Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»
Физтех-школа Прикладной Математики и Информатики
Кафедра математического моделирования и прикладной математики

Направление подготовки / специальность: 03.03.01 Прикладные математика и физика (бакалавриат)
Направленность (профиль) подготовки: Математическое моделирование, вычислительная математика и физика

УПРАВЛЕНИЕ МАКЕТАМИ НАНОСПУТНИКОВ НА АЭРОДИНАМИЧЕСКОМ СТОЛЕ С ПОМОЩЬЮ МАШИННОГО ОБУЧЕНИЯ

(бакалаврская работа)

Студент:
Толканев Артем Андреевич

_____
(подпись студента)

Научный руководитель:
Иванов Данил Сергеевич,
канд. физ.-мат. наук, доц.

_____
(подпись научного руководителя)

Консультант *(при наличии)*:
_____
_____
(подпись консультанта)

Москва 2019

Оглавление

Аннотация

Выпускная бакалаврская работа посвящена разработке алгоритма управления макетом наноспутника на аэродинамическом столе с использованием машинного обучения. Макет управляется с помощью имитаторов двигателей, однако на него действуют возмущения как со стороны аэродинамической силы, возникающей под воздействием стационарных воздушных потоков, так и гравитационные возмущения вследствие неровности поверхности стола. Целью работы является создание и исследование алгоритма управления макетом, который позволяет адаптивно учитывать возмущения при движении по заданной траектории. Управление макетом построено на основе методов машинного обучения, а именно глубокой нейронной сети с последующим ее обучением. Процесс получения результата можно разделить на 2 этапа. Первый этап - это предобучение на PD-регуляторе, при котором построенная нейросеть обучается имитировать PD-регулятор при отсутствии возмущающих сил. Вторая часть - обучение на столе с действующими на нем стационарными возмущениями. В работе исследуется время, необходимое для обучения и точность отслеживания требуемого относительного движения.

Введение

Можно сформулировать два способа решения проблемы управления спутником. Первый из них - это математическое моделирование и вывод закона управления для сформулированной динамической системы. Другой способ заключается в предположении значительных неизвестных возмущений в динамической модели и реализации адаптивного управления, способного отслеживать изменения в модели среды. Алгоритмы управления полетом обычно проверяются на испытательном стенде, например, на аэродинамическом столе [Segal et al., 2014]. Однако стол так же создает возмущения, аналогов которых в космосе нет. В этой работе для предлагается построить адаптивное управление на основе машиного обучения для минимизации создаваемых столом помех. В последние несколько лет методы машинного обучения были успешно применены для решения различных задач, в частности, методы обучения с подкреплением [Jiang et al., 2018] [Willis et al., 2016] [Izzo and Pettazzi, 2007]. В процессе движения макет изучает среду и на основе полученных отклонений от предполагаемой траектории корректирует свое управление. Построение алгоритма управления представлено в работе и приведены результаты моделирования на компьютерной модели аэродинамичесского стола.

1. Постановка задачи

Рассмотрим движение макета наноспутника по заданной заранее траектории с известным начальным положением по аэродинамическому столу с неизвестными возмущениями.

классная картинка с макетом на столе

Макет наноспутника представляет собой автономный аппарат массой около 5.2 kg, совершающим движения по столу с помощью 4 имитаторов двигателей ввиде вентиляторов. При решении задачи предполагалось, что мы знаем как управлять вентиляторами, чтобы совершать перемещения по столу, при условии отсутствия возмущений. Ускорение, подающееся лопастями, имеет ограничение в 0.37 $\frac{m}{s^2}$.

Аэродинамический стол представляет из себя поверхность размерами 198cm на 148cm с отверстиями диаметром 1 mm, размещенными с интервалом 20 mm, из которых дует воздух. Из-за неравномерных потоков воздуха и неровности стола при движении макета появляется возмущающее ускорение. При решении задачи считаем, что возмущения стационарны. Направления и величины возмущающих ускорений приведены на рисунках [Ivanov et al., 2018]:

