

Санкт-Петербургский государственный университет

Кафедра прикладной кибернетики

Группа 24.Б82-мм

Алгоритмы тікмік

Черняков Евгений Олегович

Отчёт по учебной практике
в форме «Сравнение»

Научный руководитель:
профессор, научный сотрудник кафедры прикладной кибернетики Мокаев Р. Н.

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	5
2. Обзор	6
2.1. Обзор существующих решений	6
2.2. Обзор используемых технологий	9
2.3. Выводы	11
3. Описание решения	13
3.1. Первая задача	14
3.2. Вторая задача	20
3.3. Третья задача	20
3.4. Некоторые типичные ошибки	20
3.5. Листинги, картинка и прочий «не текст»	22
3.6. Некоторые детали описания реализации	23
Список литературы	27

Введение

Задача стабилизации перевернутого маятника на подвижной тележке (Cart-Pole) является классическим эталоном в теории автоматического управления и обучении с подкреплением. Она моделирует неустойчивые нелинейные системы, принципы управления которыми лежат в основе работы балансирующих роботов и ракет-носителей на старте. Традиционные методы решения, такие как ПИД-регуляторы или LQR-алгоритмы, требуют точного знания математической модели объекта. Однако с развитием искусственного интеллекта всё большую популярность приобретают методы, способные обучаться управлению через взаимодействие со средой, не требуя явного задания уравнений динамики. Существующие исследования широко освещают применение глубокого обучения с подкреплением (Deep RL) для этой задачи, демонстрируя возможности сложных нейросетевых архитектур.

В то же время, в учебной и технической литературе часто наблюдается пробел в сравнительном анализе эффективности базовых стохастических методов оптимизации по сравнению с градиентными методами начального уровня. Нередко сложные алгоритмы применяются там, где достаточно простейшего случайного поиска. Остается открытym вопросом: оправдано ли усложнение алгоритма (переход от случайногo поиска к градиентной политике) для системы с малым количеством степеней свободы, и как именно соотносятся затраты на вычисления и качество стабилизации при реализации физической модели «с нуля», без использования готовых библиотек симуляции.

Целью данной работы является выявление сравнительной эффективности алгоритмов различной природы — базового случайного поиска (Basic Random Search), метода восхождения на холм (Hill Climbing) и градиентной политики (Policy Gradient) — в задаче балансировки стержня. Необходимо определить, какой из подходов обеспечивает наилучший баланс между скоростью обучения и устойчивостью удержания маятника. Работа призвана продемонстрировать, что для определенных классов задач простые безградиентные методы могут составлять

конкуренцию более сложным подходам RL, а также показать влияние точности физической симуляции на процесс обучения агентов.

Для реализации поставленной задачи в работе разрабатывается программная модель динамической системы, основанная на численном решении дифференциальных уравнений движения стержня и тележки. В данную среду интегрируются и настраиваются алгоритмы Basic Random Search, Hill Climbing и Policy Gradient. В ходе экспериментов проводится обучение агентов с фиксацией ключевых метрик: времени сходимости алгоритма и длительности удержания стержня в вертикальном положении. На основе полученных данных будет проведен сравнительный анализ и сформулированы выводы о применимости каждого из методов для задач стабилизации неустойчивых динамических систем.

1. Постановка задачи

Целью работы является проведение сравнительного анализа эффективности алгоритмов стохастической оптимизации и методов обучения с подкреплением при решении задачи стабилизации стержня на подвижном основании. Для её выполнения были поставлены следующие задачи:

1. математически описать динамику системы «тележка-стержень», составив дифференциальные уравнения движения;
2. создать программную среду, эмулирующую поведение системы во времени на основе разработанной математической модели, без использования сторонних физических движков;
3. программно реализовать алгоритмы: Basic Random Search, Hill Climbing, Policy Gradient;
4. провести серию экспериментов по обучению агентов каждым из алгоритмов, зафиксировав метрики скорости сходимости и длительности удержания стержня в вертикальном положении;
5. сравнить полученные данные, выявить преимущества и недостатки каждого алгоритма и оценить целесообразность использования градиентных методов для задач данной размерности.

2. Обзор

Целью данного раздела является формирование необходимого теоретического фундамента для понимания работы и обоснование методологического выбора, сделанного в рамках исследования. Обзор систематизирует известную информацию о механической системе «тележка-стержень», существующих подходах к ее стабилизации и принципах работы выбранных алгоритмов обучения.

Обзор был выполнен с использованием следующих методов:

- Систематический поиск: Использовались научные базы данных (например, Google Scholar, Scopus, ResearchGate) по ключевым запросам, охватывающим как классическую теорию управления, так и современное обучение с подкреплением (например, «Cart-Pole stabilization», «Inverted Pendulum control», «Policy Gradient vs Random Search»).
- Оценка релевантности: В фокусе внимания находились статьи и монографии, описывающие математическую модель системы с распределенной массой (стержень), а также работы, напрямую сравнивающие эффективность стохастической оптимизации (Random Search, Hill Climbing) с градиентными методами.
- Идентификация технологий: Изучались принципы работы выбранных алгоритмов (PG, HC, BRS) и современные методы их реализации на языке Python.

