­­­­ Московский авиационный институт

(национальный исследовательский университет)

Институт № 8 «Информационные технологии и прикладная математика»

**Лабораторная работа №3**

**по курсу «Теоретическая механика и компьютерное моделирование»**

**Составление и численное решения дифференциальных уравнений движения системы и ее анимация.**

Выполнил студент группы М8О-207Б-20

Куценко Борис Дмитриевич

Преподаватель: Чекина Евгения Алексеевна

Оценка:

Дата: 27.12.2021

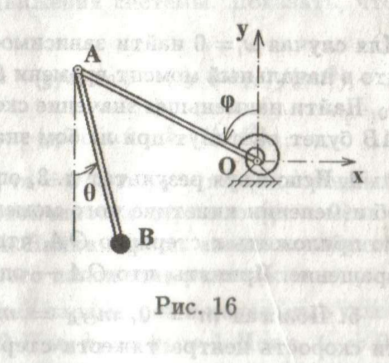
Москва, 2021

**Вариант № «16»**

**Задание:**

Необходимо составить и численно решить дифференциальные уравнения движения системы (уравнения Лагранжа второго рода), а затем реализовать анимацию движения механической системы используя язык программирования Python.

**Механическая система:**

****

**Текст программы:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from matplotlib.animation import FuncAnimation**

**import matplotlib.patches as pat**

**import sympy as sp**

**from scipy.integrate import odeint**

**import math**

**def formY(y, t, fomega\_phi, fomega\_thetta):**

**y1, y2, y3, y4 = y**

**dydt = [y3, y4, fomega\_phi(y1, y2, y3, y4), fomega\_thetta(y1, y2, y3, y4)]**

**return dydt**

**#time**

**t\_fin = 600**

**Steps = 10000**

**#constants**

**l = OA = AB = 5**

**R\_Circle = l / 10**

**k1 = 5**

**k2 = 12**

**m = 10**

**c = 20**

**g = 9.81**

**#starting postion**

**phi0 = 2**

**thetta0 = 0**

**dphi0 = 1**

**dthetta0 = 1**

**t = sp.Symbol('t')**

**phi = sp.Function('phi')(t)**

**thetta = sp.Function('thetta')(t)**

**omega\_phi = sp.Function('omega\_phi')(t)**

**omega\_thetta = sp.Function('omega\_thetta')(t)**

**#1 defining the kinetic energy**

**Vb2 = ((l \* omega\_phi) \*\* 2) + ((l \* omega\_thetta) \*\* 2) - 2 \* l \* l \* omega\_phi \* omega\_thetta \* sp.cos(phi - thetta)**

**K = m \* Vb2 / 2**

**#2 defining the potential energy**

**Pmg = m \* g \* l \* (sp.cos(phi + 3.14/2) - sp.cos(thetta))**

**P2 = c \* phi \* phi / 2**

**P = Pmg + P2**

**#Not potential force**

**Q1 = - k1 \* omega\_phi**

**Q2 = - k2 \* omega\_thetta**

**#Lagrange function**

**L = K - P**

**#equations**

**ur1 = sp.diff(sp.diff(L,omega\_phi),t)-sp.diff(L,phi) - Q1**

**ur2 = sp.diff(sp.diff(L,omega\_thetta),t)-sp.diff(L,thetta) - Q2**

**# isolating second derivatives(dV/dt and dom/dt) using Kramer's method**

**a11 = ur1.coeff(sp.diff(omega\_phi, t), 1)**

**a12 = ur1.coeff(sp.diff(omega\_thetta, t), 1)**

**a21 = ur2.coeff(sp.diff(omega\_phi, t), 1)**

**a22 = ur2.coeff(sp.diff(omega\_thetta, t), 1)**

**b1 = -(ur1.coeff(sp.diff(omega\_phi, t), 0)).coeff(sp.diff(omega\_thetta, t),**

**0).subs([(sp.diff(phi, t), omega\_phi),**

**(sp.diff(thetta, t), omega\_thetta)])**

**b2 = -(ur2.coeff(sp.diff(omega\_phi, t), 0)).coeff(sp.diff(omega\_thetta, t),**

**0).subs([(sp.diff(phi, t), omega\_phi),**

**(sp.diff(thetta, t), omega\_thetta)])**

**#we can check the result with:**

**# a11 = 1**

**# a12 = sp.cos(phi - thetta)**

**# a22 = 1**

**# a21 = sp.cos(phi - thetta)**

**# b1 = omega\_thetta \* omega\_thetta\*sp.sin(phi - thetta) + (g/l) \* sp.sin(phi) - (c\*phi + k1\* omega\_phi) / (m\*l\*l)**