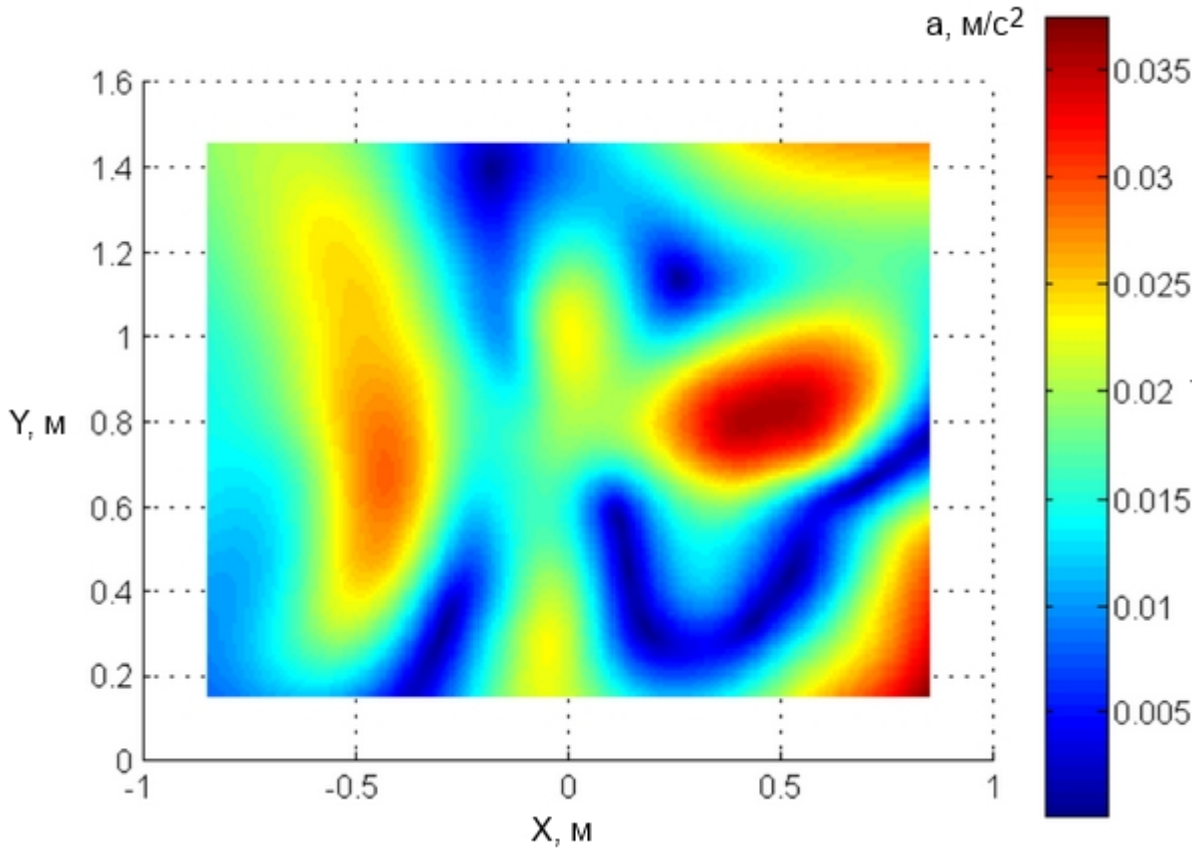

Рисунок 1 – Величины возмущающего ускорения

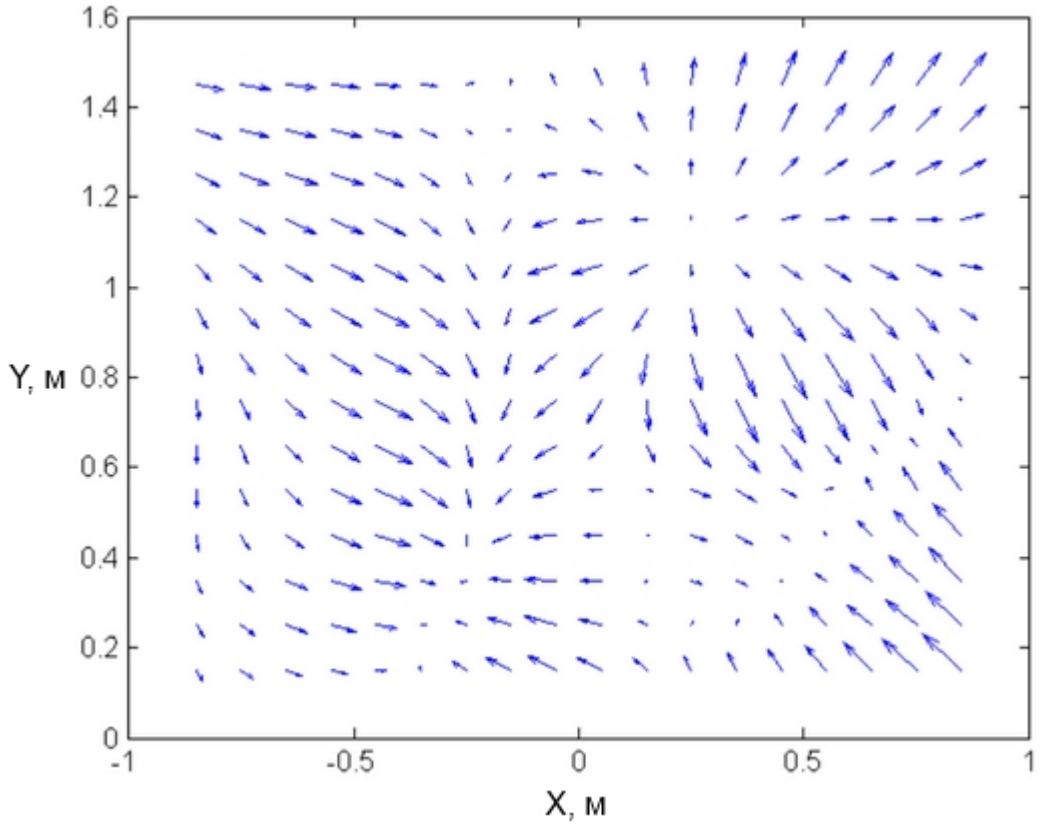

Рисунок 2 – Направления возмущающего ускорения

Введем систему координат, жестко связанную с аэродинамическим столом Oxy и запишем систему уравнений движения макета:

$$\begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{v}, \\ \dot{\boldsymbol{v}} = \boldsymbol{a_{control}} + \boldsymbol{a_{perturbation}} \\ \boldsymbol{r}|_{t=0} = \boldsymbol{r_0} \\ \boldsymbol{v}|_{t=0} = \boldsymbol{v_0} \\ |\boldsymbol{a_{control}}| \leq 0.36538 \ \frac{m}{s^2} \\ -0.82 \ m \leq x \leq 0.82 \ m \\ 0 \ m \leq y \leq 1.61 \ m \end{cases} \tag{1}$$

где $\boldsymbol{r}$ – вектор положения макета на столе ($\boldsymbol{r} = (x, y)$), $\boldsymbol{v}$ – его скорость, $\boldsymbol{a_{control}}$ – ускорение, полученное с помощью управления, $\boldsymbol{a_{perturbation}}$ – ускорение, создаваемое возмущениеями окружающей среды. $\boldsymbol{R}$ и $\boldsymbol{V}$ - векторы начальных положение и скорость макета соотвественно.

Рассмотрим траекторию, по которой должен двигаться макет. Предполагаем, что она представляет собой 4-х мерную вектор функцию, зависющую от времени, которая показывает, какими должны быть положение ($\boldsymbol{r_d}$) и скорость ($\boldsymbol{v_d}$), в заданный момент времени.

В рамках перечисленных предположений и допущений требуется построение адаптивного управления вентиляторами с помощью методов машинного обучения для достижения требуемой траектории движения макета, на который действуют неизвестные, но стационарные возмущения со стороны аэродинамического стола.

