ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«ДИНАМИКА 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

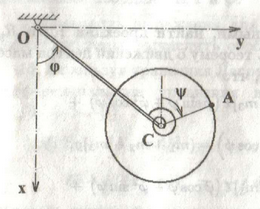
с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Вариант 9**

**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.

**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

*# Начальные условия*

phi0 *=* 0.0

psi0 *=* 0.0

phi\_dot0 *=* 0.5 */* c

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

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

*# Система уравнений*

def equations(t, y):

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

M1 *=* (m1 */* 3.0) *+* m2 *+* m3

M2 *=* (m2 */* 2.0) *+* m3

cos\_sum *=* np.cos(phi *+* psi)

A11 *=* M1 *\** L

A12 *=* m3 *\** r *\** cos\_sum

A21 *=* m3 *\** L *\** cos\_sum

A22 *=* M2 *\** r

b1 *=* ((m1 */* 2.0) *+* m2 *+* m3) *\** g *\** np.sin(phi) *-* (c */* L) *\** (phi *+* psi)

b2 *=* m3 *\** g *\** np.sin(psi) *-* (c */* r) *\** (phi *+* psi)

A *=* np.array([[A11, A12], [A21, A22]], dtype*=*float)

b *=* np.array([b1, b2], dtype*=*float)

*try*:

phi\_ddot, psi\_ddot *=* np.linalg.solve(A, b)

*except* np.linalg.LinAlgError:

phi\_ddot, psi\_ddot *=* 0.0, 0.0

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

*# Интеграция системы во времени*

t\_span *=* (0, 10)

t\_eval *=* np.linspace(t\_span[0], t\_span[1], 1000)

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]

*# Численные оценки ускорений (phi\_ddot, psi\_ddot)*

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)

)

*# Построение графиков: φ(t), ψ(t), R\_x(t), R\_y(t)*

fig1, axs *=* plt.subplots(2, 2, figsize*=*(10, 8))

*# График φ(t)*

axs[0, 0].plot(t, phi, label*=*"φ(t)")

axs[0, 0].set\_xlabel("t (c)")

axs[0, 0].set\_ylabel("φ (рад)")

axs[0, 0].set\_title("Угол φ(t)")

axs[0, 0].grid()

axs[0, 0].legend()

*# График ψ(t)*

axs[0, 1].plot(t, psi, color*=*"orange", label*=*"ψ(t)")

axs[0, 1].set\_xlabel("t (c)")

axs[0, 1].set\_ylabel("ψ (рад)")

axs[0, 1].set\_title("Угол ψ(t)")

axs[0, 1].grid()

axs[0, 1].legend()

*# График R\_x(t)*

axs[1, 0].plot(t, Rx, color*=*"green", label*=*"R\_x(t)")

axs[1, 0].set\_xlabel("t (c)")

axs[1, 0].set\_ylabel("R\_x (Н)")

axs[1, 0].set\_title("Реакция R\_x(t)")

axs[1, 0].grid()

axs[1, 0].legend()

*# График R\_y(t)*

axs[1, 1].plot(t, Ry, color*=*"red", label*=*"R\_y(t)")

axs[1, 1].set\_xlabel("t (c)")

axs[1, 1].set\_ylabel("R\_y (Н)")

axs[1, 1].set\_title("Реакция R\_y(t)")

axs[1, 1].grid()

axs[1, 1].legend()

plt.tight\_layout()

plt.show()

*# Анимация движения системы*

fig2, ax2 *=* plt.subplots(figsize*=*(6, 6))

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")