2.1. Обзор существующих решений

Обзор существующих решений направлен на систематизацию научного и инженерного опыта, накопленного в области стабилизации неустойчивых динамических систем, и, в частности, механической системы «тележка-стержень». Это позволяет избежать дублирования уже известных подходов и четко определить место данной работы среди предыдущих исследований.

2.1.1. Математическая модель «Тележка-Стержень»

Система «тележка-стержень» является классическим объектом исследования в теории автоматического управления (ТАУ) и представляет собой идеальный пример нелинейной, существенно неустойчивой системы. Математическое описание основывается на законах Ньютона или уравнениях Лагранжа. В большинстве ранних работ используется модель, где масса стержня считается точечной, однако для точной симуляции требуется модель с учетом распределенной массы (стержня), что усложняет расчеты, но повышает точность реализации.

2.1.2. Классические подходы к стабилизации

Исторически задача решалась с помощью методов, основанных на глубоком знании математической модели:

- 1. Линеаризация и LQR-регулирование (Linear Quadratic Regulator).** Этот подход является золотым стандартом в ТАУ. Преимущество LQR заключается в том, что он гарантирует оптимальность управления (минимизацию заданного квадратичного функционала) для линеаризованной системы, обеспечивая высокую скорость реакции вблизи точки равновесия. Недостаток метода в том, что он требует точной линеаризации уравнений вокруг вертикального положения и может стать неэффективным при больших начальных отклонениях стержня.
- 2. ПИД-регулирование (PID Control).** Преимуществом ПИД-регулятора является его простота и универсальность. Он не требует полного знания уравнений движения. Недостаток заключается в сложности подбора трех коэффициентов (K_p, K_i, K_d) для достижения устойчивости в широком диапазоне состояний, особенно в нелинейных зонах.
- 3. Управление на основе фазового пространства.** Некоторые решения используют переключение между различными линейными стратегиями в зависимости от текущего состояния (угла и уг-

ловой скорости), но их разработка трудоемка и слабо масштабируется на более сложные робототехнические системы.

2.1.3. Современные подходы на основе обучения с подкреплением (RL)

С развитием вычислительных мощностей и нейронных сетей методы RL стали доминирующими в задачах, требующих адаптивного управления.

1. **Value-Based методы (DQN, Q-learning).** Эти алгоритмы эффективны для дискретного пространства действий, что применимо к задаче, если управляющую силу $F(t)$ дискретизировать (например, $F \in \{-10, 0, 10\}$). Однако их сложно масштабировать, и они требуют аппроксимации функции ценности, что часто приводит к нестабильности обучения.
2. **Policy-Based методы (Policy Gradient, Actor-Critic, PPO).** Эти методы непосредственно оптимизируют политику π , что делает их идеальными для задач с непрерывным пространством действий и состояний. Преимущество Policy Gradient (PG), используемого в данной работе, заключается в его фундаментальности, позволяя агенту обучаться, лишь наблюдая за результатами своих действий (наградой), без явного моделирования среды. Недостаток PG — высокая дисперсия градиента, требующая большого числа выборок для стабилизации.

2.1.4. Обоснование выбора методологии

Данное исследование не ставит целью изобретение нового алгоритма стабилизации. Вместо этого, работа фокусируется на методологическом сравнении трех фундаментально разных классов оптимизации — наивного поиска (BRS), локального поиска (HC) и градиентной оптимизации (PG). В существующей литературе, как правило, сравниваются только передовые алгоритмы RL (A2C vs. PPO [1]), тогда как прямое

сравнение их эффективности с простыми стохастическими методами на базовой, но точно реализованной физической модели встречается редко. Этот анализ позволит определить, насколько сложность PG оправдана по сравнению с его простейшими аналогами для систем низкой размерности.

2.2. Обзор используемых технологий

Данный подраздел посвящен обзору и обоснованию выбора тех технологических решений и алгоритмов, которые будут использованы для программной реализации задачи стабилизации стержня. Цель — показать, почему выбранные алгоритмы (BRS, HC, PG) являются наилучшим набором для достижения целей исследования, а выбранные инструменты — оптимальными для реализации.

2.2.1. Сравнение и выбор алгоритмов оптимизации

В рамках работы проводится сравнительный анализ трех методов, представляющих разные уровни сложности и подходы к оптимизации функций. Обзорная литература [7, 8] показывает, что эти методы формируют логическую иерархию, позволяющую провести глубокое методологическое сравнение:

1. **Basic Random Search (BRS):** Простейший безградиентный метод.
2. **Hill Climbing (HC):** Эвристический метод, использующий локальный случайный поиск для ускорения сходимости BRS.
3. **Policy Gradient (PG):** Градиентный метод обучения с подкреплением, который теоретически должен превосходить безградиентные методы по скорости сходимости в задачах оптимизации [8].