2. Решение задачи

Для решения поставленной задачи построения адаптивного управления с использования машинного обучения будем отталкиваться от общеизвестного алгоритма управления на основе ПД-регулятора. С его помощью можно провести предобучение адаптивного алгоритма управления и добиться, чтобы на начальном этапе алгоритм работал не хуже него. Для построения адаптивного алгоритма используется нейронная сеть.

2.1. Ляпуновское управление

В этой части описывается влияние возмущений на точность контролируемого движения. Для управления используется $PD-$регулятор.

Возьмем функцию Ляпунова в виде:

$$V = \frac{1}{2}\left(\boldsymbol{e}_r^T K_1 \boldsymbol{e}_r + \boldsymbol{e}_v^T \boldsymbol{e}_v\right), \tag{2}$$

где $\boldsymbol{e}_r = \boldsymbol{r} - \boldsymbol{r}_d$, $\boldsymbol{e}_v = \boldsymbol{v} - \boldsymbol{v}_d$, K_1 – положительно-определенная диагональная матрица 2 на 2. На плоскости движение с управлением может быть описано уравнением

$$\ddot{\boldsymbol{r}} = \boldsymbol{u}, \tag{3}$$

где $\boldsymbol{u}$ – вектор ускорения, полученный управлением. Тогда

$$\dot{V} = \boldsymbol{e}_v^T K_1 \boldsymbol{e}_r + \boldsymbol{e}_v^T \dot{\boldsymbol{e}}_v = \boldsymbol{e}_v^T \left[K_1 \boldsymbol{e}_r + \dot{\boldsymbol{e}}_v\right] = \boldsymbol{e}_v^T \left[K_{\boldsymbol{i}} \boldsymbol{e}_r + \boldsymbol{u} - \ddot{\boldsymbol{r}}_{\boldsymbol{d}}\right]. \tag{4}$$

