КОСМИЧЕСКИЕ ИССЛЕДОВАНИЯ, 2024, том 62, № 5, с. 498–515

УДК 629.78,004.852

МЕТОДИКА ПОСТРОЕНИЯ УПРАВЛЕНИЯ КОСМИЧЕСКИМИ АППАРАТАМИ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

© 2024 г. М. Г. Широбоков

Институт прикладной математики им. М.В. Келдыша РАН, Москва, Россия

shirobokov@keldysh.ru

Поступила в редакцию 24.10.2023 г.
После доработки 30.10.2023 г.
Принята к публикации 03.11.2023 г.

В работе формулируется методика сведения общей задачи оптимального управления космическими аппаратами к задаче машинного обучения с подкреплением. Методика включает метод оценки качества алгоритма управления на основе неравенств теории вероятностей. Представлена авторская программная библиотека для сведения задач оптимального управления к обучению с подкреплением. Рассматривается два примера применения методики. Предлагаемая методика может представлять интерес также для построения управления общими механическими системами.

DOI: 10.31857/S0023420624050082, **EDN:** IGZREA

1. ВВЕДЕНИЕ

В настоящее время одними из наиболее актуальных задач управляемого движения космическими аппаратами являются безопасная посадка на неровную поверхность небесного тела, поддержание движения космического аппарата в окрестности малого небесного тела с плохо изученным гравитационным полем, управление движением аппарата на малых высотах в окрестности малых небесных тел, управление угловым движением с ограничениями на управляющие воздействия, управление движением в быстро меняющихся внешних условиях, управление связанным орбитальным и угловым движением аппарата в сложных динамических средах, управление движением с целью понижения риска столкновения с опасными и маневрирующими космическими объектами. Фундаментальные проблемы построения управления в этих задачах связаны с нелинейностью правых частей уравнений движения, отсутствием или бедностью результатов анализа движения, отсутствием или высокой неточностью моделей движения, а также неопределенностью в движении аппарата или параметров моделей движения.

Среди методов построения оптимального управления можно условно выделить два класса. В **первый класс** входят методы, основанные на принципе максимума Понтрягина, теории устойчивости Ляпунова, теории Флоке и теории игр [1–3]. Эти методы можно назвать *локальными* или *тактическими*, так как строится некоторое номинальное управление и управление, стабилизирующее движение в окрестности номинальной траектории. Во **второй класс** входят методы, основанные на принципе оптимальности Беллмана и динамическом программировании [4–6]. Эти методы можно назвать *глобальными* или *стратегическими*, так как здесь управление ищется в виде отображения из состояния в управляющие воздействия, причем управление должно «вести себя хорошо» в широком диапазоне значений аргумента.

В последнее время активно развивается и привлекает внимание исследователей раздел приближенного динамического программирования, называемый *машинным обучением с подкреплением* [7–9]. В обучении с подкреплением управление динамическими системами интерпретируется как взаимодействие агента с

внешней средой, от которой агент за свои действия получает вознаграждения и стремится максимизировать суммарные вознаграждения. В механике космического полета средой может являться космический аппарат, агентом — управляющее программное обеспечение на аппарате, вознаграждением могут быть точность прилета в заданную область пространства и экономия топлива. Функция управления (отображение из состояния аппарата в управляющие воздействия) параметризуется, параметры ищутся так, чтобы удовлетворялось уравнение оптимальности Беллмана и / или вознаграждение, полученное агентом за весь полет, было максимальным в среднем по возможным начальным условиям старта. Результатом обучения является функция управления, которая способна направлять аппарат в заданную точку пространства. Эта функция может быть загружена на борт космического аппарата и может управлять им во время реального полета на основе состояния аппарата или оценок состояния аппарата.

Математически строгая и основанная на ляпуновском подходе теория обучения с подкреплением для построения управления изложена в монографии [9]. В частности, там описываются методы построения функции оптимального управления для классических вариантов функционалов — интегралов от квадратичной функции управления и невязок по состоянию. Формулируются теоремы о сходимости параметрически заданной функции управления к оптимальной. Методы излагаются для случая непрерывных динамических систем. Изложенная там теория имеет свои недостатки:

1. Делаются существенные предположения о динамике, функции управления и функции Беллмана, например условие богатства входного сигнала (постоянства возбуждения, persistence of excitation). Эти условия зачастую не удается доказать или проверить, а их невыполнение сказывается на сходимости к оптимальной функции управления.

2. Рассматриваются только функции вознаграждения, квадратичные относительно управления и положительно определенные относительно вектора невязки. Требуется обратимость матрицы квадратичной формы относительно управления.

3. В формулах коррекции параметров приближенных функции управления и функции Беллмана используются правые части уравнений движения.

4. Теория, построенная в монографии, касается непрерывных динамических систем. Для дискретных систем (например, в механике космического полета — случай движения с импульсами) необходимо проводить адаптацию этих методов.

Следует отметить, что существуют положительные примеры применения ляпуновского подхода к обучению нейросетевых моделей управления космическими аппаратами [10–13]. Эти примеры не относятся к обучению с подкреплением, но методы, изложенные в них, схожи с теми, что описываются в работе [9]. Ляпуновский подход к обучению нейросетевых моделей также известен в литературе под названиями *детерминированное обучение* [14] и *нейродинамическое программирование* [15].

В последние несколько лет область обучения с подкреплением пополнилась эффективными алгоритмами, зарекомендовавшими себя в разных областях, в том числе и в механике космического полета (см. обзор литературы по теме в публикации [16] в разделе Reinforcement learning). Эти численные методы основываются на алгоритмах приближенного динамического программирования, методах оптимизации функций с большим числом параметров и теории частично наблюдаемых марковских процессов принятия решений. Преимуществом этих методов является существенное сокращение математических предположений и значительный охват возможных решаемых задач. Примеры их применения показывают, что стратегии управления, создаваемые этими методами, естественным образом способны адаптироваться к неизвестным параметрам аппарата и внешней среды [17–20].

Несмотря на наличие большого числа примеров применения современных методов обучения с подкреплением в задачах управления космическими аппаратами, литературе не хватает общей методики сведения задачи оптимального управления аппаратом к задаче обучения с подкреплением. Примеры применения методов обучения с подкреплением к конкретным задачам наполнены техническими деталями и решением локальных математических проблем. Вместе с тем обзор литературы дает понимание того, как могла бы выглядеть такая методика сведения одной задачи к другой в общем случае, причем так, чтобы известные примеры применения обучения с подкреплением стали частными случаями использования этой методики. Кроме того, техника решения задач оптимального управления методами обучения с подкреплением практически

не представлена в русскоязычной литературе. Презентация обоснованной общей методики помогла бы направить интерес к этой теме и помочь разрешить вопросы создания надежных регуляторов в задачах управления с нелинейными уравнениями движения, неаналитическими решениями и неопределенностью.

Методы обучения с подкреплением, о которых далее пойдет речь, основаны на методах оптимизации сложных нелинейных функций с большим числом параметров и опираются на разнообразные эвристические приемы, помогающие процедурам оптимизации быстрее сходиться к оптимальным решениям. Кроме того, эти методы обучения с подкреплением основываются на обработке выборок данных, получаемых в результате взаимодействия агента со средой. Эти методы являются *безмодельными* (*model-free*), соответствующие алгоритмы обучения не используют информацию о виде или структуре правых частей уравнений и информацию о виде или структуре функции вознаграждения, вместо этого параметры функции управления и функции Беллмана настраиваются исходя из опыта взаимодействия агента со средой. Методы обучения с подкреплением максимизируют средние суммы вознаграждений агента, поэтому в отдельных эпизодах может наблюдаться неэффективное взаимодействие агента со средой. Поскольку в общем случае уравнения движения нелинейны, их решения не выражаются в элементарных функциях, в системе присутствует неопределенность, а модели функции управления и функции Беллмана могут быть сложными для анализа, то нет возможности гарантированно утверждать, что предлагаемая функция управления решает задачу при всех исходах. Таким образом возникает задача оценивания надежности применения разработанных регуляторов, что можно рассматривать как задачу оценивания вероятности наступления неблагоприятных событий. Для оценивания вероятностей событий и точности этого оценивания можно воспользоваться классическими неравенствами теории вероятностей, в частности — неравенством Хефдинга. Примеры применения этих неравенств не встречаются в литературе по механике космического полета, хотя они делают результаты применения эвристических приемов заслуживающими доверие.

Наконец, необходимо продемонстрировать исследователям применение предлагаемой методики на модельных и простых примерах управления динамическими системами. Необходимо, чтобы разрабатываемая методика допускала простую программную реализацию, понятную специалистам в механике, малознакомыми с теорией обучения с подкреплением, и которую можно было бы использовать для решения широкого спектра задач.

Цель настоящей работы — на основании проведенных ранее обзоров литературы сформулировать методику сведения общей задачи оптимального управления космическими аппаратами к задаче машинного обучения с подкреплением, предложить метод оценивания качества полученного алгоритма управления, разработать программную реализацию методики и продемонстрировать ее работу на примерах.

2. МАТЕМАТИЧЕСКИЕ ОСНОВЫ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Обучение с подкреплением (reinforcement learning) — это раздел машинного обучения, разрабатываемый для решения задач, при формализации которых можно выделить среду и агента, взаимодействующего со средой и получающего от среды за свои действия вознаграждения. Во время обучения агент стремится действовать так, чтобы максимизировать суммарное вознаграждение, получаемое от среды. Агенту неизвестно, как ему следует действовать, он не знает «правильных» ответов (нет «учителя»), но пытается *методом проб и ошибок* угадать оптимальное поведение.

Различение агента и среды в задачах обучения с подкреплением условно, не всегда можно провести между ними четкие границы, но на практике это оказывается неважным. Под агентом понимают не обязательно материальное воплощение в виде робота или механизмов, чаще всего это программное обеспечение. Например, в космических системах агентом может быть программное обеспечение в бортовом компьютере космического аппарата, а сам аппарат — средой. Агент выполняет действия — посылает команды в блок управления аппаратом — и получает от аппарата обратную связь в виде изменения его состояния.

Теория обучения с подкреплением описывается на языке теории вероятностей. Будем всюду считать состояния, действия и вознаграждения случайными величинами. Случайные величины будем обозначать заглавными буквами, а их конкретные реализации — строчными.

Будем считать, что агент взаимодействует со средой в дискретные моменты времени

$k \in \{0,1,2,\ldots\}$. Пусть $\mathcal{S}$ — множество всех возможных состояний среды, $\mathcal{A}$ — множество всех возможных действий агента. Эти множества могут быть дискретными или непрерывными, конечными или бесконечными. Обозначим за $\mathbf{S}_k$ состояние среды в момент k, $\mathbf{A}_k$ — действие, которое производит агент в момент k.

