МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**Курсовая проект по дисциплине:**

**«МЕХАНИКА»**

**Проектирование механической модели катапульты**

Факультет: Институт Интеллектуальной Робототехники

Группа: 23932

|  |  |
| --- | --- |
| Студенты: | Оценка |
| Бакумова Валерия Евгеньевна |  |
| Карпачев Дмитрий Александрович |  |

Преподаватель:  *Сахнов А.Ю.*

НОВОСИБИРСК

2025

**1. Задание на курсовую работу.**

Критерии работоспособности механической модели требушета:

* Метание снаряда любой выбранной массы и формы на расстояние от 50 см до 80 см c отклонением от центральной оси основного направления полёта снаряда не более 30°.
* Нахождение модели во взведённом состоянии без приложения посторонних сил (без помощи человека)
* Механический спуск
* Целостность и устойчивость конструкции в течение 3-х попыток
* Снаряд должен быть сделан из мягкого упругого материала, либо покрыт материалом с такими свойствами.
* Масса снаряда: 0,05 кг

**2. Эскиз модели.**

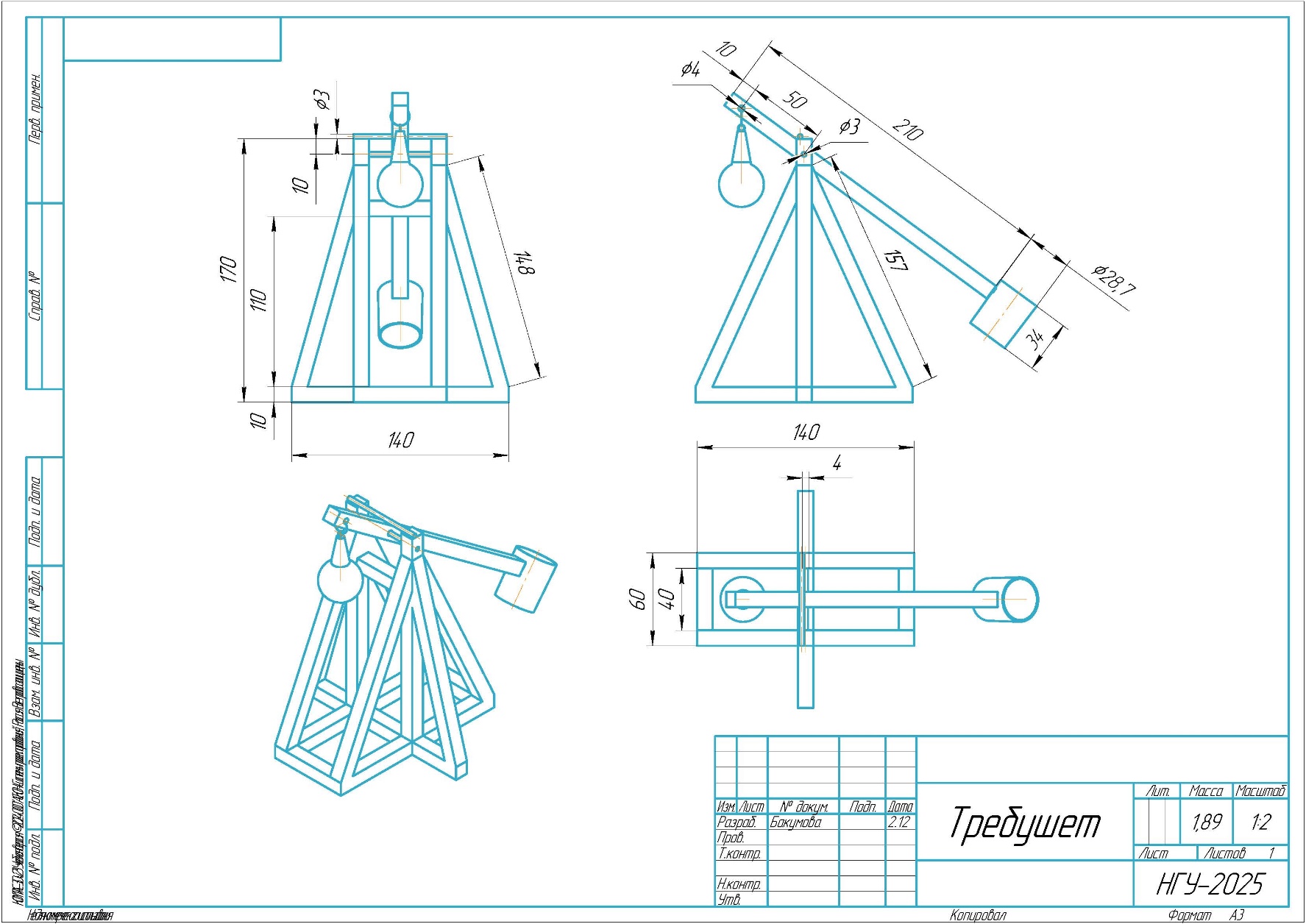


Рисунок 1 – Эскиз модели требушета

**3. Динамический анализ механической модели (Расчёт разгона снаряда)**

Таблица 1 – Параметры модели

|  |  |  |
| --- | --- | --- |
| **Параметр** | **Значение** | **Описание** |
|  | 0.08 кг | Масса груза |
|  | 0.003 кг | Масса снаряда |
|  | 0.006 кг | Масса рычага |
|  | 0.055 м | Длина малого плеча (плеча груза) |
|  | 0.15 м | Длина длинного плеча (плеча снаряда) |
|  | рад | Начальное положение рычага (угол) |
|  | 2 рад | Финальное положение рычага (угол вылета снаряда) |
|  | 0.16 м | Высота расположения оси вращения рычага над землёй |
|  | 0.008 м | Радиус снаряда |
|  |  | Собственная инерция груза как сферы радиуса 3 см |

Разгон снаряда происходит за счёт действия силы тяжести груза на малом плече. Для получения скорости воспользуемся вторым законом ньютона для вращательного движения:

1) Расчёт суммы моментов сил:

Изображение выглядит как диаграмма, линия, График

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 - Моменты от сил тяжести

Всего с модели 3 момента, которые напрямую влияют на движение рычага: момент груза, момент рычага и момент снаряда.