Для оценки алгоритмов использовались следующие критерии, релевантные целям работы: вычислительная сложность, скорость сходимости и риск застревания в локальном оптимуме.

Алгоритм	Класс	Вычисл. сложность	Риск локального оптимума
Basic Random Search	Безградиентный	Низкая	Низкий
Hill Climbing	Безградиентный	Низкая	Высокий
Policy Gradient	Градиентный (RL)	Средняя	Средний

Таблица 1: Сравнение алгоритмов оптимизации

Обоснование выбора: Данный набор выбран потому, что он позволяет ответить на ключевой вопрос исследования: оправдывает ли рост вычислительной сложности (переход от НС к PG) пропорциональный рост скорости сходимости в задаче стабилизации низкой размерности. BRS и НС выступают в роли надежных, но простых контрольных групп для оценки эффективности градиентного подхода.

2.2.2. Выбор платформы реализации

Для реализации физической модели и алгоритмов необходимо выбрать подходящую программную среду и язык. Для работ в области машинного обучения стандартным выбором является язык **Python**, несмотря на то, что **C/C++** обеспечивает более высокую производительность для чистого численного интегрирования дифференциальных уравнений [2].

Платформа	Производительность	Удобство ML/RL	Скорость разработки
C/C++	Высокая	Низкое	Низкая
Python	Средняя	Высокое	Высокая

Таблица 2: Сравнение платформ для реализации

Обоснование выбора: Для задачи стабилизации стержня, которая является системой низкой размерности, высокая производительность C/C++ не является критичной. В то же время, основная цель работы — сравнение алгоритмов обучения. Использование языка **Python**

в связи с фреймворком **PyTorch** значительно ускоряет разработку, упрощает реализацию Policy Gradient за счет автоматического дифференцирования и обеспечивает лучшую читаемость кода, что критически важно для курсовой работы.

2.2.3. Используемые библиотеки и инструменты

Для программной реализации среды и алгоритмов будет использован следующий стек технологий:

- **Язык программирования: Python.**
- **Численные расчеты и симуляция:** Библиотека **NumPy**. Используется для эффективного хранения и оперирования матрицами весовых коэффициентов агентов.
- **Реализация алгоритма Policy Gradient:** Фреймворк **PyTorch**. Выбран для удобства работы со стохастическими распределениями (сэмплирование действий агента), необходимых для алгоритма Policy Gradient.
- **Визуализация и анализ данных:** Библиотека **Matplotlib**. Будет применяться для визуализации экспериментов, построения графиков сходимости, визуализации метрик и сравнения результатов экспериментов.

2.3. Выводы

Обзор существующих решений (раздел 2.1) показал, что, хотя задача стабилизации стержня решена классическими и современными методами, отсутствует прямое сравнение выбранных алгоритмов (BRS, HC, PG) на самописной физической модели. Обзор технологий (раздел 2.2) подтвердил, что язык **Python** и его библиотеки оптимальны для реализации, поскольку приоритет отдается удобству и скорости реализации сложных алгоритмов машинного обучения перед микрооптимизацией

физической симуляции. Таким образом, теоретическая база для выполнения поставленных задач полностью сформирована.

3. Описание решения

Реализация в широком смысле: что таки было сделано. Часто это на самом деле несколько отдельных разделов, по одному на каждую «реализационную» задачу из постановки. Например, «Архитектура» и «Особенности реализации», либо по разделу на каждую крупную подзадачу. Если разделы получились недостаточно эпичными (меньше пары страниц), сделайте их подразделами одного большого раздела, как тут.

Если программной реализации нет, тут пишутся Ваши теоретические наработки, если есть, хоть в каком-то виде, то опишите, даже если серьёзно кодом надо будет заняться только в следующей части.

Если работа на текущем этапе предполагает *только* обзор, то

- но всё равно же надо было попробовать воспроизвести чужие результаты, напишите про это сюда;
- если нет, то, наверное, обзор получился годным и ценным сам по себе, источников 40–50 было рассмотрено¹, тогда ладно, раздел «Описание решения» можно не делать.

Для понимания того, как отчёт по учебной практике должен писаться, можно посмотреть видео ниже. Они про научные доклады и написание научных статей. Учебные практики и ВКР отличаются тем, что тут есть требования отдельных глав (слайдов) со списком задач и списком результатов. Но даже если Вы забьёте на требования, специфичные для ВКР, и соблюдете все рекомендации ниже, получившиеся скорее всего будет лучше, чем первая попытка 99% ваших однокурсников.