В теории обучения с подкреплением процесс взаимодействия агента со средой описывается марковским процессом принятия решений. В этом случае переходы между состояниями среды полностью определяются функцией перехода

$$p: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1],$$

которая для всех $\mathbf{s} \in \mathcal{S}$, $\mathbf{a} \in \mathcal{A}$, $\mathbf{s}' \in \mathcal{S}$ и $k \geq 0$ есть[1]

$$p(\mathbf{s},\mathbf{a},\mathbf{s}') = \mathbb{P}(\mathbf{S}_{k+1} = \mathbf{s}' \mid \mathbf{S}_k = \mathbf{s}, \mathbf{A}_k = \mathbf{a}).$$

Будем считать, что вероятность перехода между состояниями не зависит от времени (такую среду называют *стационарной*).

Вознаграждение R_k в момент времени k определяется распределением d_R, зависящем только от текущего состояния $\mathbf{S}_k$, действия $\mathbf{A}_k$ и, возможно, будущего состояния $\mathbf{S}_{k+1}$ и не зависящего от времени k. Будем также считать, что вознаграждения равномерно по исходам ограничены сверху: $|R_k| \leq R_{\max}$.

Введем начальное распределение состояний $d_0: \mathcal{S} \to [0,1]$, то есть функцию, такую, что для всех $\mathbf{s}$

$$d_0(\mathbf{s}) = \mathbb{P}(\mathbf{S}_0 = \mathbf{s}).$$

Опишем теперь правило, согласно которому в среде действует агент. *Стратегией* называется функция $\pi: \mathcal{S} \times \mathcal{A} \to [0,1]$, которая для всех $\mathbf{s} \in \mathcal{S}$, $\mathbf{a} \in \mathcal{A}$ и $k \geq 0$ равна

$$\pi(\mathbf{s},\mathbf{a}) = \mathbb{P}(\mathbf{A}_k = \mathbf{a} \mid \mathbf{S}_k = \mathbf{s}).$$

Таким образом, стратегия для каждого $\mathbf{s}$ определяет распределение на множестве действий, действие выбирается случайно в соответствии с этим распределением. Технически, действие выбирается в результате применения генератора случайных чисел, отвечающего данному распределению. Стратегия может быть детерминированной, в этом случае для всех $\mathbf{s}$ и $\mathbf{a}$ функция $\pi(\mathbf{s},\mathbf{a})$ принимает значения из множества $\{0,1\}$ и под стратегией можно понимать просто функцию $\mathbf{a} = \pi(\mathbf{s})$. Будем считать, что стратегия не зависит от времени t, а действие определяется лишь состоянием $\mathbf{s}$, в котором находится среда.

Итак, рассмотрим процесс взаимодействия агента со средой. Сначала в соответствии с начальным распределением d_0 инициализируется начальное состояние $\mathbf{S}_0 \sim d_0$ (знак ~ означает, что случайная величина слева в выражении генерируется из распределения справа в выражении). Затем для этого состояния в соответствии со стратегией агент производит действие $\mathbf{A}_0 \sim \pi(\mathbf{S}_0,\cdot)$. Далее, в соответствии с функцией перехода p среда переходит в новое состояние $\mathbf{S}_1 \sim p(\mathbf{S}_0,\mathbf{A}_0,\cdot)$. После этого агент получает вознаграждение $R_0 \sim d_R(\mathbf{S}_0,\mathbf{A}_0,\mathbf{S}_1)$. Далее агент производит новое действие $\mathbf{A}_1 \sim \pi(\mathbf{S}_1,\cdot)$, среда переходит в новое состояние $S_2 \sim p(\mathbf{S}_1,\mathbf{A}_1,\cdot)$, агент получает вознаграждение $R_1 \sim d_R(\mathbf{S}_1,\mathbf{A}_1,\mathbf{S}_2)$, и так далее. Взаимодействие со средой заканчивается либо в определенный момент времени $k = K$, либо при достижении средой некоторого особого состояния. Участок времени взаимодействия агента со средой от начального до финального состояния называется *эпизодом*.

В обучении с подкреплением агент стремится увеличивать суммарное за эпизод вознаграждение от среды. Формализуется это введением целевой функции

$$J(\pi) = \mathbb{E}\left(\sum_{k=0}^{\infty} R_k \;\middle|\; \pi \right),$$

где вертикальная черта означает, что все действия, которые производит агент, производятся им в рамках стратегии π. Так как любой эпизод конечен, то и ряд представляет собой конечную сумму (формально можно считать, что $R_k = 0$, начиная с некоторого момента времени, зависящего от исхода). Стратегия π^* называется *оптимальной*, если

$$\pi^* \in \arg\max_{\pi \in \Pi} J(\pi),$$

где Π — множество рассматриваемых стратегий. В общем случае оптимальная стратегия может не существовать, а если существует, то может быть не единственной.

В определении целевой функции участвует сумма случайных величин. Чтобы сделать эту сумму ограниченной для любого исхода, вводят величину $\gamma \in [0,1]$ и рассматривают целевую

[1] Эта формула является строгой для дискретных пространств состояния и действия, но приводится здесь в общем случае для простоты. Для непрерывных пространств состояния или действия следует вводить плотность вероятности перехода.

функцию $J(\pi) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k R_k \;\middle|\; \pi \right)$, сумма в которой конечна с вероятностью единица, если $\gamma < 1$ и $|R_k| \le R_{\max}$.

Функцией ценности состояния называется функция $v^\pi : \mathcal{S} \to \mathbb{R}$, для каждого $\mathbf{s} \in \mathcal{S}$ равная

$$v^\pi(\mathbf{s}) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k R_{k+m} \;\middle|\; \mathbf{S}_m = \mathbf{s}, \pi \right). \qquad (1)$$

Эта функция выражает среднее суммарное вознаграждение за эпизод, получаемое агентом, действуя в рамках стратегии π, стартуя из состояния $\mathbf{s}$ в момент m. Обратим внимание, что эта функция не зависит от m, и в ее определении допустимо брать $m = 0$. Независимость функции ценности от m является следствием стационарности среды, стационарности вознаграждения и марковского свойства процесса.

Функцией ценности действия называется функция $q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, для каждого $\mathbf{s}$ и $\mathbf{a}$ равная

$$q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k R_{k+m} \;\middle|\; \mathbf{S}_m = \mathbf{s}, \mathbf{A}_m = \mathbf{a}, \pi \right).$$

Эта функция выражает среднее суммарное вознаграждение за эпизод, получаемое агентом, стартуя из состояния $\mathbf{s}$ с действием $\mathbf{a}$ в момент m и далее действуя в рамках стратегии π. В этом выражении можно брать m произвольным.

В *частично наблюдаемых марковских процессах принятия решений* агент принимает решение о действиях не на основе состояний, а на основе наблюдений, которые являются функциями состояний (возможно случайными). Если наблюдение $\mathbf{o} = \varphi(\mathbf{s})$ является детерминированной и взаимно однозначной функцией состояния, то в качестве состояния можно выбрать наблюдение и ввести марковский процесс принятия решений, в котором стратегия может быть определена как

$$\pi(\mathbf{o}, \mathbf{a}) = \mathbb{P}(\mathbf{A}_k = \mathbf{a} \mid \mathbf{O}_k = \mathbf{o}), \qquad (2)$$

где $\mathbf{O}_k$ — вектор наблюдений в момент k, а функция ценности —

$$v^\pi(\mathbf{o}) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k R_{k+m} \;\middle|\; \mathbf{O}_m = \mathbf{o}, \pi \right). \qquad (3)$$

Если $\mathbf{o} = \varphi(\mathbf{s}) + \xi$, где φ — взаимно однозначная детерминированная функция состояния, а ξ — случайный вектор, то наблюдению $\mathbf{o}$ может соответствовать множество возможных состояний $\mathbf{s}$. При достаточно малой дисперсии компонент ξ, допустимо рассматривать стратегию и функцию ценности как функции наблюдения по формулам (2)–(3), если это слабо влияет на результаты методов оптимизации стратегии. Если функция ценности используется только во время оптимизации стратегии, то есть играет только вспомогательную роль, то допустимо определять ее по формуле (1).

В прочих случаях наблюдение не однозначно связано с состоянием, поэтому для сведения процесса принятия решений к марковскому процессу принятия решений кроме наблюдения следует также ввести историю наблюдений $\mathbf{h}$, которая вместе с текущим наблюдением позволяет с высокой точностью оценивать состояние:

$$\mathbf{o}, \mathbf{h} \to \varphi(\mathbf{s}) + \xi,$$

где $\varphi(\mathbf{s})$ — взаимно однозначная функция состояния, а ξ — случайный вектор с малыми дисперсиями компонент. История наблюдений $\mathbf{h}$ может состоять из одного, двух или многих наблюдений, предшествующих наблюдению $\mathbf{o}$. Этот вектор может быть равен и равносильной функции предыдущих наблюдений. В любом случае, стратегия вводится как отображение

$$\pi(\mathbf{o}, \mathbf{h}, \mathbf{a}) = \mathbb{P}(\mathbf{A}_k = \mathbf{a} \mid \mathbf{O}_k = \mathbf{o}, \mathbf{H}_k = \mathbf{h}),$$

где $\mathbf{H}_k$ обозначает вектор истории наблюдений к моменту k не включительно, а $\mathbf{h}$ — реализация этого вектора. Аналогично определяется функция ценности состояния

$$v^\pi(\mathbf{o}, \mathbf{h}) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k R_{k+m} \;\middle|\; \mathbf{O}_m = \mathbf{o}, \mathbf{H}_m = \mathbf{h}, \pi \right).$$

Наблюдению и истории здесь все еще неоднозначно соответствует состояние, но оценка этого состояния тем точнее, чем меньше дисперсия компонент ошибки ξ, которая обычно связана с частотой наблюдений и объемом истории наблюдений.

3. МЕТОДЫ ОПТИМИЗАЦИИ СТРАТЕГИЙ

Существует множество алгоритмов поиска оптимальных стратегий. К классическим методам относятся методы итерации по стратегиям и ценности с известными условиями сходимости [6, 7]. Однако область применения этих методов ограничена средами с дискретным и конечным множеством состояний и агентами с дискретным и конечным множеством действий, в то время

как для механики космического полета характерны задачи с непрерывными множествами состояний и действий. Для таких случаев разрабатываются методы приближенного динамического программирования, общую теорию о которых можно найти в монографиях [8, 15].

Обзор методов показывает, что среди них можно выделить два класса методов: основанные на использовании агентом функции перехода между состояниями (*задачи планирования, model-based methods*), и методы, в которых агент не использует эту функцию для расчета оптимальных действий (*безмодельные методы, model-free methods*). Характерными чертами первого класса методов являются эффективность и точность. Примером применения таких методов может служить работа по разработке гарантирующего синтеза управления для управления космическим аппаратом в окрестности неустойчивой точки либрации [21]. Методы же второго класса не используют специфические свойства динамики и потому более универсальны. Из наиболее известных — метод градиента глубокой детерминированной стратегии (Deep Deterministic Policy Gradient, DDPG) [22], метод асинхронного исполнителя–критика (Asynchronous Advantage Actor Critic, A3C) [23], метод оптимизации ближайшей стратегии (Proximal Policy Optimization, PPO) [24]. Перечисленные методы являются градиентными, среди безградиентных методов для оптимизации стратегий можно применять эволюционные алгоритмы [25, 26]. Обзор применения безмодельных методов к задачам механики космического полета можно найти в работе [16].

