=3)
gr.plot([-0.1, 0, 0.1], [Yo+0.7, Yo, Yo+0.7], 'black')

AB = gr.plot([Xa[0], Xb], [Ya[0], Yb], 'green')[0]
String_r = gr.plot([R, R], [Yo, h_p0 + x[0]], 'green')[0]
```

```
Alp = n.linspace(0, 2*n.pi, 100)
Xc = n.cos(Alp)
Yc = n.sin(Alp)

Block = gr.plot(R * Xc, Yo + R * Yc, 'black')[0]

Ticker = gr.plot(Xa[0] + r * Xc, Ya[0] + r * Yc, 'black')[0]

Np = 20
Yp = n.linspace(0, 1, 2*Np+1)
Xp = 0.05 * n.sin(n.pi/2*n.arange(2*Np+1))
Spring = gr.plot(R + Xp, (h_p0 + x[0]) * Yp)[0]


def run(i):
    Ticker.set_data([Xa[i] + r * Xc], [Ya[i] + r * Yc])
    AB.set_data([Xa[i], Xb], [Ya[i], Yb])
    String_r.set_data([R, R], [Yo, h_p0 + x[i]])
    Spring.set_data(R + Xp, (h_p0 + x[i]) * Yp)
    return [Ticker, AB, String_r, Spring]


anim = FuncAnimation(fgr, run, frames=step, interval=1)

