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 №3**

Выполнил(а) студент группы М8О-208Б-21

Белоусов Егор Леонидович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Зав. каф. 802, Бардин Б.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

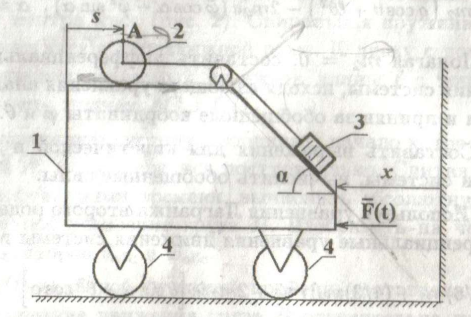
подпись, дата

с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

*Задание:* построить анимацию движения системы с помощью Python.

**Механическая система:**



**Текст программы**

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

Steps = 1001

t\_fin = 20

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

x = 1.5 \* np.sin(1.8 \* t)

s = 0.6 \* np.sin(1.2 \* t)

Alpha = np.pi / 4

BoxM, CylM, WeightM, WheelM = 10, 3, 2, 1

X0Right = 15

BoxW, BoxH = 6, 3

BoxHRight = BoxH / 4

BoxWUp = BoxW - (BoxH - BoxHRight) / np.tan(Alpha)

WheelR, CylR, BlockR = 0.5, 0.4, 0.2

WeightW, WeightH = 0.8, 2 \* BlockR

ThreadLen = 4

S0 = 1.2

InclStart = ThreadLen - BoxWUp + S0

# O - left bottom of the box

X\_O = X0Right - BoxW - x

Y\_O = 2 \* WheelR

# A - center of the cylinder

X\_A = X\_O + S0 + s

Y\_A = Y\_O + BoxH + CylR

# C1 & C2 - centers of the wheels

X\_C1 = X0Right - 4 \* BoxW / 5 - x

Y\_C1 = WheelR

X\_C2 = X0Right - BoxW / 5 - x

Y\_C2 = WheelR

# C3 - center of the block

X\_C3 = X\_O + BoxWUp - BlockR

Y\_C3 = Y\_O + BoxH + BlockR

# B1 - top of the block

X\_B1 = X\_C3

Y\_B1 = Y\_C3 + BlockR

# B2 - top-right of the block

X\_B2 = X\_B1 + BlockR \* np.cos(np.pi / 4)

Y\_B2 = Y\_B1 - BlockR \* (1 - np.sin(np.pi / 4))

# D - left bottom of the weight

X\_D = X\_O + BoxWUp + np.cos(Alpha) \* (InclStart + s)

Y\_D = Y\_O + BoxH - np.sin(Alpha) \* (InclStart + s)

# F - left center of the weight

X\_F = X\_D + WeightH / 2 \* np.sin(Alpha)

Y\_F = Y\_D + WeightH / 2 \* np.cos(Alpha)

X\_Ground = [17.5, 17.5, 0]

Y\_Ground = [6, 0, 0]

X\_Box = np.array([0, 0, BoxWUp, BoxW, BoxW, 0])

Y\_Box = np.array([0, BoxH, BoxH, BoxHRight, 0, 0])

X\_Weight = np.array([

0, WeightH \* np.sin(Alpha),

WeightH \* np.sin(Alpha) + WeightW \* np.cos(Alpha),

WeightW \* np.cos(Alpha), 0

])

Y\_Weight = np.array([

0, WeightH \* np.cos(Alpha),

WeightH \* np.cos(Alpha) - WeightW \* np.sin(Alpha),

-WeightW \* np.sin(Alpha), 0

])

psi = np.linspace(0, 2 \* np.pi, 20)

X\_Wheel = WheelR \* np.sin(psi)

Y\_Wheel = WheelR \* np.cos(psi)

X\_Block = BlockR \* np.sin(psi)

Y\_Block = BlockR \* np.cos(psi)

X\_Cyl = CylR \* np.sin(psi)

Y\_Cyl = CylR \* np.cos(psi)

fig = plt.figure(figsize=[15, 7])

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

ax.axis('equal')

ax.set(xlim=[-5, 20], ylim=[-4, 10])

ax.plot(X\_Ground, Y\_Ground, color='black', linewidth=3)

