МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Курсовая проект по дисциплине:

«МЕХАНИКА»

Проектирование механической модели требушета

Факультет: Институт Интеллектуальной Робототехники
Группа: 22932

Студенты:	Оценка
Кадиленко Иван	
Кузнецов Глеб	

Преподаватель: *Сахнов А.Ю.*

НОВОСИБИРСК
2024

1. Задание на курсовую работу.

1. Метание деформируемого снаряда любой выбранной массы и формы на расстояние от 30 см до 60 см и отклонением от центральной оси не более 30°.

2. Нахождение модели во взведённом состоянии без приложения посторонних сил (без помощи человека).

3. Механический спуск.

4. Целостность и устойчивость конструкции в течение 3-х попыток метания.

2. Эскиз модели.

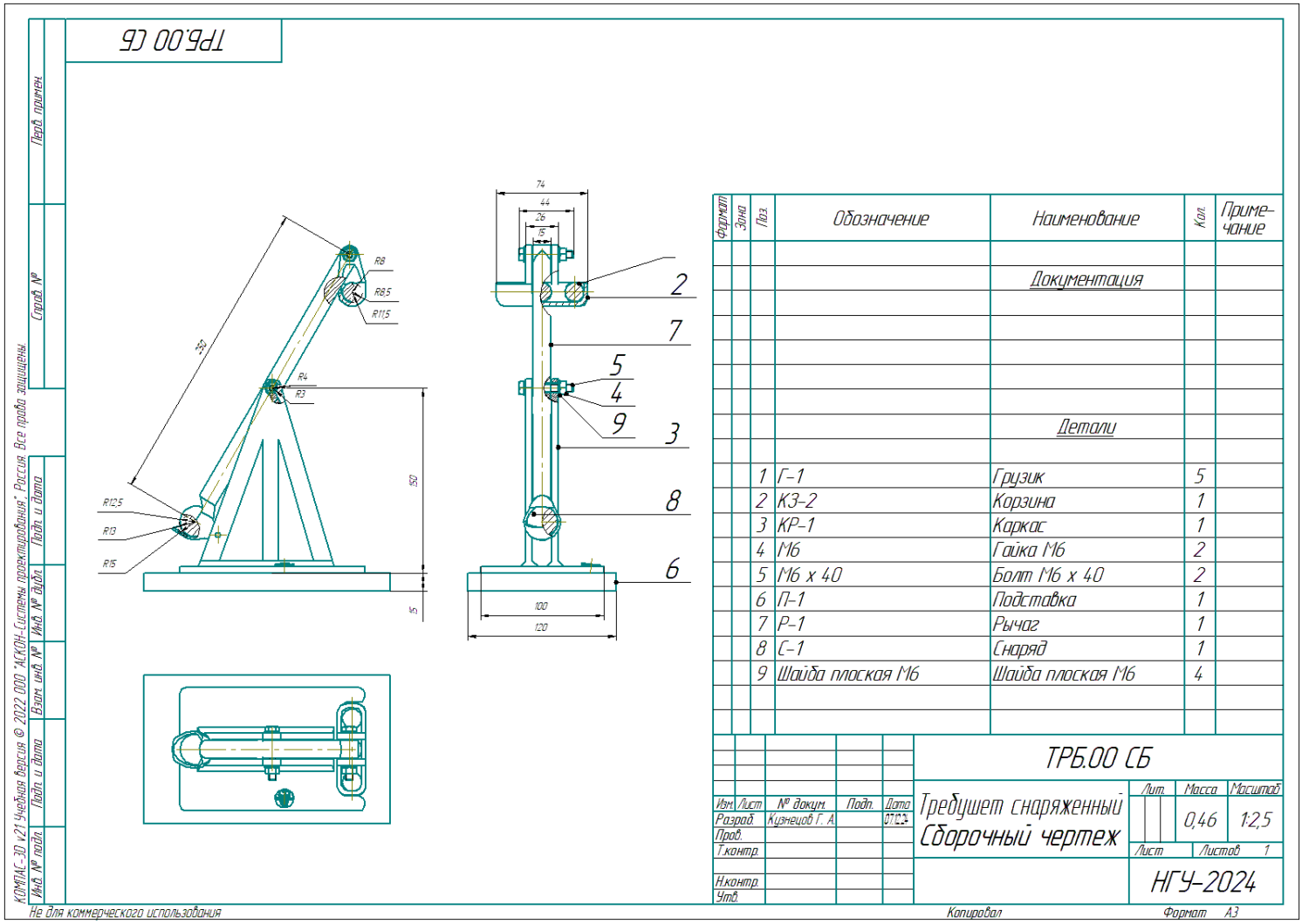

Рисунок 1. Чертёж модели.

3. Динамический анализ механической модели (Расчёт разгона снаряда)

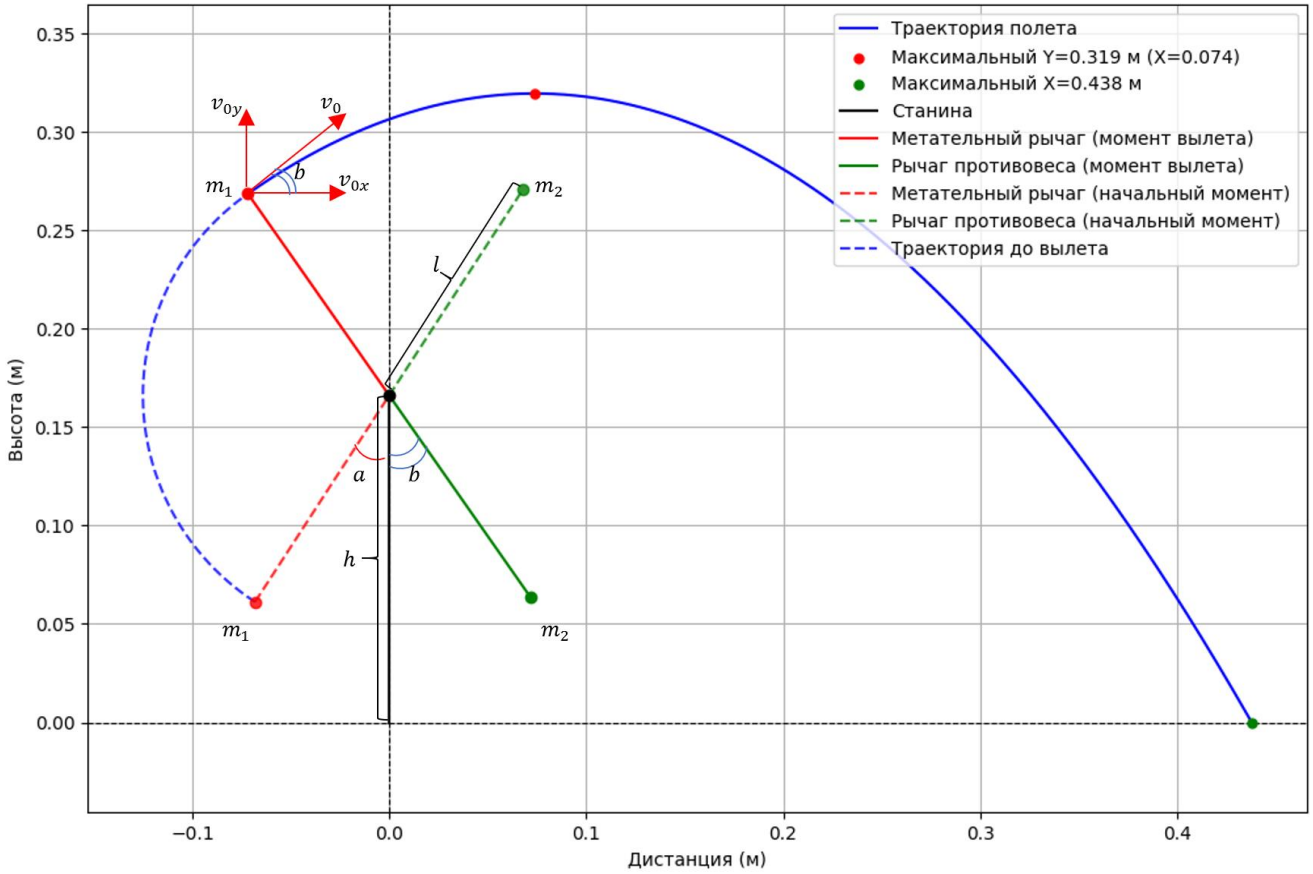

Рисунок 2. Схема работы требушета с параметрами.