Угол между рычагом и вертикалью (угол приложения сил тяжести к рычагу) -

Тогда формула момента для силы, действующей под углом для нашего случая, преобразуется в

Момент снаряда:

Момент рычага:

Момент груза:

Тогда:

Пусть

2) Расчёт инерций:

Для снаряда (материальная точка):

Для рычага и груза воспользуемся Теоремой Штейнера для учёта их собственных инерций:

Груз:

Рычаг:

Итоговая инерция системы:

3) Переходим к дифференциальному уравнению для нахождения связи и :

Базовая форма:

Тождество:

Тогда получаем дифференциальное уравнение:

Найдём константы из начальных условий и :

Тогда получаем

Исходя из того, что система стартует из состояния покоя () и движение от до уменьшает угол получаем итоговую зависимость угловой скорости от текущего угла:

Получим вектор линейной скорости снаряда через векторное произведение :

,

Вычислив матричным методом данное произведение получим:

Отсюда начальные скорости для снаряда при выпуске в :

4) Численный расчёт стартовой скорости снаряда:

Дано:

- длины плеч рычага

- массы снаряда, груза и рычага

- углы, задающие начальное и конечное положения рычага

Найти:

- ? м/с - скорости вдоль координатных осей в момент вылета снаряда

Решение:

Найдём константную часть суммы моментов:

Общая инерция системы:

Угловая скорость:

Скорость вдоль осей:

Итого:

**4. Кинематический анализ механической модели (Расчёт траектории полёта снаряда)**

Дано:

- начальные скорости (из раздела номер 4)

- длина плеча снаряда

- угол наклона рычага в момент вылета снаряда

- высота крепления оси

- радиус снаряда

Найти:

- ?

- ? с

- ? м

- ? м

Решение:

Изображение выглядит как диаграмма, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3 – График траектории полета снаряда

Стартовое положение снаряда в пространстве:

- равномерное движение по оси OX, следовательно a=0

- равноускоренное падение по оси OY

Траектория полёта снаряда:

Время и расстояние падения:

- время полёта

- дальность полёта

- расстояние полёта относительно оси вращения рычага

Численный расчёт:

Итого:

**5. Обоснование устойчивости механической модели (Определение центра тяжести)**

Найти точку центра тяжести требушета.

