



Выполнил студент

442 группы

Юдинцев Егор Викторович

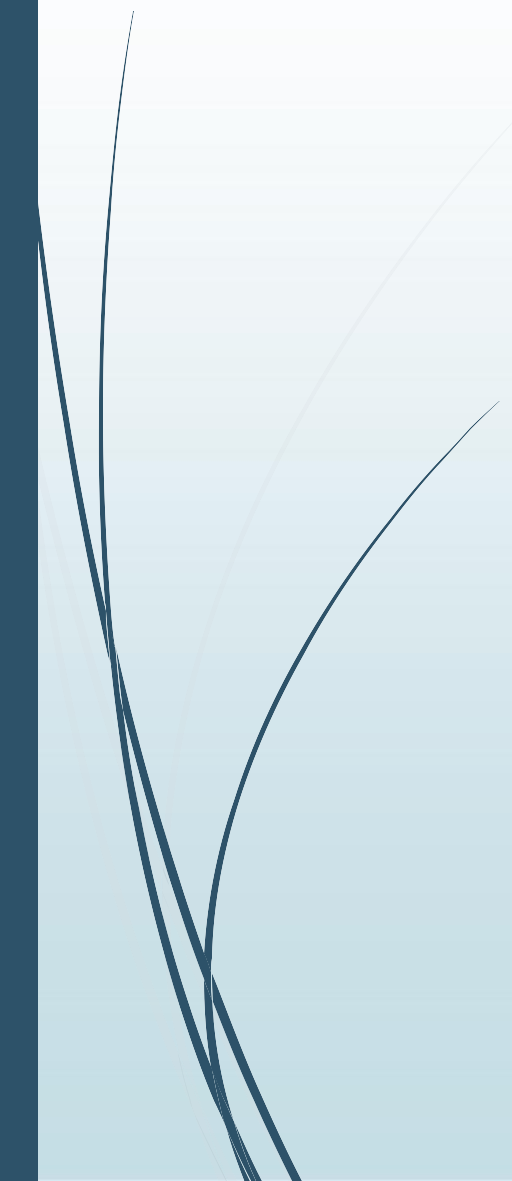
Научный руководитель:

Галяев Андрей Алексеевич

Реализация нейросетевого алгоритма поиск пути в лабиринте

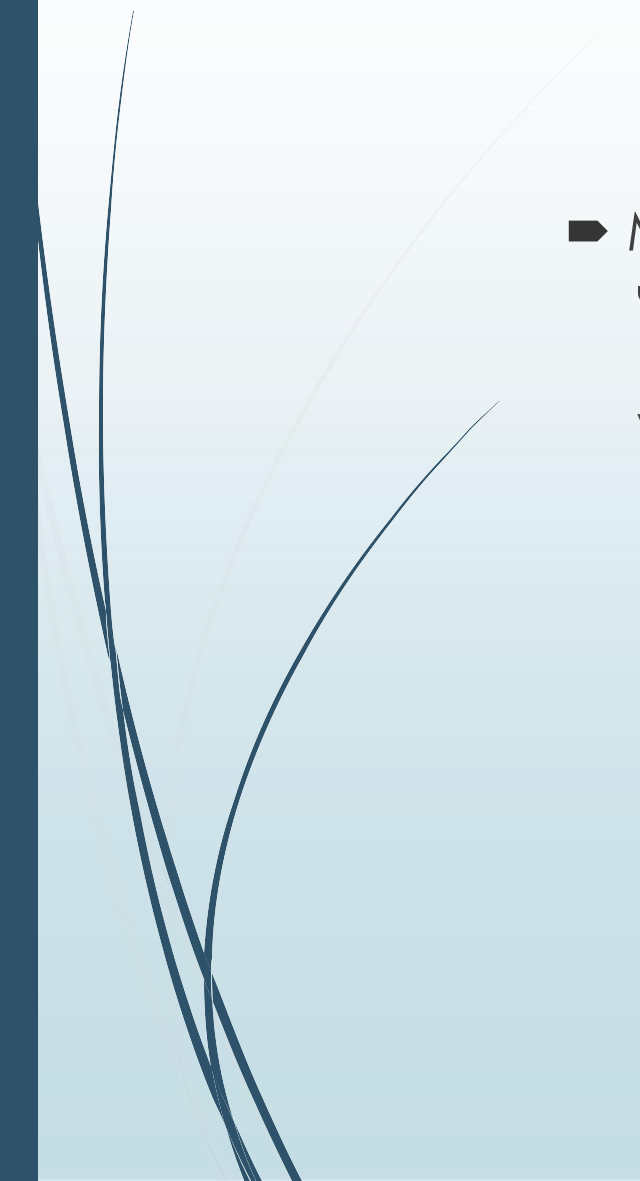


План

- 
1. Теоретическое введение
 2. Постановка задачи
 3. Программная реализация
 4. Заключение



Теоретическое введение

- Машинное обучение - это область компьютерных наук, которая часто использует статистические методы, чтобы дать компьютерам возможность "учиться" (то есть постепенно улучшать производительность в конкретной задаче) с данными, не будучи заранее явно запрограммированной.
- 