Для того, чтобы выполнялась теорема Барбашина-Красовского, необходимо, что $\dot{V} \leq 0$, то есть

$$K_1 \boldsymbol{e}_r + \boldsymbol{u} - \ddot{\boldsymbol{r}}_d = -K_2 \boldsymbol{e}_v, \tag{5}$$

где K_2 – положительно-определенная диагональная матрица 2 на 2. Тогда получаем необходимое ускорение:

$$\boldsymbol{u} = -K_1 \boldsymbol{e}_r - K_2 \boldsymbol{e}_v + \ddot{\boldsymbol{r}}_d. \tag{6}$$

$PD-$регулятор совпадает с полученным способом управления, когда $\ddot{\boldsymbol{r}}_d = \boldsymbol{0}$. Результаты данного управления представим на примере опорной траектории вида:

$$\begin{aligned} x &= A \cdot \sin(\omega t) + x_0 \\ y &= A \cdot \cos(\omega t) + y_0 \end{aligned} \tag{7}$$

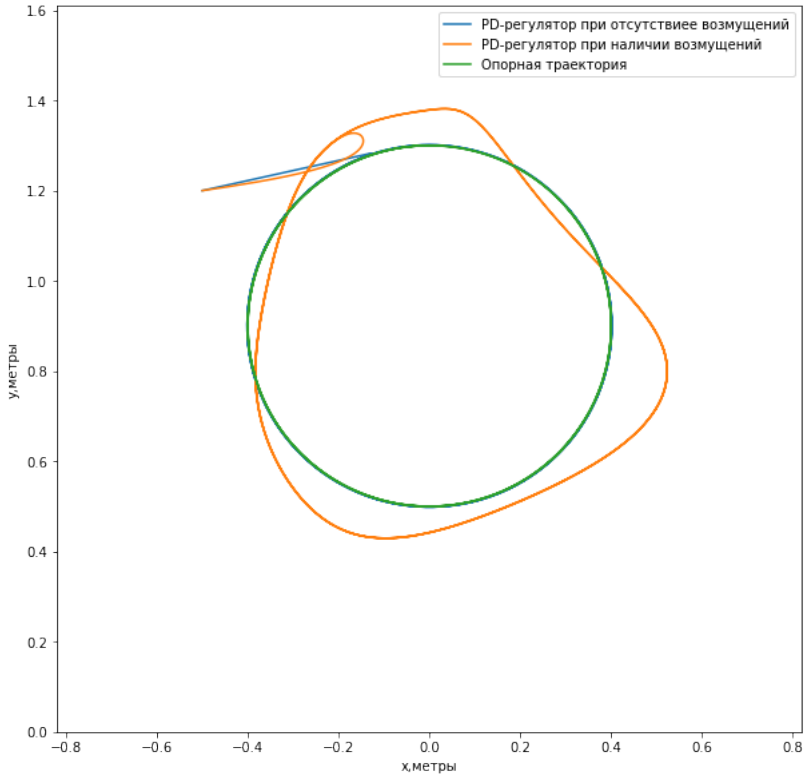

Рисунок 3 – Опорная траектория и траектории управляемого движения в случае возмущенного и невозмущенного движений.

На рис. 3 представлена опорная траектория в виде окружности, которая достигается с помощью PD-регулятора почти идеально, если на макет не действует возмущений. Однако, если действуют неизвестные возмущения на макет, то ошибки траектории могут достигать до 15 см, что может быть неудовлетворительно для тестирования того или иного алгоритма управления на аэродинамическом столе.

2.2. Управление на основе машинного обучения

В этой части будет описана используемая теория по нейросетям и обучению с подкреплением.

2.2.1. Описание нейросети

В данной работе при решении задачи используется многослойный перцептрон (МСП) [Кадурин А.,]. Он представляет собой параметрическую математическую модель в виде суперпозиции и линейных комбинаций "простых" функций, где под простыми понимается возможность подсчитать градиент по параметрам этих линейных комбинаций. МСП состоит из набора полностью связанных слоев. Рассмотрим "слой" - это набор n элементов, каждый из которых имеет *вход* и *выход*. Рассмотрим i-ый элемент этого слоя.