Данная работа посвящена разработке общей методики решения широкого класса задач и опирается на безмодельные методы. В безмодельных методах динамика среды описывается задаваемой исследователем функцией перехода, но эта функция перехода не используется агентом, и его поведение оптимизируется исходя из опыта взаимодействия со средой — цепочек вида «состояние» – «действие» – «вознаграждение» – «состояние» – «действие» – «вознаграждение» и так далее.

В современных методах обучения с подкреплением стратегия представляется в виде параметрически заданной функции с конечным числом параметров:

$$\pi = \pi(\mathbf{a}, \mathbf{s}, \theta) \text{ или } \mathbf{a} = \pi(\mathbf{s}, \theta),$$

где θ — вектор оптимизируемых параметров. Задача поиска оптимальной стратегии сводится к оптимизации функционала

$$J(\theta) = \mathbb{E}\left(\sum_{k=0}^{\infty} \gamma^k R_k \;\middle|\; \pi(\theta) \right).$$

Рассмотрим один из вариантов оптимизации этого функционала, применяемый, например, методом PPO. Математическое ожидание в выражении для $J(\theta)$ заменяется на выборочное среднее (среднее арифметическое). Производится серия испытаний Монте-Карло, в каждой серии среда инициализируется в начальном состоянии, агент при фиксированных значениях параметров θ производит действия, получает за них вознаграждения R_t, и действует до конца эпизода. Так в серии испытаний получаются реализации суммарных вознаграждений за эпизод, их среднее дает оценку $J(\theta)$. Далее значения параметров θ с использованием конкретного метода оптимизации (например, PPO) корректируются в сторону повышения значения функционала J и процесс сбора данных повторяется снова. Процесс оптимизации останавливается, когда значение функционала перестает увеличиваться.

Интересно заметить, что для коррекции значений параметров в сторону повышения значения функционала J не требуется рассчитывать производные вознаграждений, действий, управляющих воздействий или состояний по параметрам θ. Согласно теореме о градиенте стратегии (policy gradient theorem) [27] в силу стационарности среды и марковского свойства градиент J содержит только производную по стратегии π:

$$\nabla_\theta J(\theta) \propto \mathbb{E}_{\mathbf{S}\sim d_0, \mathbf{A}\sim\pi(\theta)}[q^\pi(\mathbf{S}, \mathbf{A}) \nabla_\theta \ln \pi(\theta)],$$

где $\propto$ означает пропорциональность. Это выражение используется в градиентных методах оптимизации стратегий.

4. МЕТОДИКА ПОСТРОЕНИЯ УПРАВЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Перейдем теперь к описанию методики построения управления. Пусть динамика механической системы описывается системой обыкновенных дифференциальных уравнений

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}), t \in [t_0, t_f], \quad (4)$$

где t — время, $\mathbf{x} \in \mathbb{R}^n$ — вектор состояния, $\mathbf{u} \in \mathbb{R}^m$ — вектор управления.

Рассматриваются функции управления вида $\mathbf{u} = \mathbf{u}(t, \mathbf{x})$ или, в более общем случае, вида $\mathbf{u} = \mathbf{u}(t, \mathbf{o}, \mathbf{h})$, где $\mathbf{o} = \mathbf{o}(t, \mathbf{x}) \in \mathbb{R}^k$ — вектор наблюдения, $\mathbf{h} \in \mathbb{R}^l$ — вектор, характеризующий историю

наблюдений. Назовем управление допустимым, если оно:

1) переводит каждое решение уравнений (4) с начальным условием из заданного множества $(t_0, \mathbf{x}(t_0)) \in \Omega_0$ в множество заданных краевых условий $(t_f, \mathbf{x}(t_f)) \in \Omega_f$, причем эти решения удовлетворяют промежуточным ограничениям $\{(t, \mathbf{x}(t)), t_0 < t < t_f\} \in \Omega_{\text{int}}$,

2) удовлетворяет ограничениям

$$\{(t, \mathbf{u}(t, \mathbf{x}(t))), t_0 \le t \le t_f\} \in \Omega_U$$

или

$$\{(t, \mathbf{u}(t, \mathbf{o}(t), \mathbf{h}(t))), t_0 \le t \le t_f\} \in \Omega_U.$$

Пусть определен функционал

$$\mathcal{J} = \mathcal{J}(\mathbf{u}, t_0, \mathbf{x}(t_0), t_f, \mathbf{x}(t_f)), \qquad (5)$$

который допустимому управлению $\mathbf{u}$, переводящему решение с начальным условием $(t_0, \mathbf{x}(t_0)) \in \Omega_0$ в краевое условие $(t_f, \mathbf{x}(t_f)) \in \Omega_f$, ставит в соответствие число.

Рассматривается задача поиска функции управления $\mathbf{u} = \mathbf{u}(t, \mathbf{x})$ или, в более общем случае, $\mathbf{u} = \mathbf{u}(t, \mathbf{o}, \mathbf{h})$, для каждых $(t_0, \mathbf{x}(t_0)) \in \Omega_0$ и $(t_f, \mathbf{x}(t_f)) \in \Omega_f$ оптимизирующей функционал (5).

В механике космического полета распространены задачи поиска как непрерывной функции управления, так и поиска импульсов скорости. Рассмотренная выше постановка задачи оптимального управления естественным образом применима к задачам поиска непрерывной или кусочно-непрерывной функции управления. Если же стоит задача поиска импульсов скорости, то можно считать управление $\mathbf{u}$ равным нулю между импульсами и равным импульсу скорости в момент совершения импульса.

Для того, чтобы свести поставленную задачу оптимального управления к задаче обучения с подкреплением, необходимо выполнить следующие шаги:

1. Связать понятие состояния из теории обучения с подкреплением с понятием состояния механической системы. В общем случае состоянием из теории обучения с подкреплением можно считать $\mathbf{s} = (t, \mathbf{x})$ или любую взаимно однозначную функцию от $(t, \mathbf{x})$. Если система (4) автономная, то состоянием можно считать $\mathbf{x}$ или любую взаимно однозначную функцию от $\mathbf{x}$.

2. На области начальных условий $(t_0, \mathbf{x}(t_0)) \in \Omega_0$ определить распределение вероятностей $\mathcal{D}_0$, в соответствии с которым в серии испытаний Монте-Карло будут генерироваться начальные условия для обучения стратегии. Это может быть равномерное распределение, нормальное распределение или какое-либо другое распределение. Это распределение должно отражать ожидания разработчиков и сопровождающих миссию касательно того, в каких областях пространства и с какой вероятностью будет находиться аппарат в момент начала действия управления.

3. Так как алгоритмы обучения с подкреплением, рассматриваемые в настоящей работе, оперируют средами с дискретным временем, следует перейти от динамики с непрерывным временем (4) к динамике с дискретным временем. Это значит, что необходимо определить отображение $(t_k, \mathbf{x}_k, \mathbf{u}_k) \to (t_{k+1}, \mathbf{x}_{k+1})$ для каждого дискретного шага k. Например, это отображение можно определить с использованием схемы интегрирования Эйлера

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h\mathbf{f}(t_k, \mathbf{x}_k, \mathbf{u}_k), t_{k+1} = t_k + h,$$

задав некоторый малый шаг h. В общем случае шаг на участке может вычисляться с использованием любого численного метода интегрирования. Если управление представляет собой импульсы скорости, под дискретным шагом можно понимать целую траекторию между импульсами, рассчитываемую методом интегрирования. Алгоритмам обучения с подкреплением обычно требуется информация о том, является ли новое состояние $k+1$ финальным. Например, это состояние может быть финальным, если $t_{k+1} = t_f$ или если в момент времени $t_{k+1} < t_f$ происходит событие, означающее конец траектории движения (столкновение с небесным телом, выход за допустимые границы движения и т. п.). Поэтому в общем случае следует задавать отображения вида $(t_k, \mathbf{x}_k, \mathbf{u}_k) \to (t_{k+1}, \mathbf{x}_{k+1}, d_{k+1})$, где $d_{k+1} = 0$, если состояние $k+1$ не является финальным, и $d_{k+1} = 1$, если это состояние является финальным. Интервал времени от начального состояния до финального является эпизодом в терминах обучения с подкреплением.

4. Определить функцию вознаграждения, которая будет представлять собой отображение $(t_k, \mathbf{x}_k, \mathbf{u}_k, t_{k+1}, \mathbf{x}_{k+1}, d_{k+1}) \to r_k$. Здесь при расчете сигнала вознаграждения учитывается то, в каком состоянии находилась система до управления, вектор управления, состояние, в которое перешла система, и информация о том, является ли новое состояние финальным. В качестве

функции вознаграждения можно рассматривать функции вида

$$r_k = -\alpha |\mathbf{u}_k| - \rho(t_{k+1}, \mathbf{x}_{k+1}, \Omega_f), \qquad (6)$$

где ρ — это определенное исследователем расстояние от точки $(t_{k+1}, \mathbf{x}_{k+1})$ до множества Ω_f краевых условий, а α — задаваемая постоянная. Например, если ищется управление, переводящее космический аппарат в состояние, характеризуемое положением $\mathbf{r}_f$ и скоростью $\mathbf{v}_f$, и вектор состояния представляет собой положение и скорость аппарата $\mathbf{x} = [\mathbf{r}, \mathbf{v}]$, то в качестве функции А можно взять $\rho(\mathbf{x}, \Omega_f) = |\mathbf{r} - \mathbf{r}_f| + |\mathbf{v} - \mathbf{v}_f|$. Еще один вариант функции вознаграждения:

$$r_k = -\alpha |\mathbf{u}_k| + \rho(t_k, \mathbf{x}_k, \Omega_f) - \rho(t_{k+1}, \mathbf{x}_{k+1}, \Omega_f).$$

В этом случае на каждом шаге сигнал вознаграждения содержит информацию о приближении к краевым условиям. Заметим, что в этом случае при $\alpha = 0$ суммарные вознаграждения за эпизод совпадают с величиной, на которую улучшается расстояние до краевых условий:

$$R = \sum_{k=1}^{K} r_k = \rho(t_0, \mathbf{x}_0, \Omega_f) - \rho(t_{K+1}, \mathbf{x}_{K+1}, \Omega_f),$$

где K — число шагов в эпизоде. В некоторых работах функция вознаграждения состоит из трех слагаемых:

$$r_k = r_{k,\text{int}} + r_{k,f} + r_{k,\text{good}},$$

где $r_{k,\text{int}}$ состоит из значения функционала $\mathcal{J}$ и вознаграждения за удовлетворение ограничениям на траекторию, $r_{k,f}$ — вознаграждение за удовлетворение краевым условиям (подсчитывается при $d_{k+1} = 1$), и $r_{k,\text{good}}$ — вознаграждение за следование в окрестности «хорошей траектории» (например, оптимальной траектории, полученной в упрощенной постановке).