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

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

ani *=* FuncAnimation(

fig2,

animate,

frames*=*len(t),

init\_func*=*init\_animation,

interval*=*20,

blit*=*True,

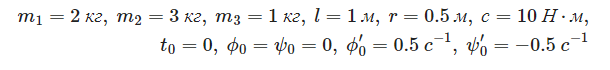
repeat*=*True,

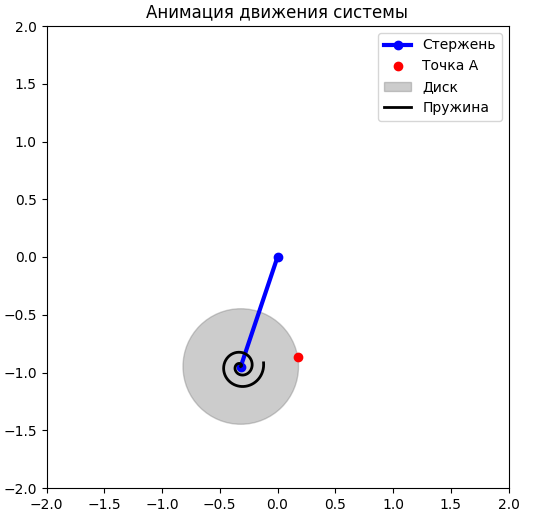
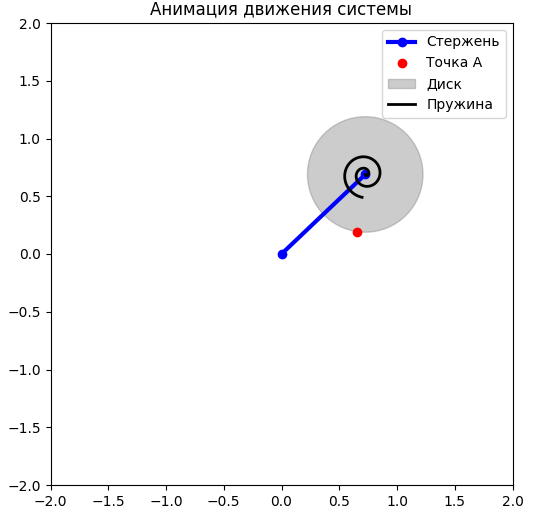
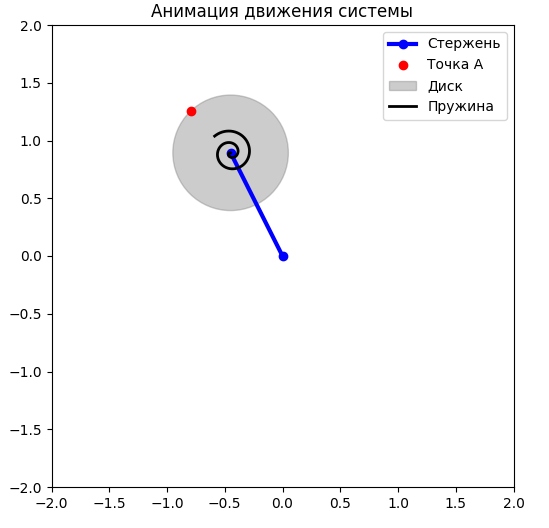
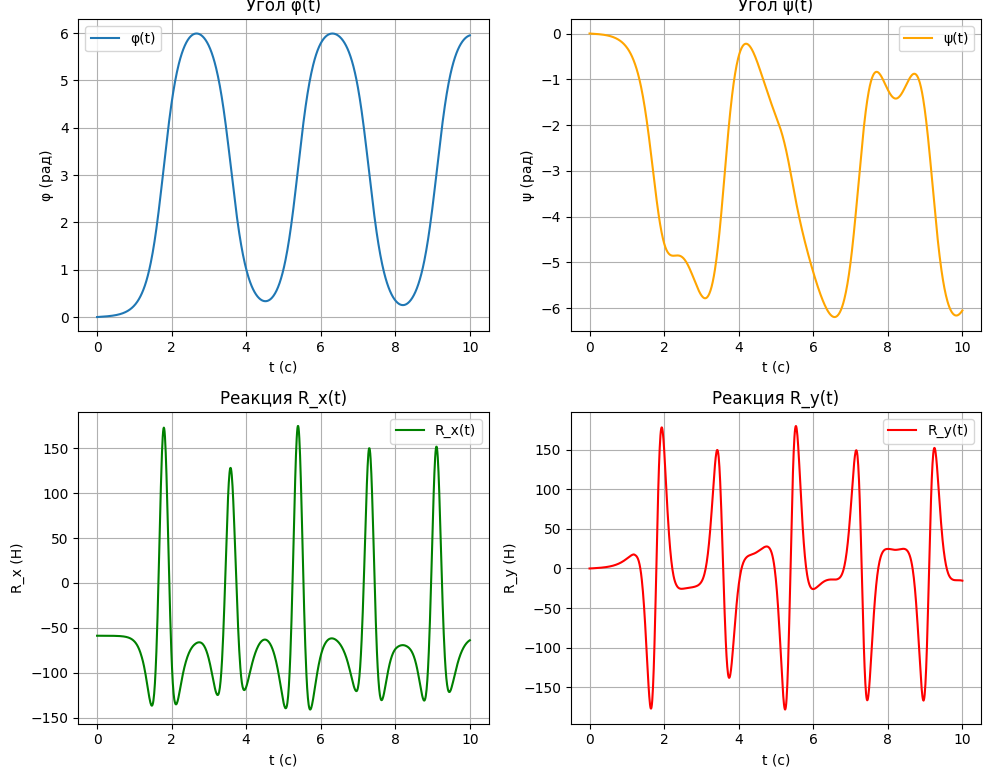
)