1. Разбиваем сложную фигуру на простые.

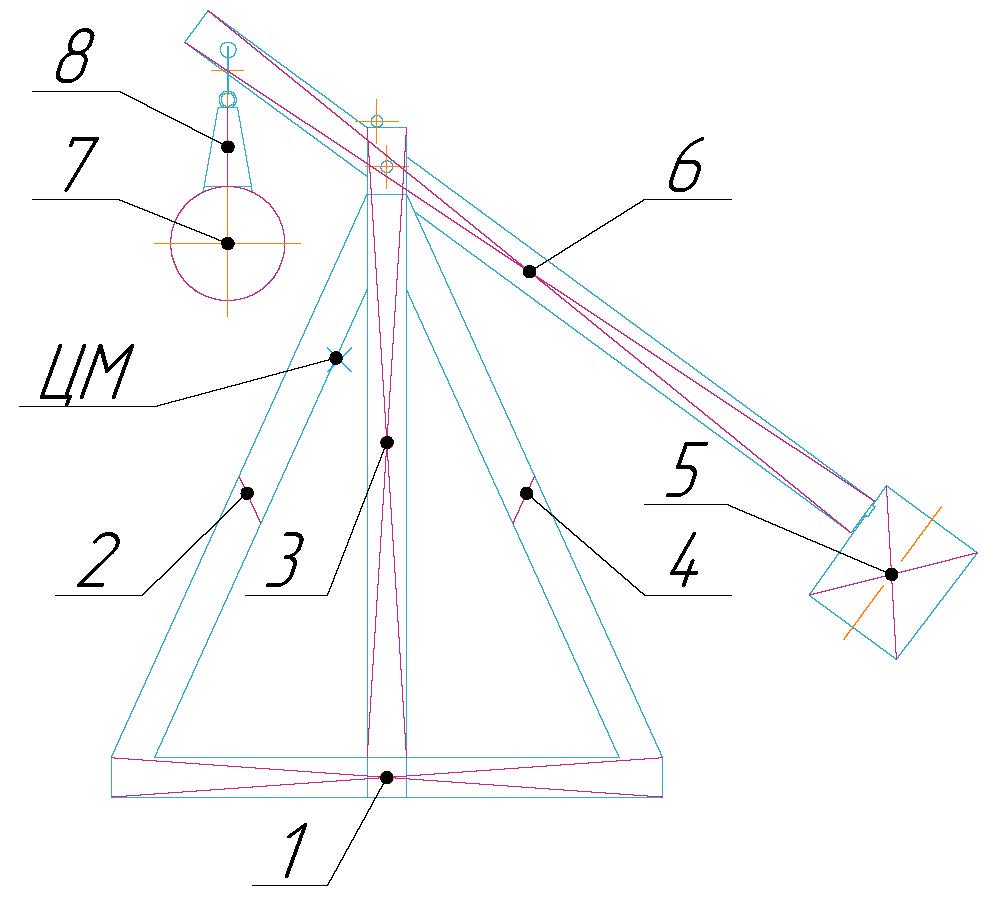


Рисунок 4 – Разбиение модели на простые фигуры.

1. Найдем координаты центров тяжести всех простых фигур:

Рассчитаем площади всех простых фигур:

1. Центр тяжести требушета считаем с учётом масс, потому что грузило сопоставимо по массе с каркасом, и без его вклада теоретический результат не совпадёт с экспериментальным.

,,

,,

Масса требушета без грузила . Масса грузила .

