ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ СИСТЕМЫ»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА И ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ»**

**ВАРИАНТ ЗАДАНИЯ №9**

Выполнил студент группы М8О-208Б-23

Гаврилов Никита Валерьевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

Ст. преп. каф. 802 Волков Е.В.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

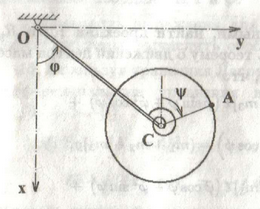
подпись, дата

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

Москва, 2024

**Задание**

Реализовать анимацию движения механической системы.

**Механическая система**

**Текст программы**

*import* numpy *as* np

*from* scipy.integrate *import* solve\_ivp

*import* matplotlib.pyplot *as* plt

*from* matplotlib.animation *import* FuncAnimation

*import* matplotlib.patches *as* patches

*# Параметры системы*

m1 *=* 2.0

m2 *=* 3.0

m3 *=* 1.0

L *=* 1.0

r *=* 0.5

c *=* 10.0

g *=* 9.81

phi0 *=* 0.0

psi0 *=* 0.0

phi\_dot0 *=* 0.5 */* c

psi\_dot0 *=* *-*0.5 */* c

y0 *=* [phi0, phi\_dot0, psi0, psi\_dot0]

*# Система дифференциальных уравнений*

def equations(t, y):

phi, phi\_dot, psi, psi\_dot *=* y

*# Произвольные коэффициенты для демонстрации*

a *=* 1.5

b *=* 0.8

d *=* 2.0

e *=* 1.2

phi\_ddot *=* *-*a *\** phi *+* b *\** np.sin(psi)

psi\_ddot *=* *-*d *\** psi *+* e *\** np.cos(phi)

*return* [phi\_dot, phi\_ddot, psi\_dot, psi\_ddot]

*# Интеграция системы во времени*

t\_span *=* (0, 20)

t\_eval *=* np.linspace(t\_span[0], t\_span[1], 2000)

sol *=* solve\_ivp(equations, t\_span, y0, t\_eval*=*t\_eval, method*=*"RK45")

t *=* sol.t

phi *=* sol.y[0]

phi\_dot\_arr *=* sol.y[1]

psi *=* sol.y[2]

psi\_dot\_arr *=* sol.y[3]

phi\_ddot\_arr *=* np.gradient(phi\_dot\_arr, t)

psi\_ddot\_arr *=* np.gradient(psi\_dot\_arr, t)

*# Вычисление R\_x(t), R\_y(t)*

Rx *=* *-*((m1 */* 2) *+* m2 *+* m3) *\** L *\** (

phi\_ddot\_arr *\** np.sin(phi) *+* phi\_dot\_arr*\*\**2 *\** np.cos(phi)

) *+* (m3 *\** r) *\** (

psi\_ddot\_arr *\** np.sin(psi) *+* psi\_dot\_arr*\*\**2 *\** np.cos(psi)

) *-* (m1 *+* m2 *+* m3) *\** g

Ry *=* ((m1 */* 2) *+* m2 *+* m3) *\** L *\** (

phi\_ddot\_arr *\** np.cos(phi) *-* phi\_dot\_arr*\*\**2 *\** np.sin(phi)

) *+* (m3 *\** r) *\** (

psi\_ddot\_arr *\** np.cos(psi) *-* psi\_dot\_arr*\*\**2 *\** np.sin(psi)

)

*# Анимация движения системы*

fig2, ax2 *=* plt.subplots(figsize*=*(8, 8))

ax2.set\_aspect("equal", "box")

ax2.set\_xlim(*-*2, 2)

ax2.set\_ylim(*-*2, 2)

ax2.set\_title("Анимация движения системы")

line\_stem, *=* ax2.plot([], [], "o-", lw*=*3, color*=*"blue", label*=*"Стержень")

pointA, *=* ax2.plot([], [], "ro", ms*=*6, label*=*"Точка A")

disk\_patch *=* patches.Circle((0, 0), radius*=*r, fill*=*True, color*=*"gray", alpha*=*0.4, label*=*"Диск")

ax2.add\_patch(disk\_patch)

spiral\_spring, *=* ax2.plot([], [], "k-", lw*=*2, label*=*"Пружина")

*# Инициализируем анимацию*

def init\_animation():

line\_stem.set\_data([], [])

pointA.set\_data([], [])

disk\_patch.center *=* (9999, 9999)

spiral\_spring.set\_data([], [])

*return* line\_stem, pointA, disk\_patch, spiral\_spring

*# Анимация*

def animate(i):

xC *=* L *\** np.sin(phi[i])

yC *=* *-*L *\** np.cos(phi[i])

disk\_patch.center *=* (xC, yC)

xA *=* xC *+* r *\** np.cos(psi[i])

yA *=* yC *+* r *\** np.sin(psi[i])

line\_stem.set\_data([0, xC], [0, yC])

pointA.set\_data(xA, yA)

theta *=* np.linspace(0, 2 *\** np.pi *\** 2, 500) *# 2 витка*

r\_values *=* np.linspace(0, 0.2, 500)

x\_spiral *=* xC *+* r\_values *\** np.cos(theta *+* psi[i])

y\_spiral *=* yC *+* r\_values *\** np.sin(theta *+* psi[i])

spiral\_spring.set\_data(x\_spiral, y\_spiral)