Введём переменные, которые потребуются для расчётов, в скобках указаны значения нашего требушета:

m_1 – Масса снаряда (0.0111 кг)

m_2 – Масса противовеса (0.0765 кг)

m_r – Масса рычага (0.019 кг)

l – Расстояние от конца рычага до оси вращения (0.125 см)

g – Ускорение свободного падения (9.81 м/с^2)

a – Стартовый угол (33°)

b – Угол вылета (35°)

b' - Угол вылета, измеренный в ту же сторону что и стартовый угол (145°)

Для нахождения скорости вылета снаряда, запишем уравнение баланса энергий для начального момента и момента вылета

$E^{\text{п}}_{Im_2} = E^{\text{п}}_{IIm_1} + E^{\text{к}}_{IIm_1} + E^{\text{к}}_{IIm_2}$

$h' = l * (\cos(a) + \cos(b))$ – Разница высоты между начальным моментом и моментом вылета

$E^{\text{п}}_{Im_2} = m_2 * g * h'$ - Потенциальная энергия снаряда в начальный момент

$E^{\text{п}}_{IIm_1} = m_1 * g * h'$ - Потенциальная энергия снаряда в момент вылета

$E^{\text{к}}_{IIm_1} = \frac{m_1 * v^2}{2}$ – Кинетическая энергия снаряда в момент вылета

$E^{\text{к}}_{IIm_2} = \frac{m_2 * v^2}{2}$ – Кинетическая энергия противовеса в момент вылета. Учитываем мы её т.к. в момент вылета противовес бьётся об ограничитель. Согласно теореме Карно, эта энергия равна кинетической энергии противовеса в момент удара об ограничитель.

Получаем следующее выражение

$$m_2 * g * h' = m_1 * g * h' + \frac{m_1 * v^2}{2} + \frac{m_2 * v^2}{2}$$

Выразим скорость

$$v^2 = \frac{2 * g * l * h' * (m_2 - m_1)}{m_2 + m_1}$$

Подставим h' и получим итоговое уравнение скорости вылета снаряда

$$v_0 = \sqrt{\frac{2 * g * l * (\cos(a) + \cos(b)) * (m_2 - m_1)}{m_2 + m_1}}$$

Подставим наши значения и получим скорость вылета снаряда

$v_0 = 1.742$ м/с

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

Введём переменные, которые потребуются для расчётов, в скобках указаны значения нашего требушета:

v_0- Скорость снаряда в момент вылета (1.742 м/с)

h - Высота оси вращения (0.15 м)

$shift$ – Смещение от точки пуска по оси X до оси вращения (0.07 м)

Рассчитаем высоту запуска

$h_l = h + l * \cos(b)$

Рассчитаем время полета снаряда до момента пересечения высоты запуска по оси Y

$v_y \rightarrow -v_y$

$t_1 = \frac{v_y - (-v_y)}{g} = \frac{2v_y}{g} = 2v_y * \frac{\sin(b)}{g}$

Найдём оставшееся время полёта снаряда

$h_l = v_y * t + \frac{gt^2}{2}$

$gt^2 + 2v * \sin(b) - 2h_l = 0$

$D = (2 * v * \sin(b))^2 + 8 * g * h_l$

$t_2 = \frac{-2*v*\sin(b)+\sqrt{D}}{2g}$

Общее время полёта

$t = t_1 + t_2$

По X нет ускорения, составим уравнение

$x = v_0 * \cos(b) * t - shift$

Уравнение по Y

$y = v_0 * \sin(b) * t - \frac{1}{2} * gt^2 + h_l$

Подставим скорость вылета снаряда и получим дальность

$x = 0.438$ м

5. Обоснование устойчивости механической модели (Определение центра тяжести)

Для нахождения центра тяжести требушета использовалась программа Компас-3D, каждому элементу конструкции был присвоен вес, в соответствии с реальными значениями.