ax2.legend()

plt.show()

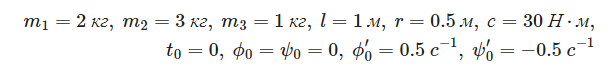
**Результат работы программы:**

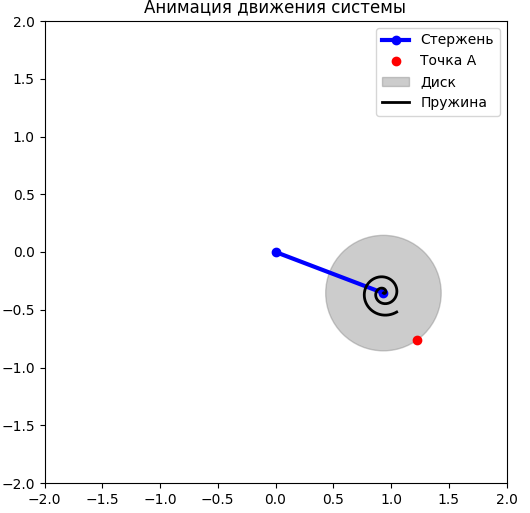
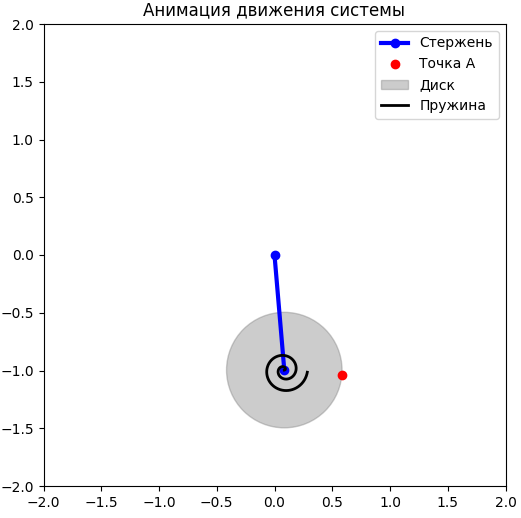
1. **Начальное состояние**

