Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра нелинейных динамических систем и процессов управления

Скиба Глеб Андреевич

Генерация походки четырёхногого робота при помощи нейронных сетей

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:
кандидат физ.–мат.
О.И.Гончаров

Москва, 2023

Оглавление

Постановка задачи

Одно из активно развивающихся направлений робототехники — создание четырехногих шагающих роботов. Такие роботы выступаю в качестве мобильной платформы (робот оснащён видеокамерами, тепловизорами, пространственными сканерами) для инспекции открытых пространств или помещений. При создании систем управления шагающими роботами одной из ключевых задач является синтез походки. Генератор походки должен уметь формировать траекторию движения основания робота и его ног с учетом его текущего положения, целевой позы, эти траектории должны быть физически непротиворечивы и оптимальны. В данной работе рассматривается задача создания генератора походок робота с использованием оптимизационных и нейросетевых методов.

Литературный обзор

При исследовании уже известных методов генерации походок робота были выделены при изучении литературы три класса методов, а именно аналитические методы генерации походки, оптимизационные методы генерации походки и методы использующие нейронные сети. Рассмотрим более подробно каждый класс методов и выделим их недостатки:

1. **Аналитические методы генерации походки.** Методы, состоящие из набора аналитических формул и эвристик, по которым рассчитывает движение, например CHAMP [1] задающий траектории движения ног кривыми Безье, параметры которых зависят от желаемой скорости или аналогичный метод Whole-Body Planning [11] для человекоподобного робота. **Достоинства**: данные методы просты в реализации. **Недостатки:** данные методы требуют ручного подбора эвристик, обычно не дают гарантий равновесия.
2. **Оптимизационные методы.** Методы создающие походки как решения нелинейной (TOWR [2]) или линейной (MIT Chetath [3]) задачи оптимизации. **Достоинства**: данные методы позволяют создавать достаточно сложные и длинные походки. **Недостатки:** линейные оптимизационные методы также опираются на эвристики (постановка ног), а нелинейные обычно не могут работать в реальном времени.
3. **Методы использующие нейронные сети.** Нейронные сети, обученные методом обучения с подкреплением, например DeepGait [4] или аналогичный метод RL Approach for Dynamically Stable Inverse Kinematics of Humanoid Robots [12] для человекоподобного робота.

Достоинства: данные методы позволяют создавать сложные, разнообразные походки произвольной длины. **Недостатки:** данные методы требуют физические модели робота и окружающего мира, их обучение долгий и очень вычислительно затратный процесс.

Математическое представление движения робота

Сначала определим вспомогательные функции:

1. $r_{base}(t)$ – функция, задающая радиус-вектор центра масс основания робота в момент времени t.
2. $r_{legK}(t)$ – функция, задающая радиус-вектор конца K-ой ноги робота в момент времени t.
3. $\theta_{base}(t)$ – функция, задающая ориентацию основания робота в Эйлеровых координатах в момент времени t.
4. $\theta_{legK}(t)$ – функция, задающая ориентацию конца K -ой ноги в Эйлеровых координатах в момент времени t.

Определим траекторию робота – она задается функцией $S(t) = (p(t), v(t), c(t))$, где:

1. Функция $p(t) = (r_{base}(t), \theta_{base}(t), r_{leg1}(t), \theta_{leg1}(t), \dots r_{leg4}(t), \theta_{leg4}(t))$ задает положение ног и основание робота в момент времени t.
2. Функция $v(t) = (r'_{base}(t),\ \theta'_{base}(t),\ r'_{leg1}(t),\ \theta'_{leg1}(t),\ \dots,\ r_{leg4}(t),\ \theta_{leg4}(t))$ задает линейную и угловую скорости ног и основания робота в момент времени t.
3. Функция $c(t)$ задает бинарные признаки контакта ног робота с поверхностью.

Тогда дискретизированная траектория определяется как множество $S = \{S(t_0), S(t_1), \dots, S(t_n)\}$, где $T = \{t_0, t_1, \dots, t_n\}$ – моменты времени дискретизации траектории робота. В дальнейшем будем иметь в виду под траекторией робота – дискретизированную траекторию робота, а также обозначим $S(t_i)$, как состояние робота в момент времени $t_i \in T$.

Таким образом задача генерации походок заключается в поиске функции $F(S(t_0), \vec{v})$, принимающей на вход $S(t_0)$ – начальное состояние робота и $\vec{v}$ – вектор смещения, соединяющий начальное состояния робота и финальное состояния робота ($S(t_n)$) и выдающее $S = \{S(t_0), S(t_1), \dots, S(t_n)\}$ – траекторию робота.

Дополнительно введем ограничение на траектории робота, которые будем рассматривать в данной работе – будем рассматривать только траектории робота, при которых центр масс робота совершает плоское движение.

Так же следует учесть тот факт, что траектория движения робота не может быть произвольной, робот не должен совершать физически невозможные движения. Траектории робота должны удовлетворять динамическим, кинематическим, равновесным ограничениям и другим. Тогда сначала запишем необходимые ограничения для траекторий:

1. Траектория робота не должно противоречить динамической модели робота – модель связывает траекторию с силами создаваемыми приводами и с силами реакции опор. Запишем это ограничение для плоского движения центра масс робота в следующем виде [13]:
$$c''^{xy} = \left(c^{xy} - p_c^{xy}\right) g^c h^{-1}$$
, где $c^{xy} = [c^x, c^y]^T$ – вектор координат центра масс робота по оси Ox и Oy, c''^{xy} – вектор ускорения центра масс робота, h – текущее расстояние от основания робота до поверхности, , g^c– сила тяжести, действующая на центр масс робота, $p_c^{xy} = \left[p_c^x, p_c^y\right]^T$ – вектор координат центра давления робота по осям Ox и Oy, где центр давление робота это точка, в которой пересекаются: линия действия равнодействующей сил реакции опоры и некоторая плоскость, проведённая в роботе, а также положение центра давление робота для плоского движения центра масс робота описывает совокупный эффект действия сил реакции опоры.
2. Траектория робот должна удовлетворять условиям равновесия – точка нулевого момента должна всегда находится в пределах выпуклого многоугольника, образованного стопами робота, где точка нулевого момента – точка, где общая сила горизонтальной инерции и силы тяжести равна нулю. Запишем это ограничение для плоского движения центра масс робота в следующем виде [13]:
$$z^{xy} = c^{xy} - \frac{h}{g^z} c''^{xy}, z^{xy} \in P(p^{xy})$$
, где $z^{xy} = [z^x, z^y]^T$ – вектор координат точки нулевого момента по осям Ox и Oy, $c^{xy} = [c^x, c^y]^T$ – вектор координат центра масс робота по осям Ox и Oy, c''^{xy}– вектор ускорения центра масс робота, h – текущее расстояние от основания робота до поверхности, g^z– сила тяжести, действующая на точку нулевого момента, $P(p^{xy})$ – выпуклый многоугольник, образованный стопами робота.