,

**6. Сравнение фактических параметров механической модели с расчётными параметрами.**

1) Оценить точность расчета расстояния полета

Результаты 3 запусков:

Абсолютная погрешность:

Относительная погрешность:

2) Оценить точность расчета центра тяжести

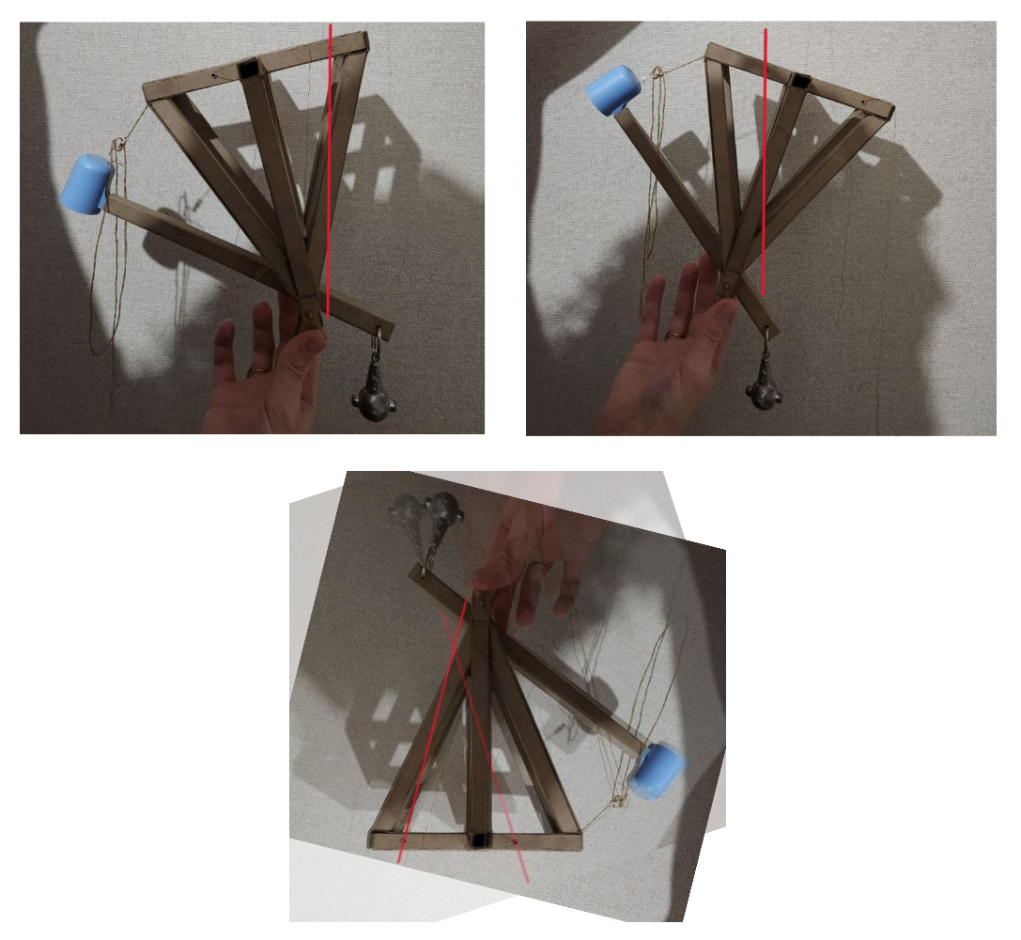


Рисунок 5 - Экспериментальное определение центра тяжести

Полученный фактический центр тяжести:

Абсолютная погрешность:

Относительная погрешность:

Относительные погрешности определения координат центра тяжести составили около 5.6% по оси X и 17.1% по оси Y, что не превышает 20%, поэтому теоретическое и экспериментальное положения центра тяжести можно считать согласующимися, а расчёты — корректными. Более высокая погрешность по оси Y объясняется тем, что положение грузила и нити в эксперименте сильнее влияет именно на вертикальную координату: небольшие изменения длины подвеса, точки крепления или угла рычага заметно смещают центр масс по высоте, но почти не изменяют его проекцию на ось X. При этом относительная погрешность расчёта дальности полёта снаряда не превышает 0.13%, что подтверждает адекватность выбранной модели движения и в целом позволяет считать курсовой проект выполненным успешно.