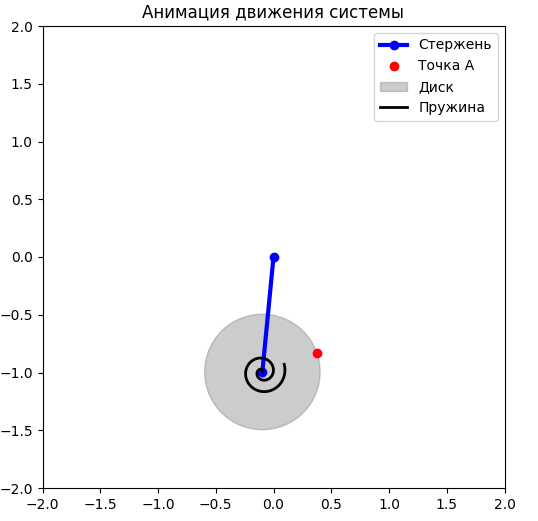
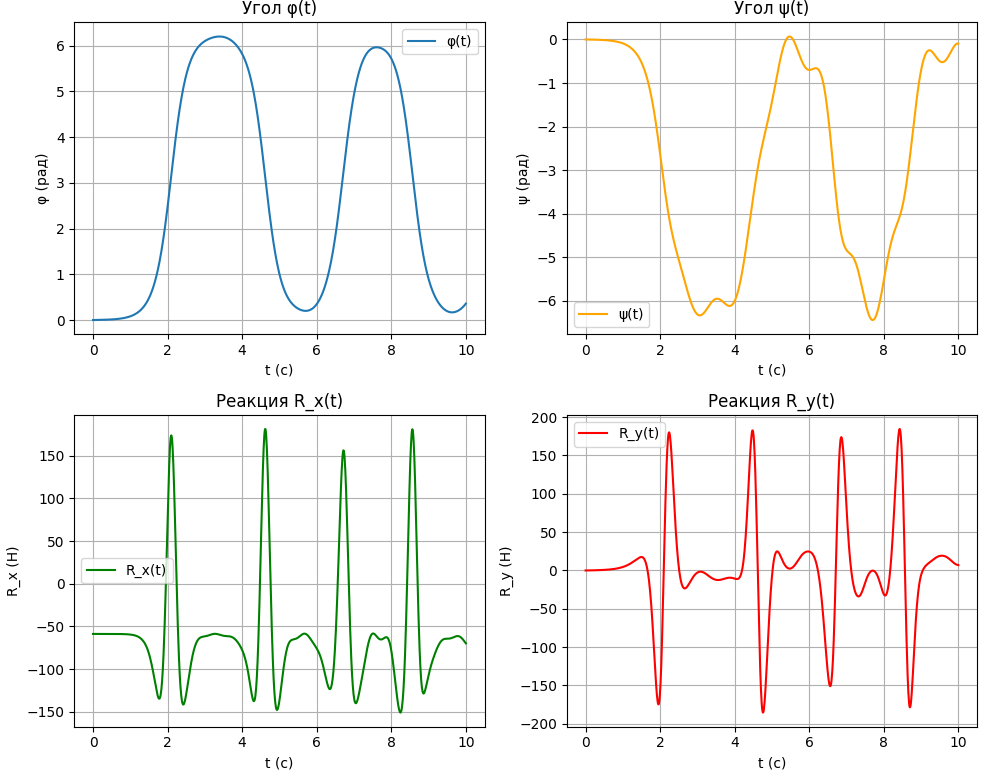