3. Траектория робот должна удовлетворять кинематическим ограничениям – зона достижимости ног ограничена шаром, с центром в точке креплении ноги к роботу, а также ноги не должны пересекаться между собой [13]. Запишем это ограничение:
$$0 < \| p_i^{xy} - c^{xy}[t_k] - p_{Bnom}^{xy} \|_2 < r$$
, где r – длина выпрямленной ноги, $p_i^{xy} = \left[p_i^x, p_i^y\right]^T$ – вектор координат ноги по осям Ox и Oy на i шаге, p_{Bnom}^{xy} – вектор координат ноги в первоначальном положении, $c^{xy}[t_k] = [c^x(t_k), c^y(t_k)]^T$ – вектор координат центра масс робота по осям Ox и Oy в момент времени t_k.

Кроме описанных выше ограничений, можно задать дополнительные ограничения на траектории:

1. Условия непроскальзывания контактов – опорные ноги робота должны иметь одинаковую скорость по отношению к основанию робота.
2. Непрерывность сил, создаваемыми приводами робота, и ограничения на скорость, ускорения и амплитуды сил приводов робота.
3. Ограничения на конусы трения ступней робота.

Описание выбранного подхода синтеза походок

В данной роботе рассмотрим подход, совмещающий два уже известных подхода генерации походок робота, а именно: оптимизационного подхода и подхода, использующего нейронные сети для генерации походки.

Основная идею данного подхода можно описать следующими шагами:

1. При помощи оптимизационного метода TOWR [2] создается база траекторий робота.
2. Траектории в базе преобразуется в формат, подходящий для нейронной сети. База преобразованных траекторий разбивается на тренировочную и тестовую выборки.
3. Нейронная сеть методом обучения с учителем обучается генерировать траектории на тренировочной выборке. Качество траекторий, созданных нейронной сетью, проверяется на тестовой выборке.

Данный подход дает следующие преимущества:

1. Обучение нейронной сети с учителем происходит быстрее обучения с подкреплением.
2. Нейронная сеть сразу учится на "качественных" траекториях, полученных оптимизационным методом.
3. Производительность нейронной сети достаточно велика, чтобы изменить траекторию и пересчитать следующие шаги, не прерывая выполнения текущего шага робота.

Метод TOWR [2] был выбран по причине того, что траектории, созданные им, не только удовлетворяет описанным ранее ограничениям, но и критерию оптимальности. Этот критерий заключается в том, что ускорение центра масс должно быть минимально возможным во время движения робота. Выполнения критерия оптимальности в траекториях, созданных TOWR, достигается следующим условием, налагаемым TOWR-ом на траектории:

$$A = \sum_{j=1}^{n} \int_{t=0}^{T_j} \left(c''^{x}(t)\right)^2 + \left(c''^{y}(t)\right)^2 dt \rightarrow min$$

, где n – количество всех шагов робота, T_j – длительность j шага, $c''^{x}(t)$ и $c''^{y}(t)$ – ускорение центра масс робота по осям Ox и Oy в момент времени t. Это условие можно интерпретировать, как условие по минимизации энергетических затрат.

Создание базы траекторий робота

Рассмотрим особенности траекторий, которые могу быть сгенерированы при помощи TOWR [2], а именно:

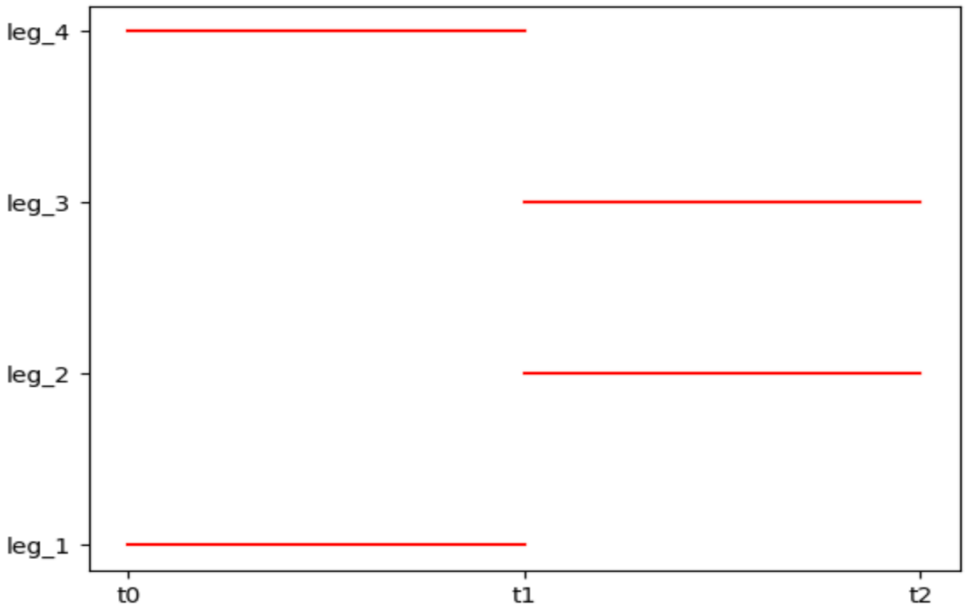
1. Траектории созданы при помощи TOWR состоят из фаз.
2. Фазы траектории – это шаблонный набор движений робота из TOWR, например:
    a. Фаза $Stand$, при которой все 4 ноги касаются поверхности движения.
    b. Фаза $Trot$, при которой робот совершает движение обеими парами противоположных ног по очереди.
    c. Существует другие фазы ($Walk1$, $Walk2$ и т.д.), но они не используется в этой работе.
3. TOWR настроен так, чтобы каждая фаза $Trot$ состояла из одинакового количества состояний робота (20 штук).

4. TOWR генерирует траектории робота в специальном режиме, таким образом эти траектории состоят из последовательности фаз: $Stand_{start}$, $Trot_1$, $Trot_2$, $Trot_3$, $Trot_4$, $Stand_{end}$.

На рисунке №1 можно увидеть диаграмму походки [14] во время фазы $Trot$. На этом рисунке на оси Ox отображены моменты времени, а на оси Oy отображены ноги робота. На этой диаграмме отмечается красным точка, если соответствующая ей нога в соответствующий момент времени перемещается в пространстве. Более подробно: во временном промежутке $[t0; t1]$ происходит перемещение только 1 и 4 н-оги, а во временном промеж $[t1; t2]$ происходит перемещение только 2 и 3 ноги.

При помощи TOWR была создана база траекторий, состоящая из 1000 траекторий. Визуализацию одной такой траектории можно увидеть на рисунках №2, №3, а именно можно увидеть график зависимости координат x и z первой ноги робота от времени, полученных из траектории, созданной TOWR. Аналогично на рисунке № 4 можно увидеть график зависимости координаты x основания робота от времени, полученных из траектории, созданной TOWR.

Рисунок №1 – Диаграмма походки ног во время фазы $Trot$



Формирование выборок и преобразования над траекториями

База траекторий была разделена на обучающую и тестовую выборки размером 800 и 200 траекторий.

Дополнительно были проведены следующие преобразования над траекториями в выборках:

1. У каждой траектории были отброшены состояния робота, соответствующие фазам $Stand$.
2. Каждая траектория разбита на 4 части – фазы траектории, соответствующей каждой фазе $Trot$ и состоящей из 20 состояний робота.