**# b2 = -omega\_phi\*omega\_phi\*sp.sin(phi - thetta) - (g/l)\*sp.sin(thetta) - k2\*omega\_thetta/(m\*l\*l)**

**print(b1)**

**print(b2)**

**detA = a11\*a22-a12\*a21**

**detA1 = b1\*a22-b2\*a12**

**detA2 = a11\*b2-b1\*a21**

**domega\_phidt = detA1/detA**

**domega\_thettadt = detA2/detA**

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

**fomega\_phi = sp.lambdify([phi, thetta, omega\_phi, omega\_thetta], domega\_phidt, "numpy")**

**fomega\_thetta = sp.lambdify([phi, thetta, omega\_phi, omega\_thetta], domega\_thettadt, "numpy")**

**y0 = [phi0, thetta0, dphi0, dthetta0]**

**sol = odeint(formY, y0, Time, args=(fomega\_phi, fomega\_thetta))**

**Phi = sol[:, 0]**

**Thetta = sol[:, 1]**

**Omega\_phi = sol[:, 2]**

**Omega\_thetta = sol[:, 3]**

**X\_A = sp.lambdify(phi, l \* sp.cos(phi + 3.14/2))**

**Y\_A = sp.lambdify(phi, l \* sp.sin(phi + 3.14/2))**

**X\_B = sp.lambdify([phi, thetta], l \* sp.cos(phi + 3.14/2) + l \* sp.sin(thetta))**

**Y\_B = sp.lambdify([phi, thetta], l \* sp.sin(phi + 3.14/2) - l \* sp.cos(thetta))**

**XA = X\_A(sol[:, 0])**

**YA = Y\_A(sol[:, 0])**

**XB = X\_B(sol[:, 0], sol[:, 1])**

**YB = Y\_B(sol[:, 0], sol[:, 1])**

**Nv= 3**

**R1 = 0.001**

**R2 = l / 12**

**fig = plt.figure(figsize=(10,10))**

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

**ax.axis('equal')**

**ax.set(xlim=[-l \* 2 - R\_Circle - l/10, l \*2 + R\_Circle + l/10], ylim=[-l\*2 - R\_Circle - l/10,l\*2 + R\_Circle + l/10])**

**V\_B = sp.lambdify([phi, thetta, omega\_phi, omega\_thetta], sp.sqrt(((l \* omega\_phi) \*\* 2) + ((l \* omega\_thetta) \*\* 2) - 2 \* l \* l \* omega\_phi \* omega\_thetta \* sp.cos(phi - thetta)))**