Пусть на *вход* подается m-мерный вектор $\boldsymbol{x}$. Тогда *выходом* (обозначим за y_i) будет

$$y_i = f(w_1 x_1 + w_2 x_2 + ... + w_m x_m), \tag{8}$$

где вектор $\boldsymbol{w}$ является обучаемым параметром нейросети и называется *весом*, а f называется *функцией активации*, она должна быть непрерывной и кусочно-дифференцируемой. Тогда *выходом* всего *слоя* (при условии, для каждого элемента слоя был одинаковый вход) будет вектор

$$y = \begin{pmatrix} f\left(w_{1,1} x_1 + \ldots + w_{1,m} x_m\right) \\ \vdots \\ f\left(w_{n,1} x_1 + \ldots + w_{n,m} x_m\right) \end{pmatrix} \tag{9}$$

Слои нейросети будут полностью связанными, если для всех слоев, кроме первого, входным вектором будет выход предыдущего слоя. Заметим, что функция активации не обязательно одинакова в разных слоях (но одна и та же для элементов одного слоя). Веса нейросети инициализируются случайно. *Входом нейросети* будет вход первого слоя нейросети. На каждый элемент первого слоя будет подаваться какое-то число, потом на каждом следующем слое происходят вычисления с помощью описанных выше уравнений и так до последнего слоя. *Выходом нейросети* будет выход последнего слоя. Из-за случайной инициализации выход нейросети будет случайным. Изменения весов на основании выхода нейросети называют *обучением нейросети*.

Обучающей выборкой назовем конечный набор пар векторов, таких, что первый вектор из пары будет подаваться на нейросеть, а второй вектор будет сравниваться по какому-то правилу с выходом нейросети. Это правило сравнения называют *функцией потерь*. Значение этой функции *Loss* используют для обучения нейросети. Функция потерь строится так, чтобы при ее уменьшении уменьшалась мера несовпадения между выходом нейросети и значениями на обучающей выборке.

Опишем процесс изменения весов нейросети, так, чтобы уменьшить среднее значение функции потерь на обучающей выборке. Из построения нейросети и функции потерь получаем, что для любого слоя нейросети функция потерь есть функция от весов данного слоя. Поэтому после вычисления функции потерь для данного входа произведем замену весов на каждом слое с помощью градиентного спуска:

$$w_{i,j} = w_{i,j} - \delta w_{i,j}, \tag{10}$$

где

$$\delta w_{i,j} = \alpha \cdot \frac{\partial Loss}{\partial w_{i,j}}\ , \tag{11}$$

α - называют *скоростью обучения*.

2.2.2. Обучение с подкреплением

Так же при решении задачи используются методы из обучения с подкреплением. [Barto,] В данной работе макет является *агентом*, который может совершать

какие-то *действия*. Все, с чем взаимодействует агент является *средой*. Это взаимодействие происходит постоянно: агент совершает действие, находясь в каком-то *состоянии*, среда реагирует на действие и сообщает агенту новое состояние. Кроме того, среда дает награду агенту - какое-то число, являющего значением функции от состояния и действия, котору агент пытается максимизировать, выбирая соответствующие действия. Пусть взаимодействие между средой и агентом происходят последовательно на каждом элементе последовательности времени $\{t\}_0^{\inf}$. При каждом t агент имеет какое-то представление о состоянии $s_t \in S$ и выбирает действие $a_t \in A$, где S и A - пространства состояний и действий агента соответственно. На следующем шаге после совершения этого действия агент получает награду $r_t \in R \in \mathbb{R}$ и переходит в новое состояние $s_{t+1} \in S$. В итоге получаем траекторию агента в виде последовательности: $s_0,\ a_0,\ r_0,\ s_1,\ \ldots$. Предположим, что взаимодействие между агентом и средой подчиняется марковскому свойству, а именно

$$\mathbb{P}(s_{t+1}, r_t | s_t, a_t, ..., s_0, a_0) = \mathbb{P}(s_{t+1}, r_t | s_t, a_t) \tag{12}$$

для всех $s \in S$ и $a \in A$.

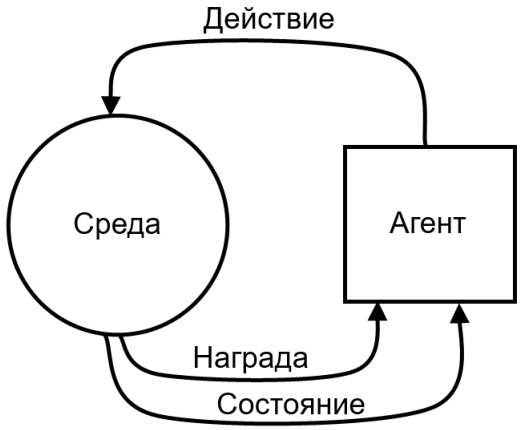

Рисунок 4 – Взаимодействие агента и среды при марковском процессе

Поставим цель перед агентом: действовать так, чтобы максимизировать не следующую полученную награда r_t, а максимизировать суммарную награду за определенный промежуток времени. Рассмотрим сумму полученных наград с определенного промежутка времени с определенным коэфициентов $\gamma \in [0; 1)$, предположив, что $r \in R$ ограничено:

$$G_t = \sum_k^{\infty} \gamma^k r_{t+k} \ . \tag{13}$$

Определим *стратегию* πагента:

$$\pi(a|s) = \mathbb{P}(a_t = a | s_t = s) \ . \tag{14}$$

Используя предположение о выполнение марковского свойства процесса определим функ-

