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.**

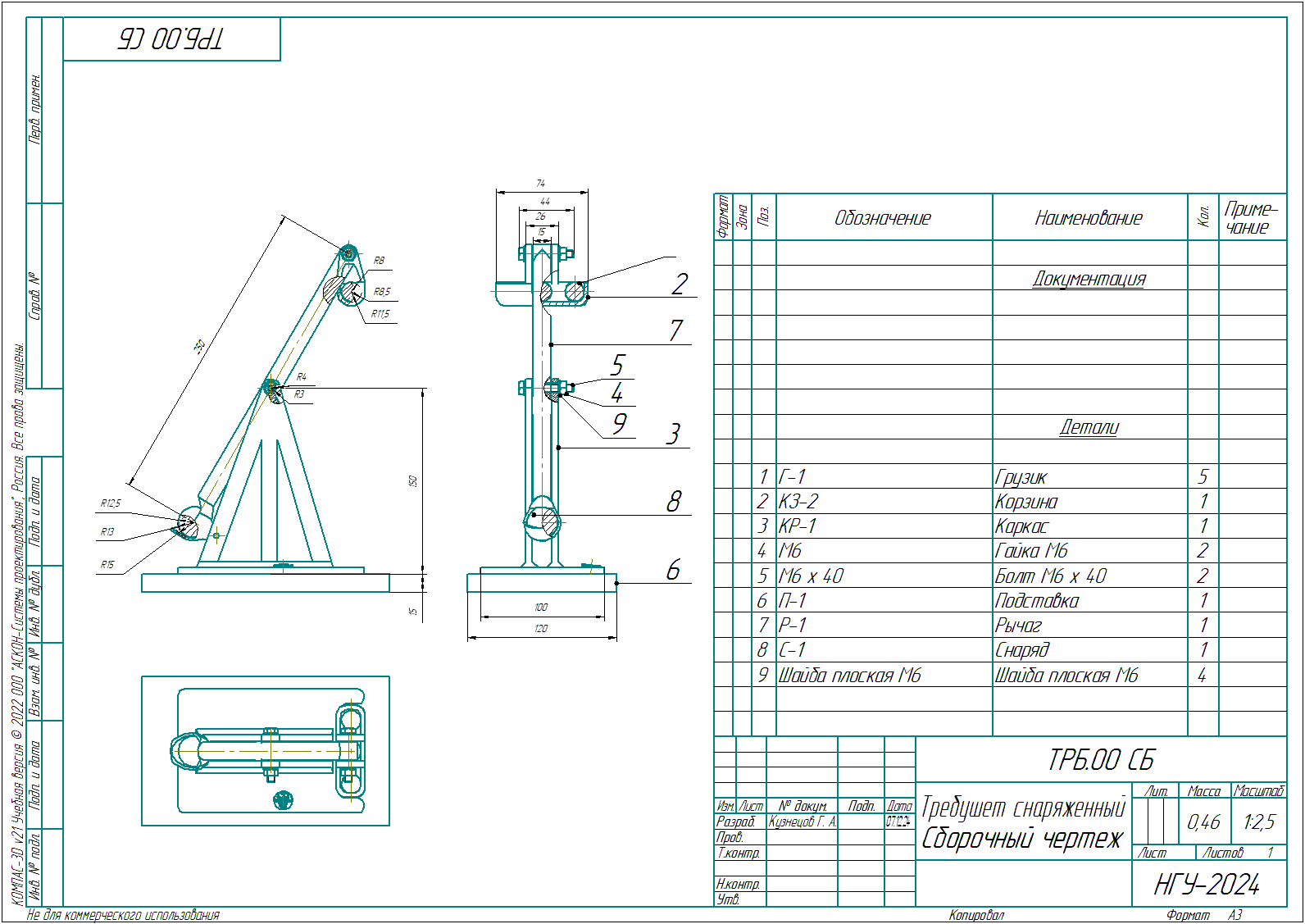
1. Метание деформируемого снаряда любой выбранной массы и формы на расстояние от 30 см до 60 см и отклонением от центральной оси не более 30°.