1. «Как сделать великолепный научный доклад» от Саймона Пейтона Джонса [5] (на английском).
2. «Как написать великолепную научную статью» от Саймона Пейтона Джонса [6] (на английском).

¹Это не шутка, в хороших работах, где целый семестр делался только обзор, оно примерно так и выходит.

3. «Как писать статьи так, чтобы люди их смогли прочитать» от Дэрэка Драйера [3] (на английском). По предыдущим ссылкам разбирается, что должно быть в статье, т.е. как она должна выглядеть на высоком уровне. Тут более низкоуровнево изложено, как должны быть устроены параграфы и т.п.
4. Или на русском от С. Григорьева [10, 9].
5. Ещё можно посмотреть How to Design Talks [4] от Ranjit Jhala, но мы думаем, что первых ссылок всем хватит.

Можно глянуть примеры хороших работ прошлых лет на сайте кафедры системного программирования². Например, работа Москаленко Ивана Николаевича за 3-й курс³.

3.1. Первая задача

Для описания динамики системы «перевернутый маятник на тележке» воспользуемся вторым законом Ньютона. Система состоит из тележки массой M , движущейся горизонтально, и стержня массой m и длиной l , закрепленного на тележке через шарнир.

Введем следующие обозначения:

- $x(t)$ — горизонтальная координата тележки;
- $\theta(t)$ — угол отклонения стержня от вертикали (по часовой стрелке);
- F — управляющая сила, приложенная к тележке;
- H, V — горизонтальная и вертикальная компоненты силы реакции в шарнире;
- $a = l/2$ — расстояние от оси шарнира до центра масс стержня;
- I — момент инерции стержня относительно точки крепления.

²Сайт кафедры СП, архив практик и ВКР: URL: <https://se.math.spbu.ru/theses.html> (дата обращения: 15 января 2024 г.).

³«Создание базы данных изображений для глобальной локализации в пространстве», URL: https://se.math.spbu.ru/thesis_download?thesis_id=1199 (дата обращения: 15 января 2024 г.).

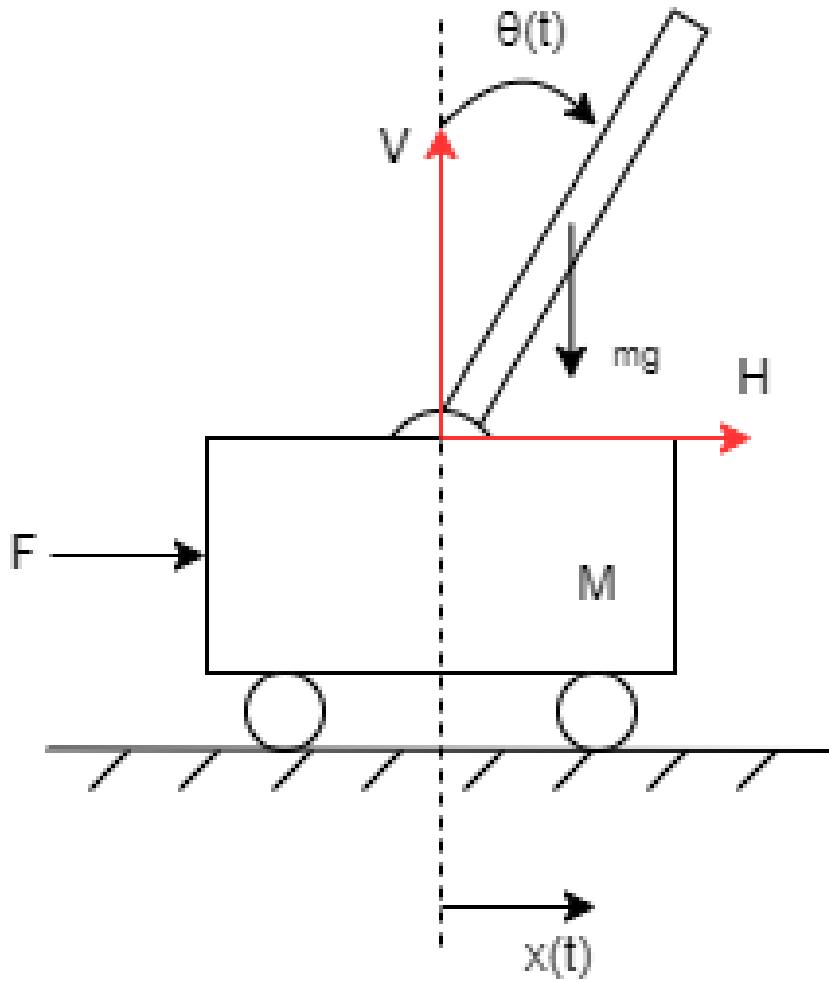


Рис. 1: Схема системы «Перевернутый маятник на тележке» с действующими силами.