Основные концепции машинного обучения

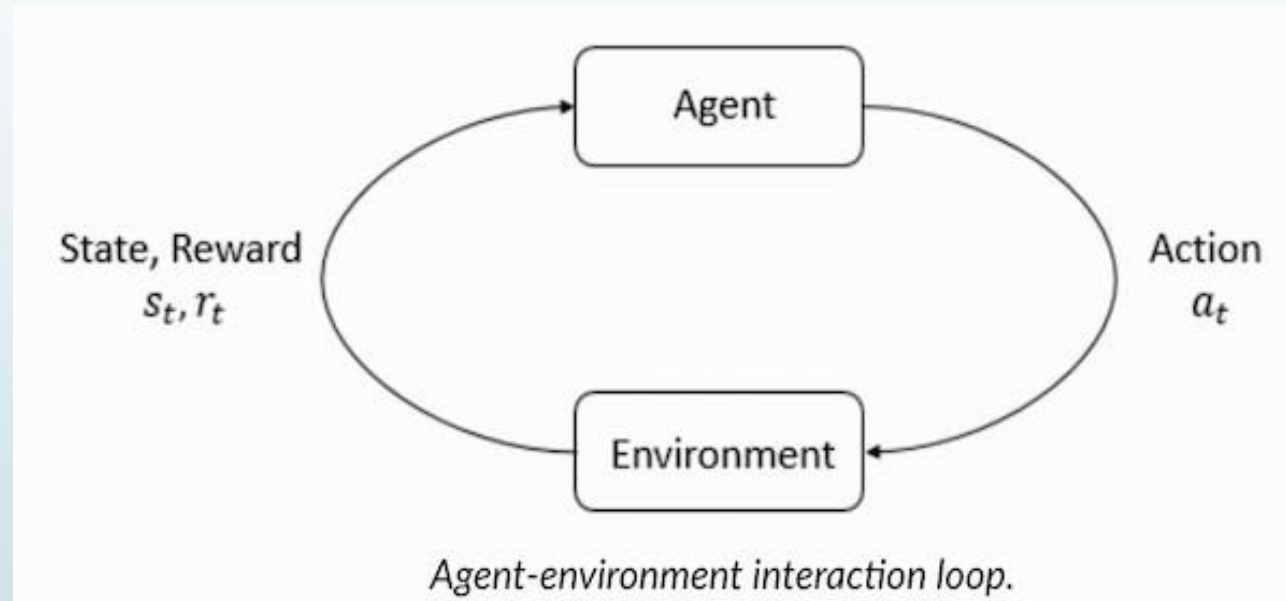
Обучение с учителем

- Агент обучается производить определённые действия на основании предварительно подготовленных выборок.

Обучение без учителя

- Агент самостоятельно формирует стратегию поведения, опираясь на изменения, производимые его действиями.

Обучение без учителя: обучение с подкреплением



Q-learning

- Watkins, 1989,
- Q – качество (англ. quality).

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Q - функция от состояния и действия.

R - вознаграждение.

Альфа характеризует темп обучения, гамма - дисконтирующий множитель.

Постановка задачи



Задача:
"Доставить объект из пункта А в пункт Б
с минимальными затратами топлива."

- Агент - дрон (БПЛА).
- Среда - окружающее пространство.

- Доступные действия:
 1. Move Up (двигаться вверх),
 2. Move Down (двигаться вниз),
 3. Move south (двигаться на юг),
 4. Move north (двигаться на север),
 5. Move west (двигаться на запад),
 6. Move east (двигаться на восток),
 7. Pickup (подобрать объект),
 8. Dropoff (сбросить объект).



Программная реализация

1. Создание среды.
2. Основной алгоритм: Q-learning.
3. Исследование алгоритма.

Используемые инструменты:

1. Python,
2. gym от OpenAI,
3. Вспомогательные библиотеки.



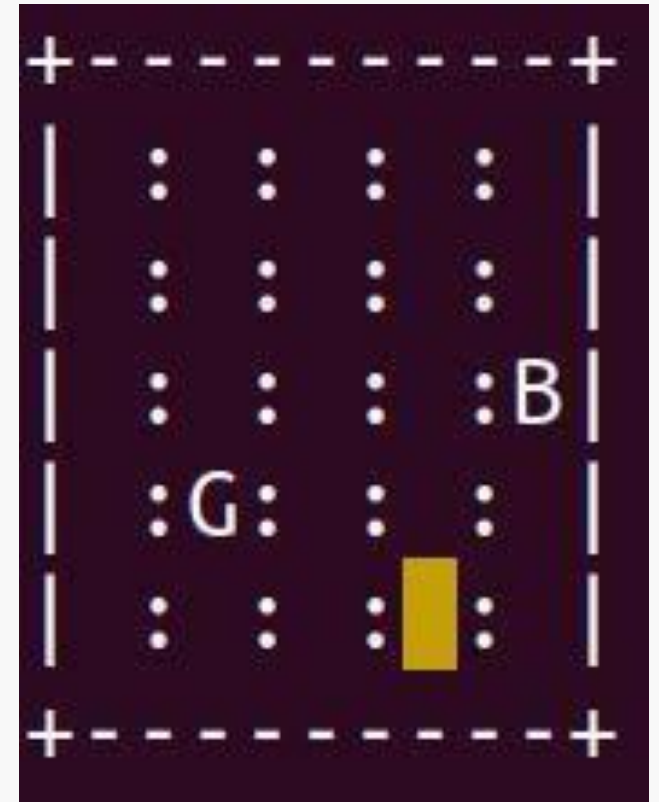
Создание среды

■ Для создания среды:

1. `labyrinth.py` - основная часть среды реализована здесь.
2. `map_generation.py` - вспомогательный файл; используется для построения символьного поля среды.

map_generation.py

- Используемые символы: +, |, :, -.
- Пункты назначения: G(reen), Y(ellow), R(ed), B(lue).
- Цель: визуализация передвижения агента в среде.



labyrinth.py


► `reward = lay_reward(lay),`

где `lay_reward` - это структура данных "ключ-значение", в которой ключ - номер слоя, а значение - вознаграждение на этом слое.

Слой	Вознаграждение
0	$-(n+1)/2$
n-1	-1

► `reward = cell_reward[lay][row][column],`

где `cell_reward` - трехмерный массив, а `lay`, `row`, `column` - индексы этого массива.


$$N = size_x \cdot size_y \cdot size_z \cdot 5 \cdot 4$$

► Шестивложенный цикл:

1. 3 пространственных параметра (lay, row, column),
2. 2 по состояниям объекта и пунктов назначения,
3. 1 по возможным действиям.