ции ценности, *ценности действия*, *производительности* при следовании фиксированной стратегии π:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}(G_t|s_t = s) \\ Q^{\pi}(s,a) &= \mathbb{E}_{\pi}(G_t|s_t = s, a_t = a) \\ J^{\pi} &= V^{\pi}(s = s_0) \end{aligned} \tag{15}$$

Цель агента - найти такую π, чтобы, начиная свое взаимодействие со средой из состояния s_0 максимизировать J.

3. Схема обучения

Поставим исходную задачу в терминах обучения с подкреплением. Состоянием $s_t \in S$ макета назовем 8-ку чисел, первые 4 из которых есть текущее положение и скорость макета на плоскости стола, а остальные 4 числа есть положение и скорость, которую мы хотели бы иметь после совершения действия:

$$s_t = [\boldsymbol{r}_t; \boldsymbol{v}_t; \boldsymbol{r}_{d,t+1} \boldsymbol{v}_{d,t+1}] \ . \tag{16}$$

Действием $a_t \in A$ назовем двойку чисел, которые являются проекциями ускорения на оси введенной системы координат, полученного с помощью управления:

$$a_t = \boldsymbol{u}_t \ . \tag{17}$$

Так как мы пытаемся построить управление, приводящее макет к опорной траектории, то выберем функцию награды, как функцию, связанную с отношением положений макета и его желаемым положением. Наградой за действие макета $R_t \in R$ из состояния возьмем отрицательное значение евклидова расстояния между двумя точками на плоскости стола - первая из них, это желаемое положение и скорость, а вторая - это положение и скорость, которые мы получили:

$$R_t = -(r_{t+1} - r_d)^2 - \alpha(v_{t+1} - v_d)^2 \ . \tag{18}$$

Теперь определим поиск оптимальной стратегии поведения макета.

3.1. Выбор основанный на функции ценности действия

Можно искать стратегию, используя функции ценности действия:

$$a = argmax_a Q(s, a) \tag{19}$$

Для поиска оптимальной стратегии необходимо найти оптимальную функцию ценности действия, которую можно получить из уравнения Беллмана:

$$Q^*(s_t, a_t) = \mathbb{E}(R_t + \gamma \max Q^*(s_{t+1}, a_{t+1})) \tag{20}$$

Так как информации о том, в каком состоянии макет окажется - отстуствует (вследствии наличия возмущений у среды) можно использовать алгоритмы поиска оптимальной функции ценности действия, путем совершения различных действий, начиная из одного состояния:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(R_t + \gamma Q(s_{t+1}, a_{t+1}) \ . \tag{21}$$

Откуда возникают две проблемы: первая - необходимость возвращаться макету в исходную позицию, чтобы гененрировать различные действия из одного состояния, что невоз-

можно, так как управление еще не может этого позволить, вторая- поиск $argmax_a Q(s,a)$, так как в данном случае пространство действий несчетно, а функция Q не обязательно выпуклая.

3.2. Выбор основанный на изменении стратегии

Можно искать стратегию, и использовать ее для генерации действия:

$$a \sim \pi(a|s) \ , \tag{22}$$

либо определить и искать детерминированную стратегию используя функции ценности действия:

$$\pi(s) = argmax_a Q(s,a) \ . \tag{23}$$

Для этого параметризуем стратегию вектором параметров π_w, $w \in \mathbb{R}^d$ и рассмотрим J^π как функцию от вектроа параметров w. Тогда

$$\nabla_w J = \frac{\partial J}{\partial \pi_w} \frac{\partial \pi_w}{\partial w} \ . \tag{24}$$

Для изменения стратегии на будем менять вектор параметров w на основе $\nabla_w J$:

$$w_{t+1} = w_t + \alpha \nabla_w J_t \tag{25}$$

Для использованиия этого метода нужно на каждом шаге вычислять величину $\nabla_w J_k$, а для этого необходимо сравнивать полученную награду при совершении действия с какой-то оценкой награды при совершении этого действия. На основании теоремы о градиенте производительности

$$\nabla_w J(\pi_w) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_w} \left[\nabla_w \log \pi_w(a|s) Q^\pi(s,a)\right] \ , \tag{26}$$

можно получать оценку $\nabla_w J_k$, в данном случае $Q^\pi(s,a)$ остается неизвестной.