**VB = V\_B(Phi, Thetta, Omega\_phi, Omega\_thetta)**

**alpha = np.linspace(0, Nv\*6.283+Phi[0]+ 3.14/2, 100)**

**X\_SpiralSpr = -(R1 + alpha \* (R2 - R1) / alpha[-1]) \* np.sin(alpha)**

**Y\_SpiralSpr = (R1 + alpha \* (R2 - R1) / alpha[-1]) \* np.cos(alpha)**

**beta = np.linspace(0, 2\*math.pi, 100)**

**X\_Circle = R\_Circle \* np.cos(beta)**

**Y\_Circle = R\_Circle \* np.sin(beta)**

**fig\_for\_graphs = plt.figure(figsize=[13, 7])**

**ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 1)**

**ax\_for\_graphs.plot(Time, Phi, color='blue')**

**ax\_for\_graphs.set\_title("Phi(t)")**

**ax\_for\_graphs.set(xlim=[0, 10])**

**ax\_for\_graphs.grid(True)**

**ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 2)**

**ax\_for\_graphs.plot(Time, Thetta, color='red')**

**ax\_for\_graphs.set\_title('Va(t)')**

**ax\_for\_graphs.set(xlim=[0, 10])**

**ax\_for\_graphs.grid(True)**

**ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2,2,3)**

**ax\_for\_graphs.plot(Time, Omega\_phi \* l, color='green')**

**ax\_for\_graphs.set\_title("phi'(t) = omega\_phi(t)")**

**ax\_for\_graphs.set(xlim=[0, 10])**

**ax\_for\_graphs.grid(True)**

**ax\_for\_graphs = fig\_for\_graphs.add\_subplot(2, 2, 4)**

**ax\_for\_graphs.plot(Time, VB, color='black')**