Заметим, что траектория, сгенерированная TOWR [2], всегда состоит из постоянного количества однотипных фаз. Такая природа траекторий позволяет нам упростить задачу генерации походок, а именно предсказывать каждую фазу траектории отдельно, а после объединять эти фазы в одну траекторию. Для того, чтобы иметь возможность решать задачу генерации походки в таком виде, разобьем каждую фазу на следующие части:

1. Состояние робота $S(t_{i_0})$ на начало фазы $Trot$.
2. Оставшиеся 19 состояний $\{S(t_{i_1}), S(t_{i_2}), \dots, S(t_{i_{18}})\}$ робота этой же фазы $Trot$.

Рисунок №2 – График зависимости координаты x первой ноги робота от времени, полученных из траектории, созданной TOWR.

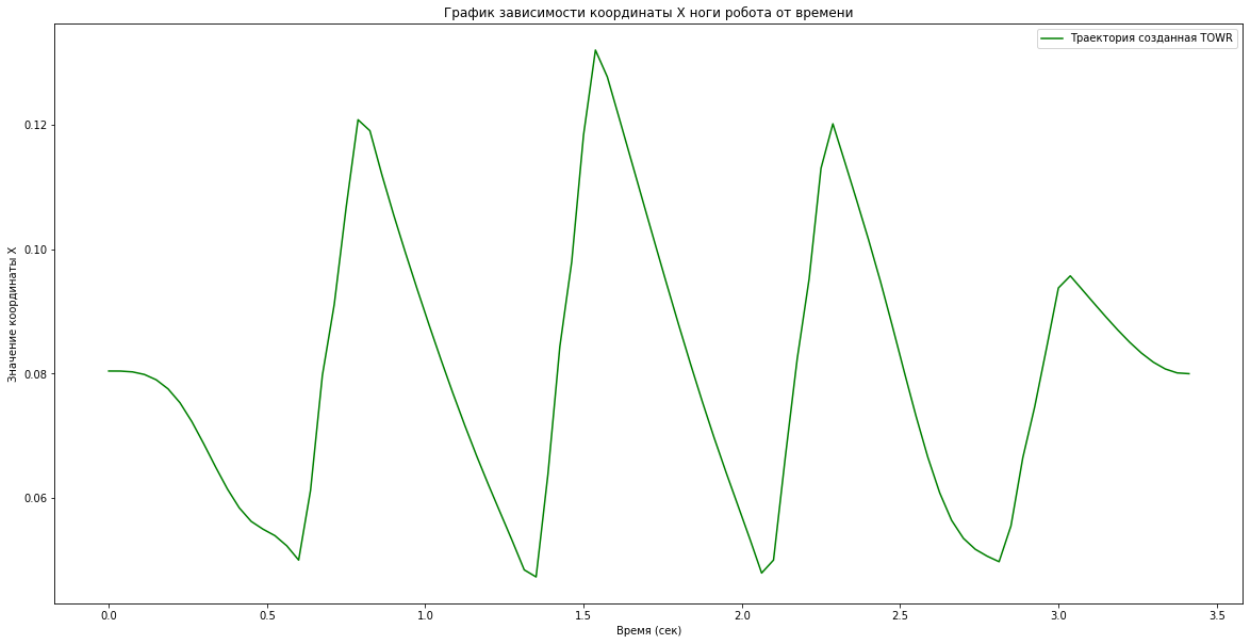


Рисунок №3 – График зависимости координаты z первой ноги робота от времени, полученных из траектории, созданной TOWR.

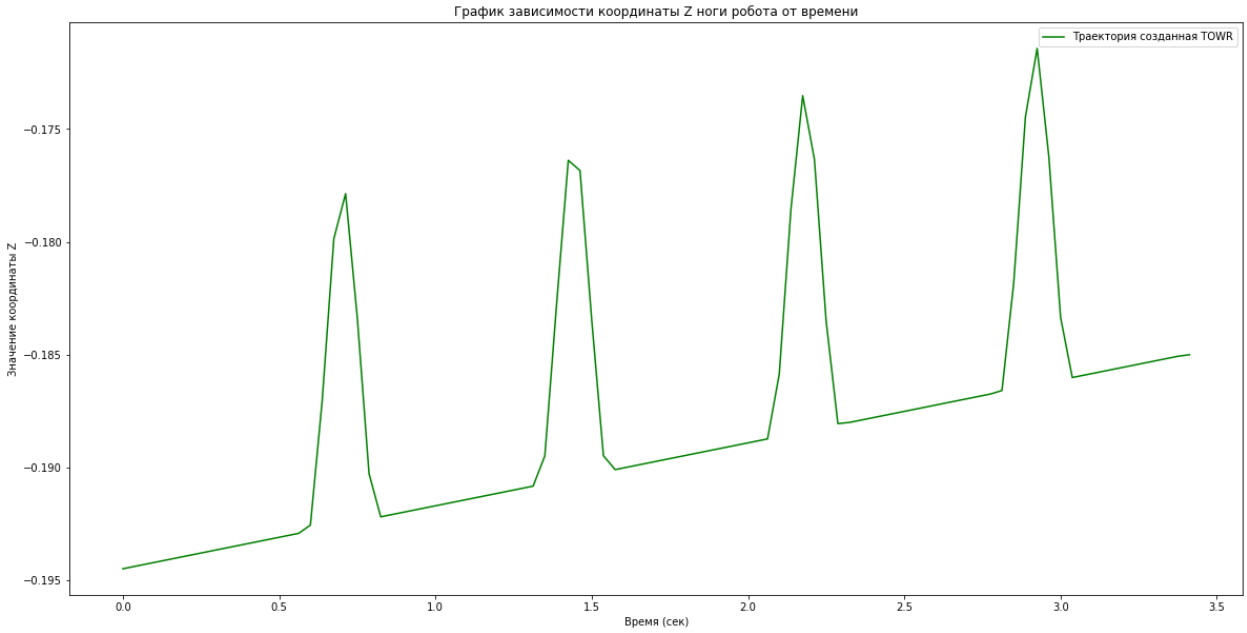
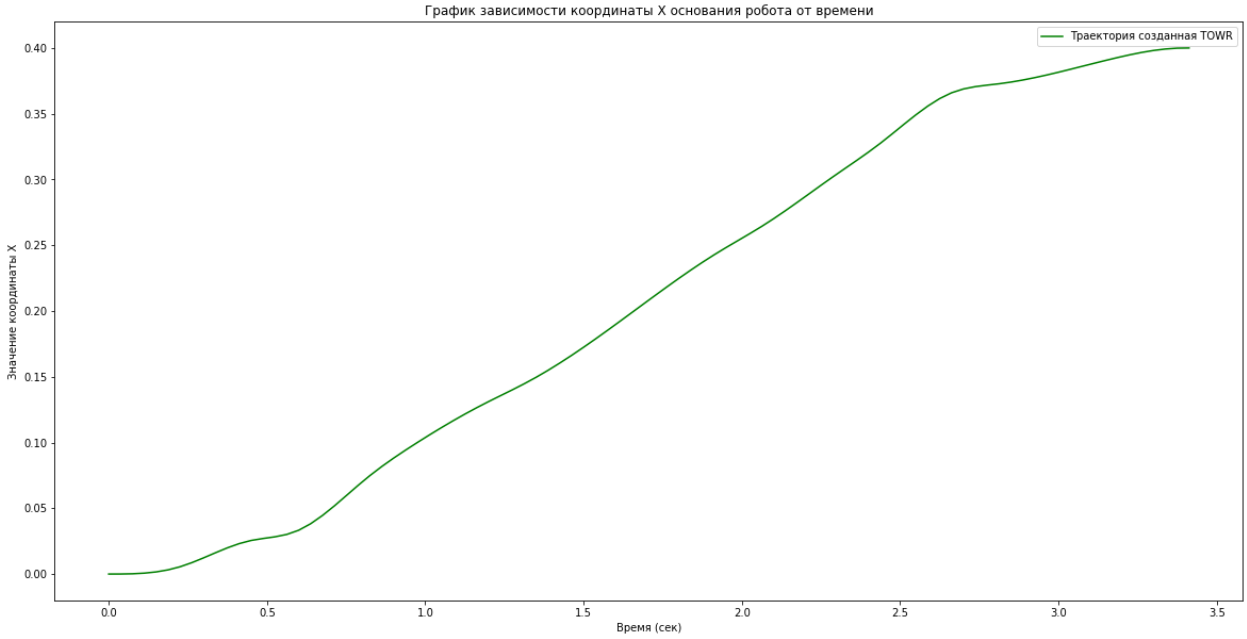