5. Определить модель восприятия, то есть отображение $(t_k, \mathbf{x}_k) \to \mathbf{o}_k$ из состояния в вектор наблюдений или измерений. В простейшем случае наблюдение может совпадать с состоянием: $\mathbf{o}_k = \mathbf{x}_k$. В общем случае это отображение моделирует работу датчиков и результат применения навигационных процедур, и потому именно наблюдение и история наблюдений используются для расчета действия и управляющих воздействий. Так наблюдением может являться оценка состояния, которая может моделироваться как $\mathbf{o}_k = \mathbf{x}_k + \xi_k$, где ξ_k — случайный вектор. Наблюдением может быть изображение, в этом случае стратегия будет отображать наблюдение-изображение непосредственно в управляющие воздействия, минуя стадию навигации (оценки состояния по изображениям).

6. Наконец, определить модель управления — параметрически заданное отображение $\mathbf{o}_k \to \mathbf{u}_k$ или $(\mathbf{o}_k, \mathbf{h}_k) \to (\mathbf{u}_{k+1}, \mathbf{h}_{k+1})$. Модель управления состоит из композиции двух отображений — из наблюдения в действие $\mathbf{o}_k \to \mathbf{a}_k$ или $(\mathbf{o}_k, \mathbf{h}_k) \to (\mathbf{a}_{k+1}, \mathbf{h}_{k+1})$ и из действия в управление $\mathbf{a}_k \to \mathbf{u}_k$. Отображение из наблюдения в действие обычно строят на основе нейросетевых моделей. Так, в случае отображений вида $\mathbf{o}_k \to \mathbf{a}_k$, чаще всего используют многослойные нейронные сети прямого распространения, например сети с одним скрытым слоем

$$\mathbf{a} = \mathbf{A}_2 \varphi_1(\mathbf{A}_1 \mathbf{o} + \mathbf{b}_1) + \mathbf{b}_2$$

или двумя скрытыми слоями

$$\mathbf{a} = \mathbf{A}_3 \varphi_2(\mathbf{A}_2 \varphi_1(\mathbf{A}_1 \mathbf{o} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3,$$

где $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, $\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$ — матрицы и векторы оптимизируемых параметров, φ_1, φ_2 — активационные функции. Выбор размеров матриц и векторов параметров, а также активационных функций остается за исследователем. Универсальные теоремы аппроксимации [28–31] утверждают, что выбором достаточно большого числа параметров и произвольных активационных функцией из широкого множества нелинейных функций можно добиться аппроксимации любой гладкой функции. Однако эти теоремы не говорят о том, сколько параметров следует брать, и каковы их значения. Выбор числа параметров и активационных функций может значительно влиять на точность аппроксимации и качество получаемой стратегии. В случаях $(\mathbf{o}_k, \mathbf{h}_k) \to (\mathbf{u}_{k+1}, \mathbf{h}_{k+1})$ можно использовать рекуррентные нейронные сети, а $\mathbf{h}_k$ считать скрытым состояние рекуррентного слоя, несущим в себе информацию об истории наблюдений. Вместо рекуррентных слоев можно использовать слои прямого распространения, но на вход сети подавать конкатенацию векторов наблюдений. В общем случае модель не обязана быть нейросетевой. Модель управления может быть построена на основе ляпуновского управления, выведенного в рамках упрощенной модели движения системы. В таком управлении чаще всего есть параметры, которые можно сделать обучаемыми (оптимизируемыми). Некоторые алгоритмы обучения с подкреплением для коррекции параметров стратегии используют также

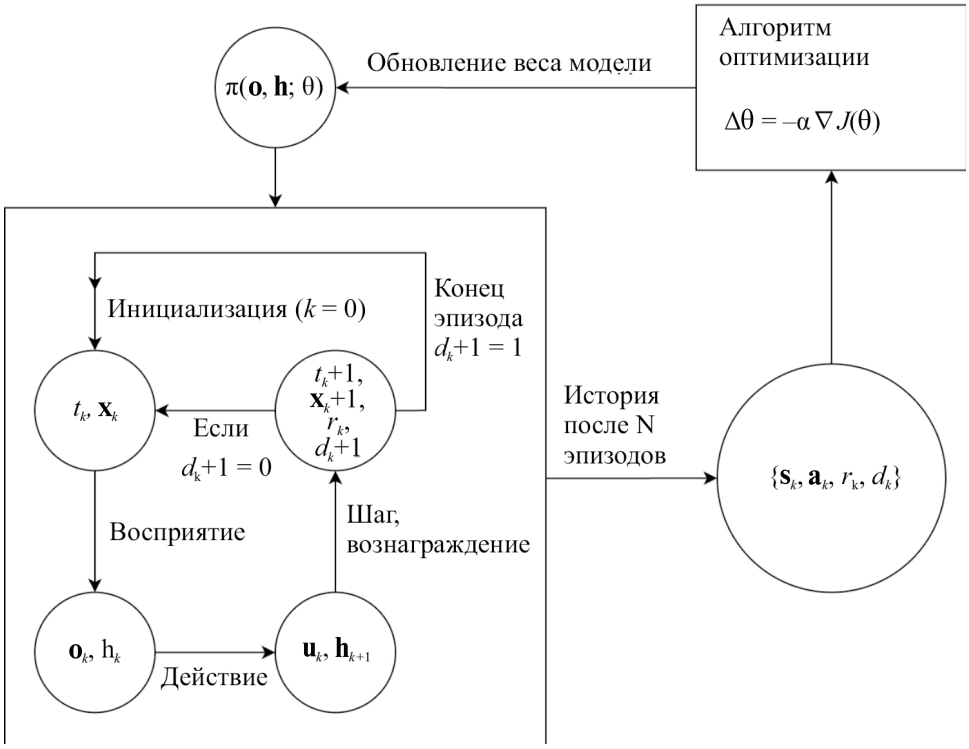

Рис. 1. Процесс обучения стратегии.

параметрическую модель функции ценности. Эту модель как правило выбирают нейросетевой, причем архитектура (число слоев, активационные функции, число нейронов в слоях) обычно совпадает с архитектурой стратегии. Что касается отображения из действия в управление, вводят функцию $\mathbf{u} = \psi(\mathbf{a})$. В механике космического полета функция ψ часто вводится, чтобы ограничивать значения, которые может принимать управление, а также в целях нормировки и масштабирования значений $\mathbf{a}$.

Процесс обучения представлен на рис. 1. Вначале инициализируются параметры модели управления $\pi(\mathbf{o}, \mathbf{h}, \theta)$, где θ содержит все оптимизируемые параметры (разбирается наиболее общий случай управления по наблюдениям и истории наблюдений). Далее начинается оценивание этой стратегии в серии испытаний Монте-Карло. Для этого в соответствии с заданным распределением $\mathcal{D}_0$ в области начальных условий Ω_0 инициализируется состояние системы $(t_0, \mathbf{x}_0)$. Затем модель восприятия сопоставляет этому состоянию наблюдение $\mathbf{o}_0$. Наблюдение подается на вход стратегии, получается вектор управления $\mathbf{u}_0$ и история наблюдений (или скрытое состояние рекуррентного слоя) $\mathbf{h}_1$. Состояние и управление подаются на вход отображения дискретного шага системы и функции вознаграждения, получаются новое состояние, вознаграждение и флаг, сигнализирующий о конце или продолжении эпизода. Если эпизод не закончен, цикл повторяется заново для нового состояния. Если эпизод закончен, инициализируется новое состояние. Этот цикл (серии испытаний Монте-Карло) продолжается много раз, в результате накапливается история взаимодействия со средой: состояния $\mathbf{s}$, действия $\mathbf{a}$, вознаграждения r и флаги завершения эпизода d. Эта история подается на вход алгоритму оптимизации, который корректирует параметры стратегии (этим алгоритмом может быть PPO, DDPG, A3C, генетический алгоритм и любой другой алгоритм оптимизации для обучения с подкреплением). Параметры стратегии корректируются, и серия испытаний Монте-Карло повторяется для исправленной стратегии.

5. ОЦЕНКА КАЧЕСТВА РАБОТЫ ФУНКЦИИ УПРАВЛЕНИЯ

Поскольку задача обучения с подкреплением формулируется в стохастической постановке, а в процессе обучения стратегии ее параметры настраиваются так, чтобы максимизировать *в среднем* суммарное вознаграждение за эпизод,

возможны исходы, при которых управление не выполняет возложенной на него задачи. Ситуация осложняется и тем, что вознаграждение представляет собой скалярную величину, содержащую вручную настраиваемую комбинацию значений функционала и функций ограничений, и поэтому оно может не в полной или не в точной мере выражать цели управления. Например, если в выражении (6) взять достаточно большое значение α, то стратегия может сойтись к пассивному управлению, ведь даже малое управление приведет к большому по величине штрафу, большему, чем невязка в краевых условиях при пассивном управлении.

Все это говорит о необходимости оценки качества работы обученной стратегии на критериях, отражающих цель построения управления. В настоящей работе предлагается оценивать вероятность неблагоприятных событий и средние значения функционала и меры удовлетворения краевых условий с использованием неравенств теории вероятностей, в первую очередь — неравенства Хефдинга [32].

Теорема. *Пусть X_1, ..., X_n — независимые случайные величины, для которых выполнено $a_i \le X_i \le b_i$ с вероятностью единица. Тогда для среднего выборочного $\bar{X} = (1/n)\sum_{i=1}^{n} X_i$ справедлива оценка*

$$\mathbb{P}(|\bar{X} - \mathbb{E}\bar{X}| \ge \varepsilon) \le 2\exp\left(-\frac{2\varepsilon^2 n^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

Следствие. Если X_i одинаково распределены, $a_i \le X_i \le b_i$ для всех i и $\mathbb{E}X_1 = \mu$, то

$$\mathbb{P}(|\bar{X} - \mu| \ge \varepsilon) \le 2\exp\left(-\frac{2\varepsilon^2 n}{(b-a)^2}\right).$$

Это неравенство означает, что проведя n независимых измерений случайной величины X, ограниченной промежутком $[a,b]$, мы получим, что вероятность отклонения среднего выборочного $\bar{X}$ от истинного математического ожидания μ более чем на ε не превосходит $p = 2\exp(-2\varepsilon^2 n/(b-a)^2)$. Неравенство Хефдинга удобно записывать в виде доверительного интервала:

$\mathbb{E}X = \bar{X} \pm \varepsilon$ с вероятностью не менее $1-p$.

Величину ε можно назвать точностью определения математического ожидания случайной величины, а $1-p$ называется уровнем доверия.

Отметим, что неравенство Хефдинга для своего использования не требует знания истинных математических ожиданий, дисперсий или моментов других порядков каких-либо случайных величин, но предполагает, что случайная величина с вероятностью единица заключена в известном промежутке. Чем меньше этот промежуток, тем меньше требуется измерений, чтобы установить интервал значений среднего с той же точностью. Сокращение промежутка в 10 раз позволяет сократить число измерений в 100 раз.