**7. Описание электронной модели механической системы**

1. import numpy as np
2. import matplotlib
3. from scipy.integrate import solve\_ivp
4. matplotlib.use('Agg')
5. import matplotlib.pyplot as plt 6

7

1. class const:
2. # trebushet params
3. h = 0.16
4. l1, l2 = 0.055, 0.15
5. l3 = (l1 + l2) / 2
6. phi\_start = 5 \* np.pi / 4
7. phi\_end = 2
8. r = 0.012
9. mg = 80 / 1000
10. ms = 3 / 1000
11. mr = 6 / 1000
12. Jg = 0.4 \* mg \* 0.03 \*\* 2

20

1. # windage
2. S = np.pi \* r \*\* 2
3. C = 0.47
4. rho = 1.225

25

1. # target zone
2. d1, d2 = 0.5, 0.8

28

1. # gravity constant
2. g = 9.81

31

32

1. # shell's inertia
2. Js = const.ms \* const.l2 \*\* 2
3. # lever's inertia
4. Jr = const.mr \* (const.l1 + const.l2) \*\* 2 / 12 + const.mr \* const.l3 \*\* 2
5. # load's inertia
6. Jg = const.Jg + const.mg \* const.l1 \*\* 2
7. # common inertia
8. J = Js + Jr + Jg
9. print("Common inertia:", J, "kg\*m^2") 42
10. # constant of moment
11. M0 = const.g \* (const.mg \* const.l1 - const.ms \* const.l2 - const.mr

\* const.l3)

1. print("Constant of moment", M0, "H") 46
2. # rotation speed by the angle
3. w = -np.sqrt(2 \* M0 / J \* (np.sin(const.phi\_end) - np.sin(const.phi\_start)))
4. print("Rotation speed:", w, "rad/s") 50
5. # vector of velocity by the coordinates
6. vox = -const.l2 \* np.sin(const.phi\_end) \* w
7. voy = const.l2 \* np.cos(const.phi\_end) \* w
8. print("Velosity X:", vox, "m/s")
9. print("Velocity Y:", voy, "m/s") 56
10. # state of rest position of the lever
11. A0 = (const.l2 \* np.cos(const.phi\_start), const.h + const.l2 \* np.sin(const.phi\_start))
12. B0 = (-const.l1 \* np.cos(const.phi\_start), const.h - const.l1 \* np.sin(const.phi\_start))

60

1. # start position of the level
2. A = (const.l2 \* np.cos(const.phi\_end), const.h + const.l2 \* np.sin(const.phi\_end))
3. x0, y0 = A
4. B = (-const.l1 \* np.cos(const.phi\_end), const.h - const.l1 \* np.sin(const.phi\_end))
5. print("Start x:", x0, "m")
6. print("Start y:", y0, "m") 67

68

1. def f(x):
2. '''
3. Trajectory of shell's movement
4. '''
5. return y0 + voy / vox \* (x - x0) - const.g \* (x - x0) \*\* 2 / (2

\* vox \*\* 2) 74

75

1. # calculating shot distance
2. D = 4 \* (voy \*\* 2 + 2 \* const.g \* (y0 - const.r))
3. T = (2 \* voy + np.sqrt(D)) / (2 \* const.g)
4. print("Flying time:", T, "s")
5. L = x0 + vox \* T
6. print("Distance", L, "m")
7. Ly = f(L)

83

84

1. def projectile\_motion(t, state, m, g, rho, C\_d, A):
2. '''
3. Equation of motion with air resistance
4. '''
5. \_, vx, \_, vy = state
6. v = np.sqrt(vx \*\* 2 + vy \*\* 2) 91
7. # power of air resistance
8. F\_d = 0.5 \* rho \* C\_d \* A \* v \*\* 2

94

1. dxdt = vx
2. dvxdt = - (F\_d / m) \* (vx / v) if v > 0 else 0
3. dydt = vy
4. dvydt = -g - (F\_d / m) \* (vy / v) if v > 0 else -g 99

100 return [dxdt, dvxdt, dydt, dvydt] 101

102

1. # intial state of movement
2. initial\_state = [A[0], vox, A[1], voy] 105
3. # 10 seconds to move
4. t\_span = (0, 10)
5. t\_eval = np.linspace(0, 10, 1000)

109

1. # solving the equation
2. solution = solve\_ivp(
3. projectile\_motion,
4. t\_span,
5. initial\_state,
6. args=(const.ms, const.g, const.rho, const.C, const.S),
7. t\_eval=t\_eval,
8. method='RK45',
9. rtol=1e-6,
10. atol=1e-9