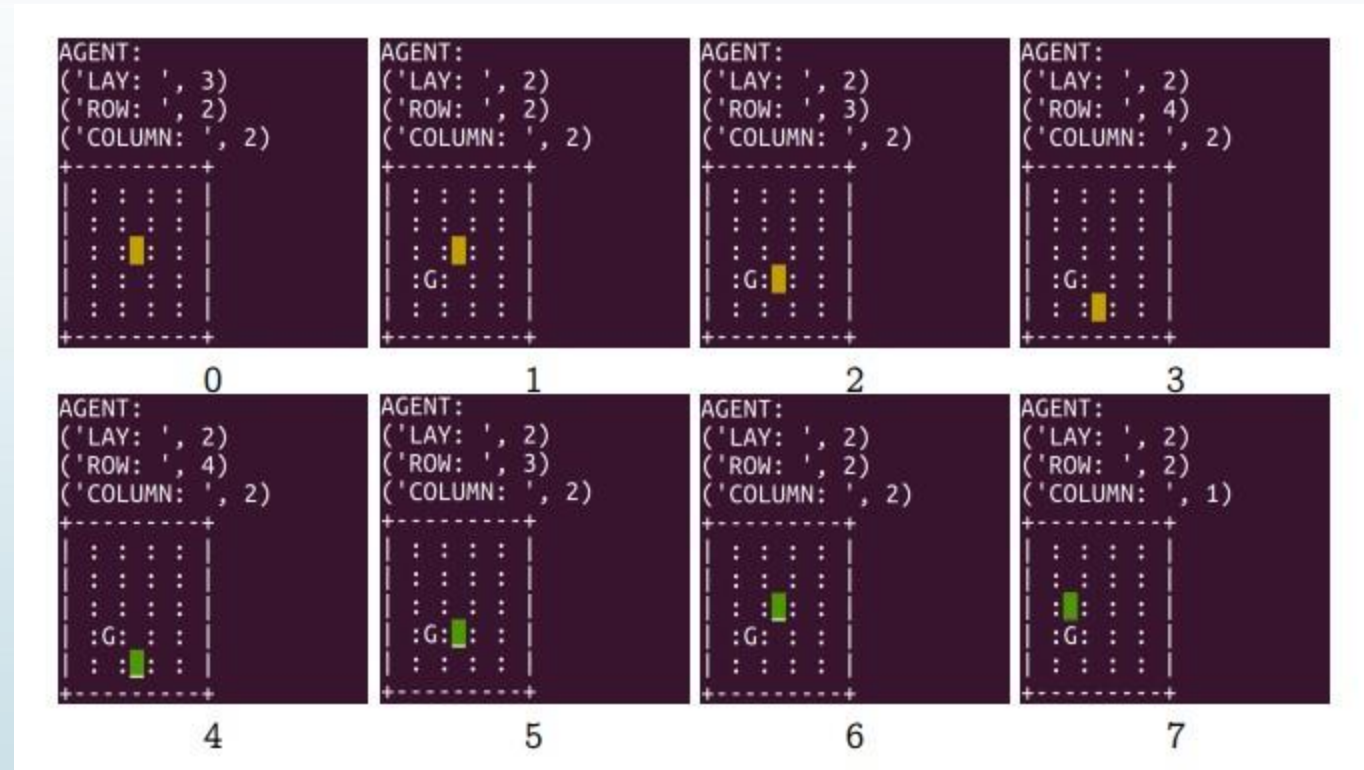
Внутри шестивложенного цикла заполняется первичная матрица вознаграждений P .

P[442]?

- (action, [(probability, nextstate, reward, done)]).
- Значения 0-7 - возможные действия агента.
- done - характеризует, удалось ли выполнить поставленную задачу.

```
(0, [(1.0, 442, -10, False)])  
(1, [(1.0, 342, -5.0, False)])  
(2, [(1.0, 442, -5.0, False)])  
(3, [(1.0, 442, -5.0, False)])  
(4, [(1.0, 942, -5.0, False)])  
(5, [(1.0, 442, -10, False)])  
(6, [(1.0, 442, -10, False)])  
(7, [(1.0, 442, -10, False)])
```

Последовательность действий агента в какой-то из ЭПИЗОДОВ.





Основной алгоритм: Q-learning.

- Для основного алгоритма:
 1. `main.py` - основная часть программы реализована здесь.
 2. `discrete.py` - вспомогательный файл от OpenAI.




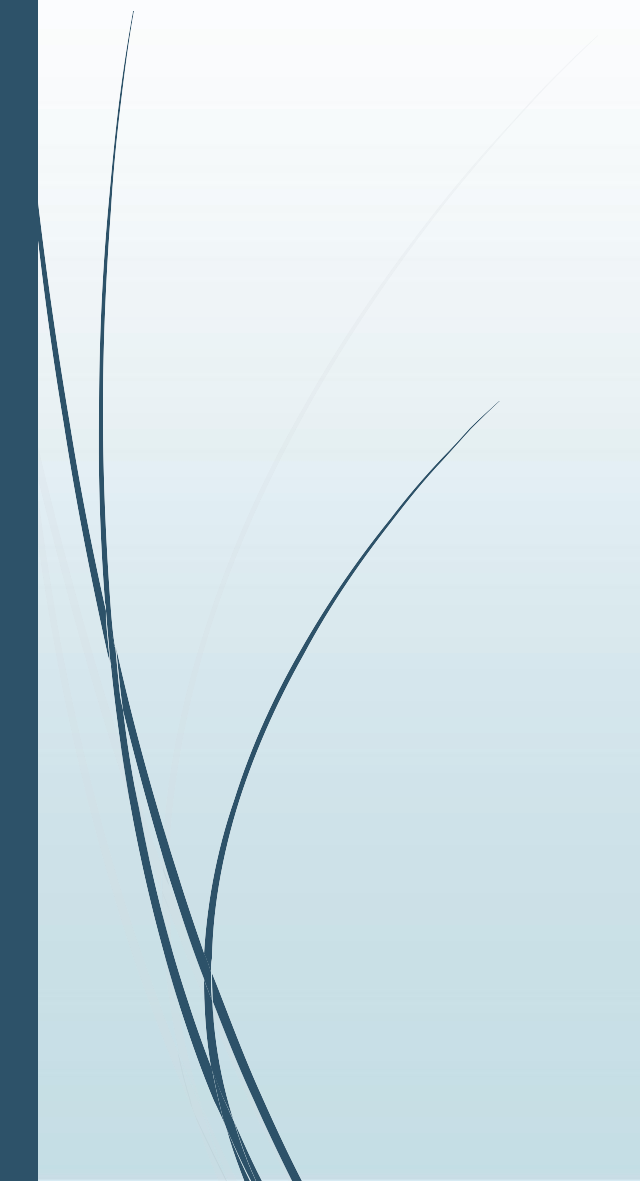
discrete.py

1. `env.reset()` - перезапускает среду; возвращает новое случайное состояние.
2. `env.step()` - продвигает развитие окружающей среды на один шаг.