**ax\_for\_graphs.set\_title("Vb(t)")**

**ax\_for\_graphs.set(xlim=[0, 10])**

**ax\_for\_graphs.grid(True)**

**OX = ax.plot([-l \* 2, l\*2], [-l/10,-l/10], 'black', linestyle = '--')**

**Draw\_Spring = ax.plot(X\_SpiralSpr, Y\_SpiralSpr, color='#666666')[0]**

**Draw\_OA=ax.plot([0, XA[0]], [0, YA[0]], color='#808080')[0]**

**Draw\_AB=ax.plot([XA[0], XB[0]], [YA[0], YB[0]], color='#808080' )[0]**

**PointB = ax.plot(XB[0], YB[0])[0]**

**PointA = ax.plot(XA[0], YA[0], color='#a0a0a0', marker='o')[0]**

**Draw\_Circle = ax.plot(X\_Circle + XB[0], Y\_Circle + YB[0], color='black', linewidth=1)[0]**

**triangle = pat.Polygon([(0,0), (-l/10, -l/10), (l/10, -l/10)], color='#d3d3d3')**

**ax.add\_patch(triangle)**

**def update(i):**

**PointB.set\_data(XB[i],YB[i])**

**Draw\_OA.set\_data([0, XA[i]], [0, YA[i]])**

**PointA.set\_data(XA[i], YA[i])**

**Draw\_AB.set\_data([XA[i], XB[i]], [YA[i], YB[i]])**

**Draw\_Circle.set\_data(X\_Circle + XB[i], Y\_Circle + YB[i])**

**alpha = np.linspace(0, Nv\*6.28+Phi[i] + 3.14/2, 100)**

**X\_SpiralSpr = -(R1 + alpha \* (R2 - R1) / alpha[-1]) \* np.sin(alpha - 1.57)**

**Y\_SpiralSpr = (R1 + alpha \* (R2 - R1) / alpha[-1]) \* np.cos(alpha - 1.57)**

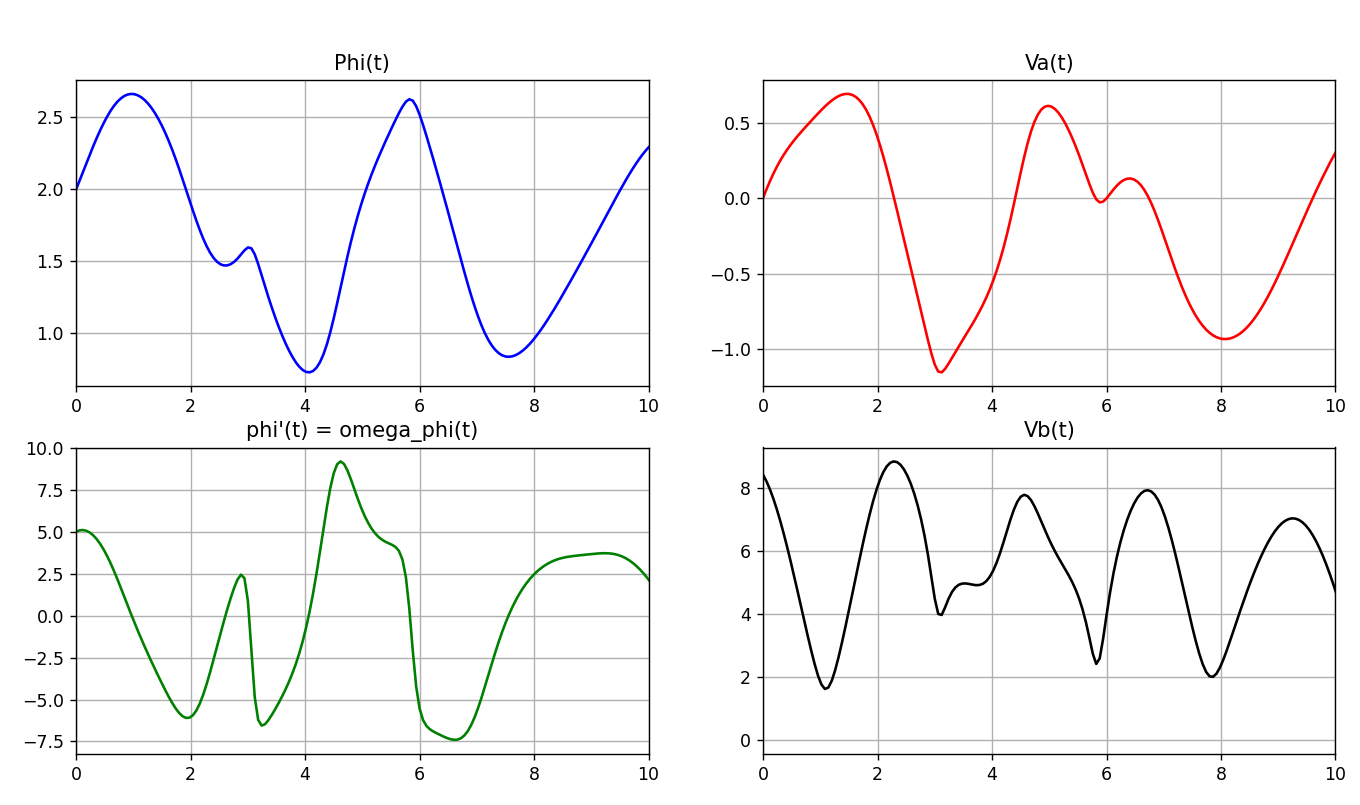
**Draw\_Spring.set\_data(X\_SpiralSpr, Y\_SpiralSpr)**