3.3. Actor-Critic метод

Учитывая предложенные теоретические предположения для формирования стратегии макета в двух предыдущих пунктов предложим объединить два подхода в так называемый Actor-Critic метод.

Определим нейросеть Actor:

На вход Actor будут подаваться состояния s макета, на выходе будут формироваться a.

Определим нейросеть Critic:

На вход Critic будут подаваться состояния s макета и выход Actor. На выходе Critic будет формироваться функция ценности Q.

На основании выхода Critic будет сравниваться с найстоящим R и обучаться. В процессе обучения Critic будет формироваться $\frac{\partial Q}{\partial a}$ по которому будет обучаться нейросеть

Actor.

3.4. Предобучение

При использовании нейросети возникает проблема: веса нейросети инициализируются случайно. Поэтому и выход нейросетей Actor и Critic будет случайным и процесс обучения может занять очень много времени. Для решений этой проблемы предлагается ввести предобучение, то есть рассмотреть нашу среду как идеальную, без наличия возмущений. Для такой среды нет необходимости использовать нейросети и можно использовать описанный в начале PD-регулятор. С помощью него обучим нейросеть Actor.

Algorithm 1 Алгоритм предобучения нейросети Actor

1: Объявляем m и n
2: **for** i: $0 \leq i \leq m$ **do**
3: Генерируем n случайных векторов s
4: Вычисляем значения на этой выборке с помощью PD- регулятора в идеальной среде
5: Вычисляем значения на этой выборке с помощью нейросети Actor
6: Ищем разницу между соотетствующими векторами по $|| \cdot ||_2$ и суммируем все полученные n значений
7: Обучаем нейросеть
8: **end for**

При моделировании использовался МСП с двумя скрытыми слоями размера 64 и 16 с функций активации $ReLU$:

$$ReLu(x) = \begin{cases} x, x > 0 \\ 0, \text{иначе} \end{cases} \tag{27}$$

на каждом из скрытых слоев, $m = 10000$ и $n = 256$.

Algorithm 2 Алгоритм предобучения нейросети Critic

Объявляем m, n, ϵ
for i: $0 \leq i \leq m$ **do**
    Генерируем n случайных векторов $[r; v]$
    if i делится на 2 **then**
        Генерируем n случайных векторов $[r_d; v_d]$
    else
        Генерируем n случайных векторов $[r_d; v_d]$ в малой окрестности ϵ
    end if
    for j: $0 \leq j \leq n$ **do**
        Генерируем необходимое ускорение a с помощью PD-регулятора
        Вычисляем R с помощью $||\cdot||_2$ между $[r; v]$ и $[r_d; v_d]$
        Вычисляем R_{critic} как выход нейросети Critic на основе $[r; v; r_d; v_d; a]$
    end for
    Вычисляем среднее квадрата отклонения R и R_critic на выборке из n элементов
    Обучаем нейросеть
end for

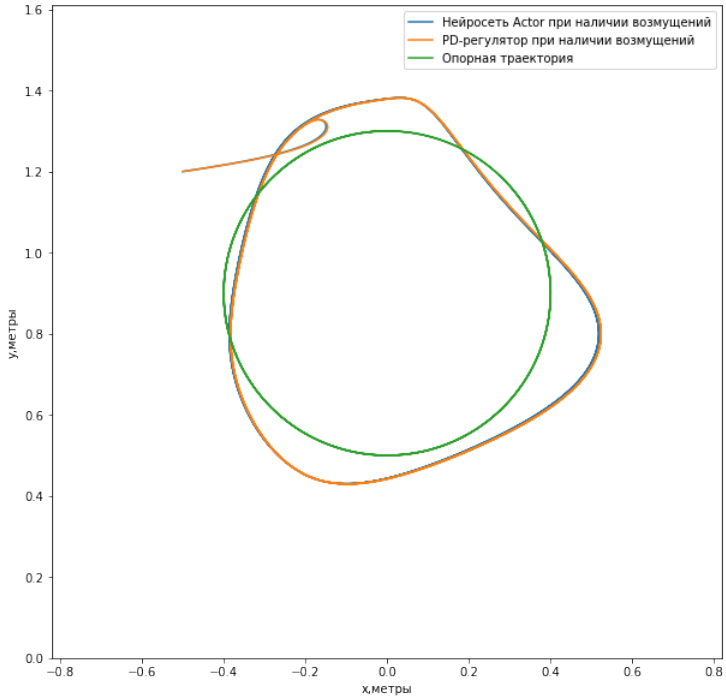

Рисунок 5 – Траектория движения макета под управлением, полученным с помощью предобученной нейросети, и траектория движения с управлением на основе ПД-регулятора

На рис. 5 представлен результат предобучения нейронной сети - траектория движения макета в возмущенной среде практически совпадает с траекторией движения под действием алгоритма на основе ПД-регулятора. Из рис. 6 видно, что разница в траекториях составлят всего несколько миллиметров.