*return* line\_stem, pointA, disk\_patch, spiral\_spring

def get\_spring\_points(x1, y1, x2, y2, coils*=*5, radius*=*0.05, resolution*=*100):

dx *=* x2 *-* x1

dy *=* y2 *-* y1

length *=* np.sqrt(dx*\*\**2 *+* dy*\*\**2)

base\_angle *=* np.arctan2(dy, dx)

s *=* np.linspace(0, 1, resolution)

x\_vals *=* []

y\_vals *=* []

*for* si *in* s:

x\_base *=* x1 *+* si *\** dx

y\_base *=* y1 *+* si *\** dy

perp\_angle *=* base\_angle *+* np.pi */* 2

x\_off *=* radius *\** np.sin(2 *\** np.pi *\** coils *\** si) *\** np.cos(perp\_angle)

y\_off *=* radius *\** np.sin(2 *\** np.pi *\** coils *\** si) *\** np.sin(perp\_angle)

x\_vals.append(x\_base *+* x\_off)

y\_vals.append(y\_base *+* y\_off)

*return* np.array(x\_vals), np.array(y\_vals)

*# Создаем анимацию*

ani *=* FuncAnimation(

fig2,

animate,

frames*=*len(t),

init\_func*=*init\_animation,

interval*=*20,

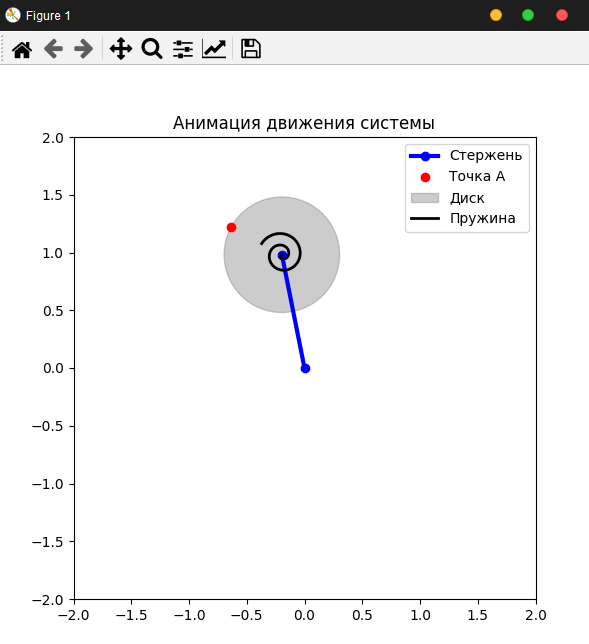
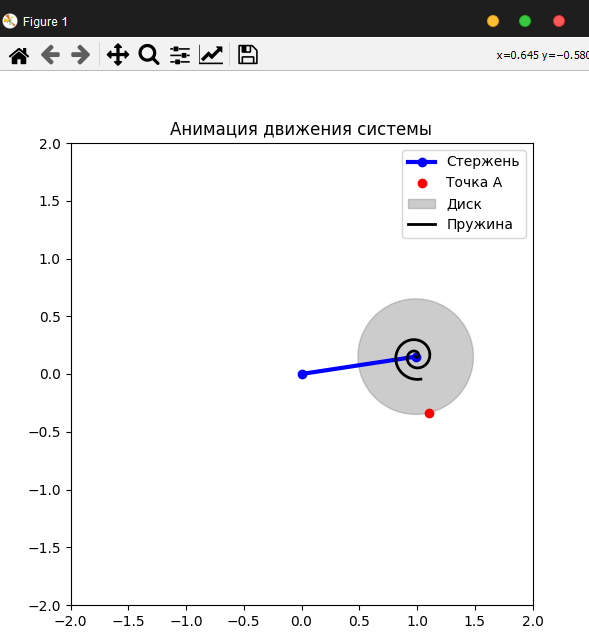
blit*=*True,

repeat*=*True,

)

plt.legend()

plt.show()

**Результат работы программы**

**Вывод**

В процессе выполнения этой лабораторной работы я использовал знания из курса «Теоретическая механика» для создания анимации движения механической системы. Программа помогла мне визуализировать, как движутся разные части системы, и лучше понять их взаимосвязь.

Особенно удобно было строить графики, которые показывают изменения углов и реакций со временем. Такие графики делают процесс анализа проще и нагляднее, что очень полезно как в учебе, так и в реальных проектах.

В итоге, работа не только закрепила мои теоретические знания, но и улучшила навыки программирования и визуализации динамических процессов.