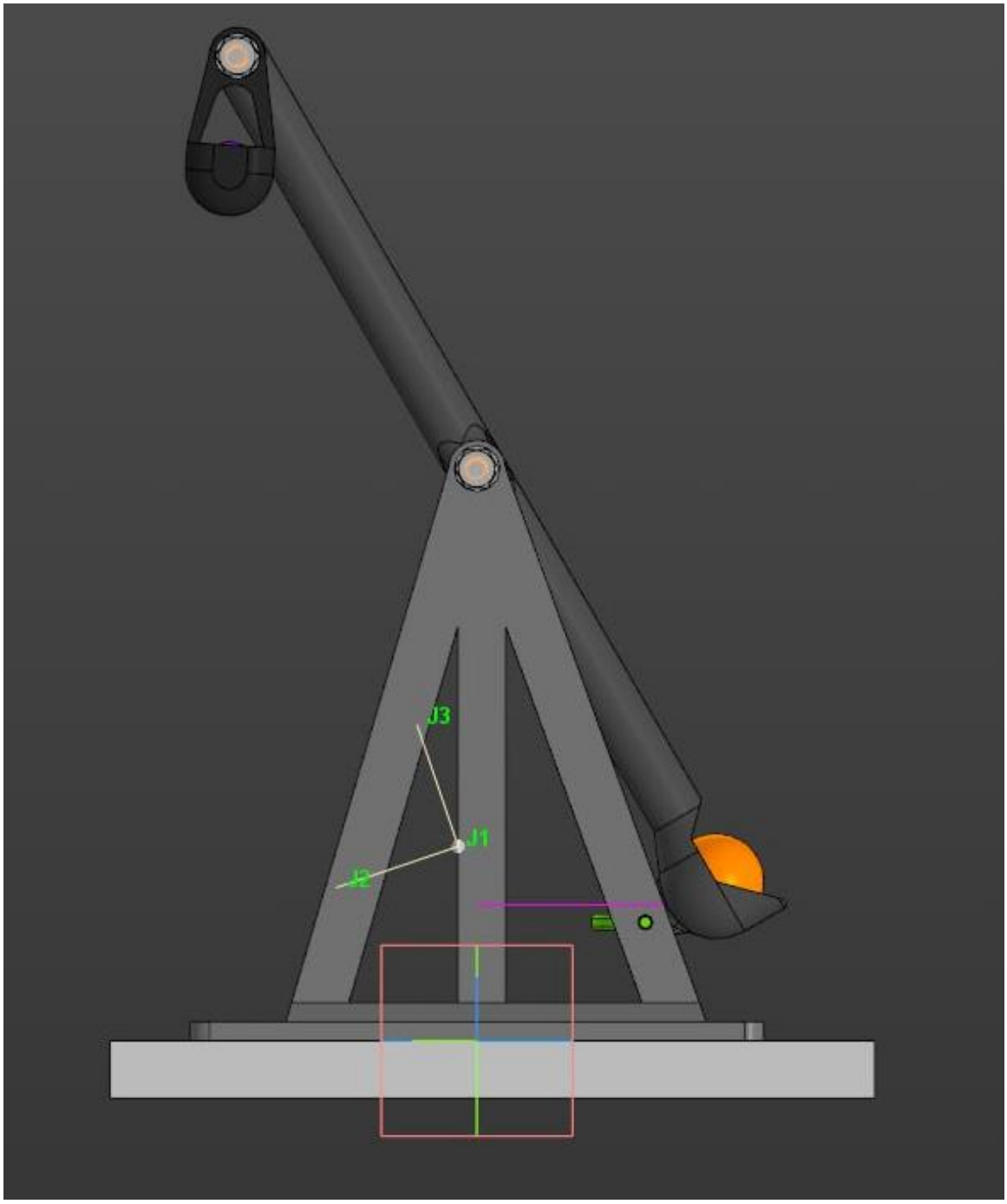

Рисунок 3. Центр тяжести требушета.