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

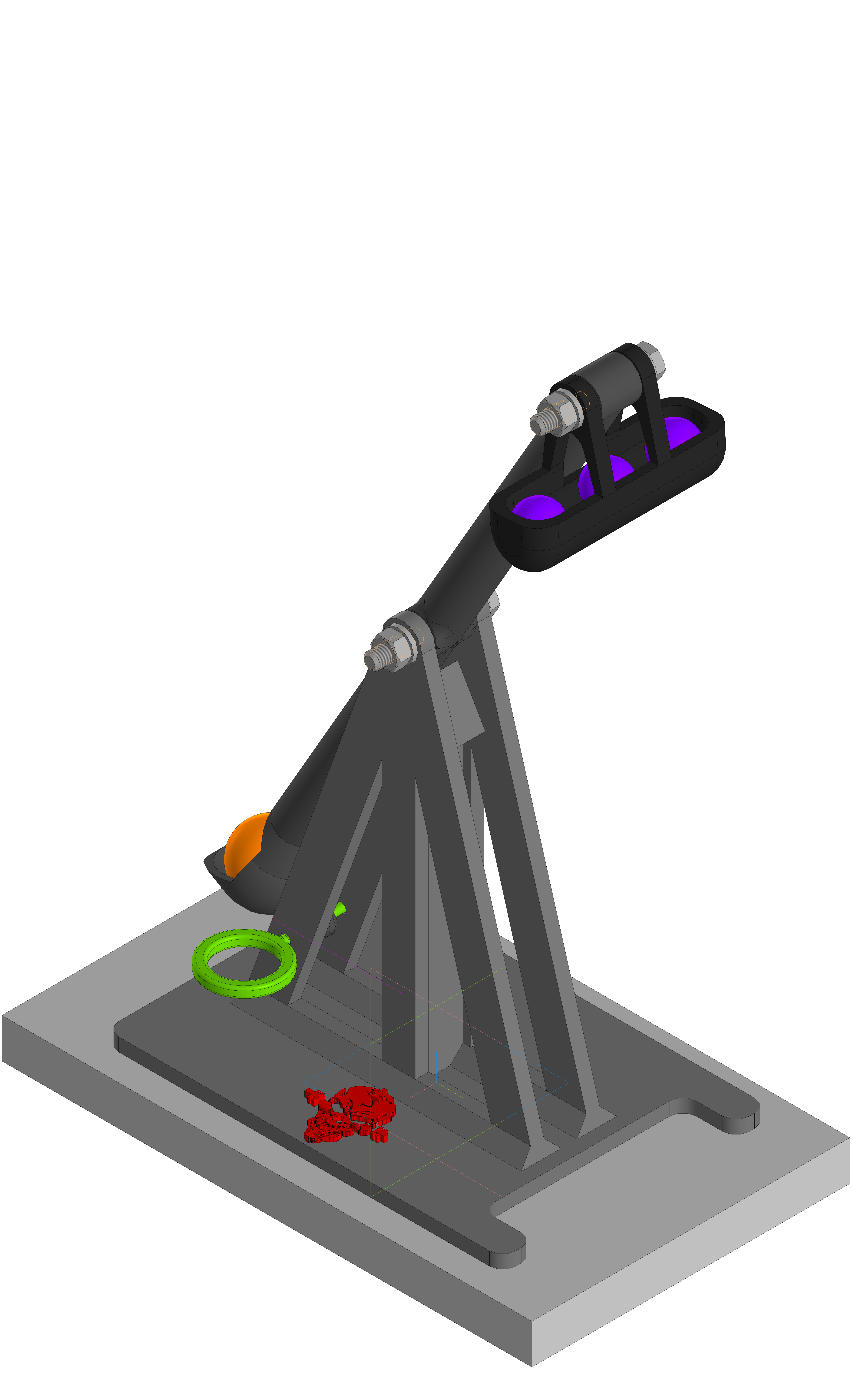
3. Механический спуск.

