# Отчет по задаче «Мюнхгаузен»

**Задача:** Программно промоделировать и изучить стрельбу из двух направленных друг на друга *пороховых* пушек.

**Условие:** В фазовом пространстве с евклидовой метрикой и декартовой системой координат R2 существует два орудия стреляющие в сторону друг друга ядрами. Между пушками существует расстояние, состоящие из двух частей: ***разницы абсцисс и ординат.*** Состояние *ядер* описывается тремя векторами: ***вектором скорости, вектором ускорения и радиус вектором.*** Требуется промоделировать и составить графики с заданной ***частотой дискретизации*** реагирования одной из пушек (стороны защиты) на стрельбу другой (стороны атаки), с учетом ***задержки*** при получении информации о запуске ядра со стороны атаки и ***ограничениями на скорость*** ядра.

**Ссылка на Github с реализацией возможного решения:**

* + 1. **Решение:** Задача была разделена да два возможных запроса:
    - Протестировать систему при заданных значениях.
    - Найти начальные координаты вектора скорости защиты, при которых будет сбито ядро атаки.

Поставленная задача требовала выполнения следующий пунктов:

* + - Создание библиотеки с реализацией работы с векторами в пространстве R2.
    - Создание алгоритма вычисления радиус векторов ядер в каждый момент времени с заданной частотой дискретизации.
    - Создание графиков с помощью полученных данных о радиус векторах ядер.

## Библиотека.

Для решения первого вопроса в предоставленном мной решении была реализована библиотека ***geom,*** возможности которой позволяют решить поставленную задачу благодаря представлении векторов в декартовой системе координат. (Собрано с помощью системы сборки CMake 3.23)

**Ссылка на заголовочный файл библиотеки:**

## Вычислительный алгоритм.

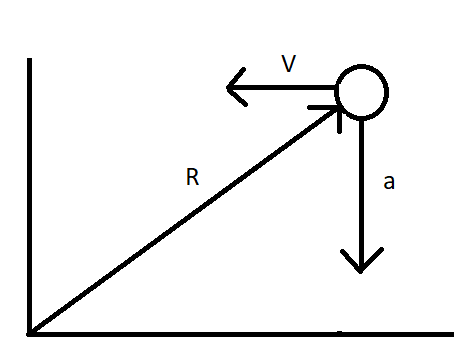
Алгоритм вычисления координат радиус векторов ядер на временном отрезке следующий:

* + - Программа считывает ***конфигурационный файл*** (подробнее в п. «Графическое представление»)
    - В зависимости от типа запроса, запускается либо поиск решения, либо тестирование заданных значений.
    - Результатом работы программы становятся CSV таблицы с координатами радиус векторов двух ядер.

Вычислительная часть представляет из себя вычисление следующих параметров каждого ядра до «падения» одного из ядер (*абсцисса одного из радиус векторов отрицательна*) или попадания ядер друг в друга (*радиус векторы ядер равны*):

R = R + V · dt

V = V + a · dt

**Ссылка на** **основной cpp:**

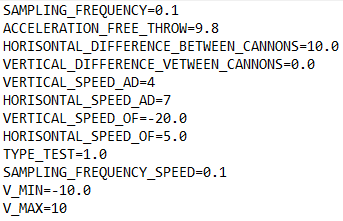
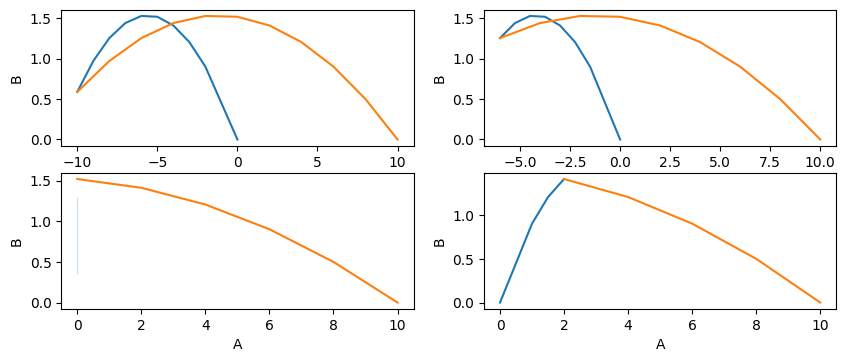
## Графическое представление.