Ниже в табл. 1 для случая $a = 0$, $b = 1$ приводятся формулы для расчета величин ε, p, n, когда даны любые две из этих величин, а также значения объема измерений n для различных значений ε, p:

$$n(\varepsilon, p) = \frac{1}{2\varepsilon^2}\ln\left(\frac{2}{p}\right),$$

$$p(\varepsilon, n) = 2\exp(-2\varepsilon^2 n),$$

$$\varepsilon(p, n) = \sqrt{\frac{1}{2n}\ln\left(\frac{2}{p}\right)}.$$

Если вместо промежутка $[0,1]$ рассматривается промежуток $[a,b]$, то объем выборки следует увеличить в $(b-a)^2$ раз, чтобы получить те же значения p, ε оценки величины.

Таблица 1. Таблица значений n, e, p в случае $a = 0$, $b = 1$

n	ε	p
150	10%	10%
26 492	1%	1%
38 005	1%	0.1%
105 967	0.5%	1.0%
119 830	0.5%	0.5%
152 019	0.5%	0.1%
3 800 452	0.1%	0.1%
495 174 378	0.01%	0.01%

Теперь применим теорему Хефдинга к оценке вероятности интересуемого исследователя события A. Примером такого события может быть то, что невязка по краевым условиям в конце эпизода управления увеличится. Другой пример — движение космического аппарата вышло в фазовом пространстве за допустимые границы, и миссия потеряна.

Пусть $\mathcal{I}(A)$ — индикатор события A, то есть $\mathcal{I}(A) = 1$, если A произошло, и $\mathcal{I}(A) = 0$, если A не произошло. Пусть p_A — неизвестная вероятность того, что A происходит. Введем независи-

мые одинаково распределенные случайные величины X_i, имеющие распределение Бернулли Be(p_A). Тогда $\mu = \mathbb{E}X_i = p_A$, и так как $0 \le X_i \le 1$, то согласно неравенству Хефдинга

$$\mathbb{P}(|\bar{X} - p_A| \ge \varepsilon) \le 2\exp\left(-2\varepsilon^2 n\right).$$

Вероятность того, что ошибка оценки p_A превысит ε, равна $p = 2\exp(-2\varepsilon^2 n)$. Например, для $\varepsilon = 0.01$, $p = 0.01$ получается $n = 26492$. Это значит, что проведя $n = 26492$ измерений случайной величины X, мы получим, что вероятность отклонения истинной вероятности события A от оценки $\bar{X}$ более чем на 1% не превосходит 1%. Таким образом, неравенство Хефдинга можно рассматривать как инструмент оценки требуемого числа испытаний для расчета вероятности события.

В заключение отметим, что вероятность событий и распределение исследуемых случайных величин зависит от распределения $\mathcal{D}_0$ на множестве начальных условий. Это распределение может быть разным во время обучения стратегии и во время ее тестирования.

6. АВТОРСКАЯ ПРОГРАММНАЯ БИБЛИОТЕКА KIAM_RL

Описанная выше методика построения управления механическими системами была воплощена автором в виде программной библиотеки kiam_rl. Библиотека написана на языке Python и на момент написания статьи состоит из двух модулей: routines.py и ppo_hyperparameters_tuning.py. Далее следует описание их возможностей.

Модуль routines.py содержит базовые классы и функции для создания сред и моделей стратегий. Модуль содержит абстрактный класс RLProblem, расширяющий возможности пакета gymnasium [33], позволяющего создавать стандартные среды, с которыми оперируют популярные программные библиотеки алгоритмов обучения с подкреплением, например stable-baselines3 [34], который содержит реализации алгоритмов PPO, DDPG и др. Класс RLProblem содержит несколько методов:

1) observation_space, возвращающий gymnasium.spaces.Box-объект, определяющий множество состояний **s**;

2) action_space, возвращающий gymnasium.spaces.Box-объект, определяющий множество действий **a**;

3) initialize, возвращающий начальные время t_0 и фазовое состояние $\mathbf{x}_0$;

4) equations_of_motion, принимающий на вход время t, фазовое состояние **x**, вектор управления **u** и возвращающий правые части уравнений движения, то есть вектор $d\mathbf{x}/dt$;

5) step, принимающий на вход время t, фазовое состояние **x**, вектор управления **u** и возвращающий результаты дискретного шага: новое время t', новое состояние $\mathbf{x}'$ и флаги конца эпизода, первый из которых сигнализирует о нормальном завершении эпизода, второй — о вынужденном завершении эпизода;

6) reward, принимающий на вход момент времени t, состояние **x**, новый момент времени t', новое состояние $\mathbf{x}'$, вектор управления **u**, флаги конца эпизода и возвращающий вознаграждение;

7) action2u, принимающий на вход действие **a** и возвращающий управление **u**;

8) perception_model, принимающий на вход время t и состояние **x** и возвращающее вектор наблюдения **o**.

Пользователь создает класс, описывающий его задачу, наследуя класс RLProblem, и самостоятельно наполняет указанные методы отображениями согласно постановке его задачи. На основе этого класса с использованием Environment модуля routines.py создается стандартный gymnasium.Env-объект среды, который можно далее использовать в сочетании с популярными библиотеками алгоритмов оптимизации стратегий. Таким образом, класс RLProblem представляет собой интерфейс между механическими аспектами задачи и обучением с подкреплением и помогает создавать и использовать стандартизированные среды.

Программный вид модели управления не стандартизирован, его пользователь создает самостоятельно. Если предполагается использовать алгоритмы библиотеки stable-baselines3, то модели должны создаваться с помощью пакета pytorch [35]. В этом может помочь класс ActorCriticNetworks модуля routines.py. Пользователь создает класс, наследуя его из класса ActorCriticNetworks, и самостоятельно определяет модель стратегии в конструкторе.

При создании и оптимизации моделей стратегии пользователю необходимо задать опции алгоритмов: архитектуру, глубину и ширину нейросетевых моделей, скорость обучения, число эпизодов для оценивания функционала J и многие другие параметры. Эти параметры называются *гиперпараметрами*, чтобы отличать их от параметров модели. Результаты оптимизации

стратегии могут существенно зависеть от выбранных значений гиперпараметров, поэтому зачастую встает задача оптимизации гиперпараметров. Поиск разумных значений гиперпараметров пользователь может производить вручную. Существуют и автоматические процедуры оптимизации гиперпараметров, основанные на методах оптимизации вычислительно затратных функций, например байесовской оптимизации [36], древесно-структурированной оценки Парзена [37]. Существуют программные библиотеки, позволяющие оптимизировать гиперпараметры, например Optuna [38], Ray [39], BoTorch [40], Hyperopt [41]. Модуль ppo_hyperparameters_tuning.py использует процедуры оптимизации гиперпараметров на основе библиотеки Optuna и предполагает, что оптимизация параметров модели осуществляется алгоритмом PPO.

7. ПРИМЕРЫ ПРИМЕНЕНИЯ МЕТОДИКИ

В данном разделе демонстрируется применение методики построения управления. Рассматриваются два примера — задача стабилизации движения в простой динамической системе и астродинамическая задача поддержания движения аппарата в окрестности неустойчивой гало-орбиты вокруг точки либрации. Обе задачи были с использованием описанной выше библиотеки kiam_rl.

Простая динамическая система

Ставится задача построения управления динамической системой

$$\dot{x} = u$$

на интервале времени $t \in [0,1]$, с начальными условиями $x_0 \in [-1,1]$, краевым условием $x(1) = 0$ и управлением $u = u(x) \in [-1,1]$. Оптимизируемый функционал:

$$\mathcal{J} = \int_0^1 |x|dt \to \min,$$

его оптимизация равносильна минимизации времени достижения системой состояния $x = 0$. Заметим, что оптимальным управлением в этой задаче является функция

$$u(x) = -\text{sign}(x),$$

где sign — функция знака. Легко показать, что соответствующей функцией ценности (функцией, сопоставляющей начальному условию значение функционала $\mathcal{J}$) является функция

$$v(x) = -x^2/2.$$

Для поиска управления воспользуемся описанной выше методикой. Будем считать, что состоянием из теории обучения с подкреплением является переменная x, то есть $s = x$. На области начальных условий $\Omega_0 = [-1,1]$ определим равномерное распределение вероятностей. Дискретный шаг определим так:

$$x_{k+1} = x_k + hu_k, t_{k+1} = t_k + h,$$

и будем считать состояние x_{k+1} финальным, если $t_{k+1} = 1$. Величину шага по времени выберем $h = 0.01$. В качестве функции вознаграждения выберем

$$r_k = -|x_k|h.$$

В этом случае будет искаться управление, максимизирующее среднее значение величины

$$R = -\sum_{k=1}^{K} |x_k|h,$$

где $K = 1/h = 100$ — число шагов. Будем считать наблюдение состоянием, то есть $o = x$. В качестве модели управления рассмотрим

$$u = \max(\min(a,1),-1),$$

$$a(x,\theta) = \theta_3 \text{th}(\theta_1 x + \theta_2) + \theta_4,$$

где действие a моделируется как полносвязная нейронная сеть прямого распространения с одним скрытым слоем и гиперболическим тангенсом в роли функции активации. Для обучения этой модели воспользуемся алгоритмом обучения PPO. Этот алгоритм требует также задания модели для функции ценности, определим ее следующим образом:

$$v = v(x,\mathbf{w}) = \mathbf{A}_2 \text{th}(\mathbf{A}_1 x + \mathbf{b}_1) + b_2,$$

где матрицы и векторы $\mathbf{A}_1 \in \mathbb{R}^{5\times 1}$, $\mathbf{b}_1 \in \mathbb{R}^5$, $\mathbf{A}_2 \in \mathbb{R}^{1\times 5}$, $b_2 \in \mathbb{R}$. Это тоже полносвязная нейронная сеть с одним скрытым слоем и гиперболическим тангенсом в роли активационной функции, но с пятью нейронами на скрытом слое.

Обучение моделей произведем с помощью реализации метода PPO из библиотеки stable-baselines3. Объем выборки для аппроксимации среднего значения функционала r (опция n_steps) выберем равным 1000. Число итераций градиентного метода для коррекции весов нейросетевых моделей (опция n_epochs) выберем равным 100, а скорость обучения (опция learning_rate) – 0.01. В общем случае объем выборки для аппроксимации среднего значения

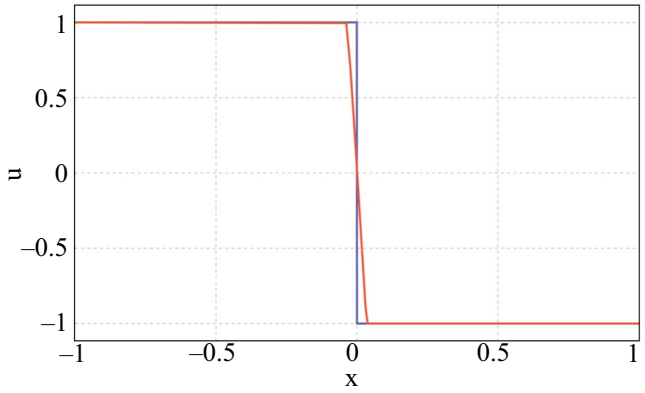

Рис. 2. Оптимальная (синий цвет) и приближенная (красный цвет) функции управления.