3.1.1. Кинематика системы

Координаты центра масс стержня (x_c, y_c) выражаются через обобщенные координаты системы x и θ :

$$x_c = x + a \sin \theta, \quad y_c = a \cos \theta. \quad (1)$$

Дважды продифференцировав эти выражения по времени, получим проекции ускорения центра масс стержня:

$$\begin{cases} \ddot{x}_c = \ddot{x} + a \cos \theta \ddot{\theta} - a \sin \theta \dot{\theta}^2, \\ \ddot{y}_c = -a \sin \theta \ddot{\theta} - a \cos \theta \dot{\theta}^2. \end{cases} \quad (2)$$

3.1.2. Уравнения динамики

Рассмотрим силы, действующие на тела системы.

1. Движение тележки. На тележку действуют управляющая сила F и горизонтальная реакция стержня H (по третьему закону Ньютона, направленная противоположно силе, действующей на стержень):

$$M\ddot{x} = F - H. \quad (3)$$

2. Движение стержня. Запишем уравнения движения центра масс стержня в проекциях на оси координат:

$$\begin{cases} m\ddot{x}_c = H, \\ m\ddot{y}_c = V - mg. \end{cases} \quad (4)$$

Подставим выражения для ускорений (2) в систему (4):

$$\begin{cases} m(\ddot{x} + a \cos \theta \ddot{\theta} - a \sin \theta \dot{\theta}^2) = H, \\ m(-a \sin \theta \ddot{\theta} - a \cos \theta \dot{\theta}^2) = V - mg. \end{cases} \quad (5)$$

Исключим силу реакции H . Сложим уравнение тележки (3) и первое уравнение стержня:

$$M\ddot{x} + m(\ddot{x} + a \cos \theta \ddot{\theta} - a \sin \theta \dot{\theta}^2) = F.$$

После группировки получаем первое уравнение движения системы:

$$(M + m)\ddot{x} + ma \cos \theta \ddot{\theta} = F + ma \sin \theta \dot{\theta}^2. \quad (6)$$

Для получения второго уравнения рассмотрим вращательное движение стержня. Запишем уравнение моментов относительно точки крепления (оси шарнира). Момент инерции стержня относительно конца равен $I = \frac{1}{3}ml^2 = \frac{4}{3}ma^2$. Уравнение вращательного движения:

$$ma \cos \theta \ddot{x} + I\ddot{\theta} = -mga \sin \theta. \quad (7)$$

3.1.3. Разрешение относительно старших производных

Объединим уравнения (6) и (7) в матричную форму относительно вектора угловых и линейных ускорений $[\ddot{x}, \ddot{\theta}]^T$:

$$\begin{pmatrix} M+m & ma \cos \theta \\ ma \cos \theta & I \end{pmatrix} \begin{pmatrix} \ddot{x} \\ \ddot{\theta} \end{pmatrix} = \begin{pmatrix} F + ma \sin \theta \dot{\theta}^2 \\ -mga \sin \theta \end{pmatrix}. \quad (8)$$

Определитель матрицы системы равен:

$$\Delta = (M+m)I - (ma \cos \theta)^2. \quad (9)$$

Решая систему методом Крамера или путем обратной матрицы, получаем явные выражения для ускорений, необходимые для численного моделирования:

$$\ddot{x} = \frac{I(F + ma \sin \theta \dot{\theta}^2) + m^2 ga^2 \cos \theta \sin \theta}{\Delta}, \quad (10)$$

$$\ddot{\theta} = \frac{-(ma \cos \theta)(F + ma \sin \theta \dot{\theta}^2) - (M+m)mga \sin \theta}{\Delta}. \quad (11)$$

Полученная система уравнений (10)–(11) позволяет сформировать **вектор состояния** системы $s = [x, \dot{x}, \theta, \dot{\theta}]^T$, который будет использоваться для реализации среды обучения с подкреплением.

3.1.4. Преобразование динамической системы в форму первого порядка

Исходная математическая модель, основанная на законах Ньютона, представляет собой систему обыкновенных дифференциальных уравнений (**ОДУ**) **второго порядка**, поскольку она выражается через вторые производные координат (ускорения \ddot{x} и $\ddot{\theta}$).

Большинство численных методов интегрирования (например, метод Рунге-Кутты) разработаны для работы с системами **ОДУ первого порядка**. Для приведения системы к требуемой форме мы используем **вектор состояния** s , который включает все обобщенные координаты

и их первые производные (скорости):

$$s = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}. \quad (12)$$

Производная этого вектора \dot{s} представляет собой систему первого порядка:

$$\dot{s} = \frac{d}{dt}s = \begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{pmatrix}. \quad (13)$$

Таким образом, динамическая система принимает форму $\dot{s} = g(s, F)$, где функция $g(s, F)$ определяется:

$$g(s, F) = \begin{pmatrix} \dot{x} \\ \ddot{x}(s, F) \\ \dot{\theta} \\ \ddot{\theta}(s, F) \end{pmatrix}, \quad (14)$$

где $\ddot{x}(s, F)$ и $\ddot{\theta}(s, F)$ вычисляются по явным формулам (10) и (11).