**Результат**: Анимация показывает плавное и гармоничное движение компонентов системы без значительных затуханий.

1. **Увеличение жёсткости пружины**

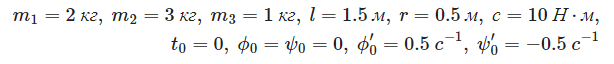


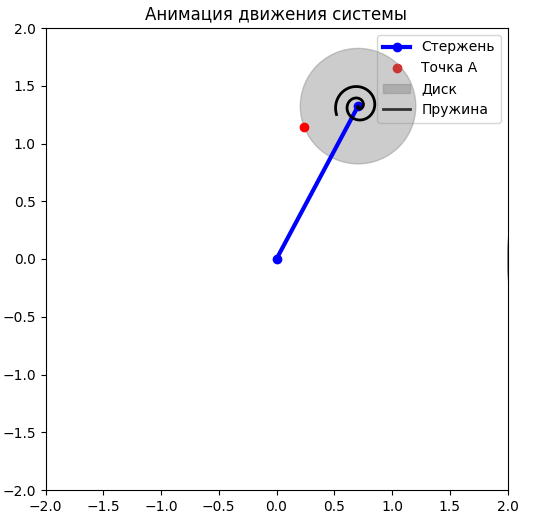
****

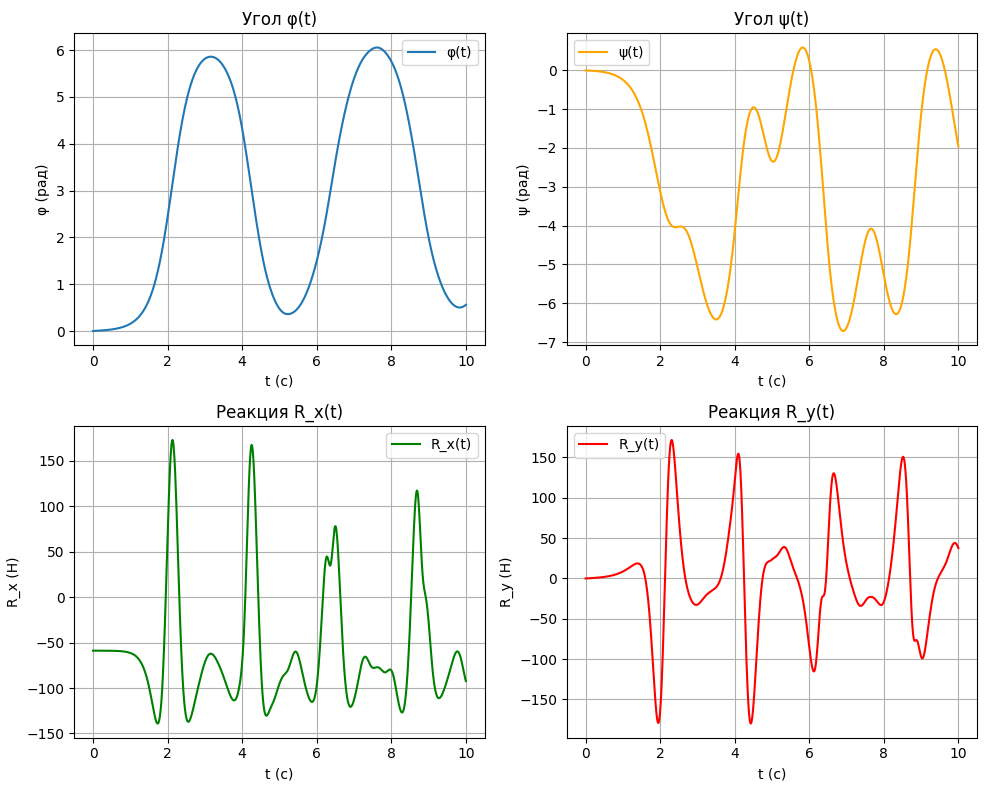
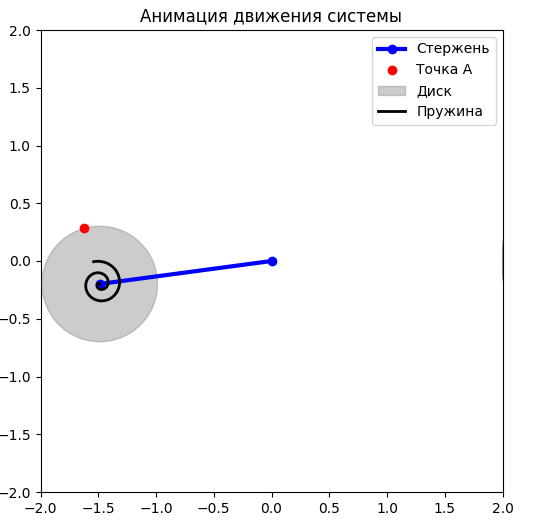
****

**Результат**: система стала немного быстрее возвращаться в положение равновесия. Колебания стали более частыми и с меньшей амплитудой.