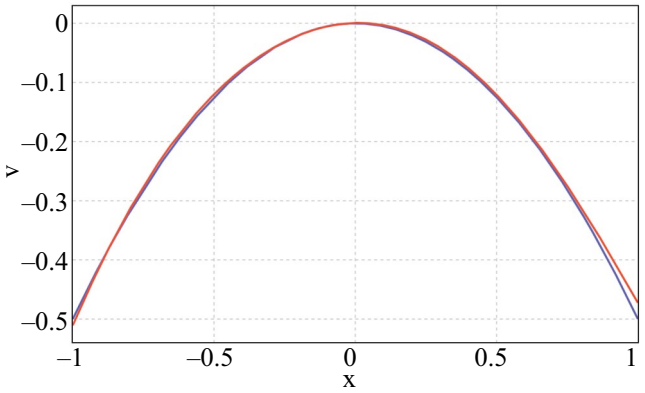

Рис. 3. Оптимальная (синий цвет) и приближенная (красный цвет) функции ценности.

функционала влияет на устойчивость процесса сходимости, а коэффициент скорости сходимости и число итераций градиентного метода оптимизации влияют на устойчивость и скорость сходимости к оптимальному решению. Обучение будем производить на центральном процессоре и завершим, когда число дискретных шагов достигнет 100000 (опция total_timesteps).

В результате оптимизации получается модель

$$u(x,\theta^*) = \max(\min(1.5035\cdot \text{th}(-21.8299x + \\ + 0.0612) - 0.0725, 1), -1),$$

которая приближает оптимальную функцию

$$-\text{sign}(x) = 1\cdot \text{th}(-\infty \cdot x + 0) + 0.$$

На рис. 2 и 3 показаны графики теоретически оптимальной и приближенной функций управления и ценности. Графики показывают близость приближенных функций к теоретическим. Эта близость регулируется шагом дискретизации h и богатством и удачностью выбора параметрических моделей управления и функции ценности.

Поддержание движения в окрестности гало-орбиты

Рассмотрим теперь задачу поддержания движения космического аппарата в окрестности неустойчивой гало-орбиты вокруг точки либрации L_1 системы Земля – Луна. В качестве модели движения выберем круговую ограниченную задачу трех тел, а уравнения движения запишем во вращающейся системе координат: начало правой системы координат поместим в центр Земли, ось x направим вдоль направления Земля – Луна, ось z направим вдоль угловой скорости орбитального движения Луны вокруг Земли. В качестве единицы расстояния выберем расстояние между Луной и Землей, а единицы частоты — орбитальную частоту движения Луны вокруг Земли. Уравнения движения в этом случае запишутся следующим образом:

$$\dot{x} = v_x,\ \dot{y} = v_y,\ \dot{z} = v_z,$$
$$\dot{v}_x = 2v_y + U_x,\ \dot{v}_y = -2v_x + U_y,\ \dot{v}_z = U_z,$$

где

$$U(x,y,z) = \frac{(x-\mu)^2 + y^2}{2} + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{\mu(1-\mu)}{2},$$
$$r_1 = \sqrt{x^2 + y^2 + z^2},\ r_2 = \sqrt{(x-1)^2 + y^2 + z^2},$$

а U_x, U_y, U_z означают частные производные функции $U = U(x,y,z)$ по x, y, z соответственно. Здесь $\mu = m_M/(m_E + m_M)$ — массовый параметр, m_E — масса Земли, m_M — масса Луны. Используются следующие значения массового параметра, единицы расстояния DU, единицы скорости VU и единицы времени TU:

$\mu = 1.215058446035100\cdot 10^{-2}$, DU = 384405 км,

VU = 1.024540192302405 км/с,

TU = 4.342564574695797 дней[2].

Рассматривается движение космического аппарата вблизи гало-орбиты вокруг точки либрации L_1 с максимальной z-координатой равной $z_{max} = 34981$ км (рис. 4).

Начальное условие, отвечающее этой орбите, есть

$$\mathbf{x}_{ref,0} = [x_{ref,0}, y_{ref,0}, z_{ref,0}, v_{x,ref,0}, v_{y,ref,0}, v_{z,ref,0}],$$
$$x_{ref,0} = 0.826890333820514,\ y_{ref,0} = 0,\ z_{ref,0} = 0.091,$$

[2]Большое число знаков после запятой приводится для воспроизводимости результатов.

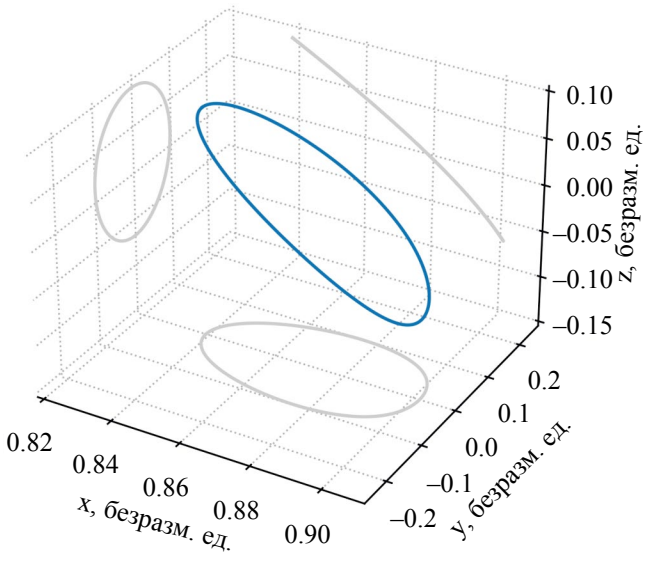

Рис. 4. Гало-орбита (синий цвет), в окрестности которой рассматривается движение аппарата. Серым цветом показаны проекции орбиты на плоскости *xy*, *xz*, *yz*.

$v_{x,\mathrm{ref},0} = 0,\ v_{y,\mathrm{ref},0} = 0.205889408677437,\ v_{z,\mathrm{ref},0} = 0.$

Период орбиты равен $P_{\mathrm{ref}} = 2.78227853520921$ безразмерных единиц времени, то есть приблизительно 12 дней. Параметризуем точки орбиты параметром $\tau \in [0, P_{\mathrm{ref}}]$ так, что $\mathbf{x}_{\mathrm{ref}}(\tau)$ — фазовое состояние на орбите в момент времени t = τ и $\mathbf{x}_{\mathrm{ref}}(0) = \mathbf{x}_{\mathrm{ref},0}$. Под окрестностью орбиты будем понимать область фазового пространства

$$\Omega_{\mathrm{vic}} = \{\mathbf{x} = [\mathbf{r}, \mathbf{v}] \in \mathbb{R}^6 : \exists \tau \in [0, P_{\mathrm{ref}}] \| \mathbf{r} - \mathbf{r}_{\mathrm{ref}}(\tau)| \le R_{\mathrm{vic}},$$
$$|\mathbf{v} - \mathbf{v}_{\mathrm{ref}}(\tau)| \le V_{\mathrm{vic}}\},$$

где $\mathbf{r}_{\mathrm{ref}}(\tau)$ и $\mathbf{v}_{\mathrm{ref}}(\tau)$ — положение и скорость в векторе $\mathbf{x}_{\mathrm{ref}}(\tau)$, а размеры окрестности по положению и скорости выбраны равными $R_{\mathrm{vic}} = 100$ км, $V_{\mathrm{vic}} = 0.1$ м/с. Наконец, будем считать, что состояние аппарата в моменты управления известно со среднеквадратичными ошибками $\sigma_r = 1$ км по положению и $\sigma_v = 0.01$ м/с по скорости.

Управление осуществляется следующим образом. Дана навигационная оценка состояния космического аппарата в окрестности орбиты. На основании этой оценки в соответствии с законом управления рассчитывается импульс скорости, нацеливающий аппарат в окрестность орбиты через четверть витка. Далее процесс управления повторяется. Задача состоит в поиске закона управления, поддерживающего таким образом движение в окрестности орбиты.

Сформулируем и решим эту задачу в терминах теории обучения с подкреплением. Состояние из теории обучения с подкреплением определим как фазовый вектор механической системы, то есть $\mathbf{s} = \mathbf{x} = [x, y, z, v_x, v_y, v_z]$.

Инициализация состояния в окрестности орбиты происходит следующим образом. Орбита дискретизируется, то есть представляется в виде конечного числа точек $\mathbf{x}_{\mathrm{ref}}(\tau_i)$, $i = 1,\ldots,1000$, с равномерным по τ разбиением. Случайным образом выбирается одна из 1000 точек $\mathbf{x}_{\mathrm{ref}}(\tau_i)$ и в соответствии с равномерным распределением в шаре радиуса 100 км вокруг нее генерируется отклонение по положению $\delta\mathbf{r}$, а в шаре радиуса 0.1 м/с — скорость $\delta\mathbf{v}$. Начальное состояние определяется как $\mathbf{s} = \mathbf{x}_{\mathrm{ref}}(\tau_i) + [\delta\mathbf{r}, \delta\mathbf{v}]$.

Траектория аппарата получается численным интегрированием уравнений движения. Для этого используется реализация метода Рунге – Кутты 8-го порядка с адаптивным шагом DOP853 [42]. Дискретный шаг представляет собой интегрирование уравнений движения на интервале времени $[0, P_{\mathrm{ref}}/4]$. Будем считать на этапе обучения, что эпизод состоит из одного шага. Из-за неустойчивости движения выбор большего числа шагов в эпизоде приводит к быстрому удалению траектории от орбиты и процесс обучения на получаемых данных затрудняется. Тестирование обученной стратегии можно осуществлять на эпизодах с большим числом шагов.

Функция вознаграждения определяется следующим образом:

$$r_1 = \max(-1, -|\Delta\mathbf{x}_1| \cdot 10^3),$$

где $|\Delta\mathbf{x}_1|$ — это минимальное расстояние в фазовом пространстве от проинтегрированного состояния $\mathbf{x}_1$ до орбиты, то есть

$$|\Delta\mathbf{x}_1| = |\mathbf{x}_1 - \mathbf{x}_{\mathrm{ref}}(\tau_{i^*})|, i^* = \arg\min_i |\mathbf{x}_1 - \mathbf{x}_{\mathrm{ref}}(\tau_i)|.$$

Нормировочный коэффициент 10^3 и ограничение снизу -1 выставляются для того, чтобы значения вознаграждения лежали в интервале $[-1, 0]$. Нормировка и ограничение вознаграждения положительно влияют на процесс обучения и аппроксимацию функции ценности. Кроме того, ограничение функции вознаграждения позволяет определять объем выборки для доверительного оценивания суммарных вознаграждений (см. раздел 5), и чем меньше дисперсия суммарного вознаграждения, тем более устойчивым является процесс обучения. Так как эпизод состоит из одного шага, то суммарное вознаграждение за эпизод $R = r_1$.