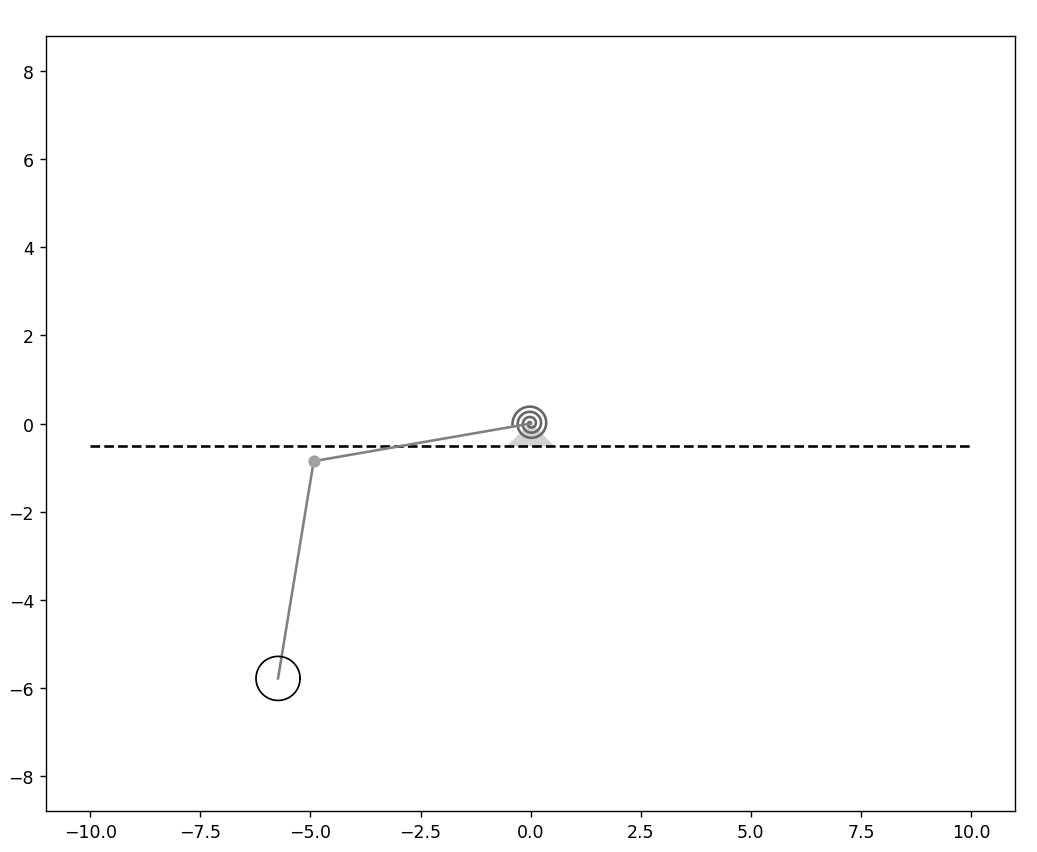
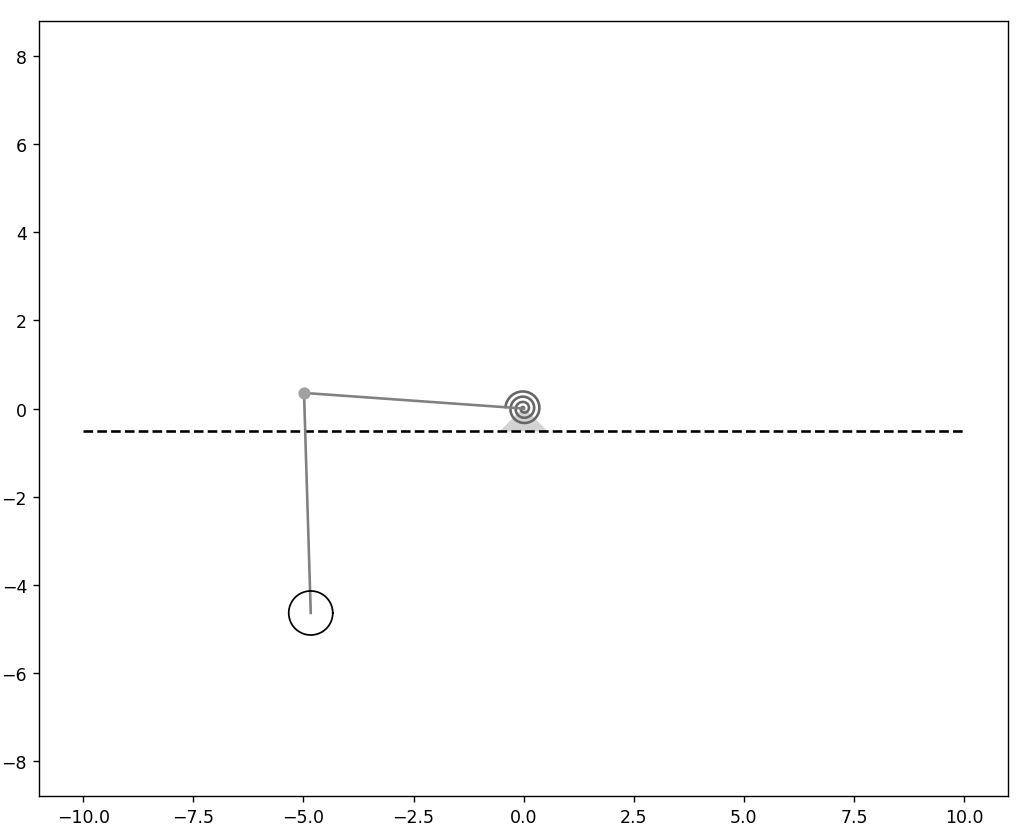
**return [PointB, Draw\_OA, Draw\_Spring, Draw\_AB, PointA, Draw\_Circle]**