main.py

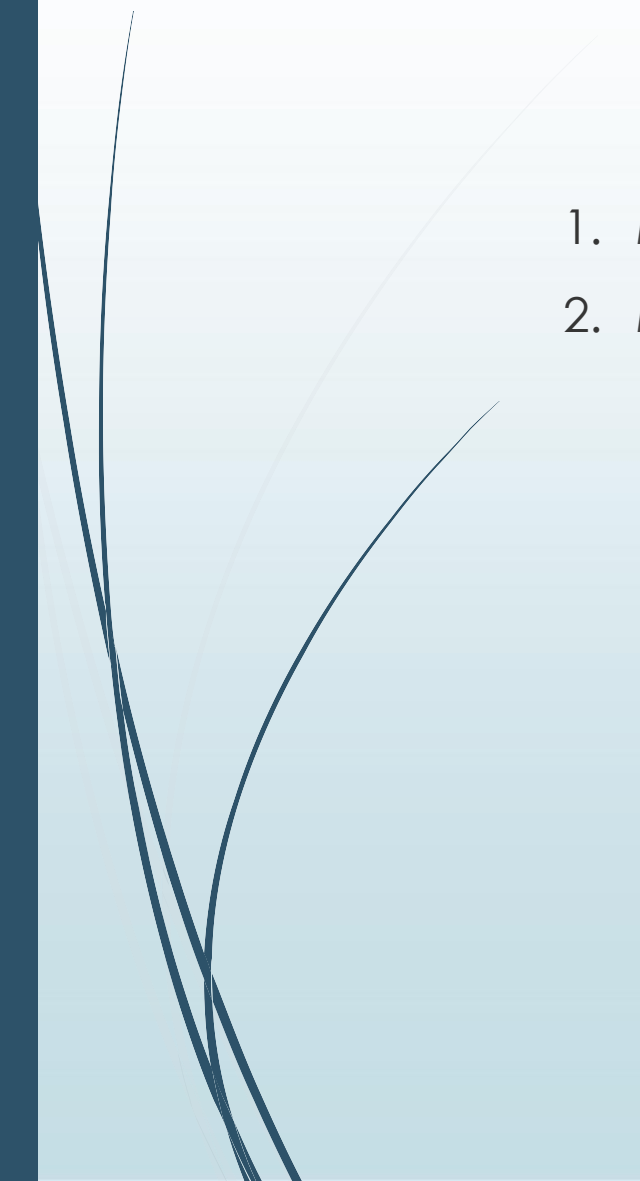
1. env - сердце OpenAI.
2. env = gym.make("Labyrinth") - создание среды.

Параметр	Пояснение
total_episodes	Количество эпизодов в обучении
total_test_episodes	Количество контрольных эпизодов
max_steps	Максимальное число шагов внутри каждого эпизода
alpha	Темп обучения
gamma	Дисконтирующий множитель
epsilon	"Любопытность" агента

- 
- 
1. Инициализация Q-таблицы с нулями,
 2. Цикл по количеству эпизодов:
 - (a) Перезапускаем среду (метод `env.reset()`),
 - (b) Внутренний цикл по количеству максимальных шагов:
 - i. Взаимодействие агента со средой: выполнение доступных для агента действий, в зависимости от 'любопытности' агента,
 - ii. Переход к новому состоянию по результатам взаимодействия со средой,
 - iii. Обновление значений Q-таблицы по формуле, которая была представлена выше,
 - iv. Новое состояние становится текущим состоянием,
 - v. Если агент не выполнил поставленную задачу, то алгоритм повторяется с п.(i),
 - vi. Если выполнил, то уменьшаем 'любопытность' агента и начинаем новый эпизод с п.(a).



Исследование алгоритма.

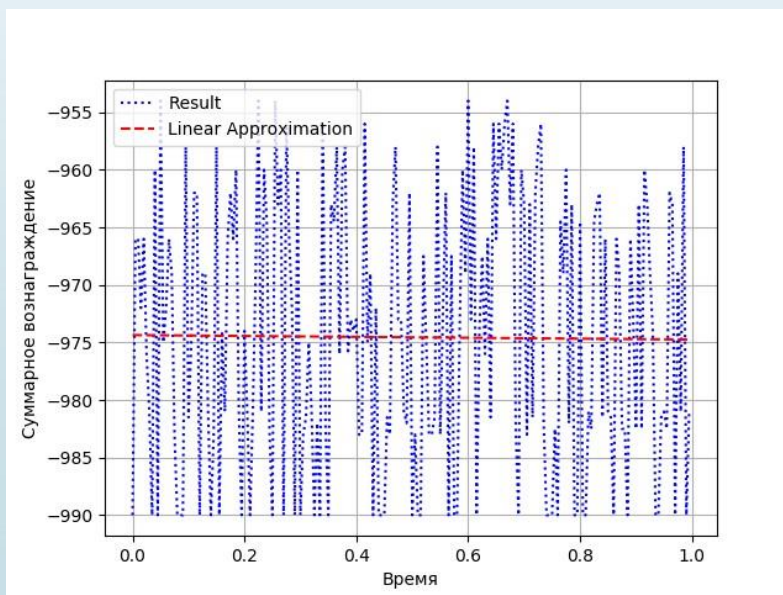
1. Модель, в которой не учитываются физические параметры.
 2. Модель, в которой учитывается высота слоя.
- 

Модель без физических параметров.

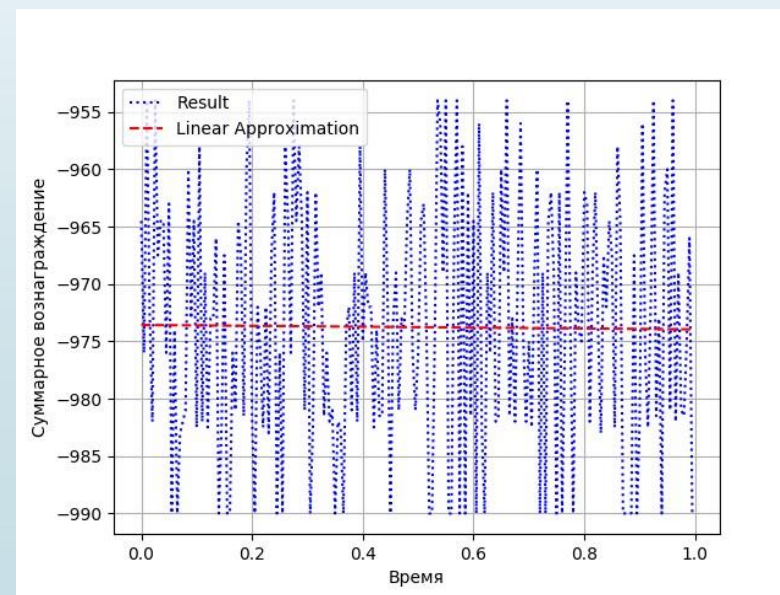
Фиксируем:

1. $\gamma = 0.75$,
2. $\alpha = 0$.