3.1.5. Достаточность вектора состояния для динамики и RL

Для обучения с подкреплением агент оперирует **вектором наблюдения (observation_space)**, который в данном случае совпадает с вектором состояния системы:

$$\mathbf{s}_{RL} = [x, \dot{x}, \theta, \dot{\theta}]^T. \quad (15)$$

1. Достаточность для динамики (Марковское свойство): Система описывается обыкновенными дифференциальными уравнениями (ОДУ) второго порядка. Для однозначного прогнозирования траектории системы в будущем необходимо знать текущее положение и скорость по каждой степени свободы. Вектор \mathbf{s}_{RL} включает все обобщен-

ные координаты (x, θ) и их первые производные $(\dot{x}, \dot{\theta})$, что обеспечивает выполнение **Марковского свойства**: будущее состояние системы полностью определяется текущим состоянием S_t и управляющим воздействием F_t .

2. Пригодность для RL: Вектор s_{RL} является **вектором наблюдения** (Observation), который предоставляет агенту всю информацию, необходимую для принятия оптимальных решений:

- x, \dot{x} : Необходимы для удержания тележки в пределах рабочей зоны.
- $\theta, \dot{\theta}$: Являются критическими параметрами. Агент использует θ для оценки отклонения от вертикали и $\dot{\theta}$ для прогнозирования будущей траектории стержня (гашение колебаний и стабилизация).

3.1.6. Численное интегрирование методом Рунге-Кутты 4-го порядка

Для симуляции движения системы (переход $s_t \rightarrow s_{t+\Delta t}$) используется **Метод Рунге-Кутты 4-го порядка (RK4)**. Этот метод обеспечивает высокий порядок точности ($O(\Delta t^4)$ на шаге) и хорошую устойчивость при моделировании нелинейных динамических систем.

RK4 является одношаговым методом, использующим четыре оценки наклона (производной $g(s, F)$) для вычисления приращения вектора состояния на временном интервале Δt .

Пусть s_t — состояние в момент времени t , F — управляющая сила, и $g(s, F)$ — функция правой части системы \dot{s} . Шаг интегрирования Δt :

1. K_1 (**Наклон в начале интервала**):

$$\mathbf{K}_1 = \Delta t \cdot g(s_t, F)$$

2. K_2 (**Наклон в середине, с использованием K_1**):

$$\mathbf{K}_2 = \Delta t \cdot g(s_t + \mathbf{K}_1/2, F)$$

3. K_3 (**Уточненный наклон в середине, с использованием**

K_2):

$$\mathbf{K}_3 = \Delta t \cdot g(s_t + \mathbf{K}_2/2, F)$$

4. K_4 (**Наклон в конце интервала, с использованием K_3 :**

$$\mathbf{K}_4 = \Delta t \cdot g(s_t + \mathbf{K}_3, F)$$

5. **Итоговое состояние $s_{t+\Delta t}$:**

$$s_{t+\Delta t} = s_t + \frac{1}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4). \quad (16)$$

Этот численный метод реализуется в функции `step` симуляционной среды, обеспечивая эволюцию состояния системы под действием управляющей силы F и внутренних физических законов.

3.2. Вторая задача

3.3. Третья задача

3.4. Некоторые типичные ошибки

Здесь мы будем собирать основные ошибки, которые случаются при написании текстов. В интернетах тоже можно найти коллекции типичных косяков⁴.

Рекомендуется по-умолчанию использовать красивые греческие буквы σ и φ , а именно φ вместо ϕ . В данном шаблоне команды для этих букв переставлены местами по сравнению с ванильным \TeX'ом .

Также, если работа пишется на русском языке, необходимо, чтобы работа была написана на *грамотном* русском языке даже если автор из ближнего зарубежья⁵. Это включает в себя:

- разделы должны оформляться с помощью `\section{...}`, а также `\subsection` и т. п.; не нужно пытаться нумеровать вручную;
- точки после окончания предложений должны присутствовать;

⁴<https://www.read.seas.harvard.edu/~kohler/latex.html> (дата обращения: 16 декабря 2022 г.).

⁵Теоретически, возможен вариант написания текстов на английском языке, но это необходимо обсудить в первую очередь с научным руководителем.