Рисунок №4 – График зависимости координаты x основания робота от времени, полученных из траектории, созданной TOWR.



Обзор архитектур нейронных сетей

Задачу генерации походок можно представить в виде задачи генерации последовательности векторов, представляющих положения робота после начала движения, по входному вектору, описывающему начальное положение робота. Для решения таких задач существует следующие архитектуры нейронных сетей: полносвязные нейронные сети, рекуррентные нейронные сети и трансформеры [10]. В данной роботе будут рассматриваться только

полносвязные и рекуррентные сети (их рассмотрим подробнее дальше). Хоть архитектура трансформеров является на данный момент лучшей в большинстве задач машинного обучения, но рассматриваться не будет по причине большой вычислительной затратности обучения таких сетей и необходимости создания очень больших обучающих выборок для достижения удовлетворительного качества предсказаний сети.

Полносвязные нейронные сети

Перед тем как определить полносвязную нейронную сеть введем несколько вспомогательных понятий:

1. Обозначим линейным слоем линейное преобразование над входящими данными вида
$$F(x)_{lin} = Wx + b$$
, где $x \in \mathbb{R}^n$ – вектор входящих данных, $W \in \mathbb{R}^{n \times m}$ и $b \in \mathbb{R}^m$ – обучаемые параметры линейного слоя.
2. Обозначим активационным слоем нелинейную дифференцируемую функцию применяемую поэлементно к входным данным. На данный момент самыми популярными активационными функциями являются $ReLU(x) = \max(0, x)$, $Sigmoid(x) = \frac{1}{1+e^{-x}}$ и $Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.

Определим полносвязную нейронную сеть как комбинацию линейных и активационных слоев, в которой присутствует как минимум один линейный слой и как минимум один активационный слой, находящийся после линейного слоя.

Пусть полносвязная нейронная сеть имеет вид $F(x) = F\big(Sigmoid(F(x)_{lin_2})\big)_{lin_1} = W_2(Sigmoid(W_1 x + b_1)) + b_2$, тогда по теореме Цыбенко[9] для любой непрерывной функции действительных переменных f на $[0,1]^n$ (или на другом компактном подмножестве $\mathbb{R}^n$) и для $\forall \varepsilon > 0$ существую такие параметры линейных слоев сети W_1, W_2, b_1, b_2, что выполняется: $|F(x) - f(x)| < \varepsilon$ или иными словами по теореме Цыбенко[9] полносвязная нейронная сеть является универсальным аппроксиматором.

Рекуррентные нейронные сети

Рекуррентные нейронные сети – вид нейронных сетей для обработки последовательности входных данных, где соединения между слоями образуют циклы, позволяя выходу нейронной сети для предыдущего вектора последовательности входных данных влиять на выход нейронной сети для

текущего вектора последовательности входных данных. Основная идея рекуррентной нейронной сети заключается в том, что сеть агрегирует информацию из уже обработанной последовательности входных данных в вектор контекста. На основании вектора контекста и текущего вектора входных данных рекуррентная нейронная сеть выдает новое предсказание.

В этой работе мы будим рассматривать только один вид архитектур рекуррентных нейронных сетей, а именно Long-short term memory или LSTM [5] (рисунок №5). LSTM специально разработана, чтобы решать главную проблему рекуррентных сетей, а именно проблему долговременных зависимостей.

Проблему долговременных зависимостей можно описать так: пусть у нас есть достаточно большая последовательность входных данных. Рекуррентная нейронная сеть агрегирует информацию из этой последовательности в вектор контекста. Чем больше входная последовательность, тем меньше вклад первых векторов последовательности в вектор контекста, иными словами сеть “хорошо помнит” новые входные вектора и “плохо помнит” старые. Предположим, что до предсказания нового выхода сети зависят от первого и последнего вектора входных данных, но из-за того, что сеть “плохо помнит” первый вектор входных данных, то сеть уже не может “качественно” предсказать новый выход и чем больше последовательность входных данных, тем менее “качественны” новые предсказания сети.

Основная идея LSTM позволяющая решать (к сожалению, только частично) проблему долговременных зависимостей – наличие двух векторов контекста. В один вектор контекста (вектор кратковременной памяти) сеть агрегирует информация из нескольких последних векторов входных данных, при помощи этого вектора контекста сеть “хорошо помнит” последнии вектора входа, но также “быстро забывает” их при получении новых входных векторов. Во второй вектор контекста (вектор долговременной памяти) сеть уже агрегирует информацию из вектора кратковременной памяти таким образом, чтобы сеть “хорошо помнила” как можно больше входных векторов.

Более формально архитектуру LSTM можно описать следующими способом: пусть вектора $\{x_0, x_1, \dots, x_n\}$ – вектора входных данных, вектора h_i, c_i – вектора кратковременной и долговременной памяти на i-ом шаге. Введем вспомогательные функции:

$$f(x_t) = Sigmoid\left(W_f x_t + U_f h_t + b_f\right)$$

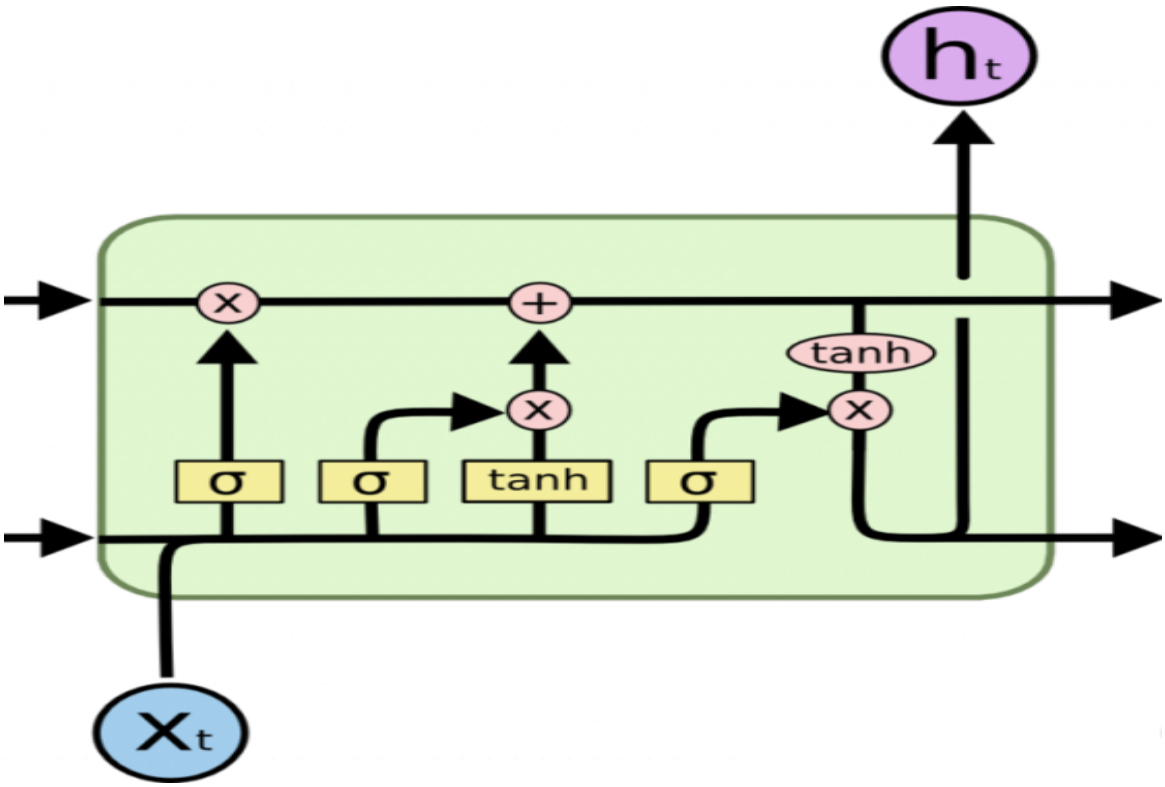
$$g(x_t) = Sigmoid\left(W_g x_t + U_g h_t + b_g\right)$$
$$o(x_t) = Sigmoid(W_o x_t + U_o h_t + b_o)$$

, где $W_f, W_g, W_o, h_t, h_g, h_o, b_f, b_g, b_o$ – обучаемые параметры LSTM. Теперь определим формулы векторов контекста следующим образом:

$$c_t = f(x_t) \circ c_{t-1} + g(x_t) \circ Tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = g(t) \circ Tanh(c_t)$$

, где W_c, U_c, b_c – обучаемые параметры LSTM, и $\circ$ – произведение Адамара.

Рисунок №5 – Архитектура LSTM



Эксперименты по генерации траекторий

Используя информацию о траекториях робота и об архитектурах нейронных сетей, приступим к описанию экспериментов.

Сначала, в эксперименте №1, при помощи архитектуры LSTM [5] будем обучать модель предсказывать полную траекторию движения робота опираясь на положение робота и вектора смещения $\vec{v}$, соединяющего начальное положение робота и финальное положение робота.

Обученная модель в эксперименте №1 не дала удовлетворительных результатов на тестовой выборке, и траектории, созданные этой моделью, оказались непригодными для исполнения на роботе. Точное описание эксперимента находится в подпункте “Эксперимент №1. Генерация полной траектории робота”, точное описание результатов находится в таблице №1.

Из-за неудачи в эксперименте №1, воспользуемся структурой траекторий, созданных TOWR [2] и будем обучать модель на основе архитектуры полносвязной нейронной сети предсказывать лишь фазу движения робота опираясь на положение робота и вектора смещения $\vec{v}$ в эксперименте №2. Полученные фазы движения будем объединять в одну траекторию робота.

Обученная модель в эксперименте №2 дала уже заметно лучший результат на тестовой выборке, чем модель из эксперимента №1, но все же траектории, созданные моделью из эксперимента №2, оказались не пригодны для исполнения на работе. Точное описание эксперимента находится в подпункте “Эксперимент №2. Генерация фазы движения робота”, точное описание результатов находится в таблице №1.

Из-за неудачи в эксперименте №2, модифицируем этот эксперимент и будем обучать 5 моделей на основе архитектуры полносвязной нейронной сети предсказывать траектории движения каждой ноги и основания робота для фазы движения робота опираясь на положение робота и вектора смещения $\vec{v}$. Полученные траектории ног и основания робота будем объединять в фазу движения, а полученные фазы будем объединять в траекторию робота.

Обученные модели в эксперименте №3 дали хороший результат на тестовой выборке. Эти модели могут генерировать исполнимые траектории роботов. Точное описание эксперимента находится в подпункте “Эксперимент №3. Генерация траекторий ног и основания робота в фазе движения робота”, точное описание результатов находится в таблице №1.

В следующих подпунктах приводится полное описание всех экспериментов, так же на рисунках №5 и №6 можно наблюдать значения целевой функции и метрики оценки качества для всех экспериментов соответственно.

Эксперимент №1. Генерация полной траектории робота

1. В качестве архитектуры модели выбрана LSTM.

2. Модель LSTM состоит из 5 слоев (каждый слой LSTM [5] получает на вход токен из предыдущего слоя LSTM). Размер векторов контекста LSTM равен 256. Модель имеет стандартную Xavier [6] инициализацию.
3. В качестве целевой функции выбрана взвешенная средняя абсолютная ошибка (WMAE) между состоянием робота, которое предсказано сетью, и состоянием робота из сгенерированной траектории. Целевая функциця и метрика оценки качества совпадают.
4. Обучение происходит 12 эпохи. Оптимизатор – Adam [7] и LR = 0,003. Размер батча выбран batch_size = 32.
5. На вход модели подается:
    a. Начальное состояния робота $S(t_0)$ траектории робота из обучающей выборки.
    b. Вектор смещения $\vec{v}$, соединяющий начальное состояния робота и финальное состояния робота.
6. Модель обучается предсказывать оставшиеся состояния робота траектории.

Эксперимент №2. Генерация фазы движения робота

1. В качестве архитектуры модели выбрана полносвязная нейронная сеть.
2. Модель состоит из 2 полносвязных слоев и одного скрытого слоя размерностью 1024. Модель имеет стандартную Kaiming [8] инициализацию.
3. В качестве целевой функции выбрана взвешенная средняя абсолютная ошибка (WMAE) между состоянием робота, которое предсказано сетью, и состоянием робота из сгенерированной траектории. Целевая функция и метрика оценки качества совпадают.
4. Обучение происходит 64 эпохи. Оптимизатор – Adam [7] и LR = 0,003. Размер батча выбран batch_size = 32.
5. На вход модели подается:
    a. Состояние робота $S(t_{f_0})$ в начале фазы траектории.
    b. Вектор смещения $\vec{v}$, соединяющий начальное состояния робота и финальное состояния робота.
6. Модель обучается предсказывать всю фазу траектории.
7. Траектория робота составляется из 4 фаз траекторий, предсказанных сетью.