Были получены следующие координаты центра массы:

$X = 0.0,\ \ Y = 5.8,\ \ Z = 51.3$

Найдём центр масс экспериментально при помощи метода подвешивания

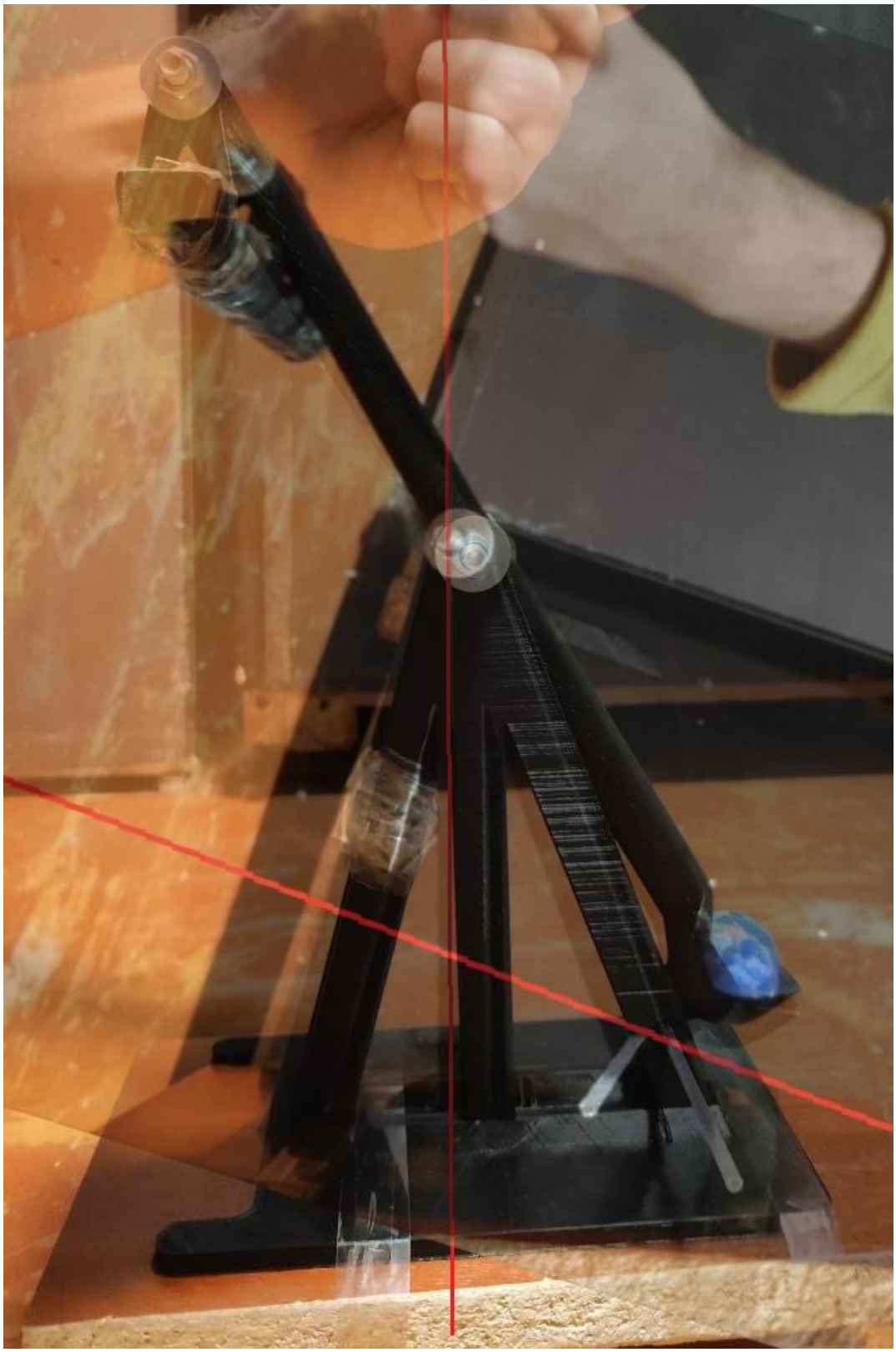

Рисунок 4. Нахождение центра тяжести методом подвешивания

Как можно увидеть, центр масс достаточно точно совпадает с расчётами

6. Сравнение фактических параметров механической модели с расчётными параметрами.