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

**2. Эскиз модели.**

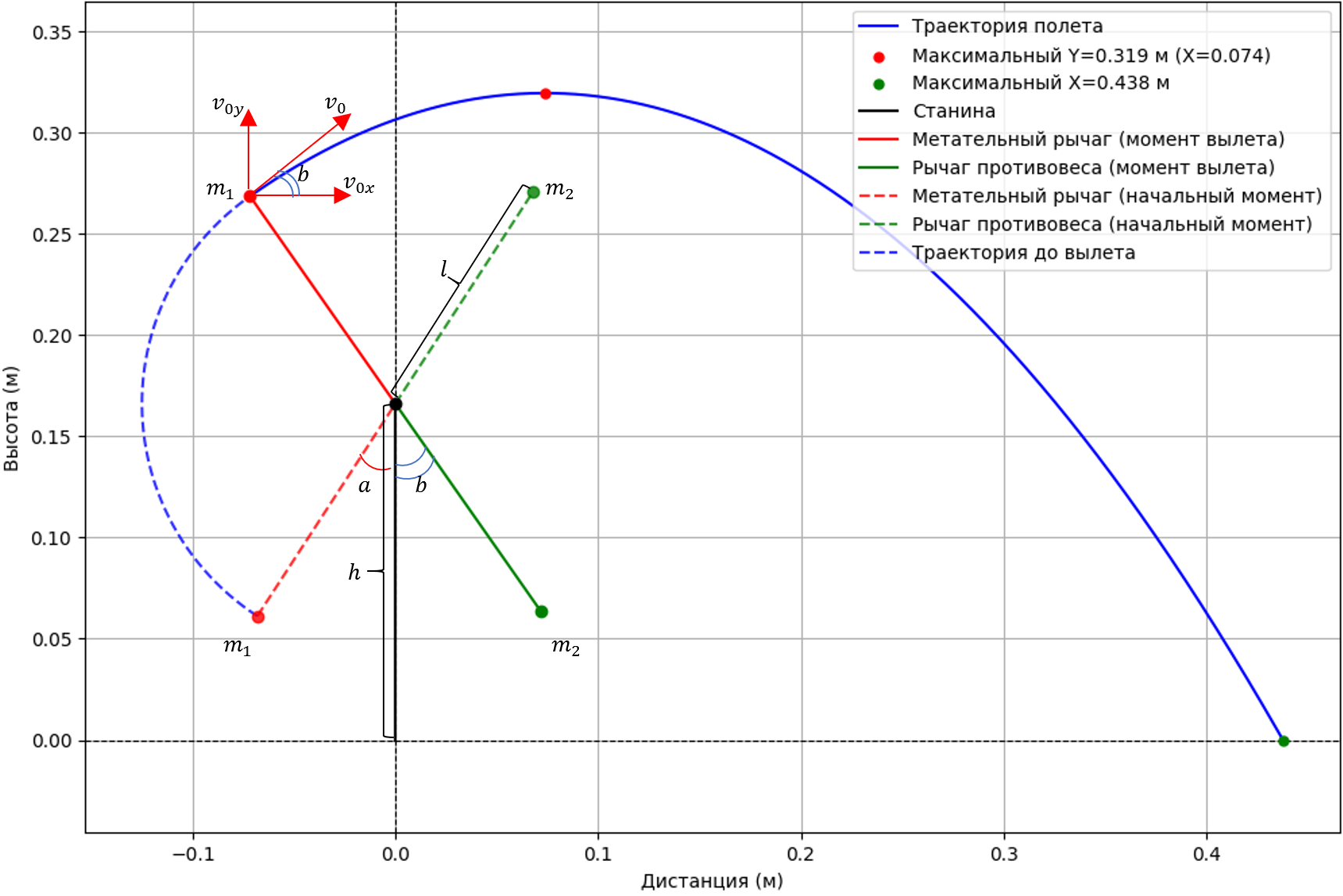


*Рисунок 1. Чертёж модели. Размеры в мм*



*Рисунок 2. Модель требушета в изометрии.*

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



*Рисунок 3. Схема работы требушета с параметрами.*

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

– Масса снаряда (0.0111 кг)

– Масса противовеса (0.0765 кг)

– Масса рычага (0.019 кг)

– Расстояние от конца рычага до оси вращения (0.125 м)

– Ускорение свободного падения (9.81 )

– Стартовый угол (33**°**)

– Угол вылета (35**°)**

- 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

– Разница высоты между начальным моментом и моментом вылета

*-* Потенциальная энергия снаряда в начальный момент

- Потенциальная энергия снаряда в момент вылета

– Кинетическая энергия снаряда в момент вылета

– 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

Выразим скорость

Подставим и получим итоговое уравнение скорости вылета снаряда

Подставим наши значения и получим скорость вылета снаряда

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

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

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

- Высота оси вращения (0.15 м)

– Смещение от точки пуска по оси Х до оси вращения (0.07 м)

Рассчитаем высоту запуска

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

Найдём оставшееся время полёта снаряда

Общее время полёта

По X нет ускорения, составим уравнение

Уравнение по Y

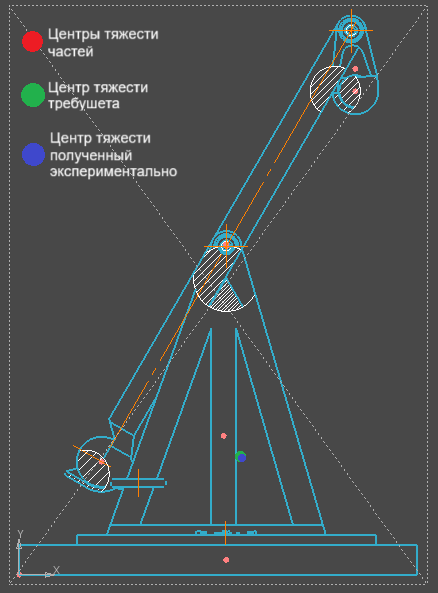
Подставим скорость вылета снаряда и получим дальность