Наблюдением считается вектор

$$\mathbf{o} = [(\mathbf{x}' - \mathbf{x}_{\text{ref}}(\tau_{i^*})) \cdot 10^3, \cos\varphi_{i^*}, \sin\varphi_{i^*}] \in \mathbb{R}^8,$$

где оценка состояния $\mathbf{x}' = \mathbf{x} + \xi$, $\xi \sim \mathcal{N}(\mathbf{0}, \Sigma_{\text{nav}})$ — вектор ошибок определения состояния, $\Sigma_{\text{nav}} = \text{diag}(\sigma_r, \sigma_r, \sigma_r, \sigma_v, \sigma_v, \sigma_v)$ — ковариационная матрица ошибок навигации, $i^* = \arg\min_i |\mathbf{x}' - \mathbf{x}_{\text{ref}}(\tau_i)|$ — номер ближайшей к $\mathbf{x}'$ точки на орбите, $\varphi_{i^*} = 2\pi \cdot \tau_{i^*} / P_{\text{ref}}$, 10^3 — нормировочный коэффициент. Первые шесть компонент этого вектора представляют собой отклонение в фазовом пространстве от ближайшей точки на орбите. Две последние компоненты определяют абсолютное положение этой точки на орбите в пространстве.

Выбор такого вектора наблюдения вызван необходимостью нормирования входного вектора в функцию стратегии для сходимости процесса обучения. Выбор $\mathbf{o} = \mathbf{x} + \xi$ в качестве вектора наблюдения приводит к тому, что значения этого вектора распределены в относительно узкой окрестности орбиты и стратегия во время обучения не способна различить близкие значения этого вектора. Поэтому здесь предлагается использовать масштабированную локальную информацию об отклонении от орбиты и информацию об абсолютном положении точки, в окрестности которой происходит наблюдение. Выбор компонент $\cos\varphi$ и $\sin\varphi$ обусловлено желанием обеспечить периодичность управления по этому углу.

Действие и импульсы скорости свяжем равенством

$$\Delta\mathbf{v} = \mathbf{a} \cdot 6 \cdot 10^{-4} \text{ / VU}.$$

Эта нормировка делается для того, чтобы значения компонент действия $\mathbf{a}$ лежали в пределах [–1,1], что способствует сходимости процесса оптимизации. Компоненты импульсов скорости лежат в пределах от –0.6 до 0.6 м/с. Нормировочный коэффициент подбирался автором вручную апостериорно. Модель для действий определим следующим образом:

$$\mathbf{a} = \mathbf{A}_2 \text{th}(\mathbf{A}_1\mathbf{o} + \mathbf{a}_1) + \mathbf{a}_2,$$

где $\mathbf{o} \in \mathbb{R}^n$ — вектор наблюдения, а матрицы и векторы $\mathbf{A}_1 \in \mathbb{R}^{16\times8}$, $\mathbf{a}_1 \in \mathbb{R}^{16}$, $\mathbf{A}_2 \in \mathbb{R}^{3\times16}$, $\mathbf{a}_2 \in \mathbb{R}^3$ — обучаемые параметры модели. Модель функции ценности определим с похожей архитектурой:

$$V = \mathbf{b}_2 \text{th}(\mathbf{B}_1\mathbf{o} + \mathbf{b}_1) + c_2,$$

где $\mathbf{B}_1 \in \mathbb{R}^{16\times8}$, $\mathbf{b}_1 \in \mathbb{R}^{16}$, $\mathbf{b}_2 \in \mathbb{R}^{16}$, $c_2 \in \mathbb{R}$. Модель действий имеет таким образом $16 \cdot 8 + 16 + 3 \cdot 16 + 3 = 195$ параметров, модель функции ценности имеет $16 \cdot 8 + 16 + 1 \cdot 16 + 1 = 161$ параметр.

Обучение моделей произведем с помощью реализации метода PPO из библиотеки stable-baselines3. Объем выборки для аппроксимации среднего значения функционала R (опция n_steps) выберем равным 10000. Число итераций градиентного метода для коррекции весов нейросетевых моделей (опция n_epochs) выберем равным 30, а скорость обучения (опция learning_rate) – 0.005. Обучение будем производить на центральном процессоре и завершим, когда число дискретных шагов достигнет 10 млн (опция total_timesteps).

На рис. 5 показано среднее суммарное вознаграждение за эпизод как функция шага. Вознаграждение в среднем растет по мере обучения моделей и в конце оптимизации колеблется на величину ±0.01 и в среднем равно 0.29. На рис. 6 изображен график зависимости от номера шага

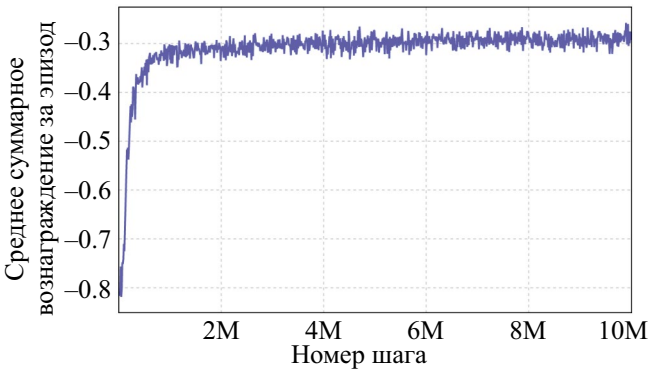

Рис. 5. Среднее вознаграждение за эпизод в зависимости от шага.

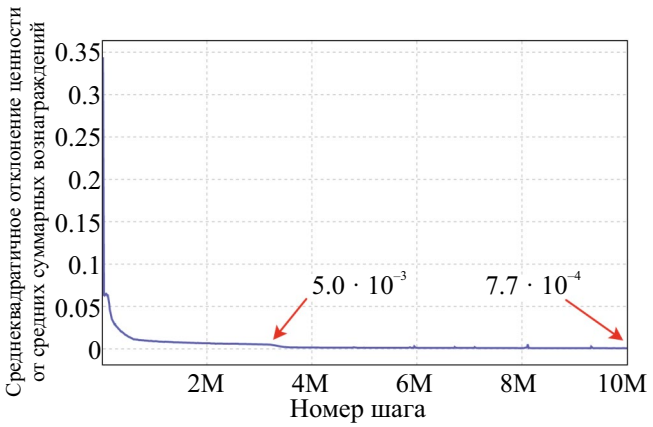

Рис. 6. Среднеквадратичное отклонение модели функции ценности от средних суммарных вознаграждений.

среднеквадратичного отклонения модели функции ценности от средних суммарных вознаграждений. Отклонение падает с ростом числа шагов, финальное значение равно $7.7\cdot10^{-4}$. Указано также значение отклонения после первого плато значений: $5.0\cdot10^{-3}$.

Оценка качества обученной модели управления производилась в эпизодах с последовательными 4 шагами (полный виток в окрестности орбиты). Оценивалась вероятность выхода аппарата за окрестность орбиты в результате маневрирования, а также затраты характеристической скорости на каждый импульс. Так как значение вероятности принадлежит интервалу [0,1], для оценивания ее с точностью 0.1% на уровне доверия 99.9% согласно табл. 1 достаточно произвести 3800452 измерения (шага), что соответствует 950113 эпизодам. Максимальное значение импульса скорости равно $\sqrt{3\cdot0.6^2}\le1.04$ м/с, поэтому для оценки затрат скорости с точностью до 0.001 м/с достаточно смоделировать $950113\cdot1.04^2=1027643$ эпизода.

В итоге было смоделировано 1 028 000 эпизодов. Результаты оценок промаха по положению мимо окрестности орбиты и затраты характеристической скорости приведены в табл. 2–5. Здесь q_0 — минимальное значение величины из встреченных; $q_{0.25}$ — величина, ниже которой находятся 25% встреченных величин; $q_{0.5}$ — медиана; $q_{0.75}$ — величина, выше которой находятся 25% встреченных величин; q_1 — максимальное значение величины из встреченных; μ — среднее арифметическое всех величин. Средние затраты характеристической скорости для каждого импульса равны:

1) первый импульс: 0.229 ± 0.001 м/с с вероятностью не менее 99.9%,

2) второй импульс: 0.265 ± 0.001 м/с с вероятностью не менее 99.9%,

3) третий импульс: 0.192 ± 0.001 м/с с вероятностью не менее 99.9%,

4) четвертый импульс: 0.143 ± 0.001 м/с с вероятностью не менее 99.9%.

Так как суммарные затраты топлива за виток ограничены величиной 4.16 м/с, то с учетом $n=1028000$ в соответствии с неравенством Хефдинга можно рассмотреть $\varepsilon = 0.008$ и

Таблица 2. Квантили и средние значения распределений промаха по положению мимо орбиты Δr и затраты характеристической скорости Δv после первого импульса

	q_0	$q_{0.25}$	$q_{0.5}$	$q_{0.75}$	q_1	μ
Δr, км	0.2811	30.3540	45.7622	61.7558	158.6183	46.5785
Δv, м/с	0.0002	0.1307	0.2074	0.3055	0.8137	0.2288

Таблица 3. Квантили и средние значения распределений промаха по положению мимо орбиты Δr и затраты характеристической скорости Δv после второго импульса

	q_0	$q_{0.25}$	$q_{0.5}$	$q_{0.75}$	q_1	μ
Δr, км	0.1526	20.4551	32.5251	46.4348	224.4787	34.5548
Δv, м/с	0.0004	0.1629	0.2531	0.3563	0.8485	0.2655

Таблица 4. Квантили и средние значения распределений промаха по положению мимо орбиты Δr и затраты характеристической скорости Δv после третьего импульса

	q_0	$q_{0.25}$	$q_{0.5}$	$q_{0.75}$	q_1	μ
Δr, км	0.2323	14.9570	23.1509	34.4176	769.9048	25.8457
Δv, м/с	0.0002	0.1077	0.1760	0.2569	0.8485	0.1916

Таблица 5. Квантили и средние значения распределений промаха по положению мимо орбиты Δr и затраты характеристической скорости Δv после четвертого импульса

	q_0	$q_{0.25}$	$q_{0.5}$	$q_{0.75}$	q_1	μ
Δr, км	0.1354	11.9981	17.7098	25.5513	3717.2211	20.2541
Δv, м/с	0.0002	0.0740	0.1216	0.1914	0.8485	0.1430

$p = 2\exp(-2\varepsilon^2 n) = 0.001$, то есть 0.829 ± 0.008 м/с с вероятностью не менее 99.9%. Экстраполяция этих значений на один год дает 25.21 ± 0.25 м/с. Для сравнения, поддержание близкой по размерам орбиты в миссии *ARTEMIS* в годовом выражении потребовало порядка 7.39 м/с [43]. Затраты на поддержание можно уменьшить, если добавить в вознаграждение затраты скорости на импульс и выбрать коэффициент перед соответствующим слагаемым. В настоящей работе такое исследование не проводится.