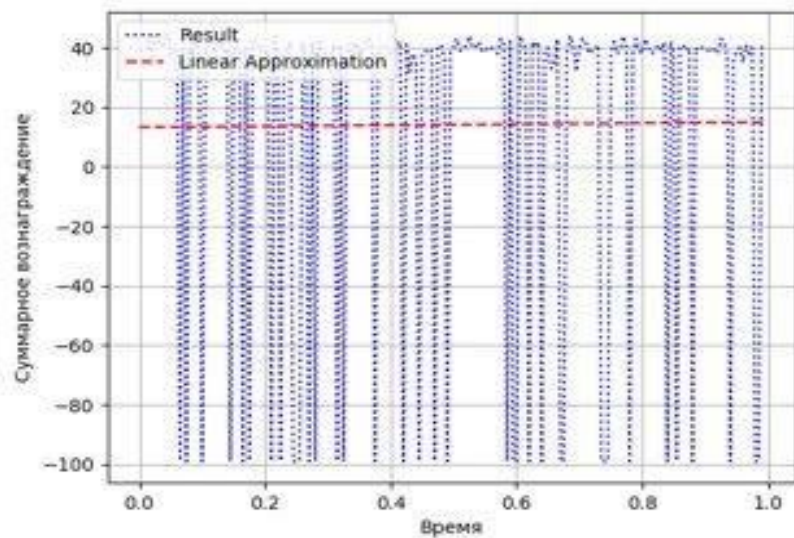
Варьируем N от 50 до 50000.