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

Проведём теоретический расчёт центра масс требушета. Учитывая симметрию требушета относительно вертикальной плоскости, достаточной найти координаты центра тяжести в проекции на данную плоскость. Формула для нахождения центра масс в общем виде для k элементов:

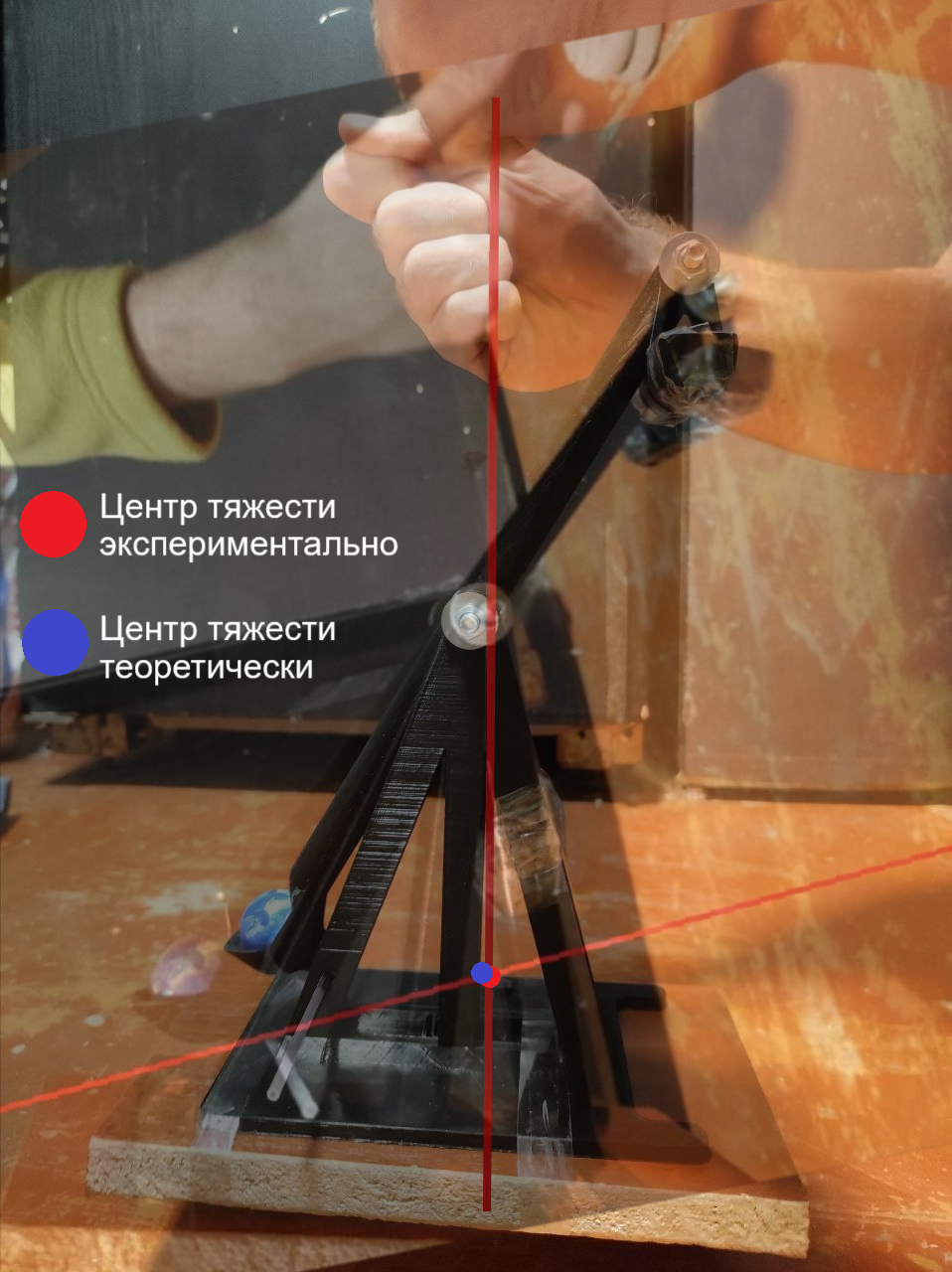
Подставим наши значения (второе слагаем умножено на 3, т.к. мы используем 3 противовеса)

Вычислим значения центра масс



*Рисунок 4. Теоретически рассчитанный центр масс требушета*

Рассчитаем центр масс экспериментально методом подвешивания.