120 )

121

1. # result trajectory with air resistance
2. x = solution.y[0]
3. y = solution.y[2] 125
4. # searching final point
5. ground\_idx = None
6. for i in range(1, len(y)):
7. if y[i] <= const.r:

130 x\_ground = x[i-1] + (const.r - y[i-1]) \* (x[i] - x[i-1]) / (y[i] - y[i-1])

131 y\_ground = const.r 132

1. x = np.append(x[:i], x\_ground)
2. y = np.append(y[:i], y\_ground)
3. ground\_idx = i
4. break

137

1. if ground\_idx is None:
2. print("Warning: Projectile didn't hit the ground within simulation time")
3. ground\_idx = -1

141

142 print("Distance with air resistance:", x[-1], "m") 143

144

1. # plotting
2. fig, ax = plt.subplots(1, 1, figsize=(10, 5))

147

1. # target zone
2. ax.axvspan(const.d1, const.d2, alpha=0.3, color='#cba6f7', label="Target zone")

150

1. # trajectories
2. X = np.linspace(x0, L, 300)
3. ax.plot(X, f(X), "g", linewidth=2, label="Trajectory of movement")
4. ax.plot(x, y, "r--", linewidth=1, alpha=0.7, label="Real trajectory

of movement") 155

1. # lever
2. ax.plot([A0[0], B0[0]], [A0[1], B0[1]], "r--o", alpha=0.4,

label="Start position")

1. ax.plot([A[0], B[0]], [A[1], B[1]], "r-o", label="Shot position") 159
2. # final position
3. ax.axvline(L, color="k", linestyle="--", label=f"Final distance =

{L:.4f}m")

1. ax.axvline(x[-1], color="k", linestyle="--", linewidth=1, label=f"Final real distance = {x[-1]:.4f}m")

163

1. # adding circles
2. circle\_end = plt.Circle((L, Ly), radius=const.r, color='green', alpha=0.5, fill=True, label="Final shell's position")
3. circle\_real = plt.Circle((x[-1], y[-1]), radius=const.r, color='red', alpha=0.3, fill=True, label="Final shell's real position")
4. circle\_start = plt.Circle(A, radius=const.r, linestyle="--", color='blue', alpha=0.3, fill=True)
5. circle0 = plt.Circle(A0, radius=const.r, linestyle="--", color='blue', alpha=0.2, fill=True)
6. ax.add\_patch(circle\_end)
7. ax.add\_patch(circle\_real)
8. ax.add\_patch(circle\_start)
9. ax.add\_patch(circle0)

173

1. ax.axis("scaled")
2. ax.set\_ylim(bottom=0)
3. ax.legend()

177

1. plt.grid(True, alpha=0.3)
2. plt.tight\_layout()
3. plt.savefig("trajectory.png", dpi=200)

181

Изображение выглядит как текст, диаграмма, линия, График

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 - Рассчитанная в коде траектория полета

**8. Список литературы**

1. Сахнов, А. Ю. Методические пособие к выполнению курсовой работы по дисциплине "Механика" / А. Ю. Сахнов ; Новосибирский государственный университет. – Новосибирск, 2025. – 6 с.

2. Мещерский, И.В. Сборник задач по теоретической механике: Учеб. пособие. – 36-е изд., исправл. / Под ред. Н. В. Бутенина, А. И. Лурье, Д. Р. Меркина. – М.: Наука. Гл. ред. физ.-мат. лит., 1986. – 448 с.

3. Правила оформления курсовых работ и ВКР [Электронный ресурс] / НГУ. – URL: https://www.nsu.ru/n/humanities-institute/students/oform-vkr/ (дата обращения: 09.12.2025)

4. Matplotlib: Visualization with Python [Электронный ресурс]. – URL: https://matplotlib.org/stable/users/index.html (дата обращения: 09.12.2025).

5. NumPy User Guide [Электронный ресурс]. – URL: https://numpy.org/doc/stable/user/index.html (дата обращения: 09.12.2025).

6. Python Documentation [Электронный ресурс]. – URL: https://docs.python.org/3/ (дата обращения: 09.12.2025).