**anima = FuncAnimation(fig, update, frames=Steps, interval=1)**

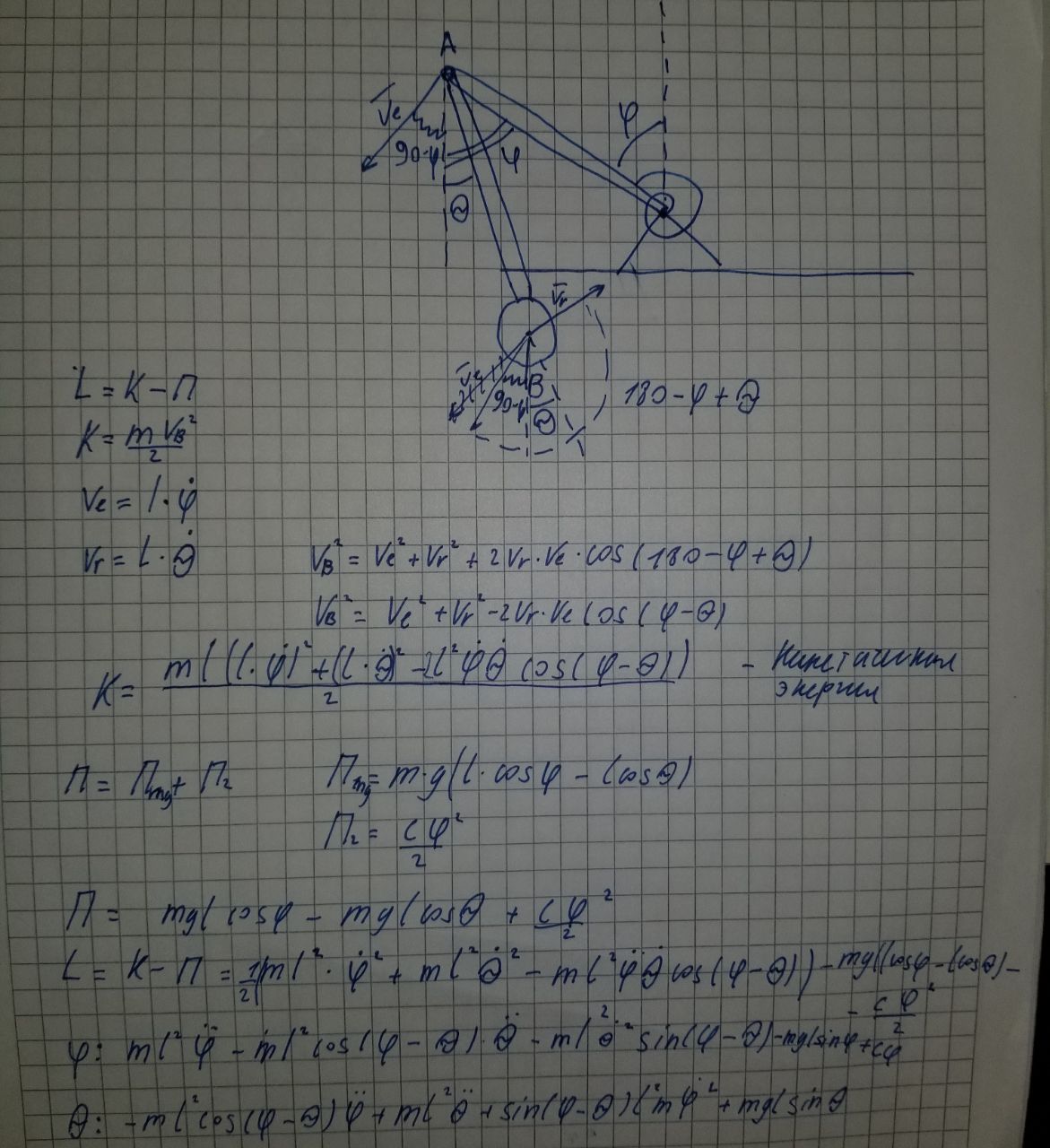
**plt.show()**

**Результат работы:**

****

**Получение уравнения Лагранжа второго рода:**

****