plot.show()
```

Результат работы:



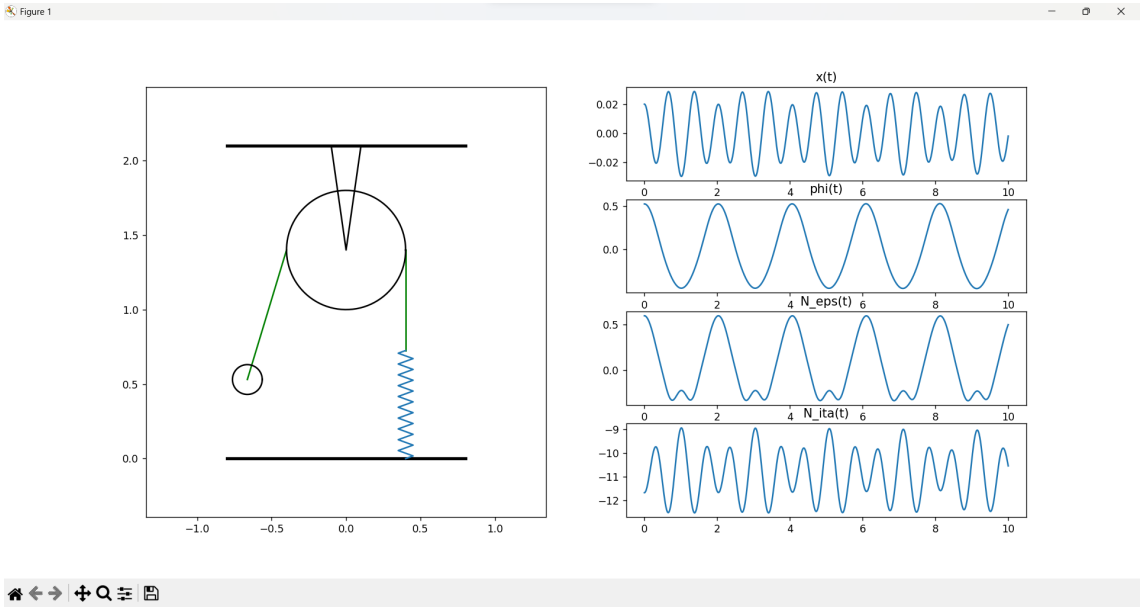


Figure 1
x(t)
phi(t)
N_eps(t)
N_ita(t)

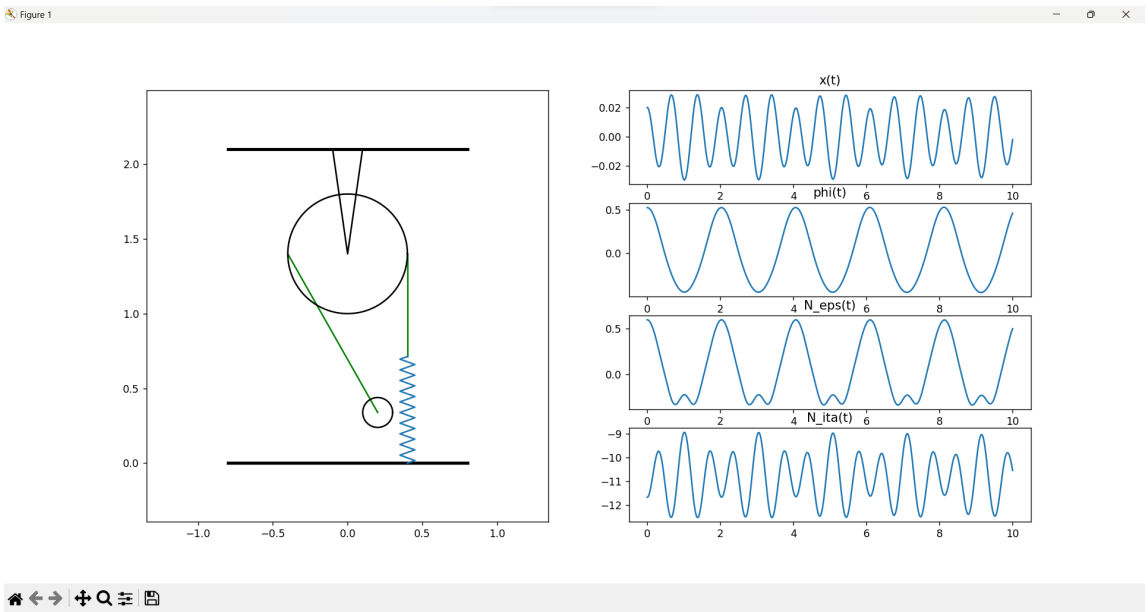


Figure 1
x(t)
phi(t)
N_eps(t)
N_ita(t)

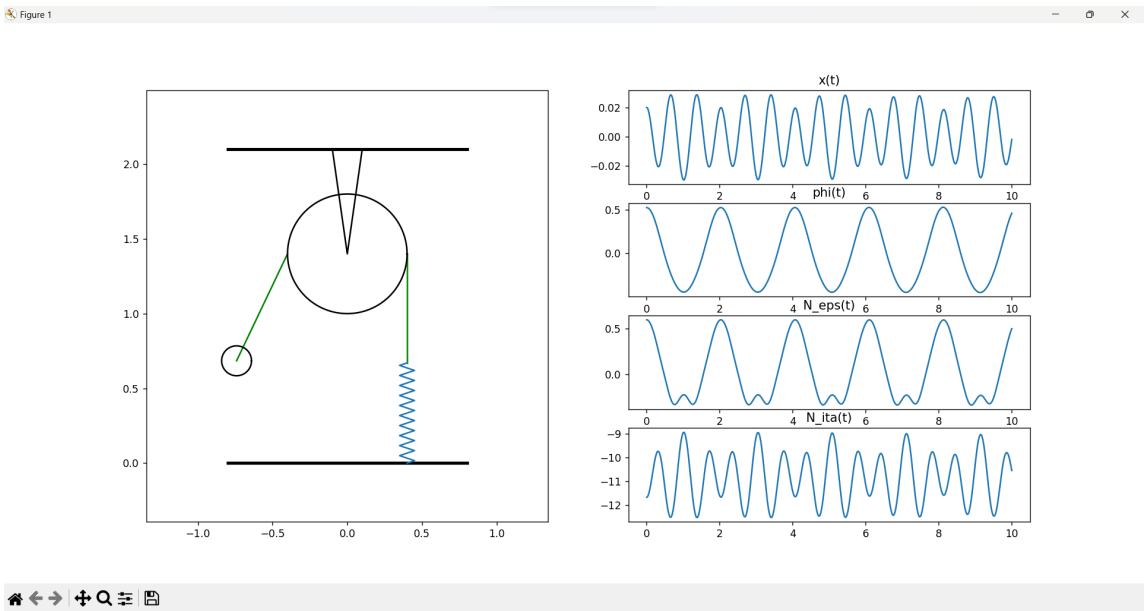


Figure 1
x(t)
phi(t)
N_eps(t)
N_ita(t)