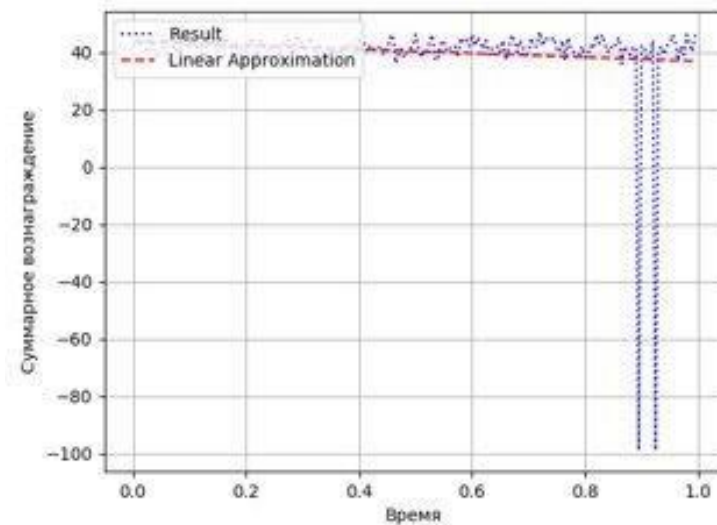
$N=50$



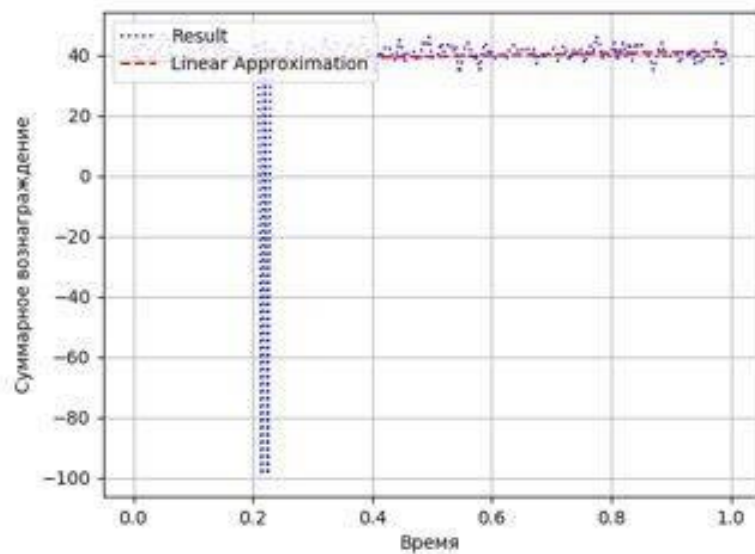
$N=50000$



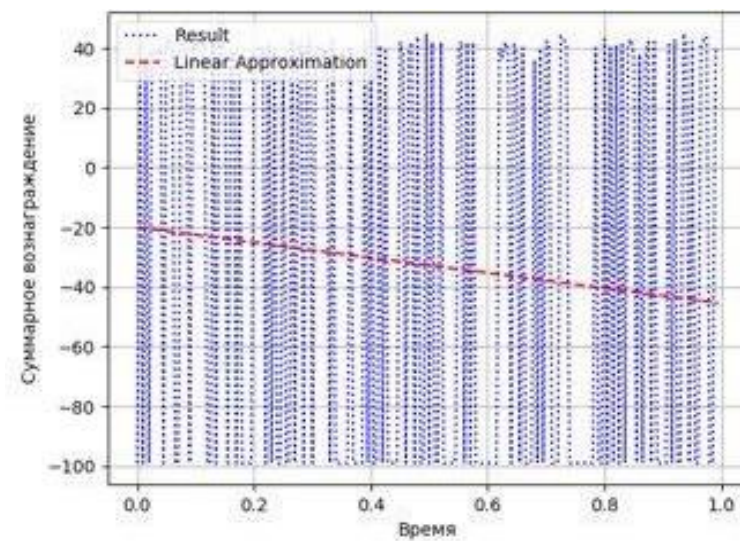
$\alpha = 0.2, N=5000$



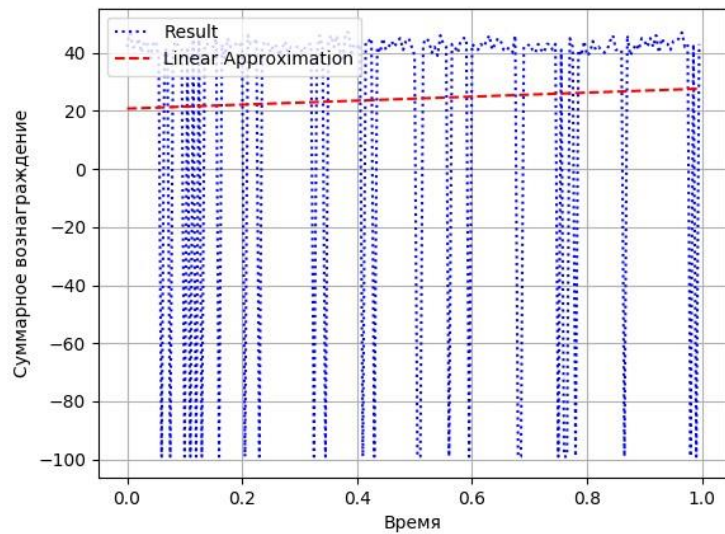
$\alpha = 0.6, N=5000$



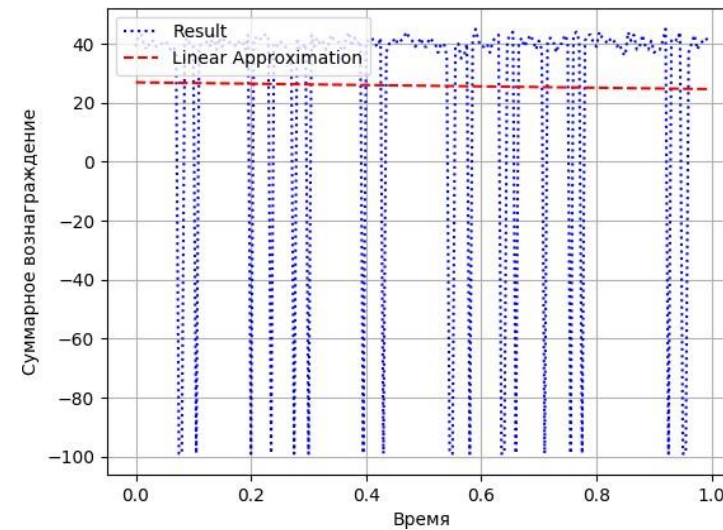
$\alpha = 1.0, N=5000$



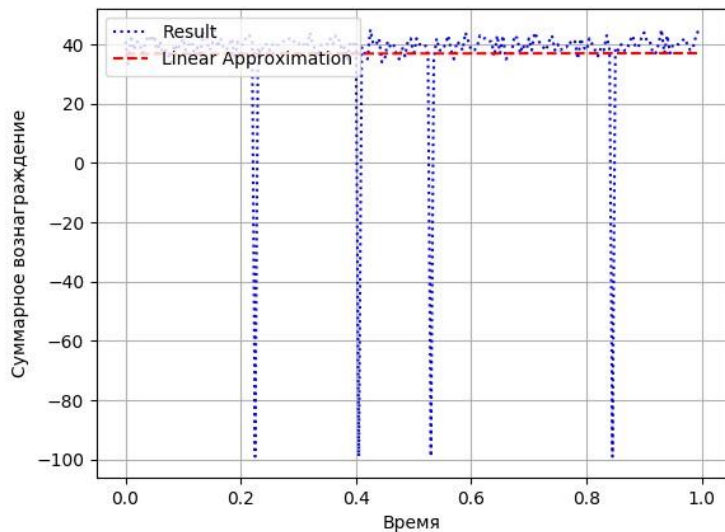
$\alpha_{cr}=1.45, N=5000$



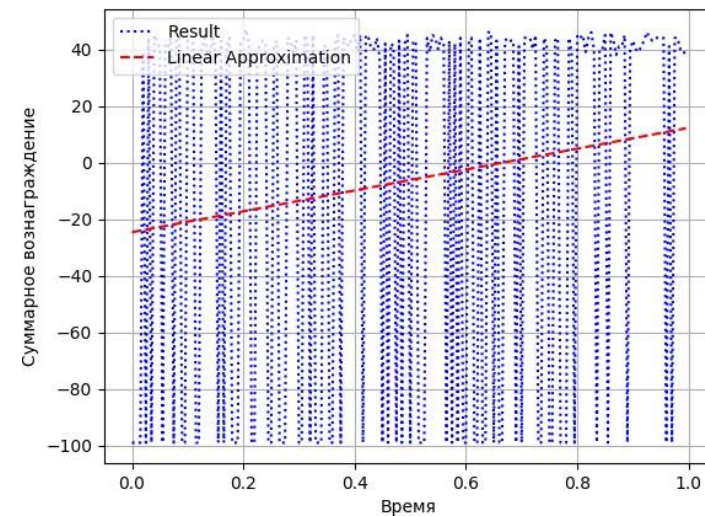
$\gamma=0.2, N=5000$



$\gamma=0.6, N=5000$



$\gamma=1.0, N=5000$



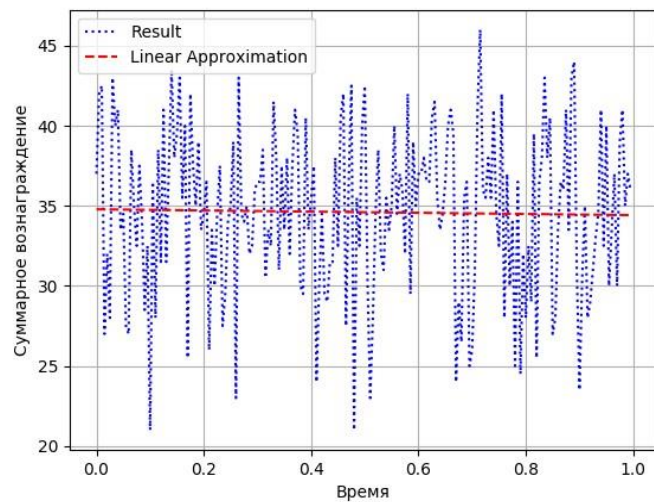
$\gamma_{cr}=1.05, N=5000$

Модель, учитывающая высоту слоя.