Проведя 3 пуска, были получены следующие результаты: 47.5, 45.0, 48.0 см.

$X_{\text{факт}} = \frac{0.475+0.45+0.48}{3} = 0.468$

Рассчитаем абсолютную и относительную погрешности

$\Delta X_{\text{абс}} = 0.468 - 0.438 = 0.03$

$\delta = \frac{0.468-0.438}{0.468} * 100\% = 6.48\%$

Такая разница в фактические и расчётные значения может быть вызвана тем, что мы учитываем удар плеча об ограничитель как абсолютно неупругий удар, что может быть не совсем верно.

7. Описание электронной модели механической системы

Программа запускается при помощи python, на вход её подаются все значения, нужные для расчёта. Если никакие значения не будут переданы, будут взяты значения нашего требушета.

Запуск: py calculation.py

Передаваемые параметры:

--m1 - Масса снаряда.

--m2 – Масса противовеса

--h – Высота оси вращения

--l – Длина плеча (от оси до края рычага).

--alpha – Начальный угол

--beta – Угол вылета

--is_shift - Учитывать ли смещение точки вылета по X относительно оси вращения.

--visualize - Выводить ли визуализацию.

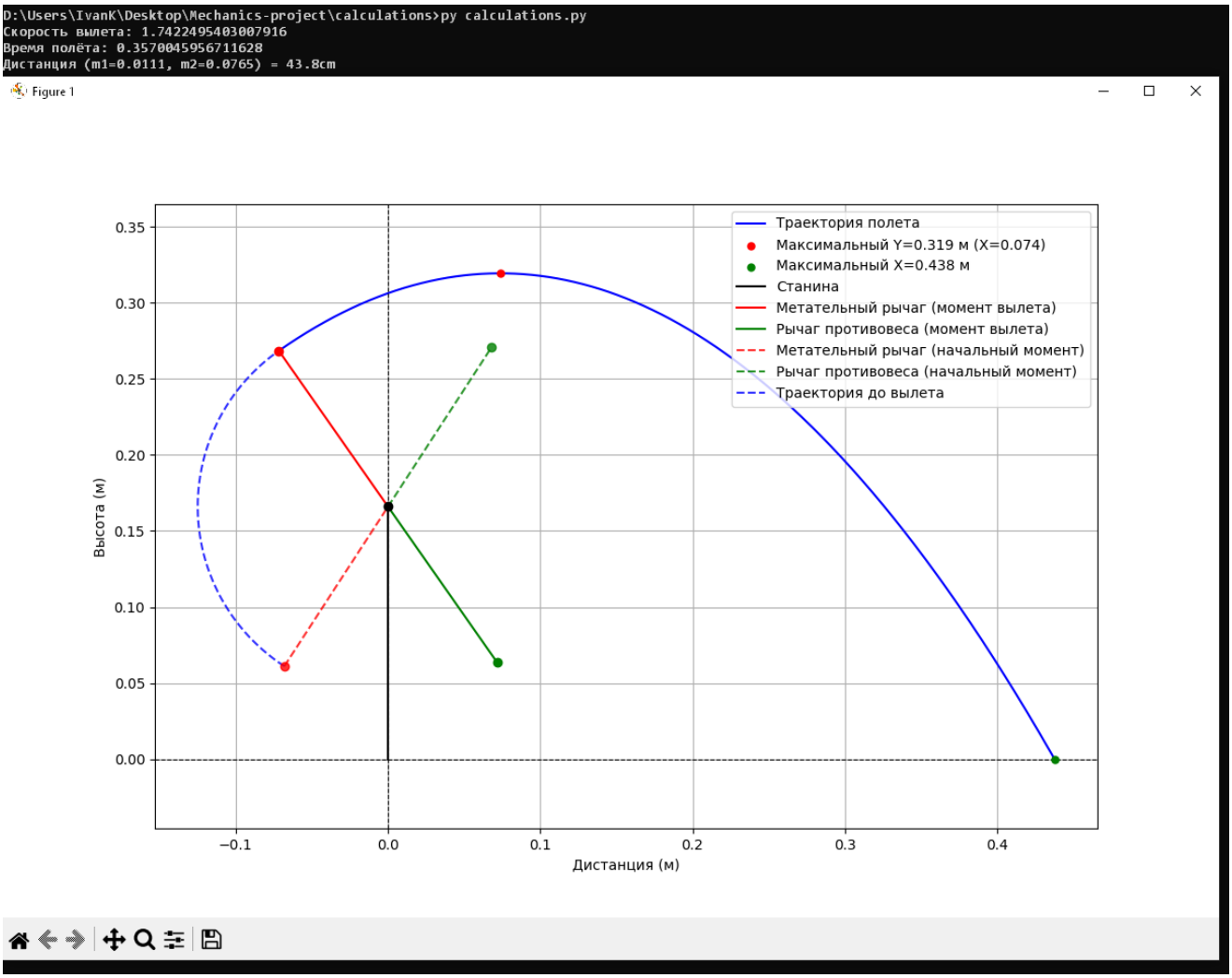

Рисунок 5. Пример работы программы.