Эксперимент №3. Генерация траекторий ног и основания робота в фазе движения робота

1. В качестве архитектуры модели выбрана полносвязная нейронная сеть.
2. Будем использовать 5 отдельных моделей для предсказания положения основания робота и каждой ноги соответственно.
3. Каждая модель состоит из 3 полносвязных слоев и 2 скрытых слоев размерностью 1024. Каждая модель имеет стандартную Kaiming [8] инициализацию.
4. В качестве целевой функции выбрана взвешенная средняя абсолютная ошибка (WMAE) между состоянием робота, которое предсказано сетью, и состоянием робота из сгенерированной траектории. Целевая функция и метрика оценки качества совпадают.
5. Обучение происходит 64 эпохи. Оптимизатор – Adam [7] и LR = 0,003. Размер батча выбран batch_size = 32.
6. На вход модели подается:
    a. Состояние робота $S(t_{f_0})$ в начале фазы траектории.
    b. Вектор смещения $\vec{v}$, соединяющий начальное состояния робота и финальное состояния робота.
7. Модели обучаются предсказывать движение основания робота или ног для всей этой фазы траектории.
8. Фаза движения составляется из предсказаний 5 моделей. Траектория робота составляется из 4 фаз траекторий.

Рисунок №5 – Значение целевой функции для всех экспериментов

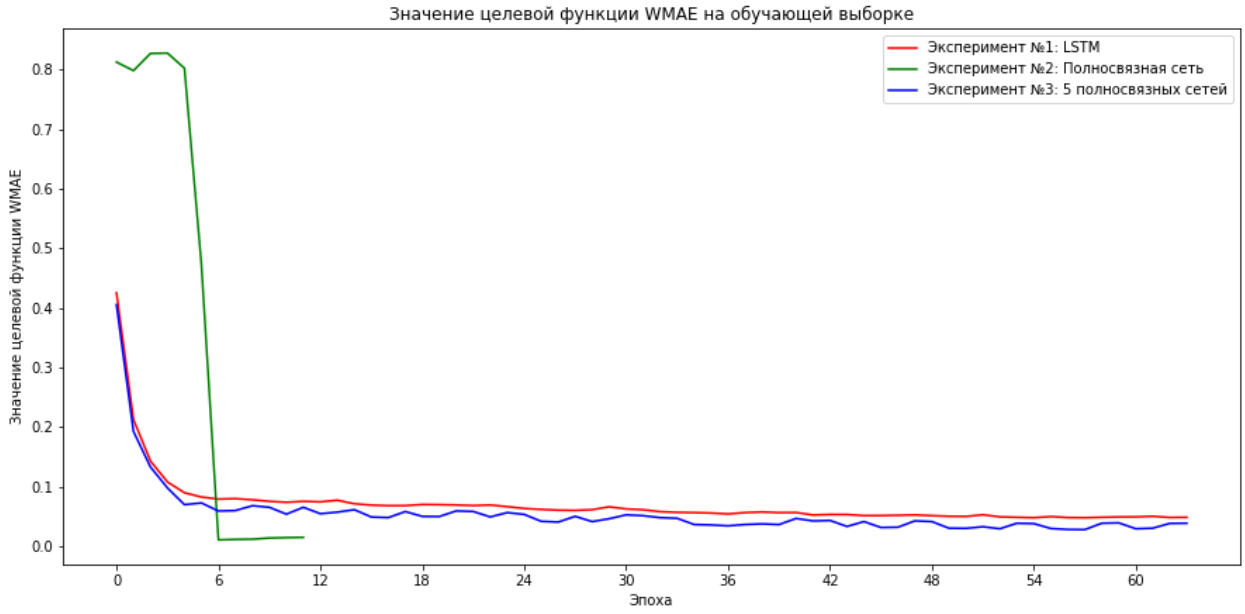
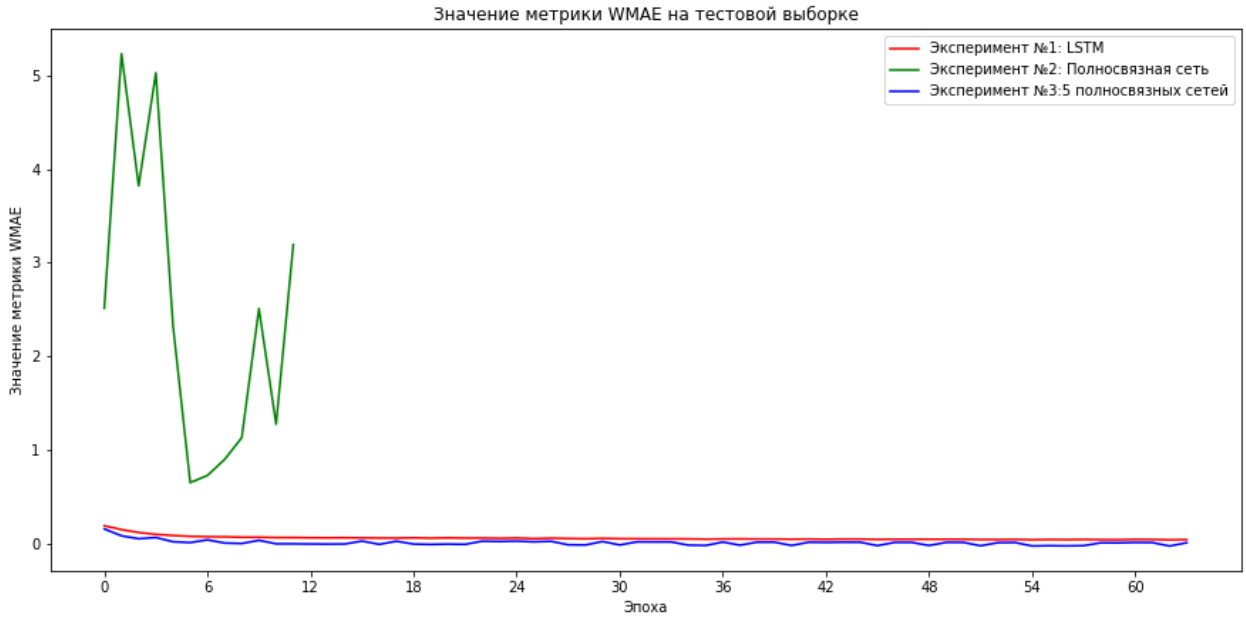


Рисунок №6 – Значение метрики оценки качества для всех экспериментов



Результаты экспериментов

В таблице №1 мы можем наблюдать результаты экспериментов. Заметим, что:

1. Хоть модель в эксперименте №2 дала хороший результат метрики оценки качества на тестовой выборке, но этого оказалось недостаточно, чтобы сгенерированные ее траектории можно было исполнять роботом.
2. Модели в эксперименте №3 дали хороший результат метрики оценки качества на тестовой выборке и траектории, сгенерированные ими, оказались пригодными для исполнения на роботе.
3. Модель LSTM [5] в эксперименте №1 не смогла добиться результатов сравнимых с результатами экспериментов №2 и №3 предположительно из-за слишком большой длины траектории.
4. Эксперименты показали, что нейронные сети работают заметно быстрее, чем оптимизационные методы.

Также на рисунках №7, №8, №9 можно визуально наблюдать отличия траектории, созданной моделями из эксперимента №3, и траектории, созданной TOWR, а именно на рисунке №7 и №8 можно увидеть график зависимости координат x и z первой ноги робота от времени, полученных из траектории, а на рисунке №9 можно увидеть график зависимости координаты x основания робота от времени, полученных из траектории (робот находится в центре координат и перемещается в точку, сдвинутую

от центра координат по оси Ox, на одинаковое расстояние в обеих траекториях).