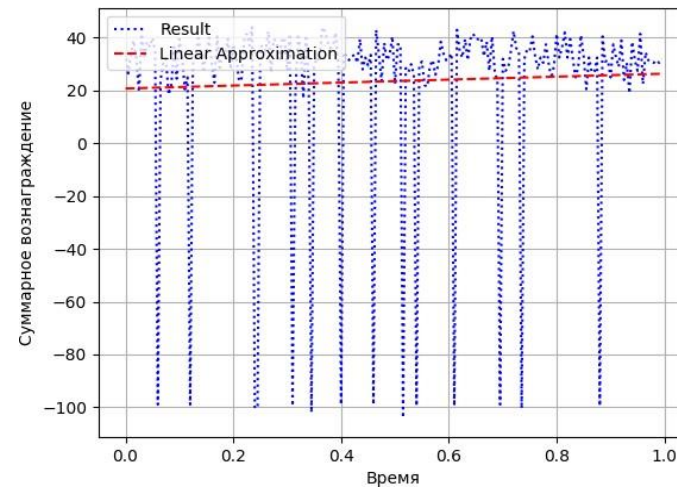
► Фиксируем:

1. $\gamma = 0.85$,
2. $N = 5000$.

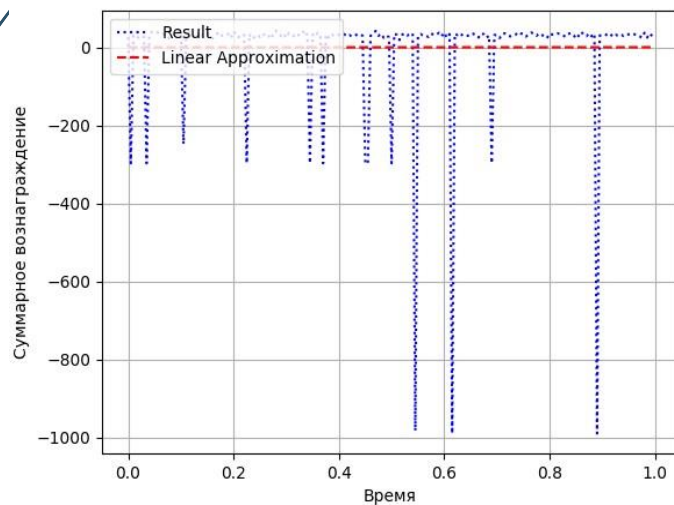
► Варьируем α .