- пробелы после запятых и точек, в конце слова перед скобкой;
- неразрывные пробелы там, где нужны пробелы, но переносить на другую строку нельзя, например т.~е., А.~Н.~Терехов, что-то~\cite{?}, что-то~\ref{?};
- дефис там, где нужен дефис (обозначается с помощью одиночного «минуса» (англ. dash) на клавиатуре);
- двойной дефис там, где он нужен; а именно при указании промежутка в цифрах: в 1900–1910 г. г., стр. 150–154;
- тире (т. е. --- — тройной минус) на месте тире, а не что-то другое; в русском языке тире не может «съезжать» на новую строку, поэтому стоит использовать такой синтаксис: До~--- после;
- даты стоит писать везде одинаково; чтобы об этом не следить, можно пользоваться заклинанием \DTMdate{2022-12-16};
- правильные кавычки должны набираться с помощью пакета `csquotes`: для основного языка в `polyglossia` стоит использовать команду `\enquote{текст}`, для второго языка стоит использовать `\foreignquote{язык}{текст}`; правильные кавычки в русской типографии — <<ёлочки>>, ни в коем случае не "скандинавские лапки";
- все перечисления должны оформляться с помощью `\enumerate` или `\itemize`; пункты перечислений должны либо начинаться с заглавной буквы и заканчиваться точкой, либо начинаться со строчной и закачиваться точкой с запятой; последний пункт перечислений всегда заканчивается точкой.
- Перед выкладыванием финальной версии необходимо просмотреть лог `LATEX`'а, и обратить внимание на сообщения вида *Overfull \hbox (1.29312pt too wide) in paragraph*. Обычно, это означает, что

текст выползает за поля, и надо подсказать, как правильно разделять слова на слоги, чтобы перенос произошел автоматически. Это делается, например, так: `соломо\ -волокуша`.

- *Обязательно используйте инструменты автоматической проверки правописания!* Не посыпайте текст даже научному руководителю на проверку, если не прогнали спеллчекер⁶, желательно, умеющий проверять пунктуацию — у научника куча времени уйдёт на комментарии вида «тут нужна запятая» и не останется сил посоветовать что-нибудь по делу. А ещё научник будет шокирован уровнем грамотности современной молодёжи, впадёт в депрессию и не будет отвечать вам неделю.

3.5. Листинги, картинка и прочий «не текст»

Различный «не текст» имеет свойство отображаться не там, где он написан в текстовом виде в LATEX, поэтому у него должна быть самодостаточная понятная подпись `\caption{...}`, уникальная метка `\label{...}`, чтобы на неё можно было бы ссылаться в тексте с помощью `\ref{...}` (более того, ссылка из текста обязательна). Ниже Вы сможете увидеть таблицу ??, на которую мы сослались буквально только что.

«Не текста» может быть довольно много — чтобы не засорять корневую папку, хорошим решением будет складывать весь «не текст» в папку `figures`. Заклинание `\includegraphics{}` уже знает этот путь и будет искать там файлы без указания папки. Команда `\input{}` умеет ходить по путям, например `\input{figures/my_awesome_table.tex}`. Кроме того, листинги кода можно подтягивать из файла с помощью команды `\inputminted{file}`.

⁶Например, для браузера можно использовать плагин LanguageTool, для VS Code — Code Spell Checker с расширением для русского (но он вроде не умеет пунктуацию, так что следите за запятыми).

Листинг 1: Название для листинга кода. Достаточно длинное, чтобы люди, которые смотрят картинку сразу после названия статьи (т. е. все люди), смогли разобраться и понять к чему в статье листинги, картинки и прочий «не текст».

```
let x = 5 in x + 1
```

3.5.1. Выделение куска листинга с помощью tikz

Это бывает полезно в текстах, а ещё чаще — в презентациях. Пример сделан на основе вопроса на STACKEXCHANGE⁷. Заодно тут показывается альтернативный minted пакет, lstlistings, если не хотите ставить Python и пакет pygments. В этом случае закомментируйте всё, что связано с minted, в matmex-diploma-custom.cls. И внимательно следите за тем, чтобы везде использовался либо только lstlistings, либо minted — смешивание их в одном документе приведёт к странным ошибкам.

3.6. Некоторые детали описания реализации

Описание реализации — очень важный раздел для будущих программных инженеров, т.е. почти для всех. Важно иметь всегда, даже если Вы написали прототип на коленке или немного скриптов.

В процессе работы можно сделать огромное количество косяков, неполный список которых ниже.

1. Реализация должна быть. На публично доступную реализацию обязательна ссылка (в заключении, но можно продублировать тут). Если код под NDA, то об этом, во-первых, должно быть сказано явно, во-вторых, на защиту должны выноситься другие результаты (например, архитектура), чтобы комиссия имела возможность оценить хоть что-то, и, в третьих, должна быть справка от работодателя, что Вы правда что-то сделали.