Для взаимодействия с программой была выбрана среда разработки **Jupyter.** Благодаря ей, а так же **python** и библиотекам **seaborn** и **pandas** удалось реализовать работу с конфигурационным файлом, а так же построения графиков, демонстрирующих траекторию полёта ядер, в зависимости от типа запроса. Реализация описана в файле graph.ipynb.

## Результат исследования.

Протестировав программу в различных конфигурациях удалось получить следующие результаты. (защита, атака)

### Поиск решения (конфигурация + график):

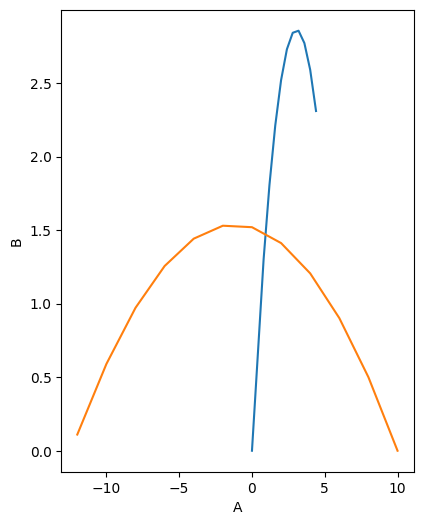
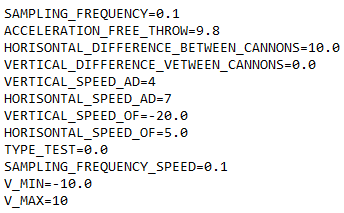
  
Координаты вектора скорости 1 решения: -10, 5

Координаты вектора скорости 2 решения: -7.5, 5

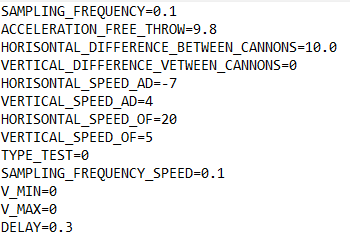
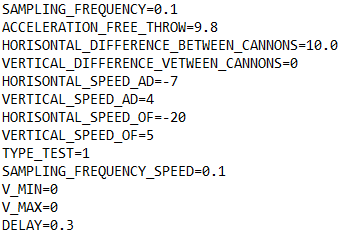
Координаты вектора скорости 3 решения: 0, 5

Координаты вектора скорости 4 решения: 5, 5

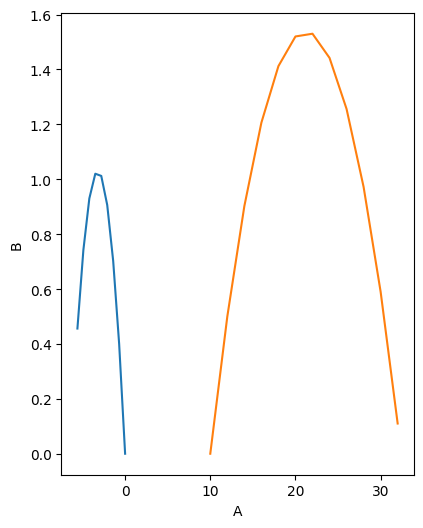
### Тестирование заданных значений (конфигурация + график):



## Поиск решения, непопадание (полёт в противоположные стороны, нулевая скорость). (конфигурация + график):



### Тестирование заданных значений (конфигурация + график):



## Вывод.

Построенная модель ищет координаты вектора скорости стороны защиты и тестирует заданные значения. Полученные в результаты можно применить в решении задач иного рода, например в моделировании поведения ПВО и последующего выбора самой безопасной траектории или нахождения начальный векторов скорости системы из n – ядер, которые не будут сталкиваться. Благодаря программе можно проводить дальнейшие, более детальные исследования.