Результаты оценки вероятности выйти за пределы окрестности орбиты по положению (отклонение более 100 км) получились следующими:

1) после первого импульса: $0.4\% \pm 0.1\%$ с вероятностью не менее 99.9%,

2) после второго импульса: $0.2\% \pm 0.1\%$ с вероятностью не менее 99.9%,

3) после третьего импульса: $0.05\% \pm 0.1\%$ с вероятностью не менее 99.9%,

4) после четвертого импульса: $0.1\% \pm 0.1\%$ с вероятностью не менее 99.9%.

Вероятность выйти за пределы окрестности орбиты на произвольном шаге равна $0.2\% \pm 0.1\%$ с вероятностью не менее 99.9%.

ЗАКЛЮЧЕНИЕ

Сформулирована методика сведения общей задачи оптимального управления космическим аппаратом к задаче машинного обучения с подкреплением. Методика состоит из нескольких шагов: 1) определение состояния и действия в терминах переменных механической задачи; 2) определение распределения начальных фазовых состояний; 3) определение дискретного шага системы; 4) определение функции вознаграждения; 5) определение модели восприятия; 6) определение модели управления. Приведены строгие подходы к оценке математического ожидания случайных величин и вероятности наступления событий на основе неравенства Хефдинга. Рассмотрены два примера применения методики: 1) к простой динамической системе, 2) к задаче поддержания неустойчивой орбиты вокруг точки либрации. В первом примере показана близость получаемых решений к теоретическим значениям. Во втором примере приведены результаты обучения моделей управления, строго оценены средние затраты характеристической скорости и вероятность неудачи поддержания орбиты.

СПИСОК ЛИТЕРАТУРЫ

1. *Понтрягин Л.В.* Принцип максимума в оптимальном управлении. Москва: Едиториал УРСС, 2004.
2. *Александров В.В., Болтянский В.Г., Лемак С.С. и др.* Оптимальное управление движением. Москва: ФИЗМАТЛИТ, 2005.
3. *Егоров А.И.* Основы теории управления. Москва: ФИЗМАТЛИТ, 2004.
4. *Беллман Р., Калаба Р.* Динамическое программирование и современная теория управления. Москва: Наука, 1969.
5. *Bertsekas D.P.* Dynamic programming and optimal control. Volume I. Belmont: Athena Scientific, 2005.
6. *Bertsekas D.P.* Dynamic programming and optimal control. Volume II. Belmont: Athena Scientific, 2007.
7. *Саттон Р.С., Барто Э.Г.* Обучение с подкреплением. Москва: Бином. Лаборатория знаний, 2017.
8. *Bertsekas D.P.* Reinforcement learning and optimal control. Belmont: Athena Scientific, 2019.
9. *Kamalapurkar R., Walters P., Rosenfeld J. et al.* Reinforcement Learning for Optimal Feedback Control. A Lyapunov-Based Approach. Cham: Springer, 2018.
10. *Gurfil P., Idan M., Kasdin N.J.* Adaptive neural control of deep-space formation flying // J. Guidance, Control, and Dynamics. 2003. V. 26. Iss. 3. P. 491–501. DOI: https://dx.doi.org/10.2514/2.5072.
11. *Leeghim H., Choi Y., Bang H.* Adaptive attitude control of spacecraft using neural networks // Acta Astronautica. 2009. V. 64. Iss. 7–8. P. 778–786. DOI: https://dx.doi.org/10.1016/j.actaastro.2008.12.004.
12. *Zeng W., Wang Q.* Learning from adaptive neural network control of an underactuated rigid spacecraft // Neurocomputing. 2015. V. 168. P. 690–697. DOI: https://dx.doi.org/10.1016/j.neucom.2015.05.055.
13. *Li S., Jiang X.* RBF neural network based second-order sliding mode guidance for Mars entry under uncertainties // Aerospace Science and Technology. 2015. V. 43. P. 226–235. DOI: https://dx.doi.org/10.1016/j.ast.2015.03.006}{10.1016/j.ast.2015.03.006.
14. *Wang C., Hill D.J.* Deterministic learning theory for identification, recognition, and control. Boca Raton: CRC Press, 2010.
15. *Bertsekas D.P, Tsitsiklis J.N.* Neuro-Dynamic Programming. Belmont: Athena Scientific, 1996.
16. *Shirobokov M., Trofimov S., Ovchinnikov M.* Survey of machine learning techniques in spacecraft control design // Acta Astronautica. 2021. V. 186. P. 87–97. DOI: https://doi.org/10.1016/j.actaastro.2021.05.018.
17. *Gaudet B., Linares R., Furfaro R.* Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations // Acta Astronautica. 2020. V. 171. P. 1–13. DOI: https://doi.org/10.1016/j.actaastro.2020.02.036.
18. *Gaudet B., Linares R., Furfaro R.* Adaptive guidance and integrated navigation with reinforcement meta-learning // Acta Astronautica. 2020.

V. 169. P. 180–190. DOI: https://doi.org/10.1016/j.actaastro.2020.01.007.

19. *Scorsoglio A., D'Ambrosio A., Ghilardi L. et al.* Image-based deep reinforcement meta-learning for autonomous lunar landing // J. Spacecraft and Rockets. 2022. V. 59. Iss. 1. P. 153–165. DOI: https://doi.org/10.2514/1.A35072.
20. *Gaudet B., Linares R., Furfaro R.* Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning // Acta Astronautica. 2020. V. 172. P. 90–99. DOI: https://doi.org/10.1016/j.actaastro.2020.03.026.
21. *Лидов М.Л., Ляхова В.А.* Гарантирующий синтез управления для стабилизации движения космического аппарата в окрестности неустойчивых точек либрации // Космические исследования. 1992. Т. 30. № 5. С. 579–595.
22. *Silver D., Lever G., Heess N. et al.* Deterministic policy gradient algorithms // Proc. 31st International Conference on Machine Learning. 2014. V. 32. Iss. 1. P. 387–395. URL: http://proceedings.mlr.press/v32/silver14.html.
23. *Mnih V., Badia A.P., Mirza M. et al.* Asynchronous Methods for Deep Reinforcement Learning // Proc. 33rd International Conference on Machine Learning. 2016. V. 48. P. 1928–1937. URL: https://proceedings.mlr.press/v48/mniha16.html.
24. *Schulman J., Wolski F., Dhariwal P. et al.* Proximal Policy Optimization Algorithms // arXiv preprint. 2017. 1707.06347. URL: https://arxiv.org/abs/1707.06347.
25. *Moriarty D.E., Schultz A.C., Grefenstette J.J.* Evolutionary algorithms for reinforcement learning // J. Artificial Intelligence Research. 1999. V. 11. P. 241–276.
26. *Sehgal A., La H., Louis S. et al.* Deep reinforcement learning using genetic algorithm for parameter optimization // Proc. 3[d] IEEE International Conference on Robotic Computing (IRC 2019). P. 596–601. DOI: https://doi.org/10.1109/IRC.2019.00121.
27. *Sutton R.S., McAllester D.A., Singh S.P. et al.* Policy gradient methods for reinforcement learning with function approximation // Advances in Neural Information Processing Systems 12 (NIPS 1999). 1999. P. 1057–1063. URL: https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.
28. *Cybenko G.* Approximation by superpositions of a sigmoidal function // Mathematics of Control, Signals, and Systems. 1989. V. 2. Iss. 4. P. 303–314. DOI: https://doi.org/10.1007/BF02551274.
29. *Leshno M., Lin V.Ya., Pinkus A. et al.* Multilayer feedforward networks with a nonpolynomial activation function can approximate any function // Neural Networks. 1993. V. 6. Iss. 6. P. 861–867. DOI: https://doi.org/10.1016/S0893-6080(05)80131-5.
30. *Pinkus A.* Approximation theory of the MLP model in neural networks // Acta Numerica. 1999. V. 8. P. 143–195. DOI: https://doi.org/10.1017/S0962492900002919.
31. *Kidger P., Lyons T.* Universal Approximation with Deep Narrow Networks // Proc. Machine Learning Research. 2020. V. 125. P. 1–22. URL: http://proceedings.mlr.press/v125/kidger20a/kidger20a.pdf.
32. *Hoeffding W.* Probability inequalities for sums of bounded random variables // J. American Statistical Association. 1963. V. 58. Iss. 301. P. 13–30. DOI: https://doi.org/10.1080/01621459.1963.10500830.
33. Gymnasium // Веб-страница документации программной библиотеки Gymnasium (https://gymnasium.farama.org/index.html). Просмотрено: 18.09.2023.
34. Stable-Baselines3 // Веб-страница документации программной библиотеки Stable-Baselines3 (https://stable-baselines3.readthedocs.io/en/master/). Просмотрено: 18.09.2023.
35. Pytorch // Сайт программной библиотеки Pytorch (https://pytorch.org/). Просмотрено: 18.09.2023.
36. *Jones D.R., Schonlau M., Welch W.J.* Efficient global optimization of expensive black-box functions // Journal of Global optimization. 1998. V. 13. P. 455–492. DOI: https://doi.org/10.1023/A:1008306431147.
37. *Bergstra J.S., Bardenet R., Bengio Y. et al.* Algorithms for Hyper-Parameter Optimization // Advances in Neural Information Processing Systems 24 (NIPS 2011). 2011. P. 2546–2554. URL: https://papers.nips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.
38. *Akiba T., Sano S., Yanase T. et al.* Optuna: A next-generation hyperparameter optimization framework // Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining. 2019. P. 2623–2631. DOI: https://doi.org/10.1145/3292500.3330701.
39. *Liaw R., Liang E., Nishihara R. et al.* Tune: A research platform for distributed model selection and training // arXiv preprint. 2018. 1807.05118. URL: https://arxiv.org/pdf/1807.05118.pdf.
40. *Balandat M., Karrer B., Jiang D. et al.* BoTorch: A framework for efficient Monte-Carlo Bayesian optimization // Advances in Neural Information Processing Systems 33. 2020. P. 21524–21538. URL: https://proceedings.neurips.cc/paper/2020/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf.
41. *Bergstra J., Yamins D., Cox D.D.* Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures // Proc. 30th International Conference on Machine Learning. 2013. V. 28. P. 115–123. URL: http://proceedings.mlr.press/v28/bergstra13.pdf.
42. *Hairer E., Wanner G.* Solving Ordinary Differential Equations I. Nonstiff Problems. Heidelberg: Springer, 2008.
43. *Folta D.C., Pavlak T.A., Haapala A.F. et al.* Earth–Moon Libration Point Orbit Stationkeeping: Theory, Modeling, and Operations // Acta Astronautica. 2014. V. 94. Iss. 1. P. 421–433.