Таблица №1 – результаты экспериментов

	WMAE на тестовой выборке	Пригодна ли траектория к исполнению на роботе	Время генерации 1–ой траектории (на Intel i9-9980HK, на 1 ядре)
Towr[2]	0	+	264 мс
Эксперимент №1	0.647	-	12.46 мс
Эксперимент №2	0.048	-	0.57 мс
Эксперимент №3	0.012	+	1.28 мс

Рисунок №7 – График зависимости координаты x первой ноги робота от времени, полученных из траекторий, созданной TOWR и моделями из эксперимента №3.

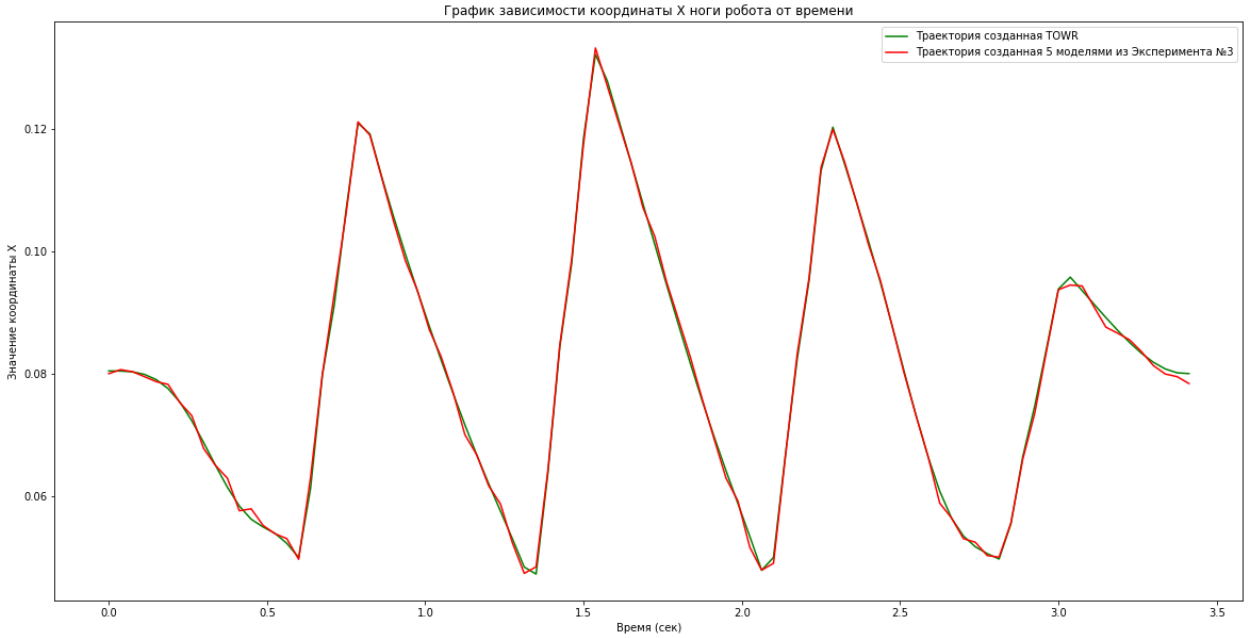


Рисунок №8 – График зависимости координаты x первой ноги робота от времени, полученных из траекторий, созданной TOWR и моделями из эксперимента №3.

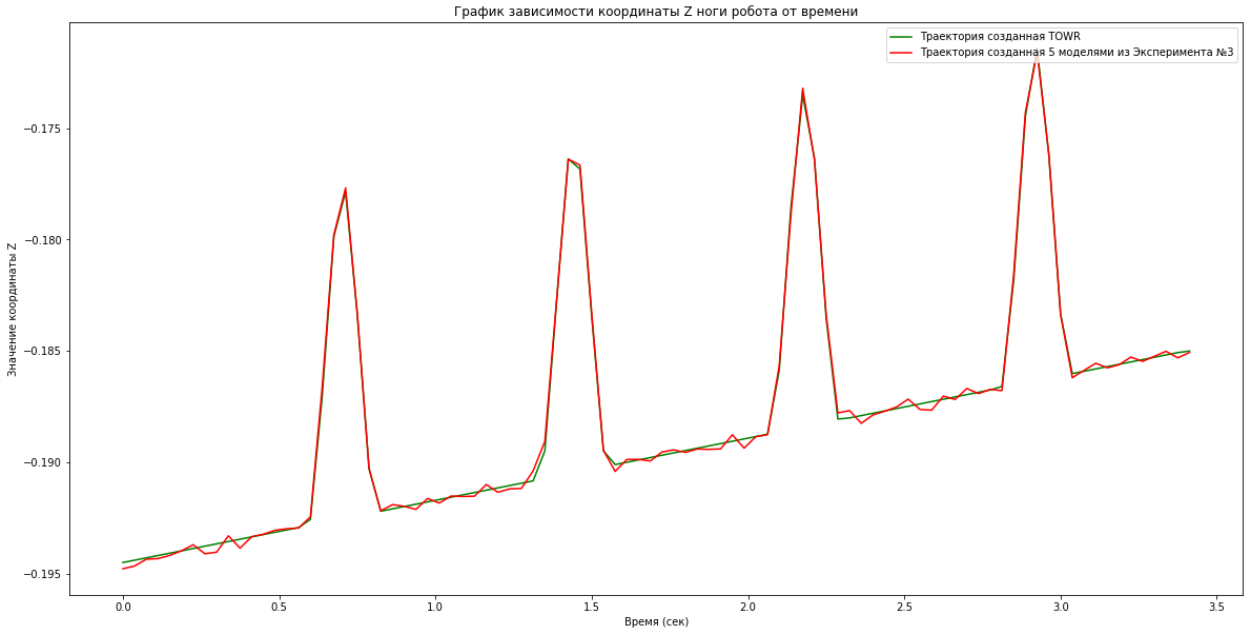
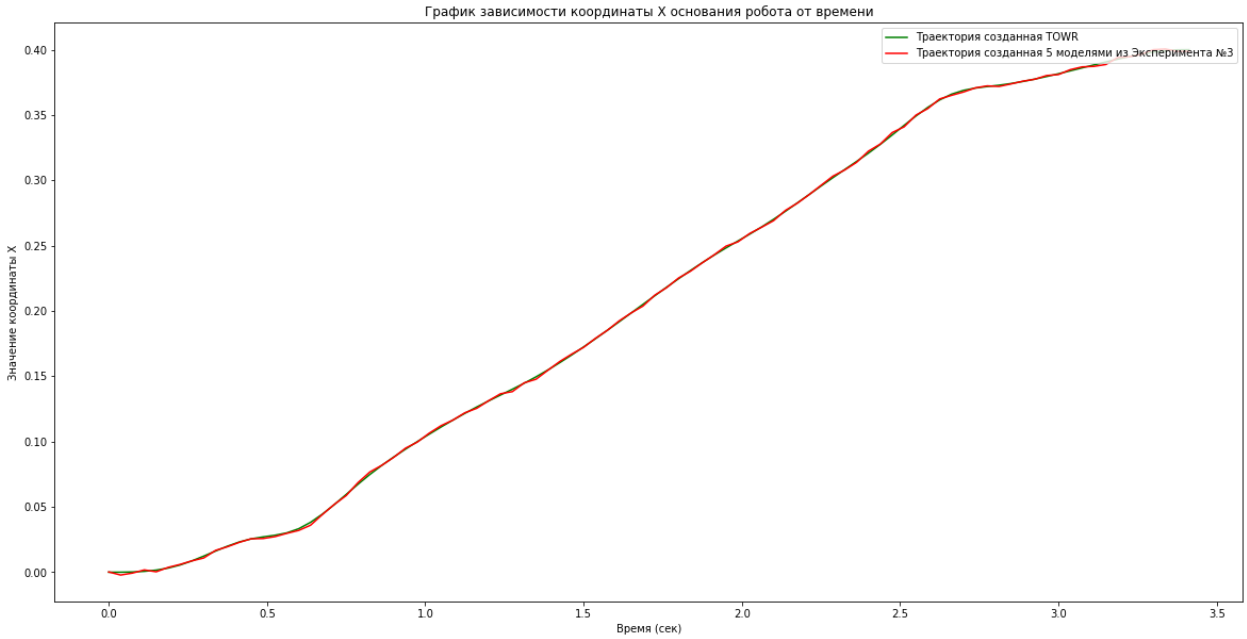