Листинг кода:

```
import math
import numpy as np
import matplotlib.pyplot as plt
import argparse

G = 9.81

def calculate_r(m1, m2, ma, h, l, a1, a2, is_shift=False, visualize=False):
    '''
    m1 - mass of projectile \n
    m2 - mass of counterweight \n
    ma - mass of lever \n
    h - height of mount \n
    l - length of lever \n
    a1 - start angle \n
    a2 - finish angle (Measured in the opposite direction from the starting point) \n
    shift - difference between middle of trebushet and position of throw \n
    visualise - visualise trajectory \n
    '''

    a1r = np.radians(a1)
    a2r = np.radians(a2)

#-----------------------------------------------------------------------------------------------
    speed = np.sqrt((2*G*l*(np.cos(a1r) + np.cos(a2r))*(m2-m1))/(m1+m2))
#-----------------------------------------------------------------------------------------------

    print('Скорость вылета:', speed)

    t1 = (2*speed*np.sin(a2r))/G
    x1 = t1*speed*np.cos(a2r)

    heigth_of_launch = h + l*np.cos(a2r)
    D = (2*speed*np.sin(a2r))**2 + 8*G*(heigth_of_launch)
    t2 = (-2*speed*np.sin(a2r) + np.sqrt(D))/(2*G)
    x2 = speed*np.cos(a2r)*t2

    print('Время полёта:', t1+t2)

    length = x1+x2
    if is_shift:
        shift = l * np.sin(a2r)
        length -= shift
    else:
        shift = 0
    print(f'Дистанция (m1={m1}, m2={m2}) = {length*100:.1f}cm')

    if visualize:
        t1_arr = np.linspace(0, t1+t2, num=10000, endpoint=False)
        x1_arr = speed*np.cos(a2r)*t1_arr - shift
        y1_arr = speed*np.sin(a2r)*t1_arr - 1/2*G*(t1_arr**2) + heigth_of_launch
        plt.figure(figsize=(12,8))
        plt.plot(x1_arr, y1_arr, color='blue', label='Траектория полета')
         plt.scatter(x1_arr[np.where(y1_arr == np.max(y1_arr))][0], np.max(y1_arr), color='red',
s=25, zorder=10, label=f'Максимальный Y={np.max(y1_arr):.3f} м (X={x1_arr[np.where(y1_arr ==
np.max(y1_arr))][0]:.3f})')
       plt.scatter(x1_arr[-1], 0, color='green', s=25, zorder=10, label=f'Максимальный X={x1_arr[-
1]:.3f} м')

        plt.plot([0, 0], [0, h], label='Станина', color="black")

          plt.plot([-l*np.sin(a2r), 0], [h+l*np.cos(a2r), h], label='Метательный рычаг (момент
вылета)', color='red')
          plt.plot([0, l*np.sin(a2r)], [h, h-l*np.cos(a2r)], label='Рычаг противовеса (момент
вылета)', color='green')

         plt.plot([-l*np.sin(a1r), 0], [h-l*np.cos(a1r), h], label='Метательный рычаг (начальный
момент)', color='red', linestyle='--', alpha=0.8)
          plt.plot([0, l*np.sin(a1r)], [h, h+l*np.cos(a1r)], label='Рычаг противовеса (начальный
момент)', color='green', linestyle='--', alpha=0.8)

        plt.plot(0, h, marker='o', color='black')
        plt.plot(-l*np.sin(a2r), h+l*np.cos(a2r), marker='o', zorder=10, color='red')