Drawed\_Wheel1 = ax.plot(X\_C1[0] + X\_Wheel, Y\_C1 + Y\_Wheel)[0]

Drawed\_Wheel2 = ax.plot(X\_C2[0] + X\_Wheel, Y\_C2 + Y\_Wheel)[0]

Drawed\_Block = ax.plot(X\_C3[0] + X\_Block, Y\_C3 + Y\_Block)[0]

Drawed\_Cyl = ax.plot(X\_A[0] + X\_Cyl, Y\_A + Y\_Cyl)[0]

Drawed\_Box = ax.plot(X\_O[0] + X\_Box, Y\_O + Y\_Box)[0]

Drawed\_Weight = ax.plot(X\_D[0] + X\_Weight, Y\_D[0] + Y\_Weight)[0]

Line\_A\_B1 = ax.plot([X\_A[0], X\_B1[0]], [Y\_A, Y\_B1], color=Drawed\_Block.get\_color())[0]

Line\_B2\_F = ax.plot([X\_B2[0], X\_F[0]], [Y\_B2, Y\_F[0]], color=Drawed\_Block.get\_color())[0]

Point\_A = ax.plot(X\_A[0], Y\_A, marker='o', markersize=5, color=Drawed\_Block.get\_color())[0]

AlphaWheel = -x / WheelR

Drawed\_WheelD1 = ax.plot([X\_C1[0] + WheelR \* np.sin(AlphaWheel[0]), X\_C1[0] - WheelR \* np.sin(AlphaWheel[0])],

[Y\_C1 + WheelR \* np.cos(AlphaWheel[0]), Y\_C1 - WheelR \* np.cos(AlphaWheel[0])])[0]

Drawed\_WheelD2 = ax.plot([X\_C2[0] + WheelR \* np.sin(AlphaWheel[0] + 1), X\_C2[0] - WheelR \* np.sin(AlphaWheel[0] + 1)],

[Y\_C2 + WheelR \* np.cos(AlphaWheel[0] + 1), Y\_C2 - WheelR \* np.cos(AlphaWheel[0] + 1)])[0]

def anima(i):

Point\_A.set\_data(X\_A[i], Y\_A)

Line\_A\_B1.set\_data([X\_A[i], X\_B1[i]], [Y\_A, Y\_B1])

Line\_B2\_F.set\_data([X\_B2[i], X\_F[i]], [Y\_B2, Y\_F[i]])

Drawed\_Box.set\_data(X\_O[i] + X\_Box, Y\_O + Y\_Box)

Drawed\_Cyl.set\_data(X\_A[i] + X\_Cyl, Y\_A + Y\_Cyl)

Drawed\_Block.set\_data(X\_C3[i] + X\_Block, Y\_C3 + Y\_Block)

Drawed\_Weight.set\_data(X\_D[i] + X\_Weight, Y\_D[i] + Y\_Weight)

Drawed\_Wheel1.set\_data(X\_C1[i] + X\_Wheel, Y\_C1 + Y\_Wheel)

Drawed\_Wheel2.set\_data(X\_C2[i] + X\_Wheel, Y\_C2 + Y\_Wheel)

Drawed\_WheelD1.set\_data([X\_C1[i] + WheelR \* np.sin(AlphaWheel[i]), X\_C1[i] - WheelR \* np.sin(AlphaWheel[i])],

[Y\_C1 + WheelR \* np.cos(AlphaWheel[i]), Y\_C1 - WheelR \* np.cos(AlphaWheel[i])])

Drawed\_WheelD2.set\_data([X\_C2[i] + WheelR \* np.sin(AlphaWheel[i] + 1), X\_C2[i] - WheelR \* np.sin(AlphaWheel[i] + 1)],

[Y\_C2 + WheelR \* np.cos(AlphaWheel[i] + 1), Y\_C2 - WheelR \* np.cos(AlphaWheel[i] + 1)])

return [Point\_A, Drawed\_Box, Drawed\_Wheel1, Drawed\_Wheel2, Drawed\_WheelD1, Drawed\_WheelD2]

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

plt.show()

**Результат работы:**

|  |  |
| --- | --- |
| **Рис.1** | **Рис.2** |
| **Рис.3** | **Рис.4** |