ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

**ОТЧЕТ**

**О ВЫПОЛОНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**«АНИМАЦИЯ ТОЧКИ»**

**ПО ДИСЦИПЛИНЕ «ТЕОРЕТИЧЕСКАЯ МЕХАНИКА»**

**ВАРИАНТ ЗАДАНИЯ №18**

Выполнил студент группы М80-203Б-23

Салихов Р.Р.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата

Проверил и принял

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

подпись, дата

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

Москва, 2024

**Задание:**

Построить заданную траекторию и анимацию движения точки, а также отобразить стрелки скорости и ускорения. Построить радиус кривизны траектории.

**Закон движения точки:**

**Текст программы**

|  |
| --- |
| import numpy as np # библиотека для мат операций  import matplotlib.pyplot as plt # для построения графиков  from matplotlib.animation import FuncAnimation # для создания анимации  import sympy as sp  # Определяю символьную переменную имеющую алгебраический смысл  t = sp.Symbol('t')  # Задаю радиус как функцию времени с гармоническим отклонением  r = 1 + 1.5 \* sp.sin(12 \* t)  # Задаю угол φ как линейную функцию времени с гармоническим отклонением  phi = 1.2 \* t + 0.2 \* sp.cos(12 \* t)  # Определяю координаты x и y в полярных координатах  x = r \* sp.cos(phi)  y = r \* sp.sin(phi)  # Вычисляю производные координат для получения компонент скорости  Vx = sp.diff(x, t) # f'(x) по времени  Vy = sp.diff(y, t) # f'(y) по времени  # Вычисляю вторые производные для получения компонент ускорения  Ax = sp.diff(Vx, t) # производная Vx по времени  Ay = sp.diff(Vy, t) # производная Vy по времени  # Вычисляю величину скорости  V = sp.sqrt(Vx\*\*2 + Vy\*\*2)  # Вычисляю кривизну траектории  kappa = sp.Abs(Vx \* Ay - Vy \* Ax) / V\*\*3  # Вычисляю радиус кривизны  rho = 1 / kappa  # Преобразовал символьные выражения в числовые функции для дальнейшего использования с numpy  F\_x = sp.lambdify(t, x, modules='numpy') # F x(t)  F\_y = sp.lambdify(t, y, modules='numpy') # F y(t)  F\_Vx = sp.lambdify(t, Vx, modules='numpy') # F Vx(t)  F\_Vy = sp.lambdify(t, Vy, modules='numpy') # F Vy(t)  F\_Ax = sp.lambdify(t, Ax, modules='numpy') # F Ax(t)  F\_Ay = sp.lambdify(t, Ay, modules='numpy') # F Ay(t)  F\_rho = sp.lambdify(t, rho, modules='numpy') # F rho(t)  # Создал массив значений времени от 0 до 2pi с 1000 точками  t\_vals = np.linspace(0, 2 \* np.pi, 1000)  # Вычисляю значения координат, скорости, ускорения и радиуса кривизны для всех точек времени  x\_vals = F\_x(t\_vals)  y\_vals = F\_y(t\_vals)  Vx\_vals = F\_Vx(t\_vals)  Vy\_vals = F\_Vy(t\_vals)  Ax\_vals = F\_Ax(t\_vals)  Ay\_vals = F\_Ay(t\_vals)  rho\_vals = F\_rho(t\_vals)  # Вычислил углы направлений вектора скорости и ускорения  Alpha\_V = np.arctan2(Vy\_vals, Vx\_vals) # угол вектора скорости  Alpha\_A = np.arctan2(Ay\_vals, Ax\_vals) # угол вектора ускорения  # Настройка графика  fig, ax = plt.subplots(figsize=(12, 12)) # создал фигуру и оси с размером 12x12 дюймов  ax.axis('equal') # установил равные масштабы по осям  ax.set\_xlim(-8, 8) # установил пределы по оси X  ax.set\_ylim(-8, 8) # установил пределы по оси Y  ax.grid(True) # включил сетку на графике  ax.plot(x\_vals, y\_vals, label='Траектория') # нарисовал траекторию движения  # Задал объекты для анимации  P, = ax.plot([], [], 'ro', label='Точка') # точка на траектории  V\_line, = ax.plot([], [], color='red', label='Скорость') # линия - вектор скорости  V\_arrow, = ax.plot([], [], color='red') # стрелка скорости  A\_line, = ax.plot([], [], color='green', label='Ускорение') # линия - вектор ускорения  A\_arrow, = ax.plot([], [], color='green') # стрелка ускорения  circle = plt.Circle((0, 0), 0, color='purple', fill=False, label='Радиус кривизны') # радиус кривизны  ax.add\_patch(circle) # добавил круг на график  # Добавляем радиус-вектор  R\_line, = ax.plot([], [], color='orange', linestyle='--', label='Радиус-вектор') # радиус-вектор  # Настройка для стрелок  a = 0.05 # полуширина стрелки  b = 0.025 # полувысота стрелки  x\_arr = np.array([-a, 0, -a]) # X для формы стрелки  y\_arr = np.array([b, 0, -b]) # Y для формы стрелки  k\_V = 0.2 # Масштаб для вектора скорости  k\_A = 0.05 # Масштаб для вектора ускорения  max\_rho = 3 # максимальный радиус кривизны для отображения  max\_center\_dist = 6 # максимальное расстояние до центра кривизны для отображения  # F поворота координат на заданный угол  def Rot2D(X, Y, Alpha):  RotX = X \* np.cos(Alpha) - Y \* np.sin(Alpha) # поворот координаты X  RotY = X \* np.sin(Alpha) + Y \* np.cos(Alpha) # поворот координаты Y  return RotX, RotY  # Функция обновления кадров анимации  def animate(i):  P.set\_data(x\_vals[i], y\_vals[i]) # обновил положение точки P  # Вычисляю конец вектора скорости  V\_end\_x = x\_vals[i] + k\_V \* Vx\_vals[i]  V\_end\_y = y\_vals[i] + k\_V \* Vy\_vals[i]  V\_line.set\_data([x\_vals[i], V\_end\_x], [y\_vals[i], V\_end\_y]) # обновил линию скорости    # Повернул стрелку скорости в нужном направлении  RotX\_V, RotY\_V = Rot2D(x\_arr, y\_arr, Alpha\_V[i])  V\_arrow.set\_data(V\_end\_x + RotX\_V, V\_end\_y + RotY\_V) # обновил стрелку скорости    # Вычислил конец вектора ускорения  A\_end\_x = x\_vals[i] + k\_A \* Ax\_vals[i]  A\_end\_y = y\_vals[i] + k\_A \* Ay\_vals[i]  A\_line.set\_data([x\_vals[i], A\_end\_x], [y\_vals[i], A\_end\_y]) # обновил линию ускорения    # Повернул стрелку ускорения в нужном направлении  RotX\_A, RotY\_A = Rot2D(x\_arr, y\_arr, Alpha\_A[i])  A\_arrow.set\_data(A\_end\_x + RotX\_A, A\_end\_y + RotY\_A) # обновил стрелку ускорения    # Вычислил величину скорости  V\_mag = np.sqrt(Vx\_vals[i]\*\*2 + Vy\_vals[i]\*\*2)  if V\_mag == 0:  circle.set\_visible(False) # скрываю круг радиуса кривизны если v=0  else:  # Вычислил нормаль к траектории  Nx = -Vy\_vals[i] / V\_mag  Ny = Vx\_vals[i] / V\_mag  # Нашёл центр кривизны  x\_c = x\_vals[i] + Nx \* rho\_vals[i]  y\_c = y\_vals[i] + Ny \* rho\_vals[i]    # Вычислил расстояние до центра кривизны  center\_dist = np.sqrt((x\_c - x\_vals[i])\*\*2 + (y\_c - y\_vals[i])\*\*2)    # Проверил условия отображения круга радиуса кривизны  if rho\_vals[i] <= max\_rho and center\_dist <= max\_center\_dist:  circle.set\_visible(True) # отобразил круг  circle.center = (x\_c, y\_c) # установил центр круга  circle.radius = rho\_vals[i] # установил радиус круга  else:  circle.set\_visible(False) # скрыл круг, если условия не выполнены    # Обновляем радиус-вектор  R\_line.set\_data([0, x\_vals[i]], [0, y\_vals[i]]) # обновил радиус-вектор    # Вернул обновлённые объекты для перерисовки  return P, V\_line, V\_arrow, A\_line, A\_arrow, circle, R\_line  # Создал анимацию с заданными параметрами  ani = FuncAnimation(fig, animate, frames=len(t\_vals), interval=20, blit=True)  # Добавил описание к графику  ax.legend()  # Вывел график  plt.show() |

**Результат работы программы:**

|  |  |
| --- | --- |
|  |  |
|  |  |

**Вывод:**

В ходе выполнения лабораторной работы была реализована анимация движения точки по заданной траектории, а также отображение вектора скорости, ускорения и радиуса кривизны траектории. Основной целью работы было проанализировать и визуализировать динамику движения точки, используя математические функции, которые описывают её траекторию, скорость и ускорение.

Реализованная анимация позволяет наблюдать изменение положения точки, а также динамику скорости и ускорения в процессе её движения. Радиус кривизны, который меняется по мере движения точки, наглядно демонстрирует изменение кривизны траектории.

Таким образом, данное задание позволило на практике продемонстрировать важные механические характеристики движения точки, такие как скорость, ускорение и кривизна траектории, а также научиться визуализировать эти характеристики с помощью программных инструментов, таких как Python и библиотека Matplotlib.