        plt.plot(l*np.sin(a2r), h-l*np.cos(a2r), marker='o', zorder=10, color='green')
        plt.plot(-l*np.sin(a1r), h-l*np.cos(a1r), marker='o', zorder=10, color='red', alpha=0.8)
        plt.plot(l*np.sin(a1r), h+l*np.cos(a1r), marker='o', zorder=10, color='green', alpha=0.8)

        pre_release_t = np.linspace(np.radians(270-a1), np.radians(90+a2), num=1000)
```

```
        pre_release_x = np.cos(pre_release_t) * l
        pre_release_y = h + np.sin(pre_release_t) * l
           plt.plot(pre_release_x,  pre_release_y,  label='Траектория  до  вылета',  color='blue',
linestyle='--', alpha=0.8)

        plt.axhline(0, color='black', linestyle='--', linewidth=0.8)
        plt.axvline(0, color='black', linestyle='--', linewidth=0.8)
        plt.xlabel('Дистанция (м)')
        plt.ylabel('Высота (м)')
        plt.legend(loc='upper right')
        plt.grid()
        plt.axis('equal')
        plt.show()

    return length

def main():
    ball = 0.0168
    jump_ball = 0.0111
    fixing = 0.009 + 0.0025 + 0.0025 * 2 #= 0.0165
    backet = [0.010, 0.027, 0.043, 0.060, 0.077]
    lever_mass = 0.019
    frame_mass = 0.066

    # Длины в метрах
    lever_length = 0.125
    frame_heigth = 0.151
    bruss_heigth = 0.015

    # Углы в градусах
    alpha = 33
    beta = 35

    parser = argparse.ArgumentParser()
    parser.add_argument("--m1", default=jump_ball,
                        help="Масса снаряда.", type=float)

    parser.add_argument("--m2", default=backet[3]+fixing,
                        help="Масса противовеса.", type=float)

    parser.add_argument("--ma", default=lever_mass,
                        help="Масса рычага.", type=float)

    parser.add_argument("--h", default=frame_heigth + bruss_heigth,
                        help="Высота оси вращения.", type=float)

    parser.add_argument("--l", default=lever_length,
                        help="Длина плеча (от оси до края рычага).", type=float)

    parser.add_argument("--alpha", default=alpha,
                        help="Начальный угол.", type=float)

    parser.add_argument("--beta", default=beta,
                        help="Угол вылета.", type=float)

    parser.add_argument("--is_shift", default=True,
                        help="Учитывать ли смещение точки вылета по X относительно оси вращения.",
type=bool)

    parser.add_argument("--visualize", default=True,
                        help="Выводить ли визуализацию.", type=bool)

    m1 = parser.parse_args().m1
    m2 = parser.parse_args().m2
    ma = parser.parse_args().ma
    h = parser.parse_args().h
    l = parser.parse_args().l
    a1 = parser.parse_args().alpha
    a2 = parser.parse_args().beta
    is_shift = parser.parse_args().is_shift
    visualize = parser.parse_args().visualize

    calculate_r(m1, m2, ma, h, l, a1, a2, is_shift, visualize)

if __name__ == "__main__":
           main()
```