*Рисунок 5. Экспериментально рассчитанный центр тяжести требушета.*

Какможно увидеть мы получили достаточно точное совпадение центров тяжести*.*

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

Проведя 3 пуска, были получены следующие результаты: 0.475 м, 0.45 м, 0.48 м.

Рассчитаем абсолютную и относительную погрешности

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

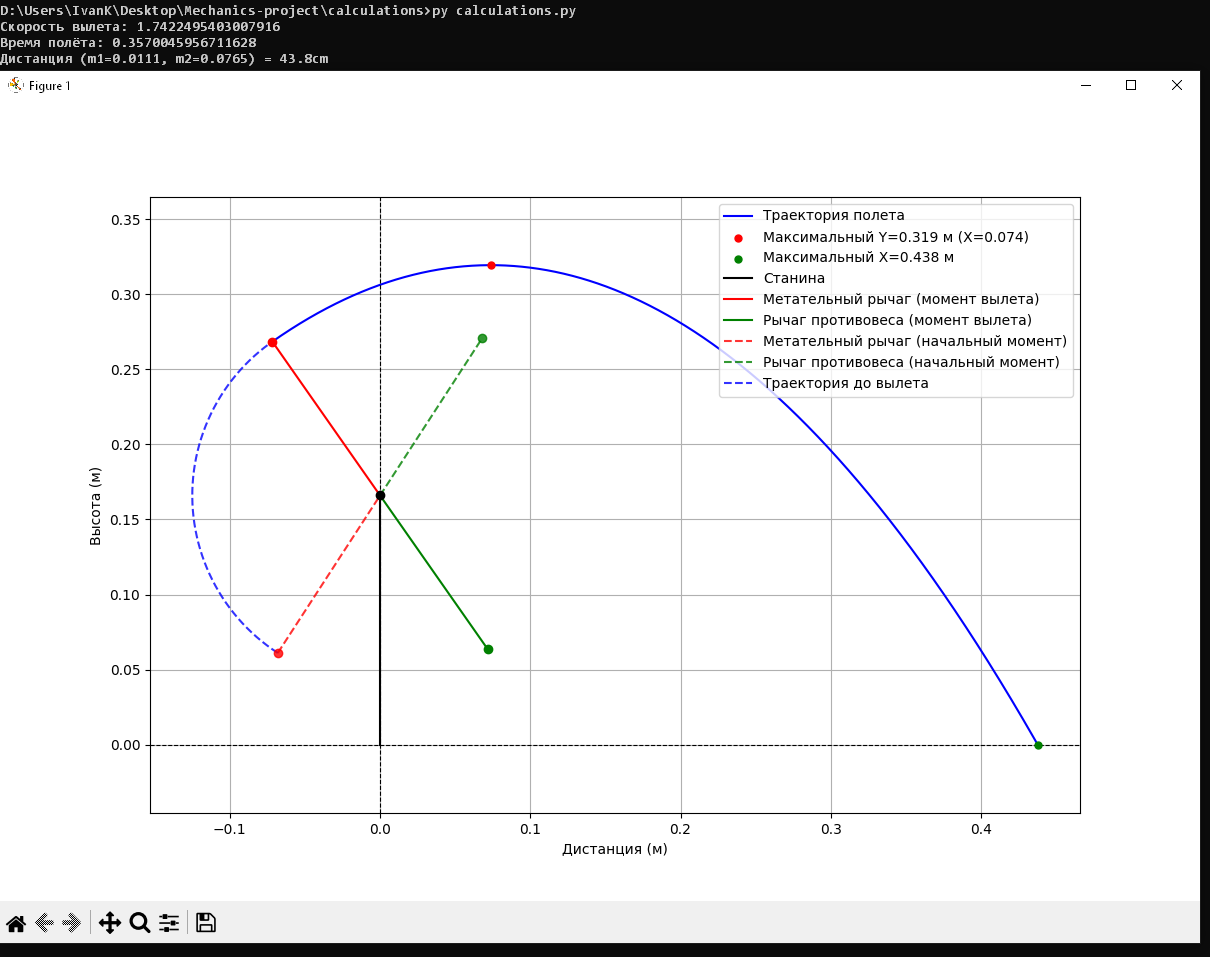
--l – Длина плеча (от оси до края рычага).

--alpha – Начальный угол

--beta – Угол вылета

--is\_shift - Учитывать ли смещение точки вылета по X относительно оси вращения.

--visualize - Выводить ли визуализацию.



*Рисунок 6. 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()