⁷Вопрос про рамку вокруг листинга на StackExchange, <https://tex.stackexchange.com/questions/284311> (дата обращения: 16 декабря 2022 г.).

```
#include <stdio.h>
#include <math.h>

/** A comment in English */
int main(void)
{
    double c = -1;
    double z = 0;

    // Это комментарий на русском языке
    printf ("For c=%lf:\n", c);
    for (int i = 0; i < 10; i++ ) {
        printf( "z=%d=%lf\n", i, z);
        z = pow(z, 2) + c;
    }
}
```

Рис. 2: Пример листинга с помощью пакета `listings` и TIKZ декорации к нему, оформленные в окружении `figure`. Обратите внимание, что рисунок отображается не там, где он в документе, а может «плавать».

- Рецензент обязан оценить код (о возможности должен побеспокоиться обучающийся).

2. Код реализации должен быть написан защищающимся целиком.

- Если проект групповой, то нужно явно выделить, какие части были модифицированы защищающимся. Например, в предыдущих разделах на картинке архитектуры нужно выделить цветом то, что Вы модифицировали.
- Нельзя пускать в негрупповой проект коммиты от других людей, или людей не похожих на Вас. Например, в 2022 году защищающийся-парень делал коммиты от сценического псевдонима, который намекает на женский «гендер». (Нет, это не шутка.) На тот момент в российской культуре это выглядело странно.
- Возможна ситуация, что вы используете конкретный ник в интернете уже лет пять, и желаете писать ВКР под этим ником на GitHub. В принципе, это допустимо, но если Вы встретите преподавателя, который считает наоборот, то Вам придется грамотно отмазываться. В Вашу пользу могут сыграть те факты, что к нику на GitHub у Вас приписаны настоящие имя и фамилия; что в репозитории у вас видна домашка за первый курс; и что Ваш преподаватель практики сможет подтвердить, что Вы уже несколько лет используете это ник; и т.п.

3. Если Вы получаете диплом о присвоении квалификации программиста, код должен соответствовать.

- (a) Не стоит выкладывать код одним коммитом.
- (b) Не стоит выкладывать код аккурат перед защитой.
- (c) Лучше хоть какие-то тесты, чем совсем без них. В идеале нужно предъявлять процент покрытия кода тестами.

- (d) Лучше сделать CI, а также CD, если оно уместно в Вашем проекте.
 - (e) Не стоит демонстрировать на защите, что Вам даже не пришло в голову напустить на код линтеры и т.п.
4. Если ваша реализация по сути является прохождением стандартного туториала, например, по отделению картинок кружек от котиков с помощью машинного обучения, то необходимо срочно сообщить об этом руководителю практики/ВКР, иначе Государственная Экзаменационная Комиссия «порвёт Вас как Тузик грелку», поставит «единицу», а все остальные Ваши сокурсники получат оценку выше. (Это не шутка, а реальная история 2020 года.)

Если Вам предстоит защищать учебную практику, а эти рекомендации видятся как более подходящие для защиты ВКР, то ... отмаза не засчитывается, сразу учитесь делать нормально.

Список литературы

- [1] A2C is a special case of PPO / Shengyi Huang, Anssi Kanervisto, Antonin Raffin et al. // arXiv preprint arXiv:2205.09123. — 2022.
- [2] Brooks Jr. Frederick P. The Mythical Man-month (Anniversary Ed.). — Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995. — ISBN: [0-201-83595-9](#).
- [3] Dreyer Derek. How To Write Papers So People Can Read Them. — 2020. — URL: <https://www.youtube.com/watch?v=XpgJ31GKPWI> (дата обращения: 21 ноября 2021 г.).
- [4] Jhala Ranjit. How to Design Talks. — 2020. — URL: <https://www.youtube.com/watch?v=aFT79TmffPk> (дата обращения: 21 ноября 2021 г.).
- [5] Jones Simon Peyton. How to Give a Great Research Talk. — URL: <https://www.microsoft.com/en-us/research/academic-program/give-great-research-talk/> (дата обращения: 21 ноября 2021 г.).
- [6] Jones Simon Peyton. How to Write a Great Research Paper. — URL: <https://www.microsoft.com/en-us/research/academic-program/write-great-research-paper/> (дата обращения: 21 ноября 2021 г.).
- [7] Sutton Richard S., Barto Andrew G. Reinforcement Learning: An Introduction. — MIT Press, 1998. — ISBN: [0262193981](#).
- [8] Williams Ronald J. Simple statistical gradient-following algorithms for connectionist reinforcement learning // Machine Learning. — 1992. — Vol. 8, no. 3-4. — P. 229–256.
- [9] Григорьев С. В. Как готовить презентацию. — URL: <https://www.youtube.com/watch?v=X1NF8QgssYc> (дата обращения: 28 октября 2024 г.).

- [10] Григорьев С. В. Как (не)надо писать тексты ВКР. — URL: <https://youtu.be/hRh-7iMVliY?si=aNIJUVwC7WBk7IKo> (дата обращения: 28 октября 2024 г.).