Предобучим нейросеть Critic. При моделировании использовался МСП с двумя скрытыми слоями размера 100 и 100 с функций активации $ReLU$, на каждом из скрытых

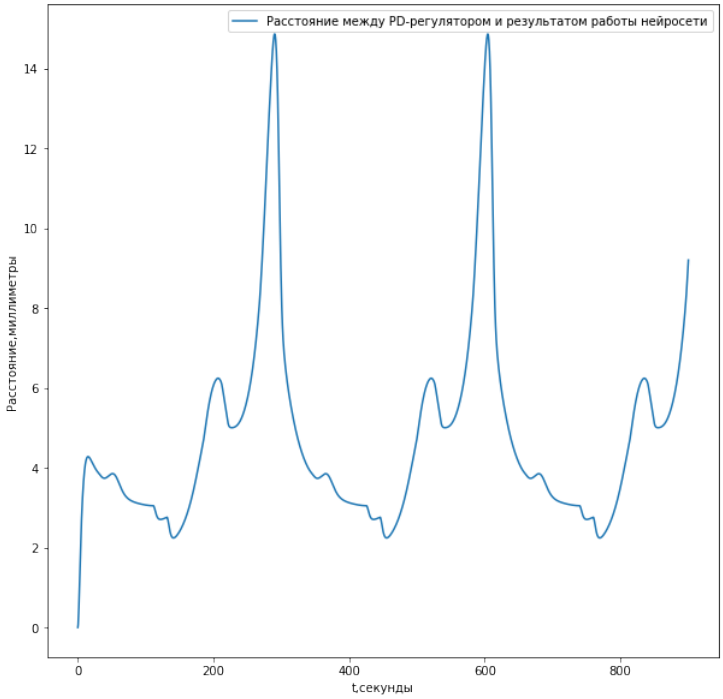

Рисунок 6 – Разница траекторий движения

слоев, $m = 10000$, $n = 256$ и $\epsilon = 0.05$.

Заключение

Оценка с помощью нейросети в реальном времени возмущений, воздействующих на макеты микроспутников на испытательном стенде с воздушной подвеской, позволяет значительно улучшить характеристики алгоритма управления движением макета. Преимущество обучения с подкреплением заключается в том, что исследователи не могут точно знать модели движения макета и возмущения на испытательном стенде, и, тем не менее, коррекции ошибки контролируемого движения будут достаточными. Недостатком является то, что нейронная сеть требует времени для обучения и требует вычислительной мощности для обучения в режиме реального времени. В настоящей работе был предложен алгоритм адаптивного управления с помощью машинного обучения, проведено предобучения двух нейронных сетей Actor и Critic, которые показывают значительное улучшение тректории движения макета по сравнению с траекторией под действием ПД-регулятора. В продолжение работы будет продемонстрирована работа алгоритма управления на основе обучения с подкреплением. Также планируется реализация алгоритма на макете микроспутника на стенде в ИПМ им М.В. Келдыша РАН.

Список использованных источников

[Barto,] Barto, R. S. S. A. G. *Introduction to Reinforcement Learning.*

[Ivanov et al., 2018] Ivanov, D., Koptev, M., Mashtakov, Y., Ovchinnikov, M., Proshunin, N., Tkachev, S., Fedoseev, A., and Shachkov, M. (2018). Determination of disturbances acting on small satellite mock-up on air bearing table. *Acta Astronautica*, 142:265–276.

[Izzo and Pettazzi, 2007] Izzo, D. and Pettazzi, L. (2007). Autonomous and distributed motion planning for satellite swarm. *Journal of Guidance, Control, and Dynamics*, 30(2):449–459.

[Jiang et al., 2018] Jiang, X., Li, S., and Furfaro, R. (2018). Integrated guidance for mars entry and powered descent using reinforcement learning and pseudospectral method. *Acta Astronautica.*

[Segal et al., 2014] Segal, S., Carmi, A., and Gurfil, P. (2014). Stereovision-based estimation of relative dynamics between noncooperative satellites: Theory and experiments. *IEEE Transactions on Control Systems Technology*, 22(2):568–584.

[Willis et al., 2016] Willis, S., Izzo, D., and Hennes, D. (2016). Reinforcement learning for spacecraft maneuvering near small bodies. In *AAS/AIAA Space Flight Mechanics Meeting*, pages 14–18.

[Кадурин А.,] Кадурин А., Николенко Сергей Игоревич, *Глубокое обучение. Погружение в мир нейронных сетей.*