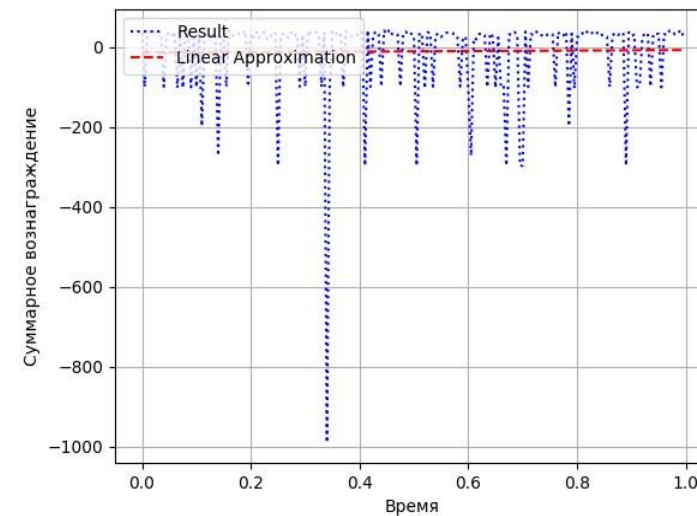
$\alpha = 0.4, N=50000$



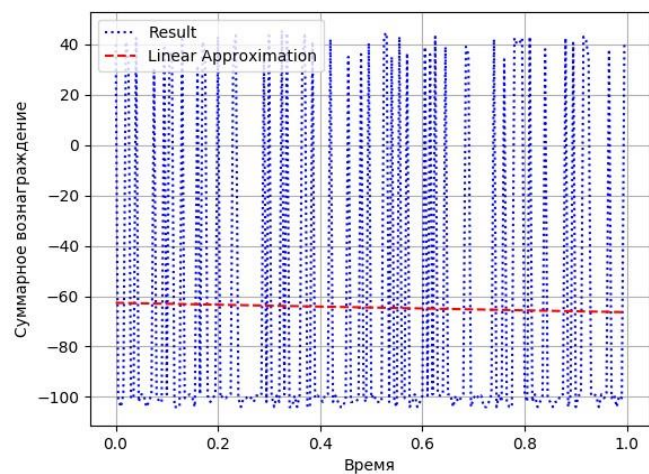
$\alpha = 0.6, N=50000$



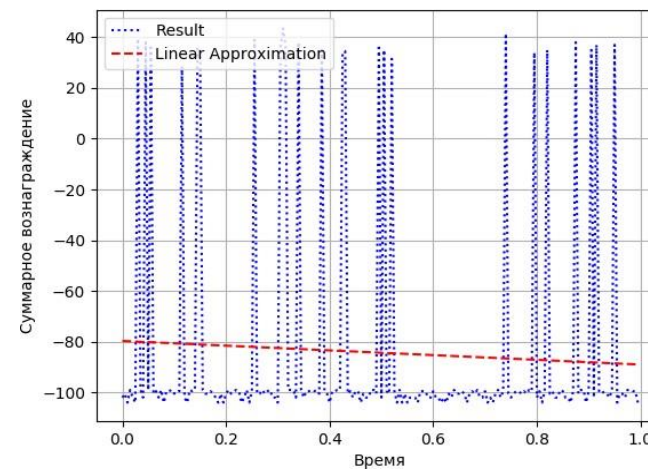
$\alpha = 0.4, N=5000$



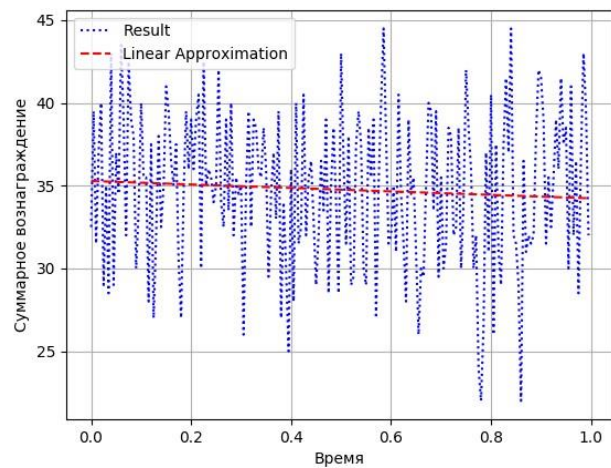
$\alpha=0.6, N=5000$



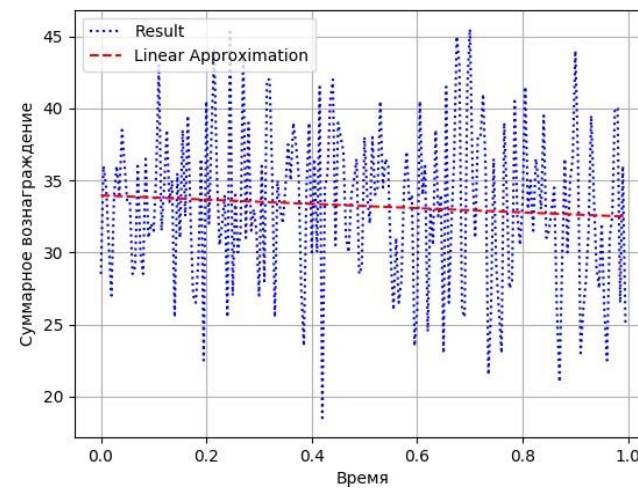
$\gamma=0.2, N=50000$



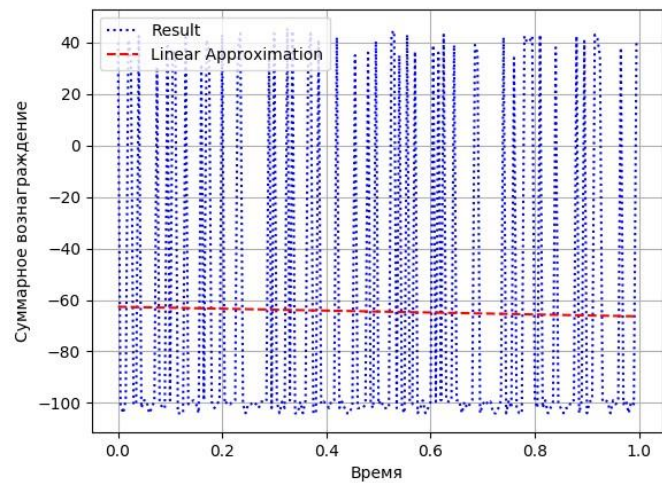
$\gamma=0.4, N=50000$



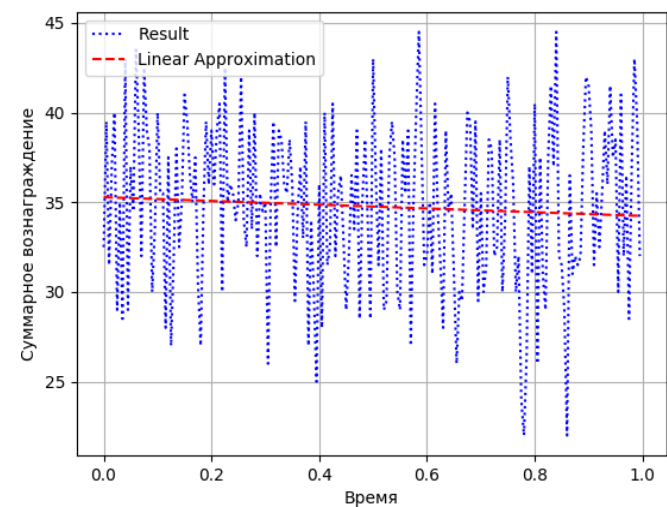
$\gamma=0.8, N=50000$



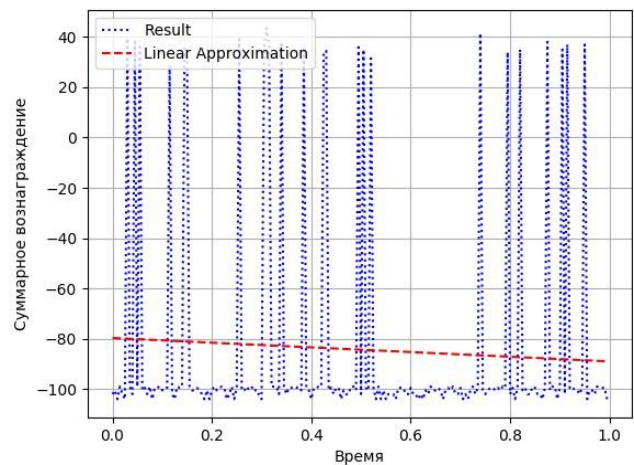
$\gamma=1.0, N=50000$



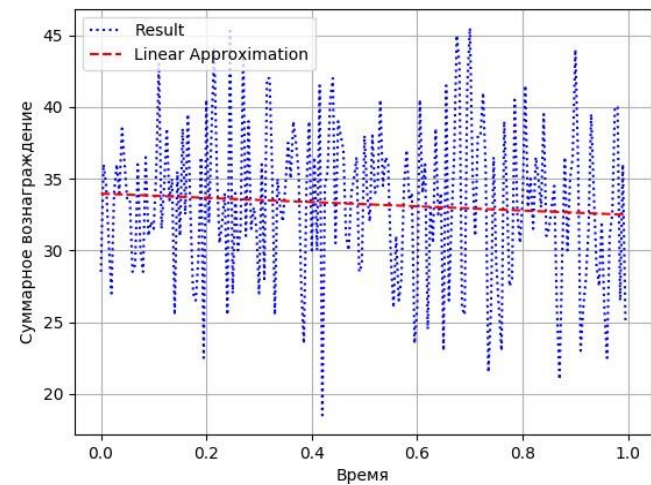
$\alpha=0.63$, $\gamma=0.8$, $N=50000$



$\alpha=0.63$, $\gamma=0.8$, $N=50000$



$\alpha=0.8$, $\gamma=0.73$, $N=50000$



$\alpha=0.8$, $\gamma=0.73$, $N=50000$

Заключение

Параметр	Модель I	Модель II
alpha	0.63	0.77
gamma	0.74	0.86
alpha_cr	1.45	1.42
gamma_cr	1.05	1.03

Модель I - без учета физических параметров среды.

Модель II - с учетом высоты слоя.



Интересные направления развития работы.

1. Усложнить модель. Внести более полный учет параметров воздуха.
2. Реализовать метод сопряженных градиентов для поиска параметров Q-learning'a.
3. Использовать реальные физические данные воздуха.
4. Реализовать 3D визуализацию перемещения агента в среде.
5. Реализовать Deep Q-learning, который является более эффективным алгоритмом, чем Q-learning.



Спасибо за внимание.