1. **Увеличение длины стержня**

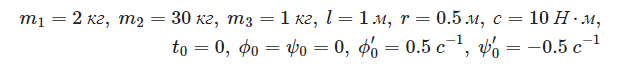


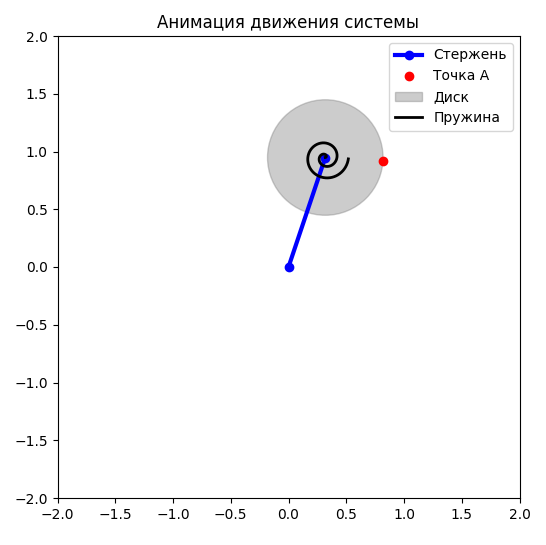
****

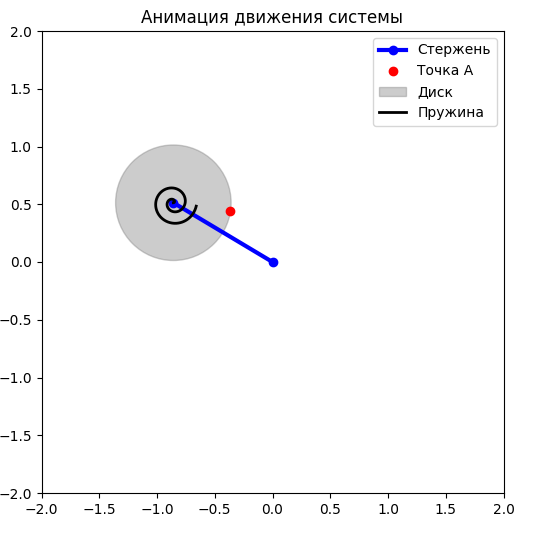
****

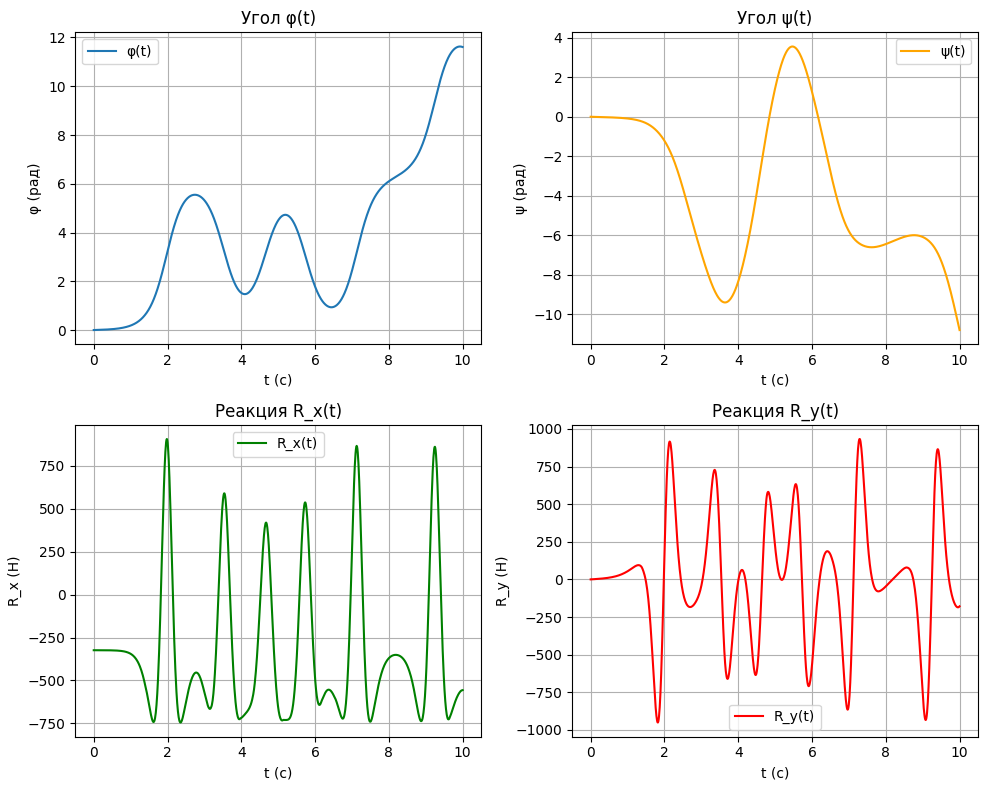
**Результат**: замедление колебаний системы. Система стала более инертной, колебания стали более плавными и длительными.

1. **Увеличение массы диска**



****

****

**Результат**: снизилась частота колебаний системы. Колебания затухают дольше, а движение стержня становится менее подвижным.

**Вывод**

В ходе выполнения данной работы была успешно интегрирована система дифференциальных уравнений движения механической системы с двумя степенями свободы с использованием языка программирования Python. Разработанная программа позволила решить систему уравнений численно и визуализировать поведение системы посредством анимации и построения графиков законов движения и реакций сил.

Полученные результаты продемонстрировали, как изменения параметров системы влияют на её динамику. Анимация наглядно показала движение компонентов системы, а графики позволили проанализировать изменения угловых координат и реакций сил во времени при различных условиях. Это способствовало лучшему пониманию взаимосвязей между физическими характеристиками системы и её поведением.