Рисунок №9 – График зависимости координаты x основания робота от времени, полученных из траекторий, созданной TOWR и моделями из эксперимента №3.



Выводы

В результате выполнения работы были рассмотрены уже существующие методы решения задачи генерации походки, а также был предложен новый метод, совмещающий оптимизационный и нейросетевой метод. Были проведены эксперименты по обучению нейронных сетей (полносвязных нейронных сетей и сетей архитектуры LSTM [5]) методом обучения с учителем, эти сети были обучены, созданных при помощи оптимизационного метода TOWR [2]. Обученные сети сравнивались между собой с и методом TOWR по времени создания новых траекторий, а также на сколько созданные ими траектории отличаются от траекторий TOWR по метрике WMAE.

В ходе выполнения работы были сделаны следующие выводы:

1. **Лучший результат дал эксперимент №3 с 5 полносвязными моделями**, которые отдельно предсказывали движения основания робота и ног робота соответственно. Данный подход позволяет создавать траектории **очень близкие** к траекториям TOWR.
2. Подход совмещения оптимизационных методов и методов, основанных на использовании нейронных сетей добился ускорения по генерации походок более чем в **200 раза**.

3. Простая полносвязная сеть достигает **лучших** результатов чем более сложная сеть LSTM, но для этого необходимо, чтобы природа данных позволяла переформулировать задачу в более простом виде.

Список литературы

1. Lee J. Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control: implementation on the MIT Cheetah robot. // Thesis: S.M., Massachusetts Institute of Technology, Department of Mechanical Engineering, 2013.
2. Winkler A. W. Bellicoso C. D. Marco H. Buchli J. Gait and Trajectory Optimization for Legged Systems through Phasebased End-Effector Parameterization. // IEEE ROBOTICS AND AUTOMATION LETTERS, Preprint version, 2018.
3. Di Carlo J. Wensing P. Katz E. Bledt G. Kim S. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. // 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), preprint version, 2018.
4. Tsounis V. Alge M. Lee J. Farshidian F. Hutter M. DeepGait: Planning and Control of Quadrupedal Gaits using Deep Reinforcement Learning. // IEEE ROBOTICS AND AUTOMATION LETTERS, preprint version, 2020.
5. Sak H. Senior A. Beaufays F. Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. // http://imat-relpred.yandex.ru.
6. Xavier G. Yoshua B. Understanding the difficulty of training deep feedforward neural networks. // DIRO, Universit de Montreal, Montreal, Quebec, Canada. Preprint version. 2014.
7. Kingma D. Ba J. Adam: A Method for Stochastic Optimization. // Published as a conference paper at the 3rd International Conference for Learning Representations. San Diego. 2015.
8. Kaiming H. Xiangyu Z. Shaoqing R. Jian S. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. // https://arxiv.org/abs/1502.01852.
9. Cybenko, G. V. Approximation by Superpositions of a Sigmoidal function // Mathematics of Control Signals and Systems. 1989.
10. Vaswani A. Shazeer N. Parmar N. Uszkoreit J. Jones L. Gomez A. N. Kaiser L. Polosukhin I. “Attention Is All You Need” // Conference on Neural Information Processing Systems, preprint version, 2017.
11. Cognetti M. Fioretti V. Oriolo G. Whole-Body Planning for Humanoids along Deformable Tasks. // IEEE International Conference on Robotics and Automation (ICRA). Stockholm, Sweden. 2016.
12. Phaniteja S. Dewangan P. Guhan P. Sarkar A. Krishna K. M. A Deep Reinforcement Learning Approach for Dynamically Stable Inverse Kinematics of Humanoid Robots // https://arxiv.org/abs/1801.10425

13. Winkler A.W. Farshidian F. Neunert M. Pardo D. Buchli J. Online Walking Motion and Foothold Optimization for Quadruped Locomotion // IEEE International Conference on Robotics and Automation (ICRA). 2017.
14. Siciliano B. Khatib O. Springer Handbook of Robotics. // Library of Congress. 2008.

Приложение А

Доступ к коду можно получить по ссылке https://github.com/pansershrek/RosGateGenerator. Код состоит из скриптов, написанных на языках Python3.9 и Bash, также был использован фреймворк pytorch для реализации нейронных сетей. Для запуска необходимо операционная система macOS или Linux. Чтобы программа запустилась необходимо установить указанные в requirements.txt библиотеки.

Код состоит из четырех частей:

1. Утилита для запуска симуляции робота (setup.sh).
2. Утилита для генерации траекторий робота при помощи TOWR [2] (run_trajectory_generator.sh).
3. Утилита для запуска исполнения траекторий робота на симуляции робота (run_trajectory_executor.sh).
4. Утилита для обучения нейронной сети и создания траектории при помощи нейронной сети на основании положения робота и вектора смещения $\vec{v}$ (run_main.sh).

Утилита для запуска робота представляет из себя Bash скрипт, который настраивает окружение симуляции робота и запускает ее. Код данного скрипта находится в папке ros_experiments и имеет имя setup.sh.

Утилита для генерации траекторий робота при помощи TOWR представляет из себя Bash обертку на Python скриптами. Эта утилита приминает на вход количество траекторий, которых нужно создать, и путь к папке, в которую нужно записать траектории. Данная утилита работает корректно только при запущенной симуляции робота. Ее код находится в папке ros_experiments и имеет имя run_trajectory_generator.sh.

Утилита для запуска исполнения траекторий робота на симуляции робота представляет из себя Bash обертку на Python скриптами. Эта утилита принимает на вход путь к траектории, которую нужно исполнить. Данная утилита работает корректно только при запущенной симуляции робота. Ее код находится в папке ros_experiments и имеет имя run_trajectory_executor.sh.

Утилита для обучения нейронной сети и создания траектории при помощи нейронной сети на основании положения робота и вектора смещения $\vec{v}$ представляет из себя Bash обертку на Python скриптами. Эта утилита считывает информацию из конфигурационного файла config.json и запускает обучение нейронной сети или создает траекторию робота в зависимости от значения параметра MODE Для запуска обучения параметр MODE должен

иметь значение TRAIN, а для запуска генерации траектории должен иметь значение INFERENCE. Код этой утилиты лежит в папке main и имеет имя run_main.sh.